



## **LatticeECP3 Family Handbook**

---

HB1009 Version 05.2, July 2013

May 2013

### Section I. LatticeECP3 Family Data Sheet

#### Introduction

Features .....	1-1
Introduction .....	1-2

#### Architecture

Architecture Overview .....	2-1
PFU Blocks .....	2-2
Slice .....	2-3
Modes of Operation.....	2-5
Routing.....	2-6
sysCLOCK PLLs and DLLs.....	2-6
General Purpose PLL.....	2-6
Delay Locked Loops (DLL).....	2-7
PLL/DLL Cascading .....	2-10
PLL/DLL PIO Input Pin Connections.....	2-10
Clock Dividers .....	2-10
Clock Distribution Network .....	2-11
Primary Clock Sources.....	2-11
Primary Clock Routing .....	2-13
Dynamic Clock Control (DCC) .....	2-13
Dynamic Clock Select (DCS) .....	2-13
Secondary Clock/Control Sources .....	2-14
Secondary Clock/Control Routing.....	2-14
Slice Clock Selection.....	2-16
Edge Clock Sources.....	2-17
Edge Clock Routing .....	2-17
sysMEM Memory .....	2-19
sysMEM Memory Block.....	2-19
Bus Size Matching .....	2-19
RAM Initialization and ROM Operation .....	2-19
Memory Cascading .....	2-19
Single, Dual and Pseudo-Dual Port Modes.....	2-20
Memory Core Reset.....	2-20
sysDSP™ Slice .....	2-20
sysDSP Slice Approach Compared to General DSP .....	2-20
LatticeECP3 sysDSP Slice Architecture Features .....	2-21
MULT DSP Element.....	2-24
MAC DSP Element.....	2-25
MMAC DSP Element.....	2-26
MULTADDSUB DSP Element.....	2-27
MULTADDSUBSUM DSP Element.....	2-28
Advanced sysDSP Slice Features .....	2-29
Cascading .....	2-29
Addition .....	2-29
Rounding.....	2-29
ALU Flags .....	2-30
Clock, Clock Enable and Reset Resources .....	2-30
Resources Available in the LatticeECP3 Family .....	2-30



Programmable I/O Cells (PIC) .....	2-31
PIO .....	2-32
Input Register Block .....	2-32
Output Register Block .....	2-34
Tristate Register Block .....	2-35
ISI Calibration .....	2-35
Control Logic Block .....	2-36
DDR Memory Support .....	2-36
Left and Right Edges .....	2-36
Bottom Edge .....	2-36
Top Edge .....	2-36
DLL Calibrated DQS Delay Block .....	2-36
Polarity Control Logic .....	2-39
DDR3 Memory Support .....	2-39
sys/O Buffer .....	2-40
sys/O Buffer Banks .....	2-40
Typical sys/O I/O Behavior During Power-up .....	2-42
Supported sys/O Standards .....	2-42
On-Chip Programmable Termination .....	2-43
Equalization Filter .....	2-44
Hot Socketing .....	2-44
SERDES and PCS (Physical Coding Sublayer) .....	2-44
SERDES Block .....	2-46
PCS .....	2-46
SCI (SERDES Client Interface) Bus .....	2-46
Flexible Quad SERDES Architecture .....	2-47
IEEE 1149.1-Compliant Boundary Scan Testability .....	2-48
Device Configuration .....	2-48
Soft Error Detect (SED) Support .....	2-49
External Resistor .....	2-49
On-Chip Oscillator .....	2-49
Density Shifting .....	2-50
<b>DC and Switching Characteristics</b>	
Absolute Maximum Ratings .....	3-1
Recommended Operating Conditions .....	3-1
Hot Socketing Specifications .....	3-2
Hot Socketing Requirements .....	3-2
ESD Performance .....	3-2
DC Electrical Characteristics .....	3-3
LatticeECP3 Supply Current (Standby) .....	3-4
SERDES Power Supply Requirements .....	3-5
sys/O Recommended Operating Conditions .....	3-6
sys/O Single-Ended DC Electrical Characteristics .....	3-7
sys/O Differential Electrical Characteristics .....	3-8
LVDS25 .....	3-8
Differential HSTL and SSTL .....	3-8
LVDS25E .....	3-9
LVCMOS33D .....	3-9
BLVDS25 .....	3-10
LVPECL33 .....	3-11
RSDS25E .....	3-12
MLVDS25 .....	3-13
Typical Building Block Function Performance .....	3-14
Pin-to-Pin Performance (LVCMOS25 12mA Drive) .....	3-14

Derating Timing Tables .....	3-15
LatticeECP3 External Switching Characteristics .....	3-16
LatticeECP3 Internal Switching Characteristics .....	3-25
Timing Diagrams .....	3-27
LatticeECP3 Family Timing Adders .....	3-29
LatticeECP3 Maximum I/O Buffer Speed .....	3-32
sysCLOCK PLL Timing .....	3-34
DLL Timing .....	3-35
SERDES High-Speed Data Transmitter .....	3-36
SERDES/PCS Block Latency .....	3-38
SERDES High Speed Data Receiver .....	3-39
Input Data Jitter Tolerance .....	3-39
SERDES External Reference Clock .....	3-41
PCI Express Electrical and Timing Characteristics .....	3-44
AC and DC Characteristics .....	3-44
XAUI/Serial Rapid I/O Type 3/CPRI LV E.30 Electrical and Timing Characteristics .....	3-45
AC and DC Characteristics .....	3-45
Serial Rapid I/O Type 2/CPRI LV E.24 Electrical and Timing Characteristics .....	3-47
AC and DC Characteristics .....	3-47
Gigabit Ethernet/Serial Rapid I/O Type 1/SGMII/CPRI LV E.12 Electrical and Timing Characteristics .....	3-48
AC and DC Characteristics .....	3-48
SMPTE SD/HD-SDI/3G-SDI (Serial Digital Interface) Electrical and Timing Characteristics .....	3-49
AC and DC Characteristics .....	3-49
HDMI (High-Definition Multimedia Interface) Electrical and Timing Characteristics .....	3-50
AC and DC Characteristics .....	3-50
LatticeECP3 sysCONFIG Port Timing Specifications .....	3-52
JTAG Port Timing Specifications .....	3-58
Switching Test Conditions .....	3-59
sysI/O Differential Electrical Characteristics .....	3-60
Transition Reduced LVDS (TRLVDS DC Specification) .....	3-60
Mini LVDS .....	3-60
Point-to-Point LVDS (PPLVDS) .....	3-61
RSDS .....	3-61
<b>Pinout Information</b>	
Signal Descriptions .....	4-1
PICs and DDR Data (DQ) Pins Associated with the DDR Strobe (DQS) Pin .....	4-4
Pin Information Summary (Cont.) .....	4-6
Pin Information Summary (Cont.) .....	4-7
Pin Information Summary (Cont.) .....	4-9
Package Pinout Information .....	4-10
Thermal Management .....	4-10
For Further Information .....	4-10
<b>Ordering Information</b>	
LatticeECP3 Part Number Description .....	5-1
Ordering Information .....	5-1
LatticeECP3 Devices, Green and Lead-Free Packaging .....	5-2
<b>Supplemental Information</b>	
For Further Information .....	6-1
<b>Revision History</b>	

**Section II. LatticeECP3 Family Technical Notes**
**LatticeECP3 SERDES/PCS Usage Guide**

Introduction .....	8-1
Features .....	8-1
New Features Over LatticeECP2M™ SERDES/PCS .....	8-1
Using This Technical Note .....	8-2
Standards Supported .....	8-2
Architecture Overview .....	8-3
PCS Quads and Channels .....	8-4
Per Channel SERDES/PCS and FPGA Interface Ports .....	8-4
Detailed Channel Block Diagram .....	8-5
Clocks and Resets .....	8-6
Transmit Data Bus .....	8-6
Receive Data Bus .....	8-7
Mode-Specific Control/Status Signal Descriptions .....	8-8
SERDES/PCS .....	8-9
I/O Descriptions .....	8-10
SERDES/PCS Functional Description .....	8-13
SERDES .....	8-13
Equalizer .....	8-13
Pre-Emphasis .....	8-13
Reference Clocks .....	8-14
SERDES Clock Architecture .....	8-14
Rate Modes .....	8-15
Reference Clock from an FPGA Core .....	8-16
Full Data, Div 2 and Div 11 Data Rates .....	8-17
Dynamic Switching Between Full Rate and Half Rate (DIV2) .....	8-18
Reference Clock Sources .....	8-18
Spread Spectrum Clocking (SSC) Support .....	8-18
Loss of Signal .....	8-19
Loss Of Lock .....	8-19
TX Lane-to-Lane Skew .....	8-19
SERDES PCS Configuration Setup .....	8-20
Auto-Configuration File .....	8-20
Transmit Data .....	8-20
Receive Data .....	8-20
8b10b Decoder .....	8-21
External Link State Machine Option .....	8-21
Idle Insert for Gigabit Ethernet Mode .....	8-22
Clock Tolerance Compensation .....	8-22
Calculating Minimum Interpacket Gap .....	8-26
PCS Module Generation in IPexpress .....	8-28
8-Bit and 10-Bit SERDES-Only Modes .....	8-40
Generic 8b10b Mode .....	8-40
LatticeECP3 PCS in Gigabit Ethernet and SGMII Modes .....	8-42
XAUI Mode .....	8-43
LatticeECP3 PCS in PCI Express Revision 1.1 (2.5Gpbs) Mode .....	8-44
PCI Express Beacon Support .....	8-48
SDI (SMPTE) Mode .....	8-49
Serial RapidIO (SRIO) Mode .....	8-50
Serial Digital Video and Out-Of-Band Low Speed SERDES Operation .....	8-51
Open Base Station Architecture Initiative (OBSAI) .....	8-52
Common Public Radio Interface (CPRI) .....	8-53
SONET/SDH .....	8-55

FPGA Interface Clocks.....	8-56
Case I_a: 8/10-Bit, CTC FIFO and RX/TX FIFOs Not Bypassed.....	8-59
Case I_b: 8/10-Bit, CTC FIFO Bypassed.....	8-60
Case I_c: 8/10-Bit, RX/TX FIFO Bypassed.....	8-61
Case I_d: 8/10-Bit, CTC FIFO and RX/TX FIFOs Bypassed.....	8-62
Case II_a: 16/20-bit, CTC FIFO and RX/TX FIFOs NOT Bypassed.....	8-63
Case II_b: 16/20-bit, CTC FIFO Bypassed.....	8-64
SERDES/PCS Block Latency.....	8-65
SERDES Client Interface.....	8-66
Interrupts and Status.....	8-68
SERDES Debug Capabilities.....	8-69
PCS Loopback Modes.....	8-69
ORCAstra.....	8-71
Other Design Considerations.....	8-74
16/20-Bit Word Alignment.....	8-74
SERDES/PCS RESET.....	8-79
Reset Sequence and Reset State Diagram.....	8-79
Reset Sequence Generation.....	8-79
Lock Status Signals Definitions.....	8-79
TX Reset Sequence.....	8-79
RX Reset Sequence.....	8-79
References.....	8-82
Technical Support Assistance.....	8-82
Revision History.....	8-82
Appendix A. Configuration Registers.....	8-85
Quad Registers Overview.....	8-85
Per Quad PCS Control Registers Details.....	8-85
Per Quad PCS Control Registers Details.....	8-87
Per Quad Reset and Clock Control Registers Details.....	8-89
Per Quad PCS Status Registers Details.....	8-89
Per Quad SERDES Status Registers Details.....	8-91
Channel Registers Overview.....	8-92
Per Channel PCS Control Registers Details.....	8-93
Per Channel SERDES Control Registers Details.....	8-97
Per Channel Reset and Clock Control Registers Details.....	8-100
Per Channel PCS Status Registers Details.....	8-101
Per Channel SERDES Status Registers Details.....	8-102
Appendix B. Register Settings for Various Standards.....	8-104
Per Channel Register Settings for Various Standards.....	8-104
Per Quad Register Settings for Various Standards.....	8-104
Appendix C. Attribute Cross Reference Table.....	8-105
Appendix D. Lattice Diamond Overview.....	8-110
Converting an ispLEVER Project to Lattice Diamond.....	8-110
Importing an ispLEVER Design Project.....	8-110
Adjusting PCS Modules.....	8-110
Regenerate PCS Modules.....	8-110
Using IPexpress with Lattice Diamond.....	8-111
Creating a New Simulation Project Using Simulation Wizard.....	8-112
<b>LatticeECP3 sysIO Usage Guide</b> .....	
Introduction.....	9-1
sysIO Buffer Overview.....	9-1
Supported sysIO Standards.....	9-1
sysIO Banking Scheme.....	9-3
V <sub>CCIO</sub> (1.2V/1.5V/1.8V/2.5V/3.3V).....	9-4

V <sub>CCAUX</sub> (3.3V) .....	9-5
V <sub>CCJ</sub> (1.2V/1.5V/1.8V/2.5V/3.3V) .....	9-5
Input Reference Voltage (V <sub>REF1</sub> , V <sub>REF2</sub> ) .....	9-5
VREF1 for DDR Memory Interface .....	9-5
VTT Termination Voltage .....	9-5
Hot Socketing Support .....	9-5
Mixed Voltage Support in a Bank .....	9-6
sysIO Standards Supported Per Bank .....	9-7
sysIO Buffer Configurations .....	9-8
Bus Maintenance Circuit .....	9-8
Programmable Drive .....	9-8
Programmable Slew Rate .....	9-8
Open Drain Control .....	9-8
Differential SSTL and HSTL Support .....	9-9
PCI Support with Programmable PCICLAMP .....	9-9
Differential I/O Support .....	9-9
Differential SSTL and HSTL .....	9-9
Differential LVCMOS33 .....	9-9
GTL+ Input Support .....	9-9
On-Chip Termination .....	9-10
Equalization Setting .....	9-10
Software sysIO Attributes .....	9-10
IO_TYPE .....	9-10
OPENDRAIN .....	9-11
DRIVE .....	9-12
DIFFDRIVE .....	9-12
MULTDRIVE .....	9-12
TERMINATEVTT .....	9-13
DIFFRESISTOR .....	9-13
EQ_CAL .....	9-13
PULLMODE .....	9-13
PCICLAMP .....	9-13
SLEWRATE .....	9-14
INBUF .....	9-14
FIXEDEDELAY .....	9-14
DIN/DOUT .....	9-14
LOC .....	9-14
Design Considerations and Usage .....	9-14
Banking Rules .....	9-14
Differential I/O Rules .....	9-15
Technical Support Assistance .....	9-15
Revision History .....	9-16
Appendix A. HDL Attributes .....	9-17
VHDL Synplify Pro .....	9-17
Verilog Synplicity .....	9-20
Appendix B. sysIO Attributes Using the ispLEVER Design Planner User Interface .....	9-22
Appendix C. sysIO Attributes Using the Diamond Spreadsheet View User Interface .....	9-24
<b>LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide</b>	
Introduction .....	10-1
Clock/Control Distribution Network .....	10-1
LatticeECP3 Top-Level View .....	10-1
Primary Clocks .....	10-2
Secondary Clocks .....	10-2
Edge Clocks .....	10-2

---

General Routing for Clocks .....	10-4
Additional Connectivity for Dedicated Clock Resources .....	10-4
Very Small Clock Domains.....	10-5
Static Timing Analysis of General Routing Clocks .....	10-5
Specifying Clocks in the Design Tools .....	10-5
Global Primary Clock and Quadrant Primary Clock .....	10-6
Global Primary Clock .....	10-6
Primary-Pure and Primary-DCS.....	10-6
Quadrant Primary Clock.....	10-6
Global Secondary Clock and Regional Secondary Clocks .....	10-7
Global Secondary Clocks .....	10-7
Regional Secondary Clocks.....	10-7
Secondary Region Clock Preferencing .....	10-7
sysCLOCK™ PLL .....	10-8
Functional Description.....	10-9
PLL Divider and Delay Blocks.....	10-9
PLL Inputs and Outputs .....	10-10
CLKI Input .....	10-10
RST Input.....	10-10
RSTK Input.....	10-11
CLKFB Input.....	10-11
CLKOP Output .....	10-11
CLKOS Output with Phase and Duty Cycle Select .....	10-11
CLKOK Output with Lower Frequency .....	10-11
CLKOK2 Output .....	10-11
LOCK Output.....	10-12
Dynamic Phase and Dynamic Duty Cycle Adjustment.....	10-12
Dynamic Phase Adjustment/Duty Cycle Select.....	10-12
Fine Delay Ports.....	10-13
LatticeECP3 PLL Modules .....	10-13
LatticeECP3 PLL Library Definition.....	10-14
EPLLD Design Migration from LatticeECP2 to LatticeECP3.....	10-14
Dynamic Phase/Duty Mode.....	10-14
PLL Usage in IPexpress.....	10-15
Configuration Tab.....	10-16
PLL Modes of Operation .....	10-18
PLL Clock Injection Removal .....	10-18
PLL Clock Phase Adjustment.....	10-19
IPexpress Output .....	10-19
Notes on PLL Usage .....	10-20
sysCLOCK DLL .....	10-20
DLL Overview.....	10-20
DLL Inputs and Outputs .....	10-20
DLL Attributes .....	10-22
DLL Library Definitions.....	10-22
DLL Library Element I/Os.....	10-23
DLL Modes of Operation .....	10-23
DLL Usage in IPexpress .....	10-24
DLLDEL (Slave Delay Line) .....	10-27
DQSDLL and DQSDEL .....	10-29
Clock Dividers (CLKDIV).....	10-29
CLKDIV Library Definition .....	10-29
CLKDIV Declaration in VHDL Source Code.....	10-30
CLKDIV Usage with Verilog - Example .....	10-31

---

CLKDIV Example Circuits .....	10-31
Reset Behavior.....	10-32
Release Behavior.....	10-32
CLKDIV Inputs-to-Outputs Delay Matching.....	10-33
DCS (Dynamic Clock Select) .....	10-33
DCS Library Definition.....	10-33
DCS Timing Diagrams .....	10-34
DCS Usage with VHDL - Example .....	10-35
DCS Usage with Verilog - Example .....	10-36
Oscillator (OSCF).....	10-36
OSC Library Symbol (OSCF).....	10-37
OSC Usage with VHDL - Example .....	10-37
OSC Usage with Verilog - Example .....	10-37
Setting Clock Preferences.....	10-37
Power Supplies .....	10-37
PLL/DLL Names and Preferred Pads.....	10-37
Technical Support Assistance .....	10-38
Revision History .....	10-39
Appendix A. Primary Clock Sources and Distribution .....	10-40
Appendix B. PLL, CLKIDV and ECLK Locations and Connectivity .....	10-41
Appendix C. Lattice Diamond Usage Overview .....	10-42
Converting an ispLEVER Project to Lattice Diamond .....	10-42
Importing an ispLEVER Design Project .....	10-42
Adjusting PCS Modules .....	10-42
Regenerate PCS Modules .....	10-42
Using IPexpress with Lattice Diamond.....	10-43
Creating a New Simulation Project Using Simulation Wizard .....	10-44
<b>LatticeECP3 Memory Usage Guide</b>	
Introduction .....	11-1
Utilizing IPexpress.....	11-1
IPexpress Flow.....	11-2
Utilizing the PMI .....	11-4
Memory Modules.....	11-5
Single-Port RAM (RAM_DQ) – EBR Based .....	11-5
True Dual-Port RAM (RAM_DP_TRUE) – EBR Based.....	11-11
Pseudo Dual-Port RAM (RAM_DP) – EBR Based.....	11-19
Read Only Memory (ROM) – EBR Based.....	11-22
First In First Out (FIFO) Memory.....	11-26
Dual Clock First-In-First-Out (FIFO_DC) Memory.....	11-32
FIFO_DC Flags .....	11-32
Distributed Single-Port RAM (Distributed_SPRAM) – PFU-Based .....	11-41
Distributed Dual-Port RAM (Distributed_DPRAM) – PFU-Based.....	11-43
Distributed ROM (Distributed_ROM) – PFU-Based .....	11-45
Initializing Memory .....	11-47
Initialization File Formats .....	11-47
Technical Support Assistance .....	11-49
Revision History .....	11-49
Appendix A. Attribute Definitions.....	11-50
DATA_WIDTH.....	11-50
REGMODE.....	11-50
CSDECODE.....	11-50
WRITEMODE.....	11-50



**LatticeECP3 High-Speed I/O Interface**

Introduction .....	12-1
External Interface Description .....	12-2
High-Speed I/O Interface Building Blocks .....	12-3
ECLK .....	12-4
SCLK .....	12-4
DQS Lane .....	12-4
PLL .....	12-5
DLL .....	12-5
DQSDLL .....	12-5
Input DDR (IDDR) .....	12-5
Output DDR (ODDR) .....	12-5
CLKDIV .....	12-5
DELAY .....	12-5
ECLK/SCLK vs. DQS Lanes .....	12-5
Building Generic High-Speed Interfaces .....	12-6
Types of High-Speed DDR Interfaces .....	12-6
Using IPexpress to Build High-Speed DDR Interfaces .....	12-10
Building SDR Modules .....	12-11
Building DDR Generic Modules .....	12-13
High-Speed DDR Interface Details .....	12-17
GIREG_RX.SCLK .....	12-18
GDDR1_RX.ECLK.Aligned .....	12-19
GDDR1_RX.ECLK.Centered .....	12-19
GDDR1_RX.SCLK.Aligned .....	12-20
GDDR1_RX.SCLK.PLL.Aligned .....	12-21
GDDR1_RX.SCLK.Centered .....	12-22
GDDR1_RX.DQS.Aligned .....	12-22
GDDR1_RX.DQS.Centered .....	12-23
GDDR2_RX.ECLK.Aligned .....	12-24
GDDR2_RX.ECLK.Aligned (No CLKDIV) .....	12-26
GDDR2_RX.ECLK.Centered .....	12-27
GDDR2_RX.DQS.Aligned .....	12-28
GDDR2_RX.DQS.Centered .....	12-29
GDDR2_RX.ECLK.Dynamic .....	12-30
GDDR2_RX.DQS.Dynamic .....	12-31
GDDR2_RX.PLL.Dynamic .....	12-32
GOREG_TX.SCLK .....	12-33
GDDR1_TX.SCLK.Centered .....	12-34
GDDR1_TX.SCLK.Aligned .....	12-36
GDDR1_TX.DQS.Centered .....	12-39
GDDR2_TX.Aligned .....	12-40
GDDR2_TX.DQSDLL.Centered .....	12-40
GDDR2_TX.PLL.Centered .....	12-42
7:1 LVDS Implementation .....	12-43
Generic DDR Design Guidelines .....	12-43
Placement Guidelines for High-Speed DDR Interfaces .....	12-43
High-Speed Clock Bridge ("EA" Devices) .....	12-44
DQ-DQS Grouping Rules .....	12-44
I/O Logic (IOL) Site Types/Names .....	12-45
Design Rules for Fitting Multiple Interfaces into One Device .....	12-47
Clocking Guidelines for Generic DDR Interfaces .....	12-48
Common Software Errors and Solutions .....	12-49
Timing Analysis for High-Speed DDR Interfaces .....	12-50



DDR/DDR2/DDR3 SDRAM Interfaces Overview .....	12-58
Implementing DDR/DDR2/DDR3 Memory Interfaces .....	12-60
DQS Grouping.....	12-61
Memory Read Implementation .....	12-62
Memory Write Implementation .....	12-65
DDR3 Clock Synchronization Module .....	12-74
DDR Memory Interface Generation Using IPexpress .....	12-77
DDR Memory DQ/DQS Design Rules and Guidelines.....	12-81
DDR/DDR2 Pinout Guidelines .....	12-82
DDR3 Termination Guidelines .....	12-83
Termination for DQ, DQS and DM .....	12-83
Termination for CK.....	12-83
Termination for Address, Commands and Controls .....	12-83
Termination for DDR3 DIMM.....	12-84
DDR3 Interface without Termination .....	12-84
DQ, DQS and DM without Parallel VTT Termination .....	12-84
Address, Command and Control Signals without Parallel VTT Termination.....	12-84
Layout Considerations for DDR3 .....	12-84
DDR3 Pinout Guidelines .....	12-85
Pin Placement Considerations for Improved Noise Immunity .....	12-86
DDR Software Primitives and Attributes .....	12-88
DQSDLLB .....	12-89
DQSBUFE.....	12-92
DQSBUFG .....	12-92
DQSBUFE1.....	12-93
DQSBUF Logic Primitives for DDR Memory Interfaces .....	12-93
DQSBUFD.....	12-95
DQSBUFF .....	12-96
Input DDR Primitives.....	12-97
Output DDR Primitives.....	12-110
Interface ID Attribute .....	12-121
ISI Calibration.....	12-122
Migrating Designs from LatticeECP3 “E” to “EA” .....	12-123
Migrating Designs from ispLEVER 7.2 SP2 to ispLEVER 8.0.....	12-125
Technical Support Assistance .....	12-126
Revision History .....	12-127
Appendix A. Building DDR Interfaces Using IPexpress in ispLEVER 7.2 SP2 .....	12-128
DDR Generic.....	12-129
Configuration Tab.....	12-130
DDR_MEM.....	12-131
Configuration Tab.....	12-132
Appendix B. Building SDR/DDR Interfaces Using IPexpress in Diamond .....	12-134
Building SDR Modules .....	12-134
Building DDR Generic Modules .....	12-136
Building DDR Memory Interfaces.....	12-140
<b>Power Consumption and Management for LatticeECP3 Devices</b>	
Introduction .....	13-1
Power Supply Sequencing and Hot Socketing.....	13-1
Recommended Power-up Sequence .....	13-1
Power Calculator Hardware Assumptions.....	13-1
Power Calculator.....	13-1
Power Calculator and Power Equations.....	13-2
Typical and Worst Case Process Power/ICC.....	13-3
Junction Temperature .....	13-3

Maximum Safe Ambient Temperature .....	13-3
Operating Temperature Range .....	13-3
Dynamic Power Multiplier (DPM) .....	13-4
Power Budgeting .....	13-4
Activity Factor Calculation .....	13-4
Thermal Impedance and Airflow .....	13-5
Reducing Power Consumption .....	13-5
Power Calculator Assumptions .....	13-6
Technical Support Assistance .....	13-6
Revision History .....	13-6
<b>LatticeECP3 sysDSP Usage Guide</b>	
Introduction .....	14-1
sysDSP Slice Hardware .....	14-1
sysDSP Slice Software .....	14-2
Overview .....	14-2
Targeting sysDSP Slices Using IPexpress .....	14-2
Targeting the sysDSP Slice by Inference .....	14-12
Targeting the sysDSP Slice Using ispLeverDSP with Simulink .....	14-13
Targeting the sysDSP Slice by Instantiating Primitives .....	14-14
sysDSP in the Report Files .....	14-15
Map Report .....	14-15
PAR Report .....	14-16
Trace Report .....	14-17
sysDSP in the ispLEVER Design Planner .....	14-18
Pre-Mapped View .....	14-18
Floorplan View .....	14-18
Advanced Features of the sysDSP Slice .....	14-19
IPexpress Slice Module .....	14-23
Rounding .....	14-26
Example 1: Rounding Toward Zero .....	14-27
Example 2: Rounding to Positive Infinity .....	14-27
Example 3: Rounding to Negative Infinity .....	14-28
Example 4: Rounding Away from Zero .....	14-28
Example 5: Rounding Toward Even .....	14-29
Example 6: Rounding Toward Odd .....	14-29
sysDSP Slice Control Signal and Data Signal Descriptions .....	14-30
Technical Support Assistance .....	14-30
Revision History .....	14-30
Appendix A. DSP Primitives .....	14-31
Appendix B. Using IPexpress for Diamond .....	14-36
Invoking IPexpress for Diamond .....	14-36
sysDSP in Diamond .....	14-47
<b>LatticeECP3 sysCONFIG Usage Guide</b>	
Introduction .....	15-1
General Configuration Flow .....	15-2
Configuration Pins .....	15-2
Configuration Process and Flow .....	15-4
Power-up Sequence .....	15-4
Initialization .....	15-5
Loading the Configuration Memory .....	15-5
Wake-up .....	15-6
Clearing the Configuration Memory and Re-initialization .....	15-6
Reconfiguration Priority .....	15-6

FPGA Configuration Control Pin Definitions .....	15-7
Configuration Pin Management.....	15-7
Dedicated Control Pins .....	15-7
JTAG Pins .....	15-7
Dual-Purpose sysCONFIG Pins.....	15-11
Configuration Modes .....	15-14
SPI Interface .....	15-14
SPI Master Multiboot (SPIm) Mode.....	15-16
Slave SPI (SSPI).....	15-18
Slave Serial Configuration Mode (SCM) .....	15-20
Slave Parallel Mode (SPCM) .....	15-20
JTAG Mode (IEEE 1149.1 and IEEE 1532) .....	15-22
Bitstream Generation Software Usage.....	15-23
Configuration Options .....	15-23
Device Wake-Up .....	15-25
Synchronizing Wake-Up.....	15-26
Bitstream Generation Property Options .....	15-28
Run DRC (T/F) .....	15-28
Create Bitfile (T/F).....	15-28
Bitstream File Formats.....	15-28
No Header (T/F) .....	15-28
Bitstream Encryption/Decryption Flow .....	15-28
Encrypting the Bitstream .....	15-28
Programming the 128-bit Key .....	15-30
Verifying a Configuration.....	15-32
File Formats .....	15-33
Decryption Flow .....	15-37
Combining Configuration Modes.....	15-38
Multiple FPGAs, One SPI Flash.....	15-38
Chain Mode Options .....	15-39
References.....	15-40
Technical Support Assistance .....	15-40
Revision History .....	15-41
Appendix A. Configuration Memory Requirements .....	15-42
Appendix B. Lattice Diamond Usage Overview .....	15-44
Converting an ispLEVER Project to Lattice Diamond .....	15-44
Importing an ispLEVER Design Project .....	15-44
Adjusting PCS Modules .....	15-44
Regenerate PCS Modules .....	15-44
Using IPexpress with Lattice Diamond.....	15-45
Creating a New Simulation Project Using Simulation Wizard .....	15-46
Setting Global Preferences in Diamond.....	15-46
Setting Bitstream Generation Options in Diamond .....	15-48
Setting Security Options in Diamond .....	15-50
<b>LatticeECP3 Soft Error Detection (SED) Usage Guide</b>	
Introduction .....	16-1
SED Overview.....	16-1
Hardware Description.....	16-2
Signal Descriptions .....	16-2
SEDCLK.....	16-2
SEDENABLE.....	16-3
SEDCLKOUT .....	16-3
SEDSTART .....	16-4
SEDFRCERR.....	16-4

---

SEDINPROG.....	16-4
SEDDONE .....	16-4
SEDERR .....	16-4
AUTODONE.....	16-4
MCCLK_freq Attribute .....	16-4
SED Flow .....	16-5
SED Run Time .....	16-6
Sample Code .....	16-7
VHDL Example.....	16-7
Verilog Example .....	16-8
Technical Support Assistance .....	16-9
Revision History .....	16-9
<b>LatticeECP3 and LatticeECP2M High-Speed Backplane Measurements</b>	
Introduction .....	17-1
Eye Diagram Experiment .....	17-1
Backplane Specifications .....	17-2
Test Setup Parameters .....	17-2
Eye Diagram Measurements.....	17-2
Results and Conclusion .....	17-5
Data Rate Experiment.....	17-5
Backplane Specifications .....	17-5
Test Setup Parameters .....	17-5
Data Rate Measurements .....	17-6
Results and Conclusions.....	17-6
Conclusions and Design Guidelines .....	17-6
References.....	17-7
Technical Support Assistance .....	17-7
Revision History .....	17-7
Introduction .....	18-1
<b>LatticeECP3 Hardware Checklist</b>	
Power Supplies .....	18-2
LatticeECP3 SERDES/PCS Power Supplies .....	18-2
Power Estimation .....	18-3
Configuration Considerations.....	18-3
I/O Pin Assignments.....	18-4
Dedicated FPGA Inputs (Non-configuration).....	18-4
Pinout Considerations .....	18-4
LVDS Pin Assignments .....	18-5
HSTL and SSTL Pin Assignments .....	18-5
XRES Pin .....	18-5
SERDES Pin Considerations .....	18-5
Technical Support Assistance .....	18-8
Revision History .....	18-8
<b>Section III. LatticeECP3 Family Handbook Revision History</b>	
Revision History .....	19-1

---



## **Section I. LatticeECP3 Family Data Sheet**

---

DS1021 Version 02.3EA, June 2013

## Features

### ■ Higher Logic Density for Increased System Integration

- 17K to 149K LUTs
- 116 to 586 I/Os

### ■ Embedded SERDES

- 150 Mbps to 3.2 Gbps for Generic 8b10b, 10-bit SERDES, and 8-bit SERDES modes
- Data Rates 230 Mbps to 3.2 Gbps per channel for all other protocols
- Up to 16 channels per device: PCI Express, SONET/SDH, Ethernet (1GbE, SGMII, XAUI), CPRI, SMPTE 3G and Serial RapidIO

### ■ sysDSP™

- Fully cascadable slice architecture
- 12 to 160 slices for high performance multiply and accumulate
- Powerful 54-bit ALU operations
- Time Division Multiplexing MAC Sharing
- Rounding and truncation
- Each slice supports
  - Half 36x36, two 18x18 or four 9x9 multipliers
  - Advanced 18x36 MAC and 18x18 Multiply-Multiply-Accumulate (MMAC) operations

### ■ Flexible Memory Resources

- Up to 6.85Mbits sysMEM™ Embedded Block RAM (EBR)
- 36K to 303K bits distributed RAM

### ■ sysCLOCK Analog PLLs and DLLs

- Two DLLs and up to ten PLLs per device

### ■ Pre-Engineered Source Synchronous I/O

- DDR registers in I/O cells

- Dedicated read/write levelling functionality
- Dedicated gearing logic
- Source synchronous standards support
  - ADC/DAC, 7:1 LVDS, XGMII
  - High Speed ADC/DAC devices
- Dedicated DDR/DDR2/DDR3 memory with DQS support
- Optional Inter-Symbol Interference (ISI) correction on outputs

### ■ Programmable sysI/O™ Buffer Supports Wide Range of Interfaces

- On-chip termination
- Optional equalization filter on inputs
- LVTTTL and LVCMOS 33/25/18/15/12
- SSTL 33/25/18/15 I, II
- HSTL15 I and HSTL18 I, II
- PCI and Differential HSTL, SSTL
- LVDS, Bus-LVDS, LVPECL, RSDS, MLVDS

### ■ Flexible Device Configuration

- Dedicated bank for configuration I/Os
- SPI boot flash interface
- Dual-boot images supported
- Slave SPI
- TransFR™ I/O for simple field updates
- Soft Error Detect embedded macro

### ■ System Level Support

- IEEE 1149.1 and IEEE 1532 compliant
- Reveal Logic Analyzer
- ORCAstra FPGA configuration utility
- On-chip oscillator for initialization & general use
- 1.2V core power supply

**Table 1-1. LatticeECP3™ Family Selection Guide**

Device	ECP3-17	ECP3-35	ECP3-70	ECP3-95	ECP3-150
LUTs (K)	17	33	67	92	149
sysMEM Blocks (18Kbits)	38	72	240	240	372
Embedded Memory (Kbits)	700	1327	4420	4420	6850
Distributed RAM Bits (Kbits)	36	68	145	188	303
18X18 Multipliers	24	64	128	128	320
SERDES (Quad)	1	1	3	3	4
PLLs/DLLs	2 / 2	4 / 2	10 / 2	10 / 2	10 / 2
<b>Packages and SERDES Channels/ I/O Combinations</b>					
328 csBGA (10x10 mm)	2 / 116				
256 ftBGA (17x17 mm)	4 / 133	4 / 133			
484 fpBGA (23x23 mm)	4 / 222	4 / 295	4 / 295	4 / 295	
672 fpBGA (27x27 mm)		4 / 310	8 / 380	8 / 380	8 / 380
1156 fpBGA (35x35 mm)			12 / 490	12 / 490	16 / 586

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at [www.latticesemi.com/legal](http://www.latticesemi.com/legal). All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

## Introduction

The LatticeECP3™ (Economy Plus Third generation) family of FPGA devices is optimized to deliver high performance features such as an enhanced DSP architecture, high speed SERDES and high speed source synchronous interfaces in an economical FPGA fabric. This combination is achieved through advances in device architecture and the use of 65nm technology making the devices suitable for high-volume, high-speed, low-cost applications.

The LatticeECP3 device family expands look-up-table (LUT) capacity to 149K logic elements and supports up to 586 user I/Os. The LatticeECP3 device family also offers up to 320 18x18 multipliers and a wide range of parallel I/O standards.

The LatticeECP3 FPGA fabric is optimized with high performance and low cost in mind. The LatticeECP3 devices utilize reconfigurable SRAM logic technology and provide popular building blocks such as LUT-based logic, distributed and embedded memory, Phase Locked Loops (PLLs), Delay Locked Loops (DLLs), pre-engineered source synchronous I/O support, enhanced sysDSP slices and advanced configuration support, including encryption and dual-boot capabilities.

The pre-engineered source synchronous logic implemented in the LatticeECP3 device family supports a broad range of interface standards, including DDR3, XGMII and 7:1 LVDS.

The LatticeECP3 device family also features high speed SERDES with dedicated PCS functions. High jitter tolerance and low transmit jitter allow the SERDES plus PCS blocks to be configured to support an array of popular data protocols including PCI Express, SMPTE, Ethernet (XAUI, GbE, and SGMII) and CPRI. Transmit Pre-emphasis and Receive Equalization settings make the SERDES suitable for transmission and reception over various forms of media.

The LatticeECP3 devices also provide flexible, reliable and secure configuration options, such as dual-boot capability, bit-stream encryption, and TransFR field upgrade features.

The Lattice Diamond™ and ispLEVER® design software allows large complex designs to be efficiently implemented using the LatticeECP3 FPGA family. Synthesis library support for LatticeECP3 is available for popular logic synthesis tools. Diamond and ispLEVER tools use the synthesis tool output along with the constraints from its floor planning tools to place and route the design in the LatticeECP3 device. The tools extract the timing from the routing and back-annotate it into the design for timing verification.

Lattice provides many pre-engineered IP (Intellectual Property) modules for the LatticeECP3 family. By using these configurable soft core IPs as standardized blocks, designers are free to concentrate on the unique aspects of their design, increasing their productivity.

## Architecture Overview

Each LatticeECP3 device contains an array of logic blocks surrounded by Programmable I/O Cells (PIC). Interspersed between the rows of logic blocks are rows of sysMEM™ Embedded Block RAM (EBR) and rows of sys-DSP™ Digital Signal Processing slices, as shown in Figure 2-1. The LatticeECP3-150 has four rows of DSP slices; all other LatticeECP3 devices have two rows of DSP slices. In addition, the LatticeECP3 family contains SERDES Quads on the bottom of the device.

There are two kinds of logic blocks, the Programmable Functional Unit (PFU) and Programmable Functional Unit without RAM (PFF). The PFU contains the building blocks for logic, arithmetic, RAM and ROM functions. The PFF block contains building blocks for logic, arithmetic and ROM functions. Both PFU and PFF blocks are optimized for flexibility, allowing complex designs to be implemented quickly and efficiently. Logic Blocks are arranged in a two-dimensional array. Only one type of block is used per row.

The LatticeECP3 devices contain one or more rows of sysMEM EBR blocks. sysMEM EBRs are large, dedicated 18Kbit fast memory blocks. Each sysMEM block can be configured in a variety of depths and widths as RAM or ROM. In addition, LatticeECP3 devices contain up to two rows of DSP slices. Each DSP slice has multipliers and adder/accumulators, which are the building blocks for complex signal processing capabilities.

The LatticeECP3 devices feature up to 16 embedded 3.2Gbps SERDES (Serializer / Deserializer) channels. Each SERDES channel contains independent 8b/10b encoding / decoding, polarity adjust and elastic buffer logic. Each group of four SERDES channels, along with its Physical Coding Sub-layer (PCS) block, creates a quad. The functionality of the SERDES/PCS quads can be controlled by memory cells set during device configuration or by registers that are addressable during device operation. The registers in every quad can be programmed via the SERDES Client Interface (SCI). These quads (up to four) are located at the bottom of the devices.

Each PIC block encompasses two PIOs (PIO pairs) with their respective sysI/O buffers. The sysI/O buffers of the LatticeECP3 devices are arranged in seven banks, allowing the implementation of a wide variety of I/O standards. In addition, a separate I/O bank is provided for the programming interfaces. 50% of the PIO pairs on the left and right edges of the device can be configured as LVDS transmit/receive pairs. The PIC logic also includes pre-engineered support to aid in the implementation of high speed source synchronous standards such as XGMII, 7:1 LVDS, along with memory interfaces including DDR3.

The LatticeECP3 registers in PFU and sysI/O can be configured to be SET or RESET. After power up and the device is configured, it enters into user mode with these registers SET/RESET according to the configuration setting, allowing the device entering to a known state for predictable system function.

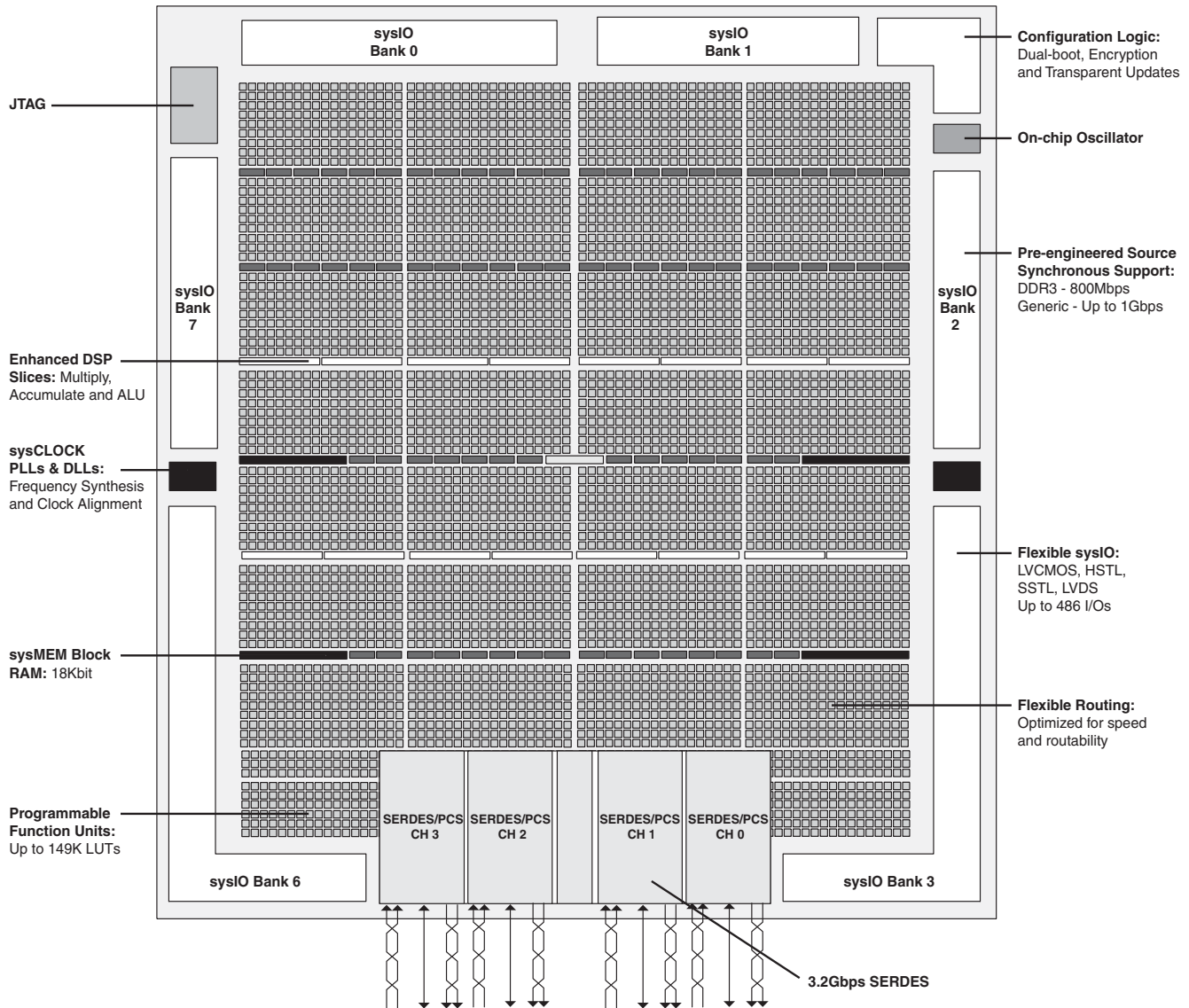
Other blocks provided include PLLs, DLLs and configuration functions. The LatticeECP3 architecture provides two Delay Locked Loops (DLLs) and up to ten Phase Locked Loops (PLLs). The PLL and DLL blocks are located at the end of the EBR/DSP rows.

The configuration block that supports features such as configuration bit-stream decryption, transparent updates and dual-boot support is located toward the center of this EBR row. Every device in the LatticeECP3 family supports a sysCONFIG™ port located in the corner between banks one and two, which allows for serial or parallel device configuration.

In addition, every device in the family has a JTAG port. This family also provides an on-chip oscillator and soft error detect capability. The LatticeECP3 devices use 1.2V as their core voltage.



**Figure 2-1. Simplified Block Diagram, LatticeECP3-35 Device (Top Level)**



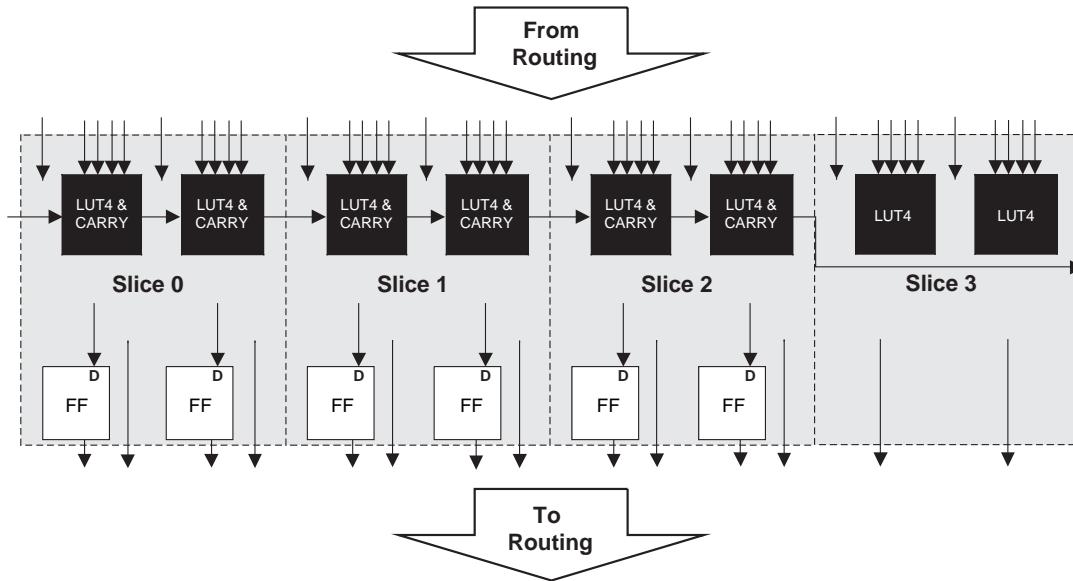
Note: There is no Bank 4 or Bank 5 in LatticeECP3 devices.

## PFU Blocks

The core of the LatticeECP3 device consists of PFU blocks, which are provided in two forms, the PFU and PFF. The PFUs can be programmed to perform Logic, Arithmetic, Distributed RAM and Distributed ROM functions. PFF blocks can be programmed to perform Logic, Arithmetic and ROM functions. Except where necessary, the remainder of this data sheet will use the term PFU to refer to both PFU and PFF blocks.

Each PFU block consists of four interconnected slices numbered 0-3 as shown in Figure 2-2. Each slice contains two LUTs. All the interconnections to and from PFU blocks are from routing. There are 50 inputs and 23 outputs associated with each PFU block.

Figure 2-2. PFU Diagram



## Slice

Slice 0 through Slice 2 contain two LUT4s feeding two registers, whereas Slice 3 contains two LUT4s only. For PFUs, Slice 0 through Slice 2 can be configured as distributed memory, a capability not available in the PFF. Table 2-1 shows the capability of the slices in both PFF and PFU blocks along with the operation modes they enable. In addition, each PFU contains logic that allows the LUTs to be combined to perform functions such as LUT5, LUT6, LUT7 and LUT8. There is control logic to perform set/reset functions (programmable as synchronous/asynchronous), clock select, chip-select and wider RAM/ROM functions.

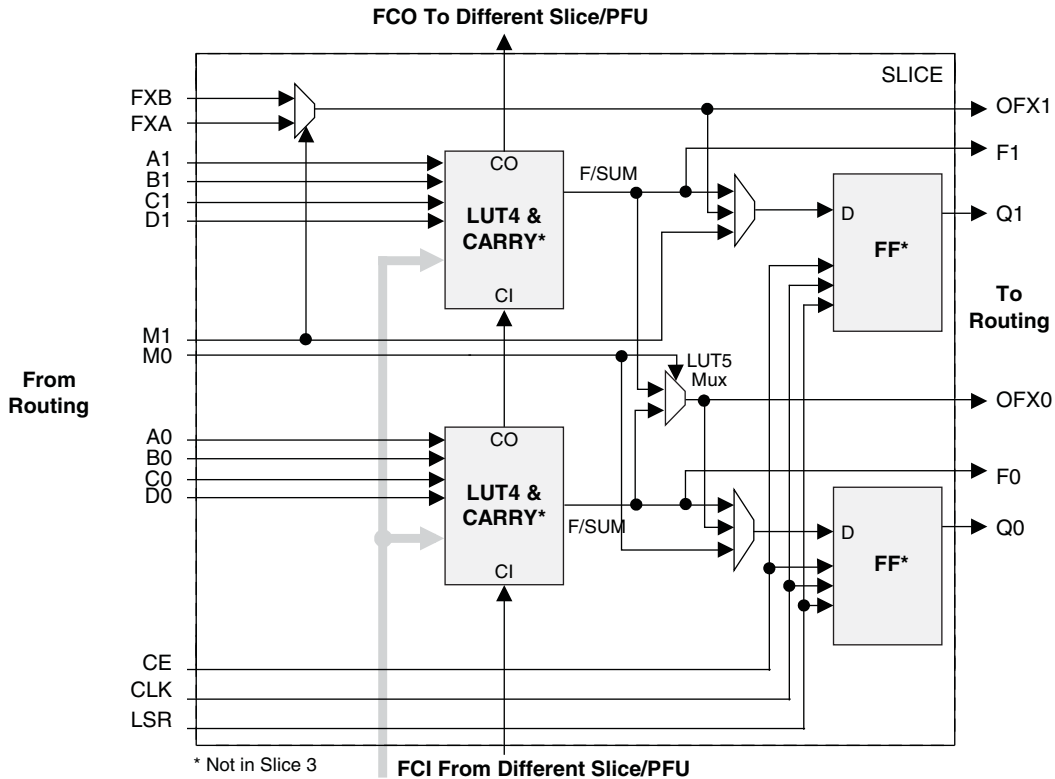
Table 2-1. Resources and Modes Available per Slice

Slice	PFU BLock		PFF Block	
	Resources	Modes	Resources	Modes
Slice 0	2 LUT4s and 2 Registers	Logic, Ripple, RAM, ROM	2 LUT4s and 2 Registers	Logic, Ripple, ROM
Slice 1	2 LUT4s and 2 Registers	Logic, Ripple, RAM, ROM	2 LUT4s and 2 Registers	Logic, Ripple, ROM
Slice 2	2 LUT4s and 2 Registers	Logic, Ripple, RAM, ROM	2 LUT4s and 2 Registers	Logic, Ripple, ROM
Slice 3	2 LUT4s	Logic, ROM	2 LUT4s	Logic, ROM

Figure 2-3 shows an overview of the internal logic of the slice. The registers in the slice can be configured for positive/negative and edge triggered or level sensitive clocks.

Slices 0, 1 and 2 have 14 input signals: 13 signals from routing and one from the carry-chain (from the adjacent slice or PFU). There are seven outputs: six to routing and one to carry-chain (to the adjacent PFU). Slice 3 has 10 input signals from routing and four signals to routing. Table 2-2 lists the signals associated with Slice 0 to Slice 2.

Figure 2-3. Slice Diagram



\* Not in Slice 3

For Slices 0 and 1, memory control signals are generated from Slice 2 as follows:  
 WCK is CLK  
 WRE is from LSR  
 DI[3:2] for Slice 1 and DI[1:0] for Slice 0 data from Slice 2  
 WAD [A:D] is a 4-bit address from slice 2 LUT input

Table 2-2. Slice Signal Descriptions

Function	Type	Signal Names	Description
Input	Data signal	A0, B0, C0, D0	Inputs to LUT4
Input	Data signal	A1, B1, C1, D1	Inputs to LUT4
Input	Multi-purpose	M0	Multipurpose Input
Input	Multi-purpose	M1	Multipurpose Input
Input	Control signal	CE	Clock Enable
Input	Control signal	LSR	Local Set/Reset
Input	Control signal	CLK	System Clock
Input	Inter-PFU signal	FC	Fast Carry-in <sup>1</sup>
Input	Inter-slice signal	FXA	Intermediate signal to generate LUT6 and LUT7
Input	Inter-slice signal	FXB	Intermediate signal to generate LUT6 and LUT7
Output	Data signals	F0, F1	LUT4 output register bypass signals
Output	Data signals	Q0, Q1	Register outputs
Output	Data signals	OFX0	Output of a LUT5 MUX
Output	Data signals	OFX1	Output of a LUT6, LUT7, LUT8 <sup>2</sup> MUX depending on the slice
Output	Inter-PFU signal	FCO	Slice 2 of each PFU is the fast carry chain output <sup>1</sup>

1. See Figure 2-3 for connection details.

2. Requires two PFUs.

## Modes of Operation

Each slice has up to four potential modes of operation: Logic, Ripple, RAM and ROM.

### Logic Mode

In this mode, the LUTs in each slice are configured as 4-input combinatorial lookup tables. A LUT4 can have 16 possible input combinations. Any four input logic functions can be generated by programming this lookup table. Since there are two LUT4s per slice, a LUT5 can be constructed within one slice. Larger look-up tables such as LUT6, LUT7 and LUT8 can be constructed by concatenating other slices. Note LUT8 requires more than four slices.

### Ripple Mode

Ripple mode supports the efficient implementation of small arithmetic functions. In ripple mode, the following functions can be implemented by each slice:

- Addition 2-bit
- Subtraction 2-bit
- Add/Subtract 2-bit using dynamic control
- Up counter 2-bit
- Down counter 2-bit
- Up/Down counter with asynchronous clear
- Up/Down counter with preload (sync)
- Ripple mode multiplier building block
- Multiplier support
- Comparator functions of A and B inputs
  - A greater-than-or-equal-to B
  - A not-equal-to B
  - A less-than-or-equal-to B

Ripple Mode includes an optional configuration that performs arithmetic using fast carry chain methods. In this configuration (also referred to as CCU2 mode) two additional signals, Carry Generate and Carry Propagate, are generated on a per slice basis to allow fast arithmetic functions to be constructed by concatenating Slices.

### RAM Mode

In this mode, a 16x4-bit distributed single port RAM (SPR) can be constructed using each LUT block in Slice 0 and Slice 1 as a 16x1-bit memory. Slice 2 is used to provide memory address and control signals. A 16x2-bit pseudo dual port RAM (PDPR) memory is created by using one Slice as the read-write port and the other companion slice as the read-only port.

LatticeECP3 devices support distributed memory initialization.

The Lattice design tools support the creation of a variety of different size memories. Where appropriate, the software will construct these using distributed memory primitives that represent the capabilities of the PFU. Table 2-3 shows the number of slices required to implement different distributed RAM primitives. For more information about using RAM in LatticeECP3 devices, please see TN1179, [LatticeECP3 Memory Usage Guide](#).

**Table 2-3. Number of Slices Required to Implement Distributed RAM**

	SPR 16X4	PDPR 16X4
Number of slices	3	3

Note: SPR = Single Port RAM, PDPR = Pseudo Dual Port RAM

---

## ROM Mode

ROM mode uses the LUT logic; hence, Slices 0 through 3 can be used in ROM mode. Preloading is accomplished through the programming interface during PFU configuration.

For more information, please refer to TN1179, [LatticeECP3 Memory Usage Guide](#).

## Routing

There are many resources provided in the LatticeECP3 devices to route signals individually or as busses with related control signals. The routing resources consist of switching circuitry, buffers and metal interconnect (routing) segments.

The LatticeECP3 family has an enhanced routing architecture that produces a compact design. The Diamond and ispLEVER design software tool suites take the output of the synthesis tool and places and routes the design.

## sysCLOCK PLLs and DLLs

The sysCLOCK PLLs provide the ability to synthesize clock frequencies. The devices in the LatticeECP3 family support two to ten full-featured General Purpose PLLs.

### General Purpose PLL

The architecture of the PLL is shown in Figure 2-4. A description of the PLL functionality follows.

CLKI is the reference frequency (generated either from the pin or from routing) for the PLL. CLKI feeds into the Input Clock Divider block. The CLKFB is the feedback signal (generated from CLKOP, CLKOS or from a user clock pin/logic). This signal feeds into the Feedback Divider. The Feedback Divider is used to multiply the reference frequency.

Both the input path and feedback signals enter the Phase Frequency Detect Block (PFD) which detects first for the frequency, and then the phase, of the CLKI and CLKFB are the same which then drives the Voltage Controlled Oscillator (VCO) block. In this block the difference between the input path and feedback signals is used to control the frequency and phase of the oscillator. A LOCK signal is generated by the VCO to indicate that the VCO has locked onto the input clock signal. In dynamic mode, the PLL may lose lock after a dynamic delay adjustment and not relock until the  $t_{LOCK}$  parameter has been satisfied.

The output of the VCO then enters the CLKOP divider. The CLKOP divider allows the VCO to operate at higher frequencies than the clock output (CLKOP), thereby increasing the frequency range. The Phase/Duty Cycle/Duty Trim block adjusts the phase and duty cycle of the CLKOS signal. The phase/duty cycle setting can be pre-programmed or dynamically adjusted. A secondary divider takes the CLKOP or CLKOS signal and uses it to derive lower frequency outputs (CLKOK).

The primary output from the CLKOP divider (CLKOP) along with the outputs from the secondary dividers (CLKOK and CLKOK2) and Phase/Duty select (CLKOS) are fed to the clock distribution network.

The PLL allows two methods for adjusting the phase of signal. The first is referred to as Fine Delay Adjustment. This inserts up to 16 nominal 125ps delays to be applied to the secondary PLL output. The number of steps may be set statically or from the FPGA logic. The second method is referred to as Coarse Phase Adjustment. This allows the phase of the rising and falling edge of the secondary PLL output to be adjusted in 22.5 degree steps. The number of steps may be set statically or from the FPGA logic.

**Figure 2-4. General Purpose PLL Diagram**

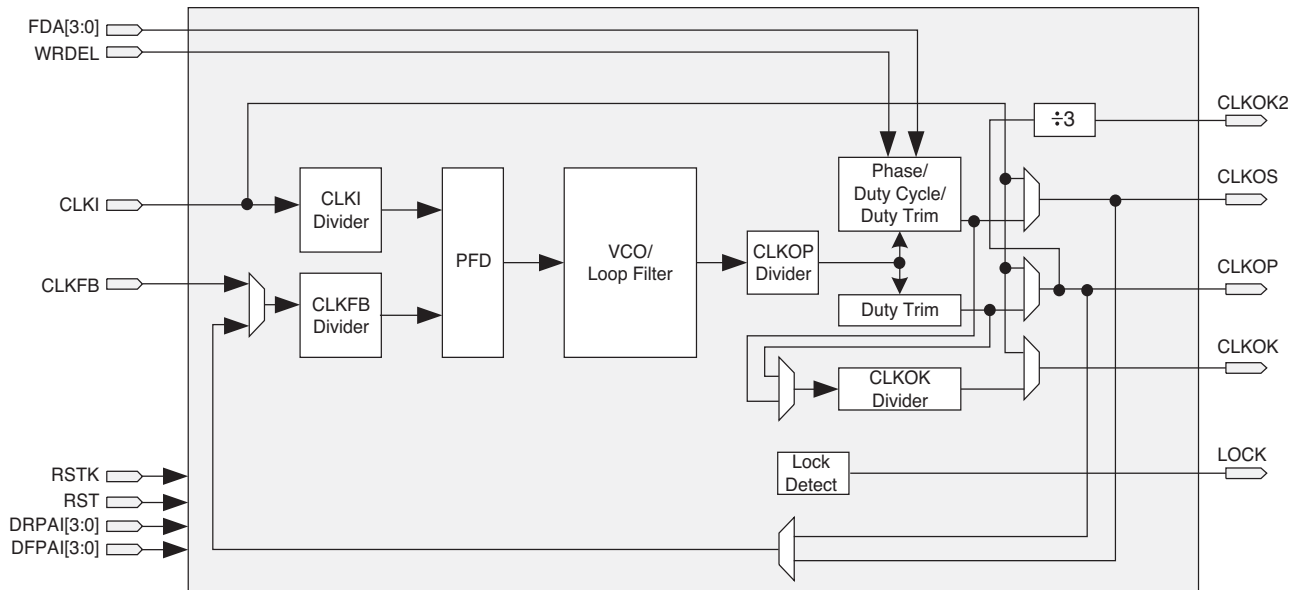


Table 2-4 provides a description of the signals in the PLL blocks.

**Table 2-4. PLL Blocks Signal Descriptions**

Signal	I/O	Description
CLKI	I	Clock input from external pin or routing
CLKFB	I	PLL feedback input from CLKOP, CLKOS, or from a user clock (pin or logic)
RST	I	"1" to reset PLL counters, VCO, charge pumps and M-dividers
RSTK	I	"1" to reset K-divider
WRDEL	I	DPA Fine Delay Adjust input
CLKOS	O	PLL output to clock tree (phase shifted/duty cycle changed)
CLKOP	O	PLL output to clock tree (no phase shift)
CLKKOK	O	PLL output to clock tree through secondary clock divider
CLKKOK2	O	PLL output to clock tree (CLKOP divided by 3)
LOCK	O	"1" indicates PLL LOCK to CLKI
FDA [3:0]	I	Dynamic fine delay adjustment on CLKOS output
DRPAI[3:0]	I	Dynamic coarse phase shift, rising edge setting
DFPAI[3:0]	I	Dynamic coarse phase shift, falling edge setting

### Delay Locked Loops (DLL)

In addition to PLLs, the LatticeECP3 family of devices has two DLLs per device.

CLKI is the input frequency (generated either from the pin or routing) for the DLL. CLKI feeds into the output muxes block to bypass the DLL, directly to the DELAY CHAIN block and (directly or through divider circuit) to the reference input of the Phase Detector (PD) input mux. The reference signal for the PD can also be generated from the Delay Chain signals. The feedback input to the PD is generated from the CLKFB pin or from a tapped signal from the Delay chain.

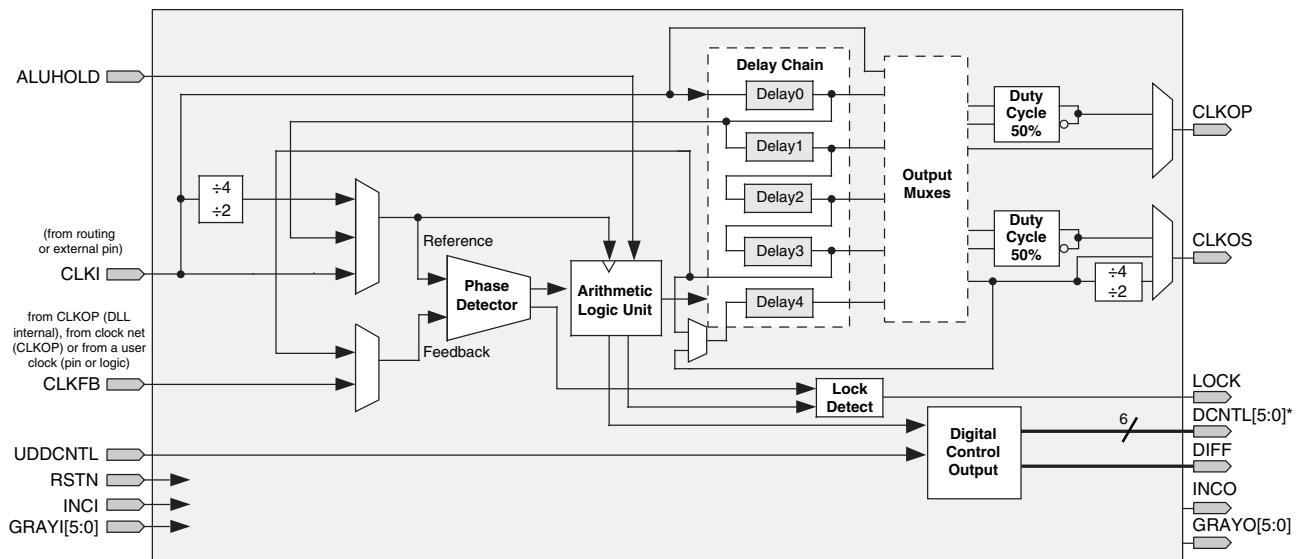
The PD produces a binary number proportional to the phase and frequency difference between the reference and feedback signals. Based on these inputs, the ALU determines the correct digital control codes to send to the delay

chain in order to better match the reference and feedback signals. This digital code from the ALU is also transmitted via the Digital Control bus (DCNTL) bus to its associated Slave Delay lines (two per DLL). The ALUHOLD input allows the user to suspend the ALU output at its current value. The UDDCNTL signal allows the user to latch the current value on the DCNTL bus.

The DLL has two clock outputs, CLKOP and CLKOS. These outputs can individually select one of the outputs from the tapped delay line. The CLKOS has optional fine delay shift and divider blocks to allow this output to be further modified, if required. The fine delay shift block allows the CLKOS output to phase shifted a further 45, 22.5 or 11.25 degrees relative to its normal position. Both the CLKOS and CLKOP outputs are available with optional duty cycle correction. Divide by two and divide by four frequencies are available at CLKOS. The LOCK output signal is asserted when the DLL is locked. Figure 2-5 shows the DLL block diagram and Table 2-5 provides a description of the DLL inputs and outputs.

The user can configure the DLL for many common functions such as time reference delay mode and clock injection removal mode. Lattice provides primitives in its design tools for these functions.

**Figure 2-5. Delay Locked Loop Diagram (DLL)**



**Table 2-5. DLL Signals**

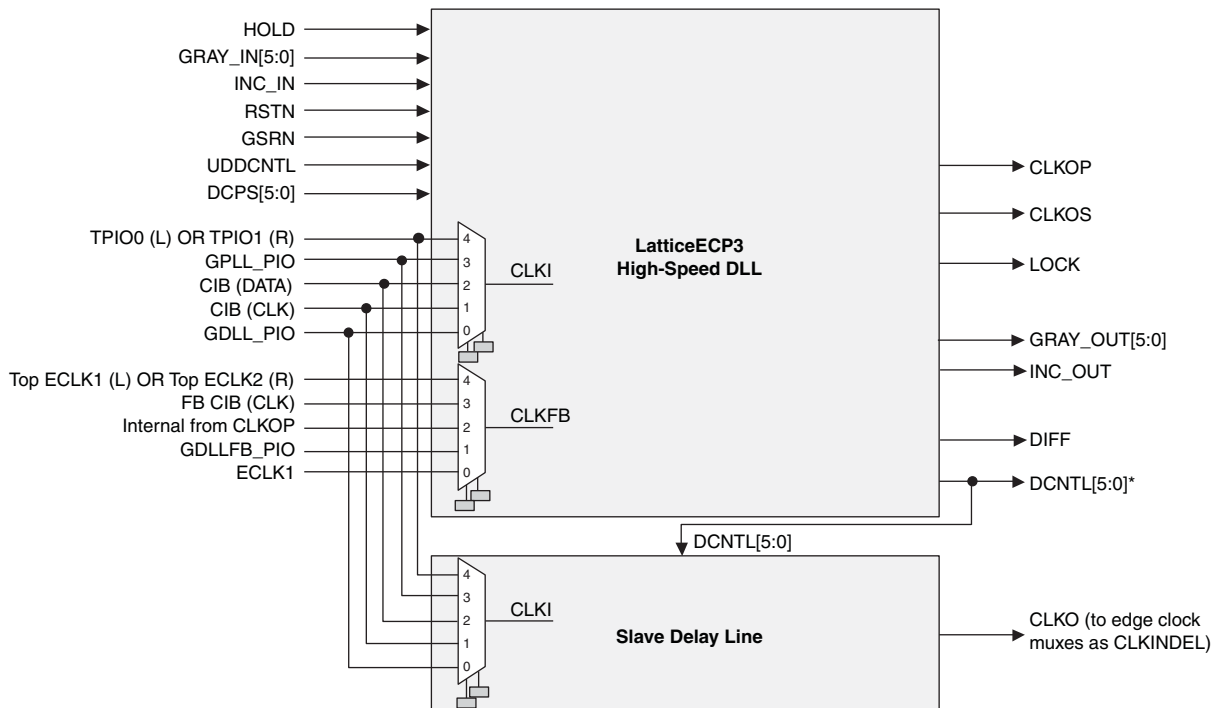
Signal	I/O	Description
CLKI	I	Clock input from external pin or routing
CLKFB	I	DLL feed input from DLL output, clock net, routing or external pin
RSTN	I	Active low synchronous reset
ALUHOLD	I	Active high freezes the ALU
UDDCNTL	I	Synchronous enable signal (hold high for two cycles) from routing
CLKOP	O	The primary clock output
CLKOS	O	The secondary clock output with fine delay shift and/or division by 2 or by 4
LOCK	O	Active high phase lock indicator
INCI	I	Incremental indicator from another DLL via CIB.
GRAYI[5:0]	I	Gray-coded digital control bus from another DLL in time reference mode.
DIFF	O	Difference indicator when DCNTL is difference than the internal setting and update is needed.
INCO	O	Incremental indicator to other DLLs via CIB.
GRAYO[5:0]	O	Gray-coded digital control bus to other DLLs via CIB

LatticeECP3 devices have two general DLLs and four Slave Delay lines, two per DLL. The DLLs are in the lowest EBR row and located adjacent to the EBR. Each DLL replaces one EBR block. One Slave Delay line is placed adjacent to the DLL and the duplicate Slave Delay line (in Figure 2-6) for the DLL is placed in the I/O ring between Banks 6 and 7 and Banks 2 and 3.

The outputs from the DLL and Slave Delay lines are fed to the clock distribution network.

For more information, please see TN1178, [LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide](#).

**Figure 2-6. Top-Level Block Diagram, High-Speed DLL and Slave Delay Line**



\* This signal is not user accessible. It can only be used to feed the slave delay line.



## PLL/DLL Cascading

LatticeECP3 devices have been designed to allow certain combinations of PLL and DLL cascading. The allowable combinations are:

- PLL to PLL supported
- PLL to DLL supported

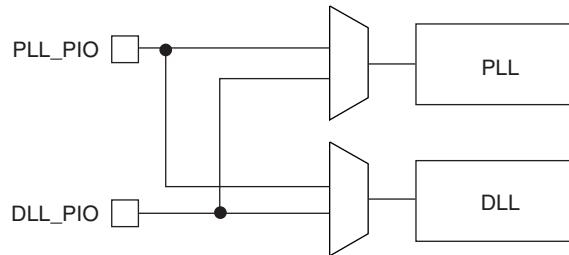
The DLLs in the LatticeECP3 are used to shift the clock in relation to the data for source synchronous inputs. PLLs are used for frequency synthesis and clock generation for source synchronous interfaces. Cascading PLL and DLL blocks allows applications to utilize the unique benefits of both DLLs and PLLs.

For further information about the DLL, please see the list of technical documentation at the end of this data sheet.

## PLL/DLL PIO Input Pin Connections

All LatticeECP3 devices contains two DLLs and up to ten PLLs, arranged in quadrants. If a PLL and a DLL are next to each other, they share input pins as shown in the Figure 2-7.

**Figure 2-7. Sharing of PIO Pins by PLLs and DLLs in LatticeECP3 Devices**

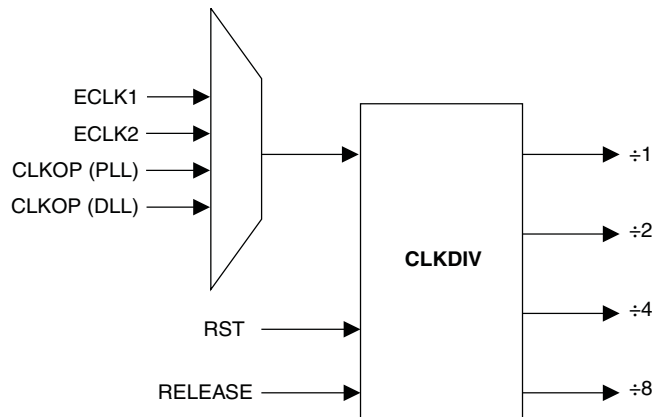


Note: Not every PLL has an associated DLL.

## Clock Dividers

LatticeECP3 devices have two clock dividers, one on the left side and one on the right side of the device. These are intended to generate a slower-speed system clock from a high-speed edge clock. The block operates in a  $\div 2$ ,  $\div 4$  or  $\div 8$  mode and maintains a known phase relationship between the divided down clock and the high-speed clock based on the release of its reset signal. The clock dividers can be fed from selected PLL/DLL outputs, the Slave Delay lines, routing or from an external clock input. The clock divider outputs serve as primary clock sources and feed into the clock distribution network. The Reset (RST) control signal resets input and asynchronously forces all outputs to low. The RELEASE signal releases outputs synchronously to the input clock. For further information on clock dividers, please see TN1178, [LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide](#). Figure 2-8 shows the clock divider connections.

Figure 2-8. Clock Divider Connections



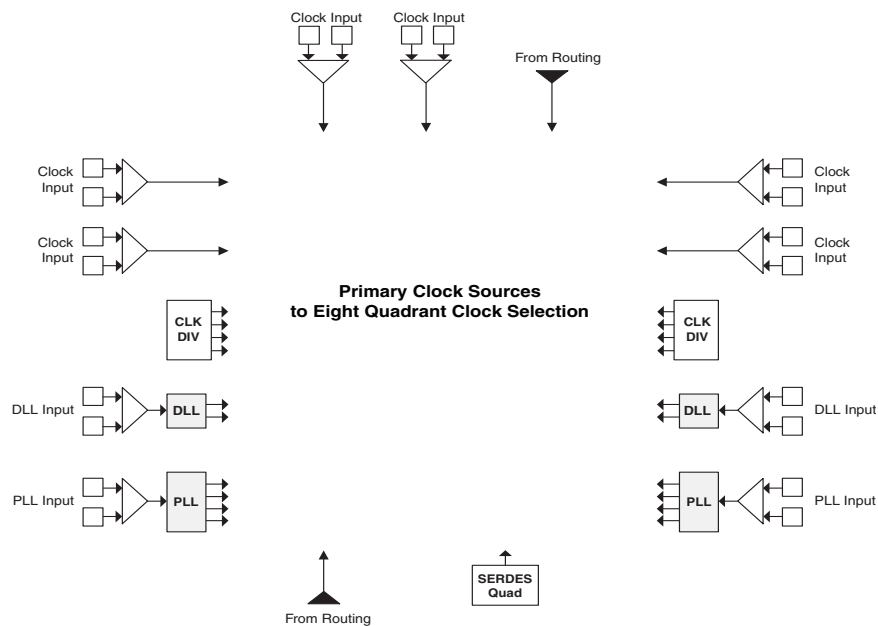
## Clock Distribution Network

LatticeECP3 devices have eight quadrant-based primary clocks and eight secondary clock/control sources. Two high performance edge clocks are available on the top, left, and right edges of the device to support high speed interfaces. These clock sources are selected from external I/Os, the sysCLOCK PLLs, DLLs or routing. These clock sources are fed throughout the chip via a clock distribution system.

## Primary Clock Sources

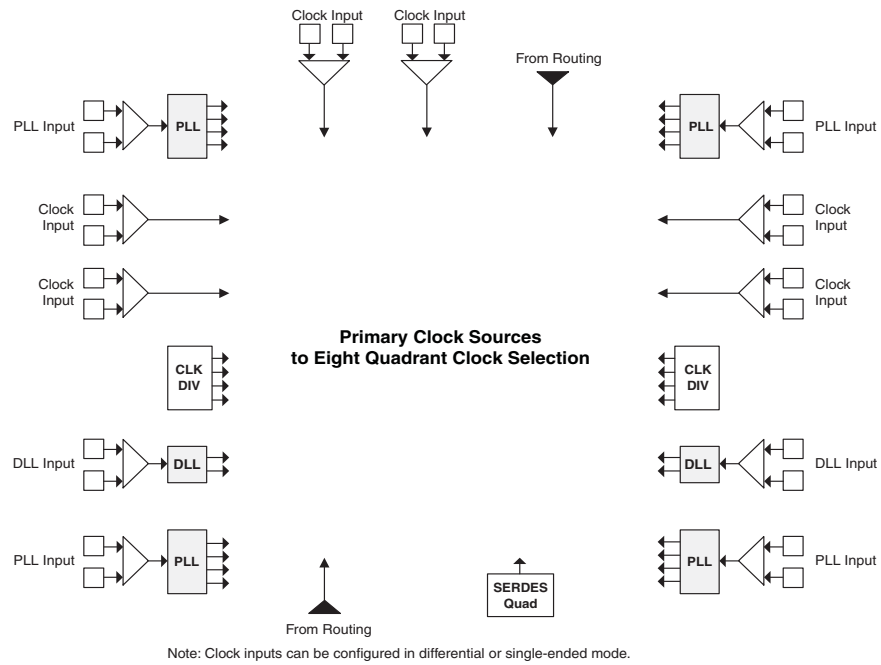
LatticeECP3 devices derive clocks from six primary source types: PLL outputs, DLL outputs, CLKDIV outputs, dedicated clock inputs, routing and SERDES Quads. LatticeECP3 devices have two to ten sysCLOCK PLLs and two DLLs, located on the left and right sides of the device. There are six dedicated clock inputs: two on the top side, two on the left side and two on the right side of the device. Figures 2-9, 2-10 and 2-11 show the primary clock sources for LatticeECP3 devices.

Figure 2-9. Primary Clock Sources for LatticeECP3-17

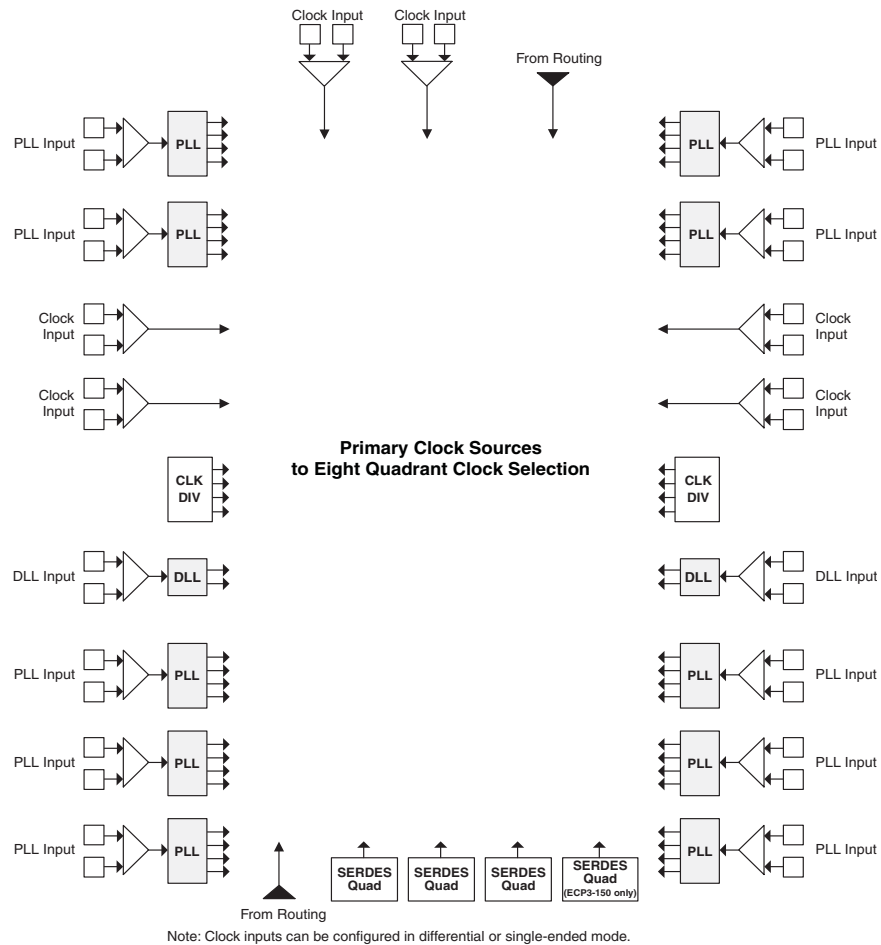


Note: Clock inputs can be configured in differential or single-ended mode.

**Figure 2-10. Primary Clock Sources for LatticeECP3-35**



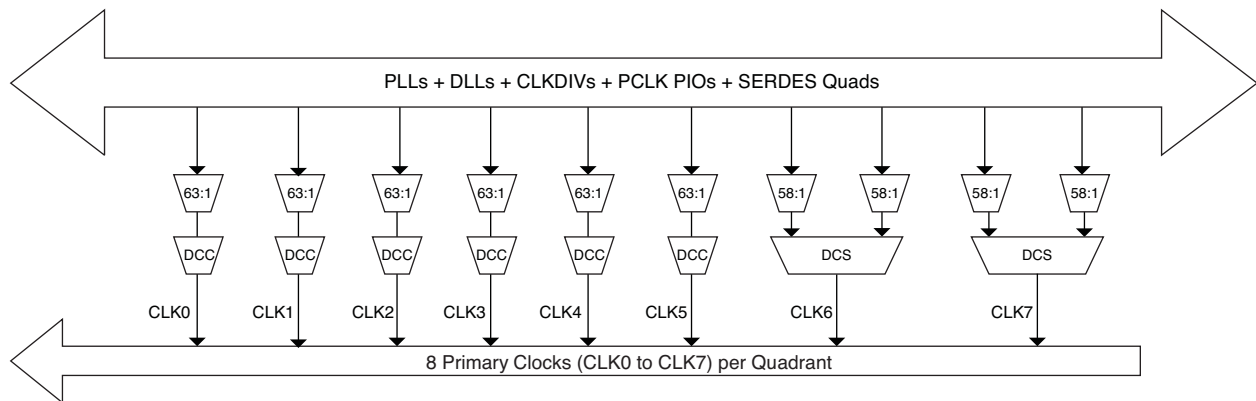
**Figure 2-11. Primary Clock Sources for LatticeECP3-70, -95, -150**



## Primary Clock Routing

The purpose of the primary clock routing is to distribute primary clock sources to the destination quadrants of the device. A global primary clock is a primary clock that is distributed to all quadrants. The clock routing structure in LatticeECP3 devices consists of a network of eight primary clock lines (CLK0 through CLK7) per quadrant. The primary clocks of each quadrant are generated from muxes located in the center of the device. All the clock sources are connected to these muxes. Figure 2-12 shows the clock routing for one quadrant. Each quadrant mux is identical. If desired, any clock can be routed globally.

**Figure 2-12. Per Quadrant Primary Clock Selection**



## Dynamic Clock Control (DCC)

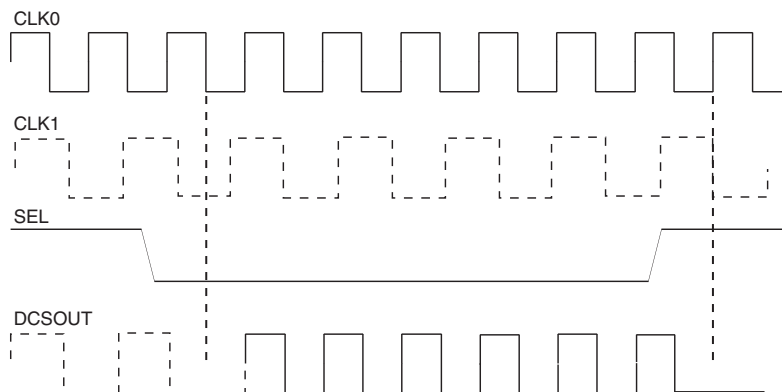
The DCC (Quadrant Clock Enable/Disable) feature allows internal logic control of the quadrant primary clock network. When a clock network is disabled, all the logic fed by that clock does not toggle, reducing the overall power consumption of the device.

## Dynamic Clock Select (DCS)

The DCS is a smart multiplexer function available in the primary clock routing. It switches between two independent input clock sources without any glitches or runt pulses. This is achieved regardless of when the select signal is toggled. There are two DCS blocks per quadrant; in total, there are eight DCS blocks per device. The inputs to the DCS block come from the center muxes. The output of the DCS is connected to primary clocks CLK6 and CLK7 (see Figure 2-12).

Figure 2-13 shows the timing waveforms of the default DCS operating mode. The DCS block can be programmed to other modes. For more information about the DCS, please see the list of technical documentation at the end of this data sheet.

**Figure 2-13. DCS Waveforms**

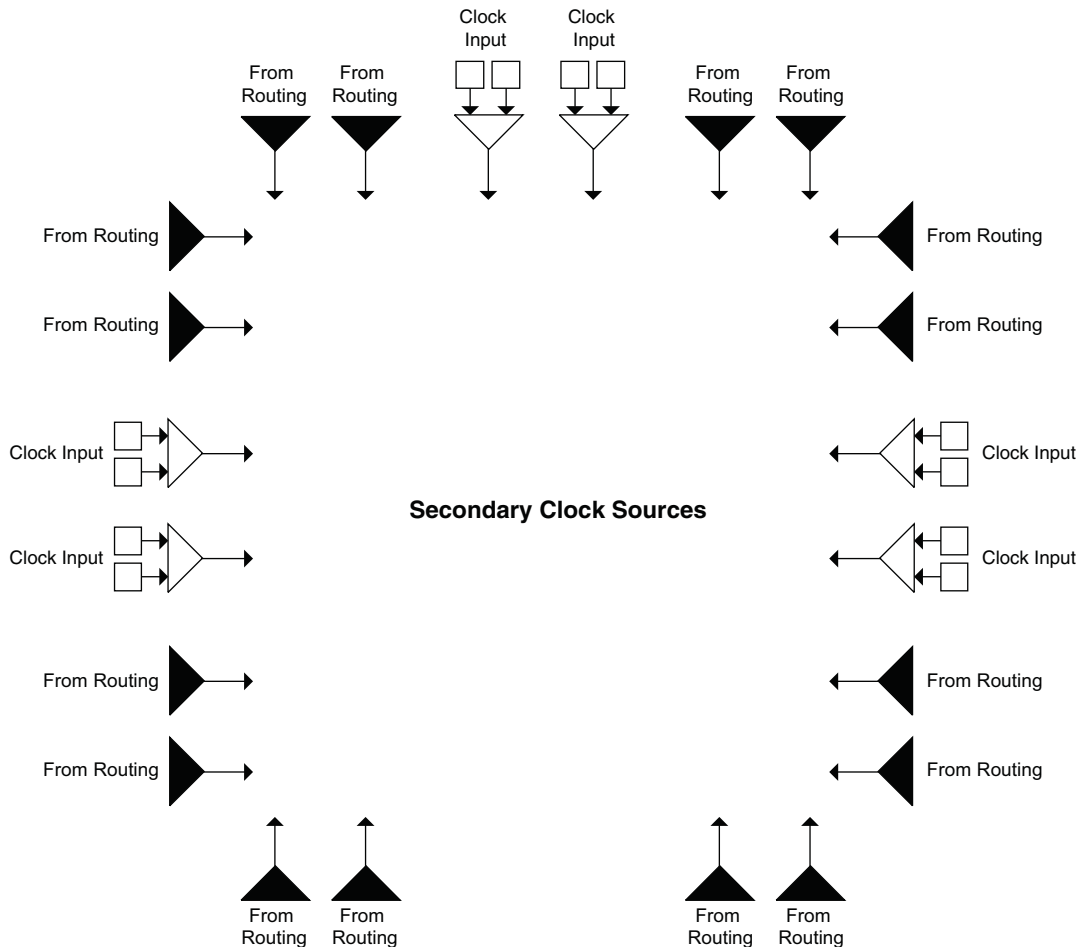


## Secondary Clock/Control Sources

LatticeECP3 devices derive eight secondary clock sources (SC0 through SC7) from six dedicated clock input pads and the rest from routing. Figure 2-14 shows the secondary clock sources. All eight secondary clock sources are defined as inputs to a per-region mux SC0-SC7. SC0-SC3 are primary for control signals (CE and/or LSR), and SC4-SC7 are for the clock.

In an actual implementation, there is some overlap to maximize routability. In addition to SC0-SC3, SC7 is also an input to the control signals (LSR or CE). SC0-SC2 are also inputs to clocks along with SC4-SC7.

**Figure 2-14. Secondary Clock Sources**



Note: Clock inputs can be configured in differential or single-ended mode.

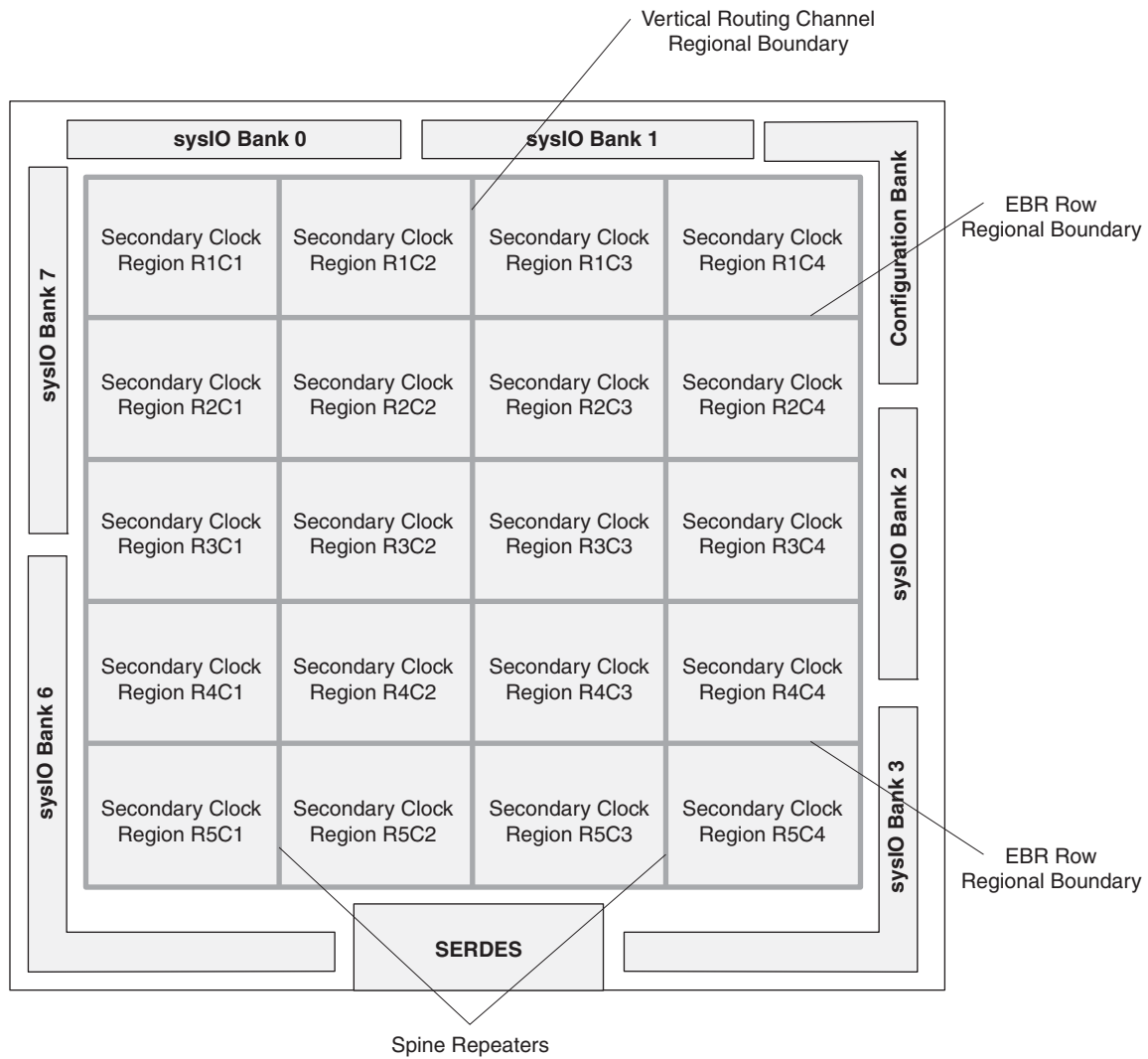
## Secondary Clock/Control Routing

Global secondary clock is a secondary clock that is distributed to all regions. The purpose of the secondary clock routing is to distribute the secondary clock sources to the secondary clock regions. Secondary clocks in the LatticeECP3 devices are region-based resources. Certain EBR rows and special vertical routing channels bind the secondary clock regions. This special vertical routing channel aligns with either the left edge of the center DSP slice in the DSP row or the center of the DSP row. Figure 2-15 shows this special vertical routing channel and the 20 secondary clock regions for the LatticeECP3 family of devices. All devices in the LatticeECP3 family have eight secondary clock resources per region (SC0 to SC7). The same secondary clock routing can be used for control signals.

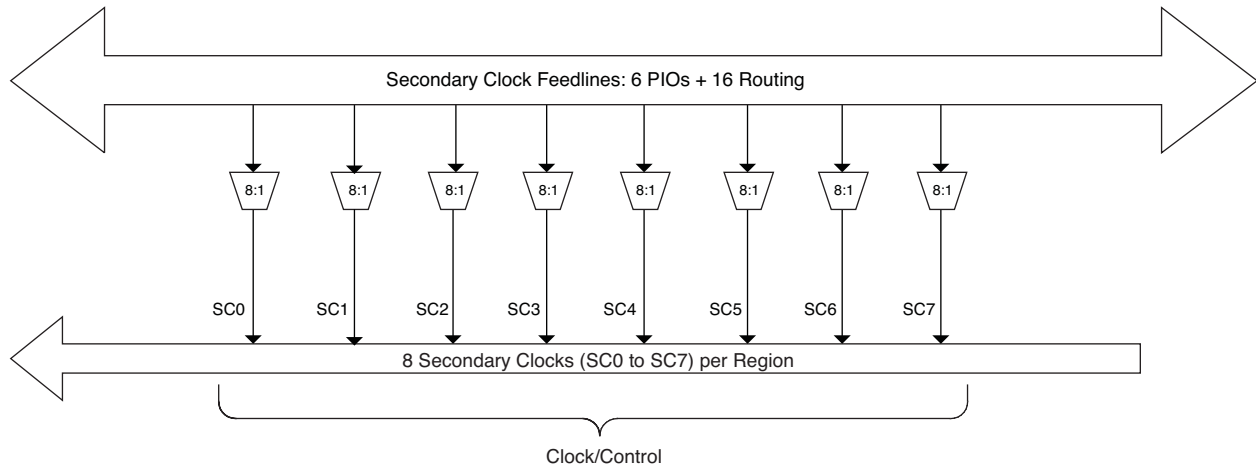
Table 2-6. Secondary Clock Regions

Device	Number of Secondary Clock Regions
ECP3-17	16
ECP3-35	16
ECP3-70	20
ECP3-95	20
ECP3-150	36

Figure 2-15. LatticeECP3-70 and LatticeECP3-95 Secondary Clock Regions



**Figure 2-16. Per Region Secondary Clock Selection**

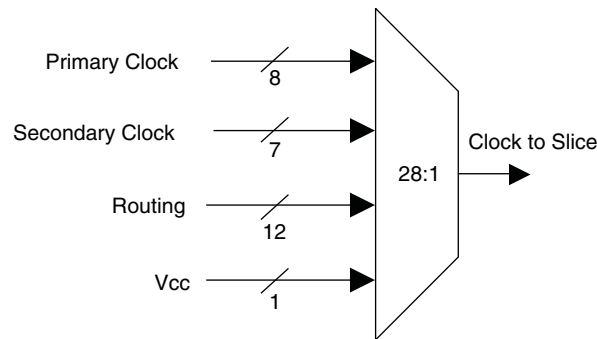


**Slice Clock Selection**

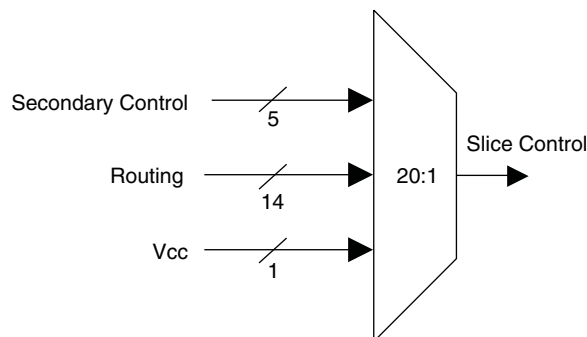
Figure 2-17 shows the clock selections and Figure 2-18 shows the control selections for Slice0 through Slice2. All the primary clocks and seven secondary clocks are routed to this clock selection mux. Other signals can be used as a clock input to the slices via routing. Slice controls are generated from the secondary clocks/controls or other signals connected via routing.

If none of the signals are selected for both clock and control then the default value of the mux output is 1. Slice 3 does not have any registers; therefore it does not have the clock or control muxes.

**Figure 2-17. Slice0 through Slice2 Clock Selection**



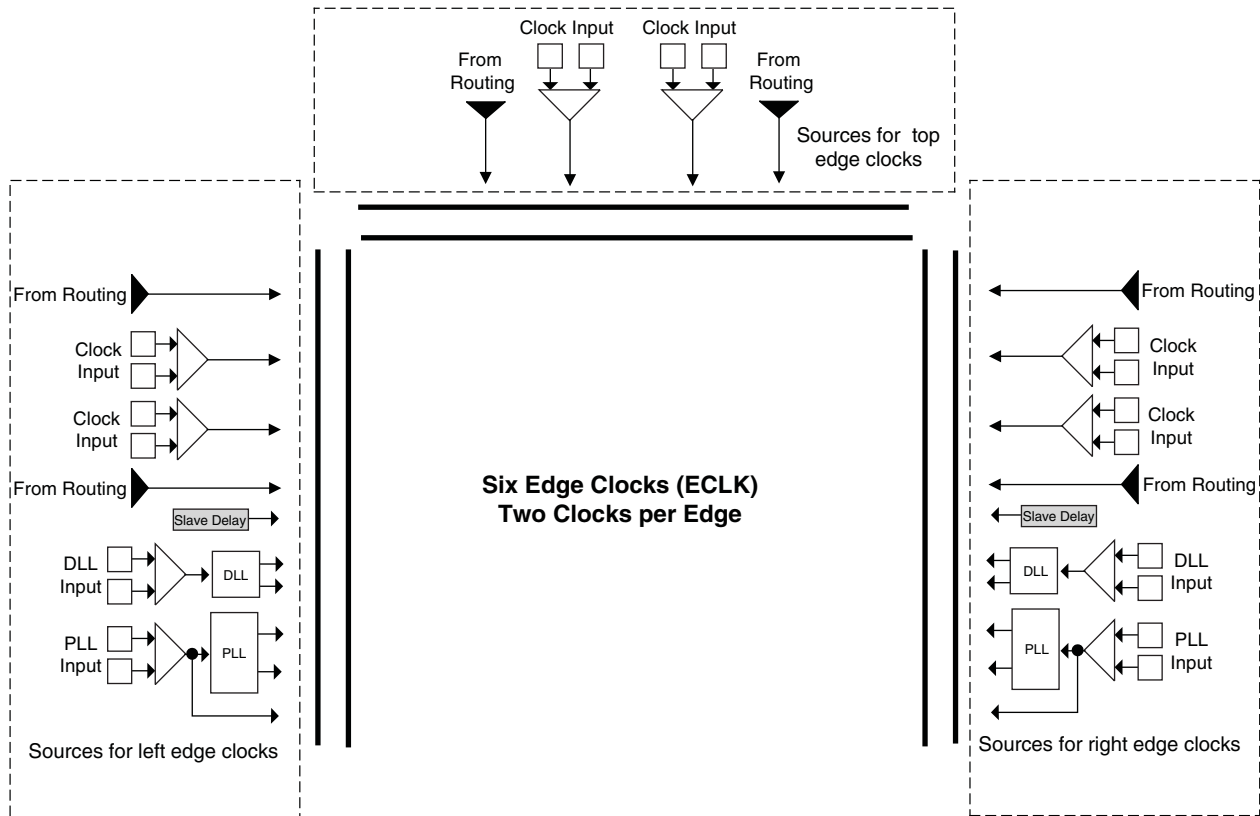
**Figure 2-18. Slice0 through Slice2 Control Selection**



## Edge Clock Sources

Edge clock resources can be driven from a variety of sources at the same edge. Edge clock resources can be driven from adjacent edge clock PIOs, primary clock PIOs, PLLs, DLLs, Slave Delay and clock dividers as shown in Figure 2-19.

**Figure 2-19. Edge Clock Sources**



**Notes:**

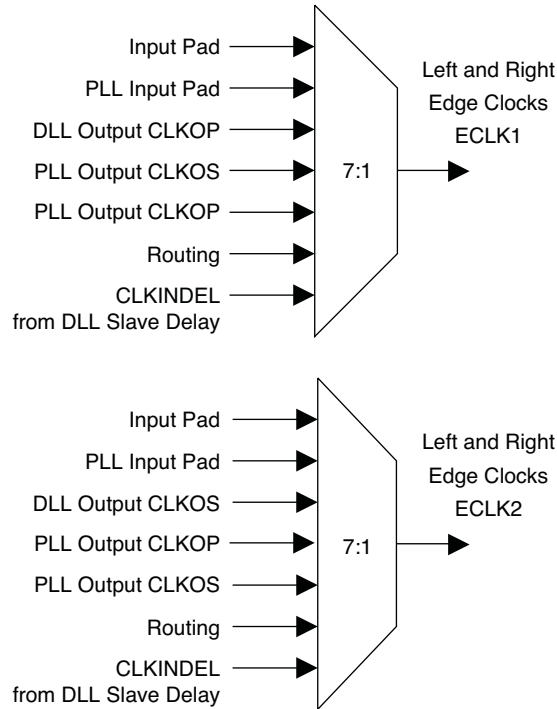
1. Clock inputs can be configured in differential or single ended mode.
2. The two DLLs can also drive the two top edge clocks.
3. The top left and top right PLL can also drive the two top edge clocks.

## Edge Clock Routing

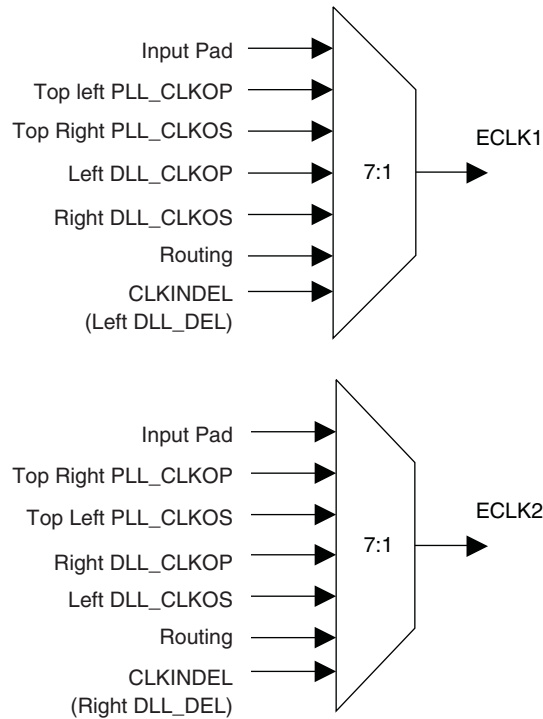
LatticeECP3 devices have a number of high-speed edge clocks that are intended for use with the PIOs in the implementation of high-speed interfaces. There are six edge clocks per device: two edge clocks on each of the top, left, and right edges. Different PLL and DLL outputs are routed to the two muxes on the left and right sides of the device. In addition, the CLKINDEL signal (generated from the DLL Slave Delay Line block) is routed to all the edge clock muxes on the left and right sides of the device. Figure 2-20 shows the selection muxes for these clocks.



**Figure 2-20. Sources of Edge Clock (Left and Right Edges)**



**Figure 2-21. Sources of Edge Clock (Top Edge)**



The edge clocks have low injection delay and low skew. They are used to clock the I/O registers and thus are ideal for creating I/O interfaces with a single clock signal and a wide data bus. They are also used for DDR Memory or Generic DDR interfaces.

The edge clocks on the top, left, and right sides of the device can drive the secondary clocks or general routing resources of the device. The left and right side edge clocks also can drive the primary clock network through the clock dividers (CLKDIV).

## sysMEM Memory

LatticeECP3 devices contain a number of sysMEM Embedded Block RAM (EBR). The EBR consists of an 18-Kbit RAM with memory core, dedicated input registers and output registers with separate clock and clock enable. Each EBR includes functionality to support true dual-port, pseudo dual-port, single-port RAM, ROM and FIFO buffers (via external PFUs).

### sysMEM Memory Block

The sysMEM block can implement single port, dual port or pseudo dual port memories. Each block can be used in a variety of depths and widths as shown in Table 2-7. FIFOs can be implemented in sysMEM EBR blocks by implementing support logic with PFUs. The EBR block facilitates parity checking by supporting an optional parity bit for each data byte. EBR blocks provide byte-enable support for configurations with 18-bit and 36-bit data widths. For more information, please see TN1179, [LatticeECP3 Memory Usage Guide](#).

**Table 2-7. sysMEM Block Configurations**

Memory Mode	Configurations
Single Port	16,384 x 1 8,192 x 2 4,096 x 4 2,048 x 9 1,024 x 18 512 x 36
True Dual Port	16,384 x 1 8,192 x 2 4,096 x 4 2,048 x 9 1,024 x 18
Pseudo Dual Port	16,384 x 1 8,192 x 2 4,096 x 4 2,048 x 9 1,024 x 18 512 x 36

### Bus Size Matching

All of the multi-port memory modes support different widths on each of the ports. The RAM bits are mapped LSB word 0 to MSB word 0, LSB word 1 to MSB word 1, and so on. Although the word size and number of words for each port varies, this mapping scheme applies to each port.

### RAM Initialization and ROM Operation

If desired, the contents of the RAM can be pre-loaded during device configuration. By preloading the RAM block during the chip configuration cycle and disabling the write controls, the sysMEM block can also be utilized as a ROM.

### Memory Cascading

Larger and deeper blocks of RAM can be created using EBR sysMEM Blocks. Typically, the Lattice design tools cascade memory transparently, based on specific design inputs.

## Single, Dual and Pseudo-Dual Port Modes

In all the sysMEM RAM modes the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

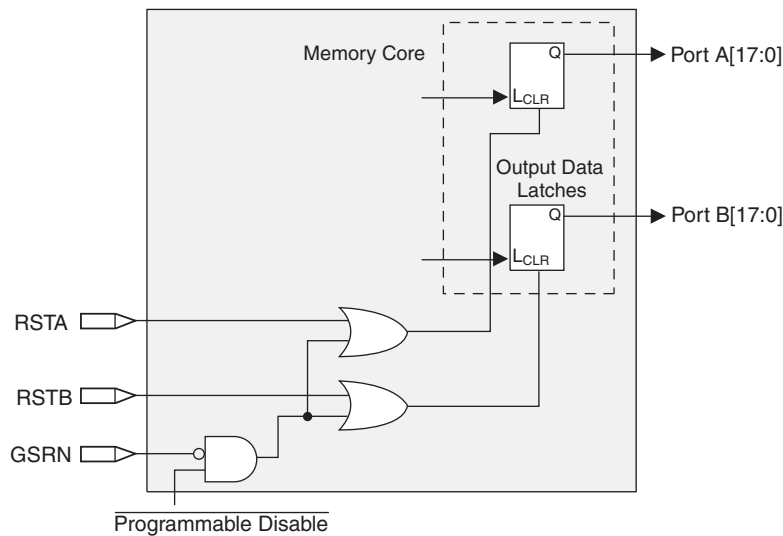
EBR memory supports the following forms of write behavior for single port or dual port operation:

1. **Normal** – Data on the output appears only during a read cycle. During a write cycle, the data (at the current address) does not appear on the output. This mode is supported for all data widths.
2. **Write Through** – A copy of the input data appears at the output of the same port during a write cycle. This mode is supported for all data widths.
3. **Read-Before-Write (EA devices only)** – When new data is written, the old content of the address appears at the output. This mode is supported for x9, x18, and x36 data widths.

## Memory Core Reset

The memory array in the EBR utilizes latches at the A and B output ports. These latches can be reset asynchronously or synchronously. RSTA and RSTB are local signals, which reset the output latches associated with Port A and Port B, respectively. The Global Reset (GSRN) signal can reset both ports. The output data latches and associated resets for both ports are as shown in Figure 2-22.

**Figure 2-22. Memory Core Reset**



For further information on the sysMEM EBR block, please see the list of technical documentation at the end of this data sheet.

## sysDSP™ Slice

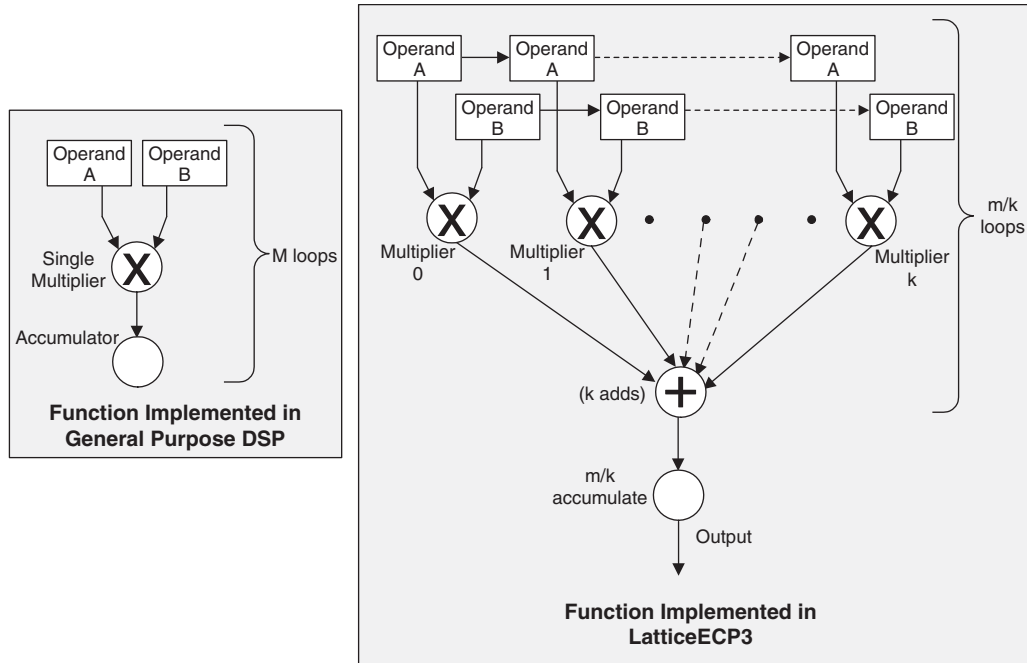
The LatticeECP3 family provides an enhanced sysDSP architecture, making it ideally suited for low-cost, high-performance Digital Signal Processing (DSP) applications. Typical functions used in these applications are Finite Impulse Response (FIR) filters, Fast Fourier Transforms (FFT) functions, Correlators, Reed-Solomon/Turbo/Convolution encoders and decoders. These complex signal processing functions use similar building blocks such as multiply-adders and multiply-accumulators.

### sysDSP Slice Approach Compared to General DSP

Conventional general-purpose DSP chips typically contain one to four (Multiply and Accumulate) MAC units with fixed data-width multipliers; this leads to limited parallelism and limited throughput. Their throughput is increased by higher clock speeds. The LatticeECP3, on the other hand, has many DSP slices that support different data widths.

This allows designers to use highly parallel implementations of DSP functions. Designers can optimize DSP performance vs. area by choosing appropriate levels of parallelism. Figure 2-23 compares the fully serial implementation to the mixed parallel and serial implementation.

**Figure 2-23. Comparison of General DSP and LatticeECP3 Approaches**



## LatticeECP3 sysDSP Slice Architecture Features

The LatticeECP3 sysDSP Slice has been significantly enhanced to provide functions needed for advanced processing applications. These enhancements provide improved flexibility and resource utilization.

The LatticeECP3 sysDSP Slice supports many functions that include the following:

- Multiply (one 18x36, two 18x18 or four 9x9 Multiplies per Slice)
- Multiply (36x36 by cascading across two sysDSP slices)
- Multiply Accumulate (up to 18x36 Multipliers feeding an Accumulator that can have up to 54-bit resolution)
- Two Multiplies feeding one Accumulate per cycle for increased processing with lower latency (two 18x18 Multiplies feed into an accumulator that can accumulate up to 52 bits)
- Flexible saturation and rounding options to satisfy a diverse set of applications situations
- Flexible cascading across DSP slices
  - Minimizes fabric use for common DSP and ALU functions
  - Enables implementation of FIR Filter or similar structures using dedicated sysDSP slice resources only
  - Provides matching pipeline registers
  - Can be configured to continue cascading from one row of sysDSP slices to another for longer cascade chains
- Flexible and Powerful Arithmetic Logic Unit (ALU) Supports:
  - Dynamically selectable ALU OPCODE
  - Ternary arithmetic (addition/subtraction of three inputs)
  - Bit-wise two-input logic operations (AND, OR, NAND, NOR, XOR and XNOR)
  - Eight flexible and programmable ALU flags that can be used for multiple pattern detection scenarios, such

- as, overflow, underflow and convergent rounding, etc.
- Flexible cascading across slices to get larger functions
- RTL Synthesis friendly synchronous reset on all registers, while still supporting asynchronous reset for legacy users
- Dynamic MUX selection to allow Time Division Multiplexing (TDM) of resources for applications that require processor-like flexibility that enables different functions for each clock cycle

For most cases, as shown in Figure 2-24, the LatticeECP3 DSP slice is backwards-compatible with the LatticeECP2™ sysDSP block, such that, legacy applications can be targeted to the LatticeECP3 sysDSP slice. The functionality of one LatticeECP2 sysDSP Block can be mapped into two adjacent LatticeECP3 sysDSP slices, as shown in Figure 2-25.

**Figure 2-24. Simplified sysDSP Slice Block Diagram**

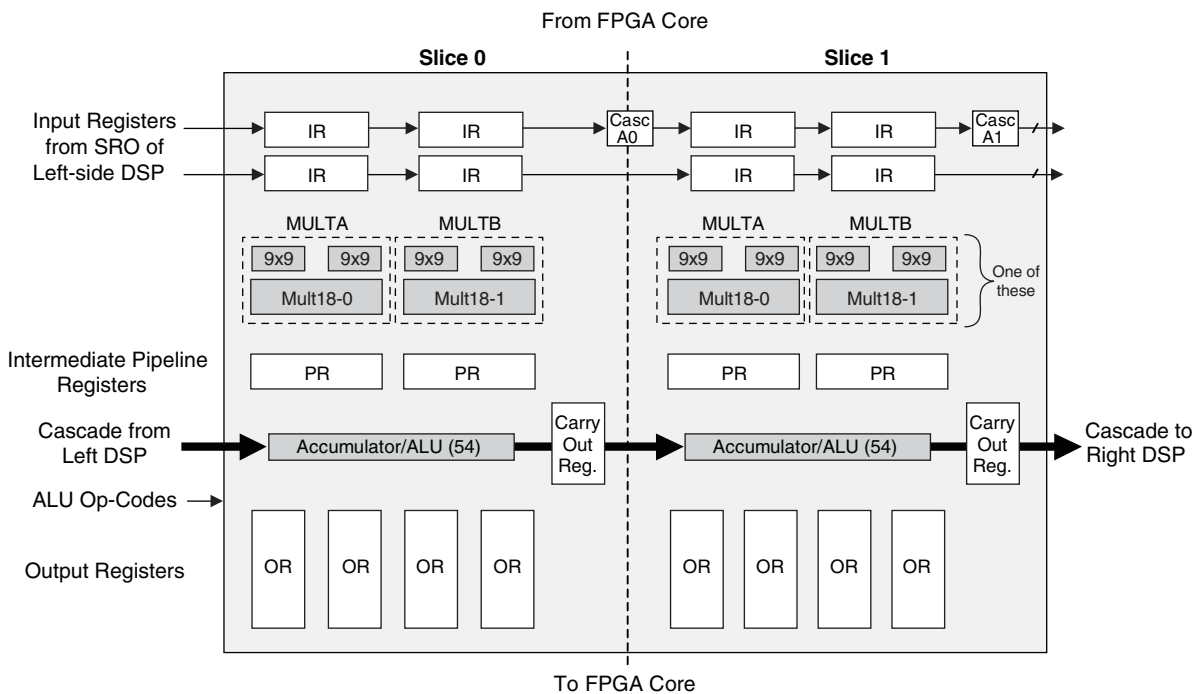
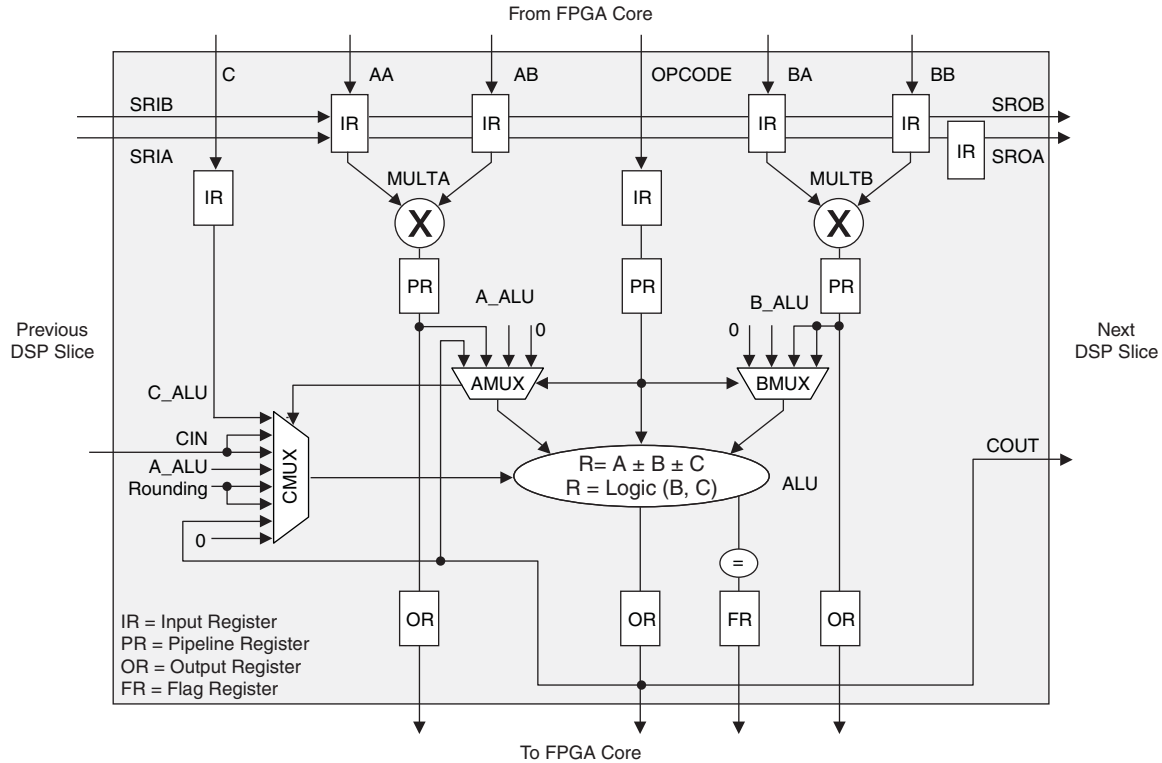


Figure 2-25. Detailed sysDSP Slice Diagram



Note: A\_ALU, B\_ALU and C\_ALU are internal signals generated by combining bits from AA, AB, BA BB and C inputs. See TN1182, LatticeECP3 sysDSP Usage Guide, for further information.

The LatticeECP2 sysDSP block supports the following basic elements.

- MULT (Multiply)
- MAC (Multiply, Accumulate)
- MULTADDSUB (Multiply, Addition/Subtraction)
- MULTADDSUBSUM (Multiply, Addition/Subtraction, Summation)

Table 2-8 shows the capabilities of each of the LatticeECP3 slices versus the above functions.

Table 2-8. Maximum Number of Elements in a Slice

Width of Multiply	x9	x18	x36
MULT	4	2	1/2
MAC	1	1	—
MULTADDSUB	2	1	—
MULTADDSUBSUM	1 <sup>1</sup>	1/2	—

1. One slice can implement 1/2 9x9 m9x9addsubsum and two m9x9addsubsum with two slices.

Some options are available in the four elements. The input register in all the elements can be directly loaded or can be loaded as a shift register from previous operand registers. By selecting “dynamic operation” the following operations are possible:

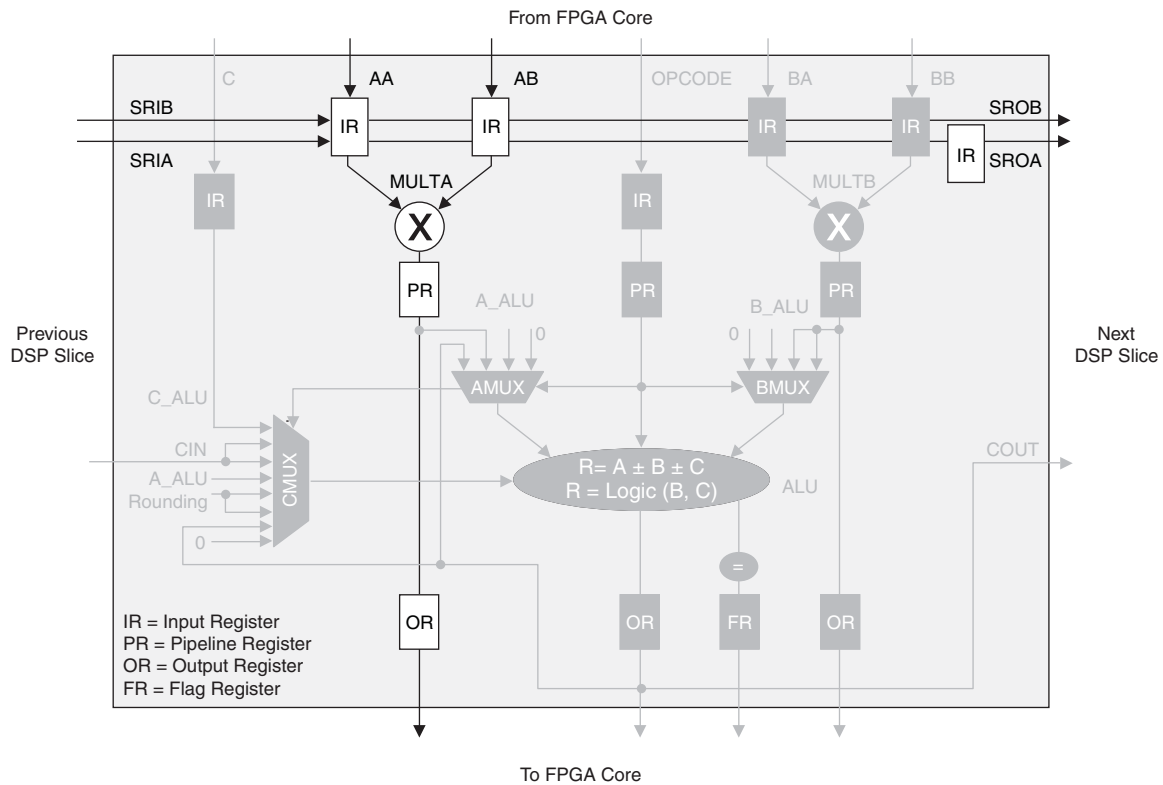
- In the Add/Sub option the Accumulator can be switched between addition and subtraction on every cycle.
- The loading of operands can switch between parallel and serial operations.

For further information, please refer to TN1182, [LatticeECP3 sysDSP Usage Guide](#).

### MULT DSP Element

This multiplier element implements a multiply with no addition or accumulator nodes. The two operands, AA and AB, are multiplied and the result is available at the output. The user can enable the input/output and pipeline registers. Figure 2-26 shows the MULT sysDSP element.

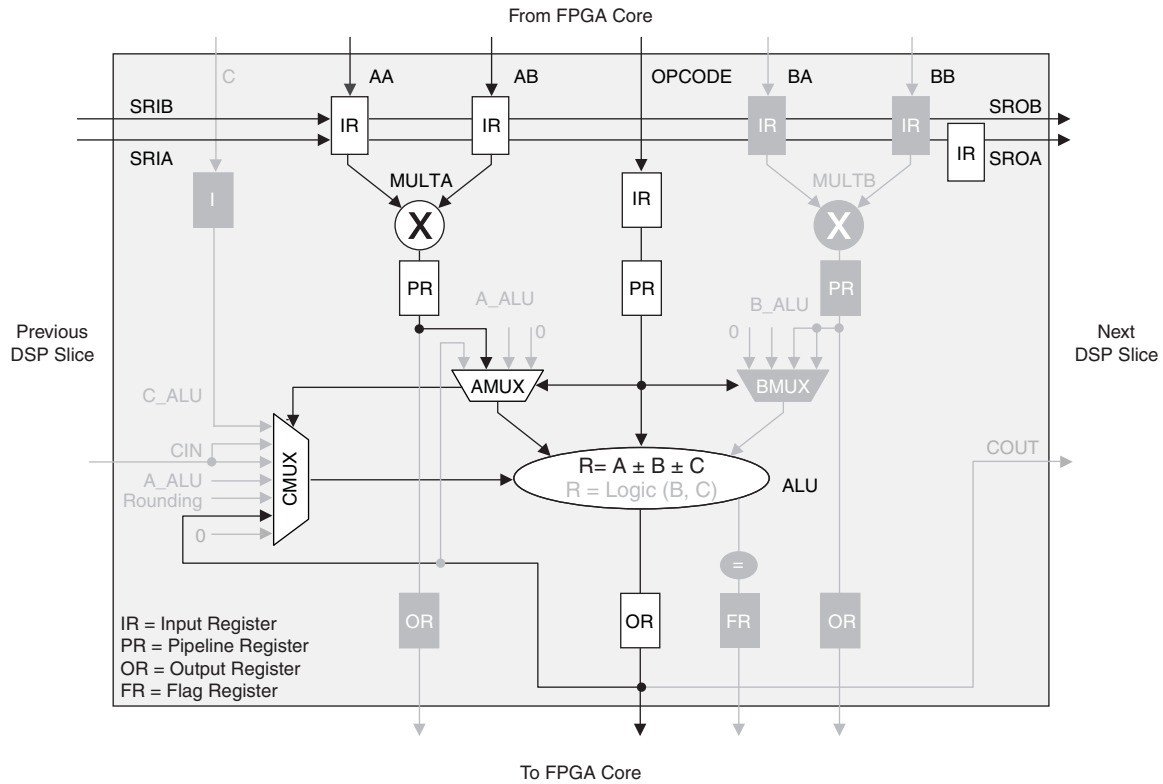
**Figure 2-26. MULT sysDSP Element**



### MAC DSP Element

In this case, the two operands, AA and AB, are multiplied and the result is added with the previous accumulated value. This accumulated value is available at the output. The user can enable the input and pipeline registers, but the output register is always enabled. The output register is used to store the accumulated value. The ALU is configured as the accumulator in the sysDSP slice in the LatticeECP3 family can be initialized dynamically. A registered overflow signal is also available. The overflow conditions are provided later in this document. Figure 2-27 shows the MAC sysDSP element.

**Figure 2-27. MAC DSP Element**

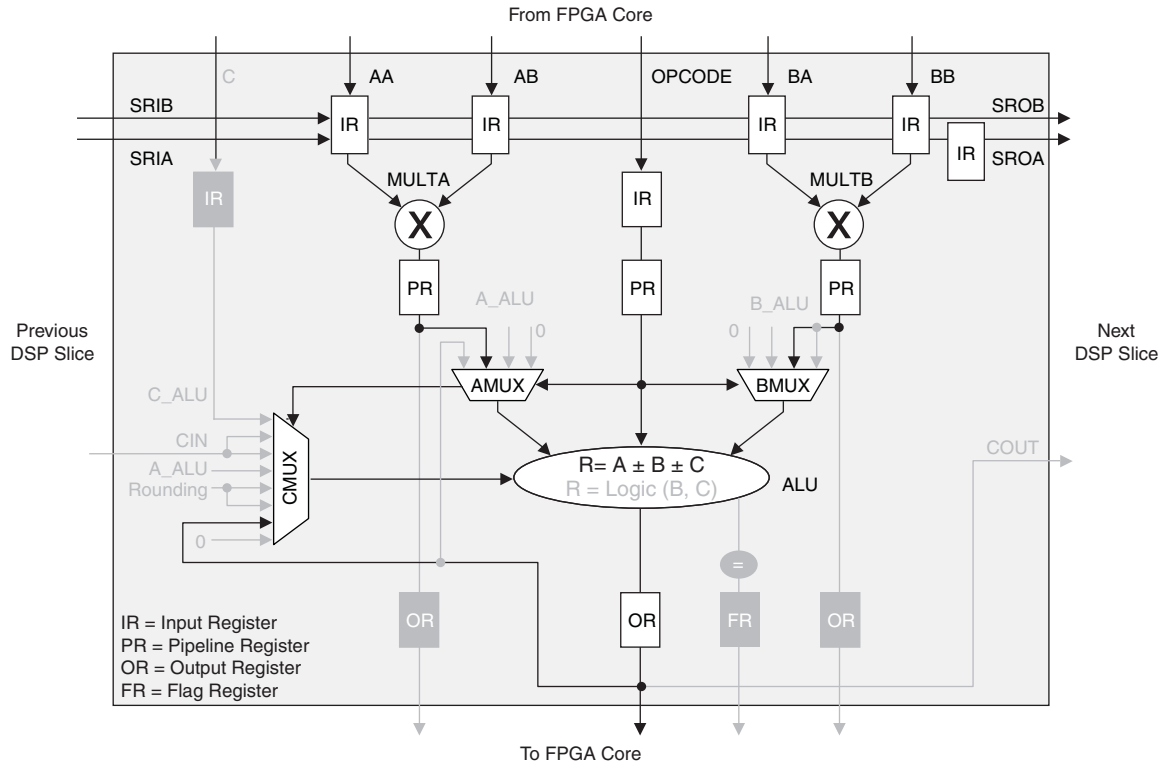




### MMAC DSP Element

The LatticeECP3 supports a MAC with two multipliers. This is called Multiply Multiply Accumulate or MMAC. In this case, the two operands, AA and AB, are multiplied and the result is added with the previous accumulated value and with the result of the multiplier operation of operands BA and BB. This accumulated value is available at the output. The user can enable the input and pipeline registers, but the output register is always enabled. The output register is used to store the accumulated value. The ALU is configured as the accumulator in the sysDSP slice. A registered overflow signal is also available. The overflow conditions are provided later in this document. Figure 2-28 shows the MMAC sysDSP element.

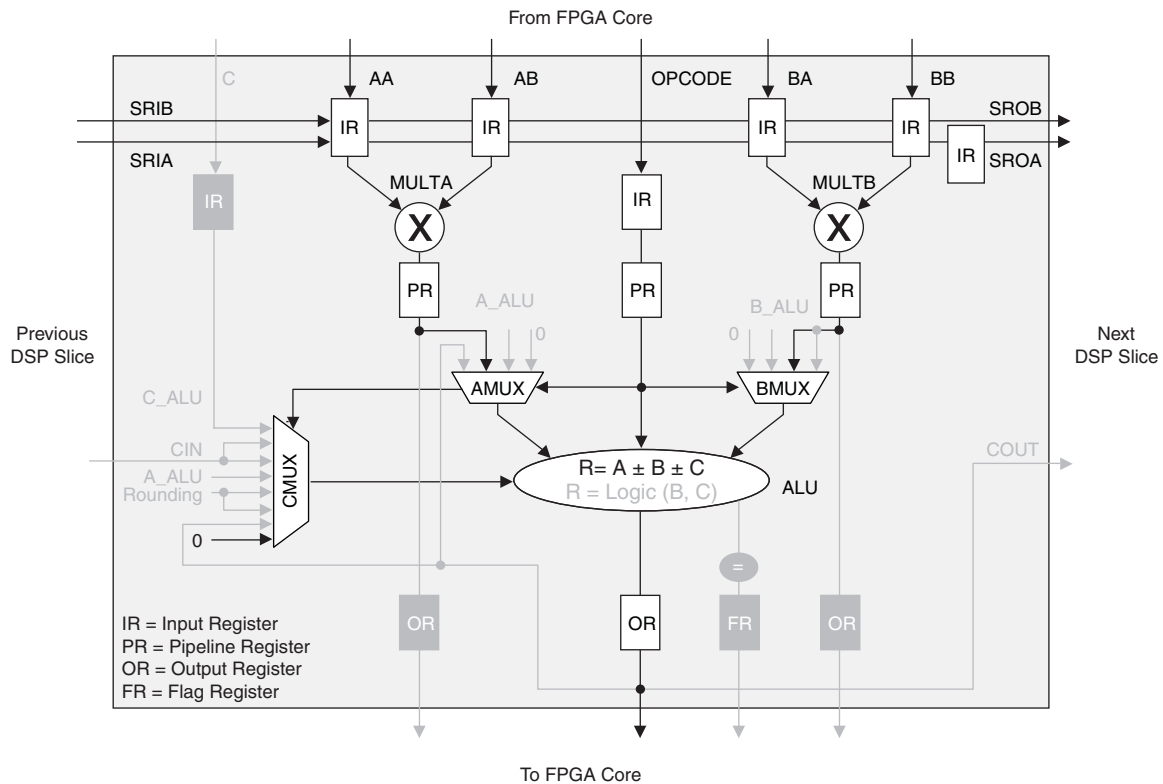
**Figure 2-28. MMAC sysDSP Element**



**MULTADDSUB DSP Element**

In this case, the operands AA and AB are multiplied and the result is added/subtracted with the result of the multiplier operation of operands BA and BB. The user can enable the input, output and pipeline registers. Figure 2-29 shows the MULTADDSUB sysDSP element.

**Figure 2-29. MULTADDSUB**



### MULTADDSUBSUM DSP Element

In this case, the operands AA and AB are multiplied and the result is added/subtracted with the result of the multiplier operation of operands BA and BB of Slice 0. Additionally, the operands AA and AB are multiplied and the result is added/subtracted with the result of the multiplier operation of operands BA and BB of Slice 1. The results of both addition/subtractions are added by the second ALU following the slice cascade path. The user can enable the input, output and pipeline registers. Figure 2-30 and Figure 2-31 show the MULTADDSUBSUM sysDSP element.

**Figure 2-30. MULTADDSUBSUM Slice 0**

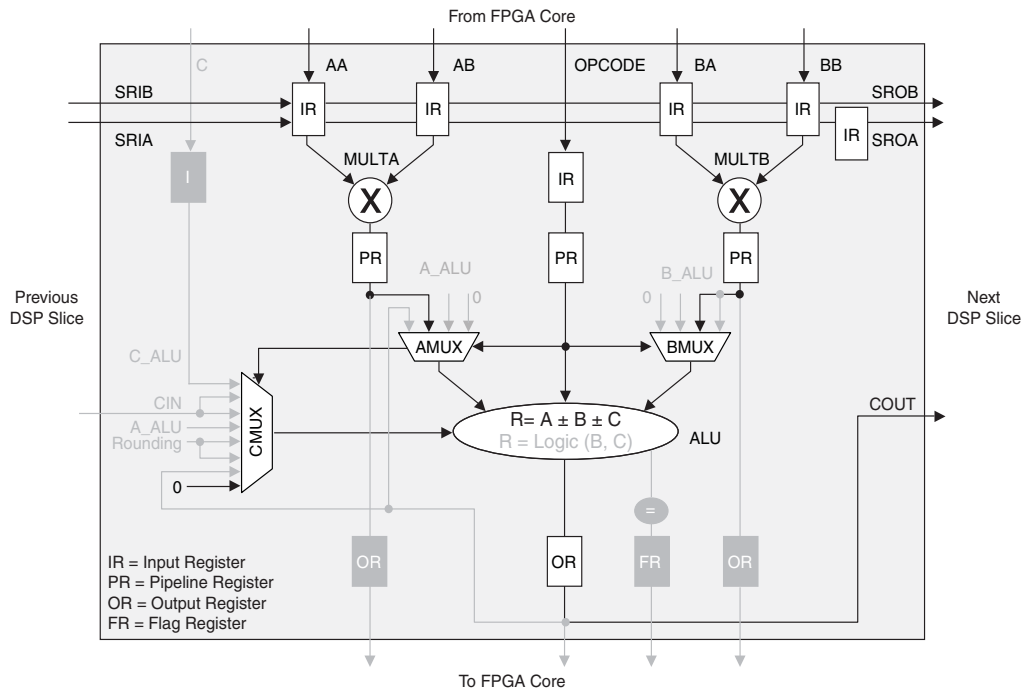
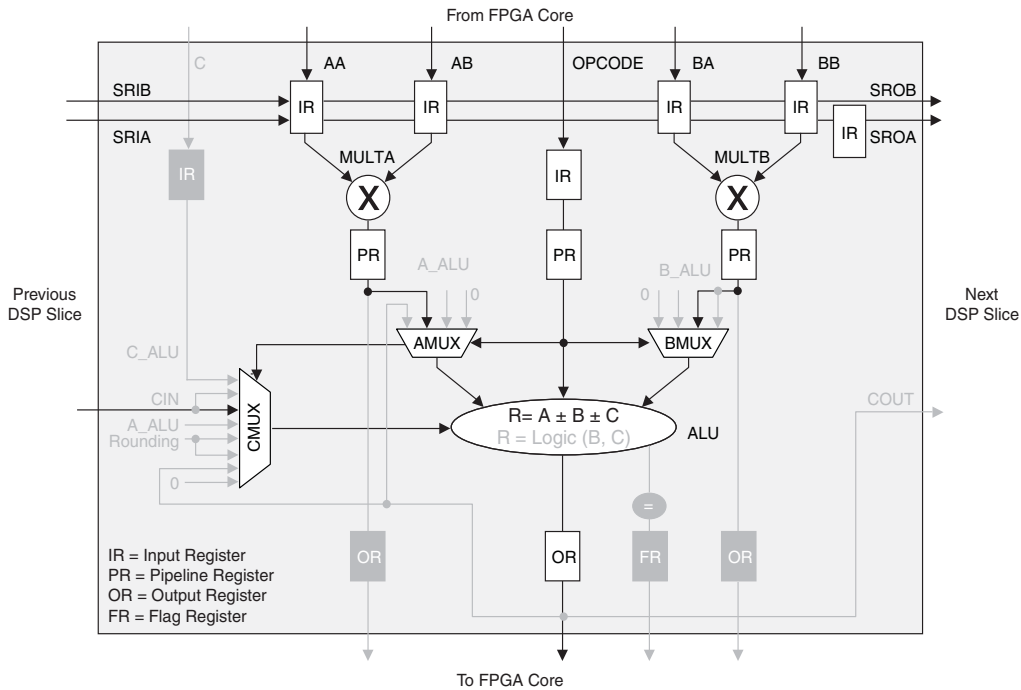


Figure 2-31. MULTADDSUBSUM Slice 1



## Advanced sysDSP Slice Features

### Cascading

The LatticeECP3 sysDSP slice has been enhanced to allow cascading. Adder trees are implemented fully in sysDSP slices, improving the performance. Cascading of slices uses the signals CIN, COUT and C Mux of the slice.

### Addition

The LatticeECP3 sysDSP slice allows for the bypassing of multipliers and cascading of adder logic. High performance adder functions are implemented without the use of LUTs. The maximum width adders that can be implemented are 54-bit.

### Rounding

The rounding operation is implemented in the ALU and is done by adding a constant followed by a truncation operation. The rounding methods supported are:

- Rounding to zero (RTZ)
- Rounding to infinity (RTI)
- Dynamic rounding
- Random rounding
- Convergent rounding

## ALU Flags

The sysDSP slice provides a number of flags from the ALU including:

- Equal to zero (EQZ)
- Equal to zero with mask (EQZM)
- Equal to one with mask (EQOM)
- Equal to pattern with mask (EQPAT)
- Equal to bit inverted pattern with mask (EQPATB)
- Accumulator Overflow (OVER)
- Accumulator Underflow (UNDER)
- Either over or under flow supporting LatticeECP2 legacy designs (OVERUNDER)

## Clock, Clock Enable and Reset Resources

Global Clock, Clock Enable and Reset signals from routing are available to every sysDSP slice. From four clock sources (CLK0, CLK1, CLK2, and CLK3) one clock is selected for each input register, pipeline register and output register. Similarly Clock Enable (CE) and Reset (RST) are selected at each input register, pipeline register and output register.

## Resources Available in the LatticeECP3 Family

Table 2-9 shows the maximum number of multipliers for each member of the LatticeECP3 family. Table 2-10 shows the maximum available EBR RAM Blocks in each LatticeECP3 device. EBR blocks, together with Distributed RAM can be used to store variables locally for fast DSP operations.

**Table 2-9. Maximum Number of DSP Slices in the LatticeECP3 Family**

Device	DSP Slices	9x9 Multiplier	18x18 Multiplier	36x36 Multiplier
ECP3-17	12	48	24	6
ECP3-35	32	128	64	16
ECP3-70	64	256	128	32
ECP3-95	64	256	128	32
ECP3-150	160	640	320	80

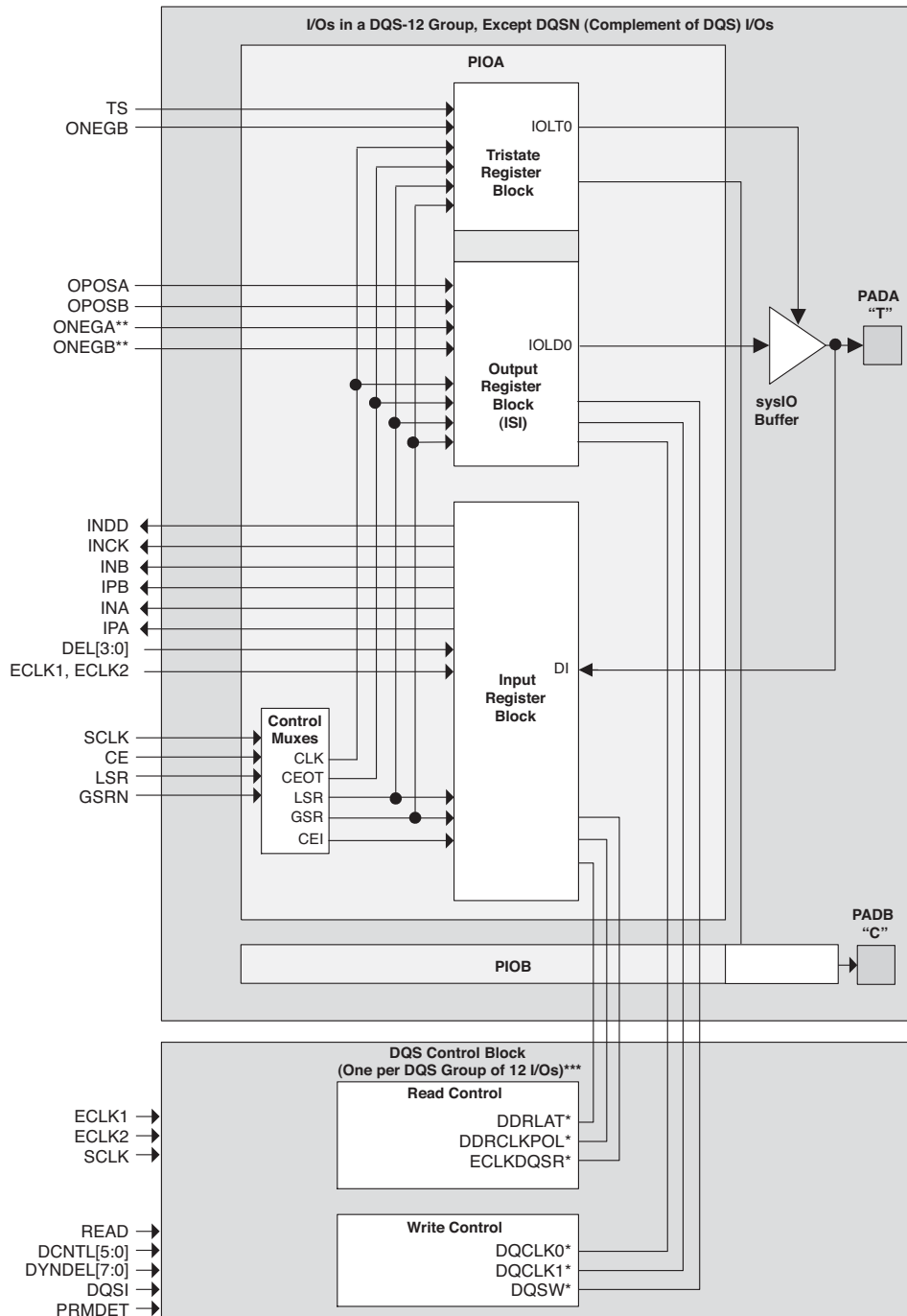
**Table 2-10. Embedded SRAM in the LatticeECP3 Family**

Device	EBR SRAM Block	Total EBR SRAM (Kbits)
ECP3-17	38	700
ECP3-35	72	1327
ECP3-70	240	4420
ECP3-95	240	4420
ECP3-150	372	6850

## Programmable I/O Cells (PIC)

Each PIC contains two PIOs connected to their respective sysI/O buffers as shown in Figure 2-32. The PIO Block supplies the output data (DO) and the tri-state control signal (TO) to the sysI/O buffer and receives input from the buffer. Table 2-11 provides the PIO signal list.

Figure 2-32. PIC Diagram



\* Signals are available on left/right/top edges only.  
 \*\* Signals are available on the left and right sides only  
 \*\*\* Selected PIO.

Two adjacent PIOs can be joined to provide a differential I/O pair (labeled as “T” and “C”) as shown in Figure 2-32. The PAD Labels “T” and “C” distinguish the two PIOs. Approximately 50% of the PIO pairs on the left and right edges of the device can be configured as true LVDS outputs. All I/O pairs can operate as LVDS inputs.

**Table 2-11. PIO Signal List**

Name	Type	Description
INDD	Input Data	Register bypassed input. This is not the same port as INCK.
IPA, INA, IPB, INB	Input Data	Ports to core for input data
OPOSA, ONEGA <sup>1</sup> , OPOSB, ONEGB <sup>1</sup>	Output Data	Output signals from core. An exception is the ONEGB port, used for tristate logic at the DQS pad.
CE	PIO Control	Clock enables for input and output block flip-flops.
SCLK	PIO Control	System Clock (PCLK) for input and output/TS blocks. Connected from clock ISB.
LSR	PIO Control	Local Set/Reset
ECLK1, ECLK2	PIO Control	Edge clock sources. Entire PIO selects one of two sources using mux.
ECLKDQSR <sup>1</sup>	Read Control	From DQS_STROBE, shifted strobe for memory interfaces only.
DDRCLKPOL <sup>1</sup>	Read Control	Ensures transfer from DQS domain to SCLK domain.
DDRLAT <sup>1</sup>	Read Control	Used to guarantee INDDR2 gearing by selectively enabling a D-Flip-Flop in datapath.
DEL[3:0]	Read Control	Dynamic input delay control bits.
INCK	To Clock Distribution and PLL	PIO treated as clock PIO, path to distribute to primary clocks and PLL.
TS	Tristate Data	Tristate signal from core (SDR)
DQCLK0 <sup>1</sup> , DQCLK1 <sup>1</sup>	Write Control	Two clocks edges, 90 degrees out of phase, used in output gearing.
DQSW <sup>2</sup>	Write Control	Used for output and tristate logic at DQS only.
DYNDEL[7:0]	Write Control	Shifting of write clocks for specific DQS group, using 6:0 each step is approximately 25ps, 128 steps. Bit 7 is an invert (timing depends on input frequency). There is also a static control for this 8-bit setting, enabled with a memory cell.
DCNTL[6:0]	PIO Control	Original delay code from DDR DLL
DATAVALID <sup>1</sup>	Output Data	Status flag from DATAVALID logic, used to indicate when input data is captured in IOLOGIC and valid to core.
READ	For DQS_Strobe	Read signal for DDR memory interface
DQSI	For DQS_Strobe	Unshifted DQS strobe from input pad
PRMBDET	For DQS_Strobe	DQSI biased to go high when DQSI is tristate, goes to input logic block as well as core logic.
GSRN	Control from routing	Global Set/Reset

1. Signals available on left/right/top edges only.

2. Selected PIO.

## PIO

The PIO contains four blocks: an input register block, output register block, tristate register block and a control logic block. These blocks contain registers for operating in a variety of modes along with the necessary clock and selection logic.

### Input Register Block

The input register blocks for the PIOs, in the left, right and top edges, contain delay elements and registers that can be used to condition high-speed interface signals, such as DDR memory interfaces and source synchronous interfaces, before they are passed to the device core. Figure 2-33 shows the input register block for the left, right and top edges. The input register block for the bottom edge contains one element to register the input signal and no DDR registers. The following description applies to the input register block for PIOs in the left, right and top edges only.

Input signals are fed from the sysI/O buffer to the input register block (as signal DI). If desired, the input signal can bypass the register and delay elements and be used directly as a combinatorial signal (INDD), a clock (INCK) and, in selected blocks, the input to the DQS delay block. If an input delay is desired, designers can select either a fixed delay or a dynamic delay DEL[3:0]. The delay, if selected, reduces input register hold time requirements when using a global clock.

The input block allows three modes of operation. In single data rate (SDR) the data is registered with the system clock by one of the registers in the single data rate sync register block.

In DDR mode, two registers are used to sample the data on the positive and negative edges of the modified DQS (ECLKDQSR) in the DDR Memory mode or ECLK signal when using DDR Generic mode, creating two data streams. Before entering the core, these two data streams are synchronized to the system clock to generate two data streams.

A gearbox function can be implemented in each of the input registers on the left and right sides. The gearbox function takes a double data rate signal applied to PIOA and converts it as four data streams, INA, IPA, INB and IPB. The two data streams from the first set of DDR registers are synchronized to the edge clock and then to the system clock before entering the core. Figure 2-30 provides further information on the use of the gearbox function.

The signal DDRCLKPOL controls the polarity of the clock used in the synchronization registers. It ensures adequate timing when data is transferred to the system clock domain from the ECLKDQSR (DDR Memory Interface mode) or ECLK (DDR Generic mode). The DDRLAT signal is used to ensure the data transfer from the synchronization registers to the clock transfer and gearbox registers.

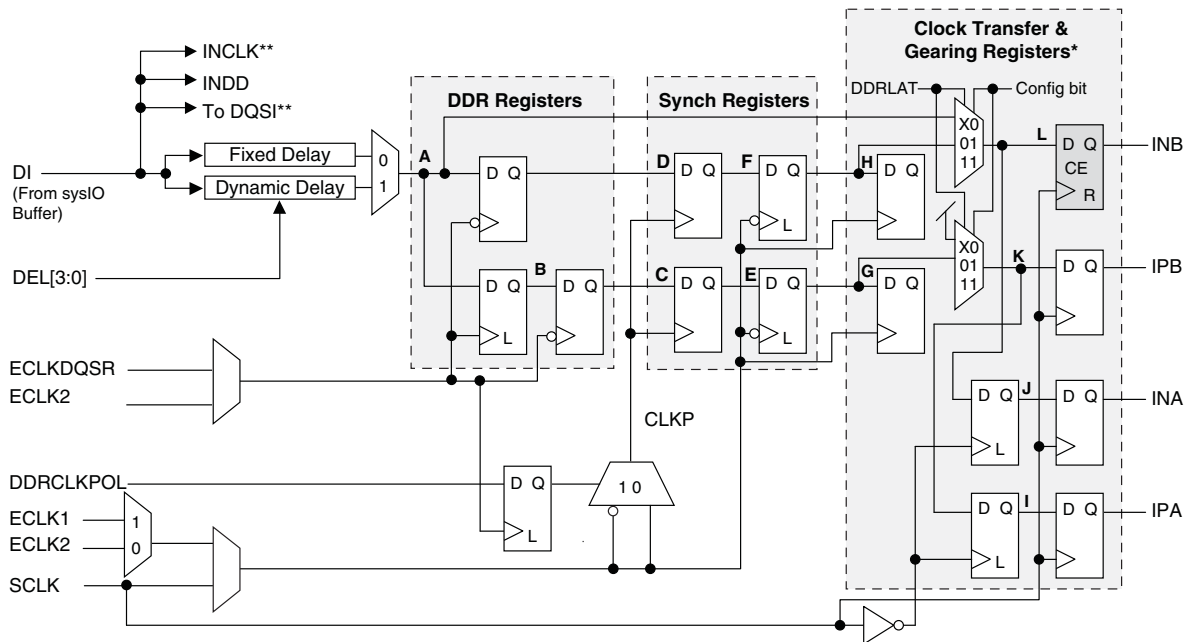
The ECLKDQSR, DDRCLKPOL and DDRLAT signals are generated in the DQS Read Control Logic Block. See Figure 2-37 for an overview of the DQS read control logic.

Further discussion about using the DQS strobe in this module is discussed in the DDR Memory section of this data sheet.

Please see TN1180, [LatticeECP3 High-Speed I/O Interface](#) for more information on this topic.



Figure 2-33. Input Register Block for Left, Right and Top Edges



\* Only on the left and right sides.  
 \*\* Selected PIO.  
 Note: Simplified diagram does not show CE/SET/REST details.

## Output Register Block

The output register block registers signals from the core of the device before they are passed to the sys/O buffers. The blocks on the left and right PIOs contain registers for SDR and full DDR operation. The topside PIO block is the same as the left and right sides except it does not support ODDR2 gearing of output logic. ODDR2 gearing is used in DDR3 memory interfaces. The PIO blocks on the bottom contain the SDR registers but do not support generic DDR.

Figure 2-34 shows the Output Register Block for PIOs on the left and right edges.

In SDR mode, OPOSA feeds one of the flip-flops that then feeds the output. The flip-flop can be configured as a Dtype or latch. In DDR mode, two of the inputs are fed into registers on the positive edge of the clock. At the next clock cycle, one of the registered outputs is also latched.

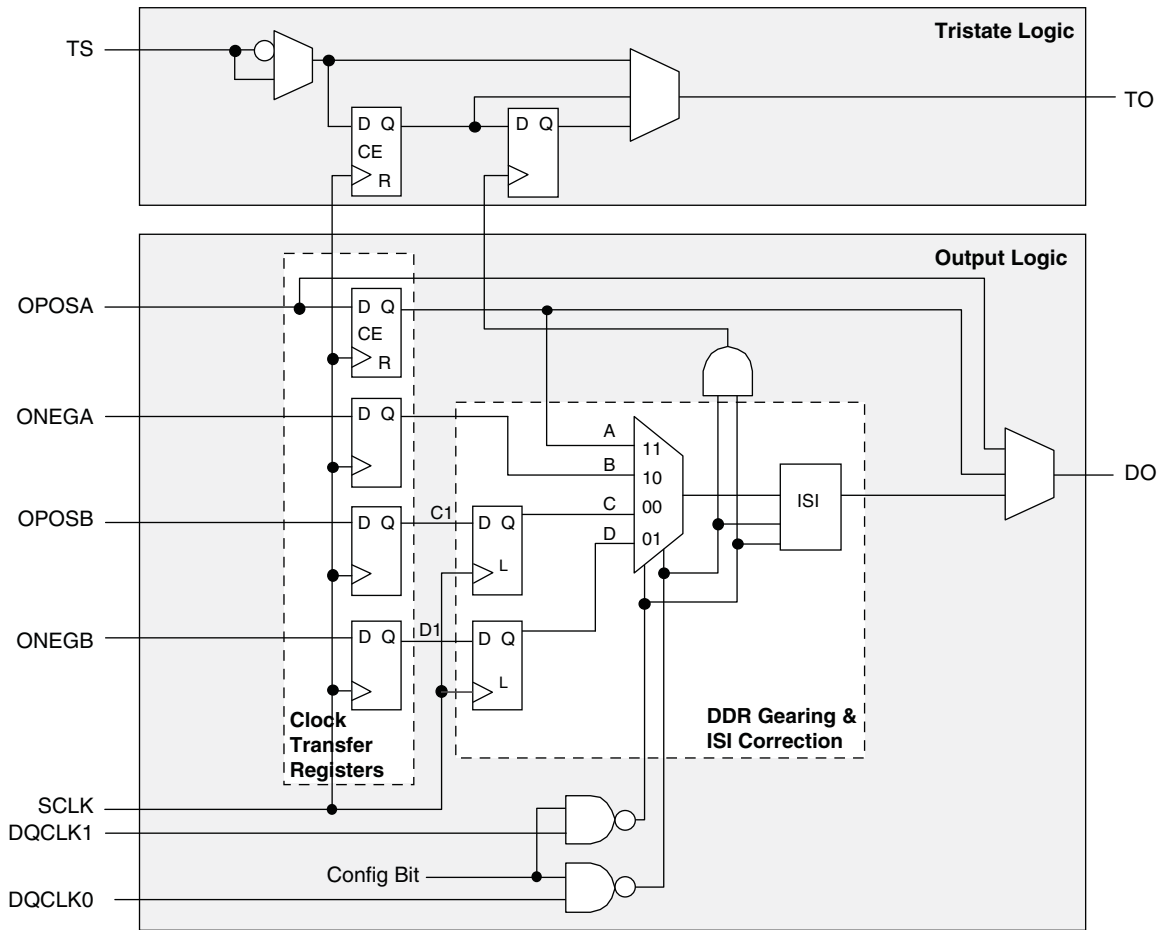
A multiplexer running off the same clock is used to switch the mux between the 11 and 01 inputs that will then feed the output.

A gearbox function can be implemented in the output register block that takes four data streams: OPOSA, ONEGA, OPOSB and ONEGB. All four data inputs are registered on the positive edge of the system clock and two of them are also latched. The data is then output at a high rate using a multiplexer that runs off the DQCLK0 and DQCLK1 clocks. DQCLK0 and DQCLK1 are used in this case to transfer data from the system clock to the edge clock domain. These signals are generated in the DQS Write Control Logic block. See Figure 2-37 for an overview of the DQS write control logic.

Please see TN1180, [LatticeECP3 High-Speed I/O Interface](#) for more information on this topic.

Further discussion on using the DQS strobe in this module is discussed in the DDR Memory section of this data sheet.

Figure 2-34. Output and Tristate Block for Left and Right Edges



### Tristate Register Block

The tristate register block registers tri-state control signals from the core of the device before they are passed to the sysI/O buffers. The block contains a register for SDR operation and an additional register for DDR operation.

In SDR and non-gearing DDR modes, TS input feeds one of the flip-flops that then feeds the output. In DDRX2 mode, the register TS input is fed into another register that is clocked using the DQCLK0 and DQCLK1 signals. The output of this register is used as a tristate control.

### ISI Calibration

The setting for Inter-Symbol Interference (ISI) cancellation occurs in the output register block. ISI correction is only available in the DDRX2 modes. ISI calibration settings exist once per output register block, so each I/O in a DQS-12 group may have a different ISI calibration setting.

The ISI block extends output signals at certain times, as a function of recent signal history. So, if the output pattern consists of a long strings of 0's to long strings of 1's, there are no delays on output signals. However, if there are quick, successive transitions from 010, the block will stretch out the binary 1. This is because the long trail of 0's will cause these symbols to interfere with the logic 1. Likewise, if there are quick, successive transitions from 101, the block will stretch out the binary 0. This block is controlled by a 3-bit delay control that can be set in the DQS control logic block.

For more information about this topic, please see the list of technical documentation at the end of this data sheet.

## Control Logic Block

The control logic block allows the selection and modification of control signals for use in the PIO block.

## DDR Memory Support

Certain PICs have additional circuitry to allow the implementation of high-speed source synchronous and DDR, DDR2 and DDR3 memory interfaces. The support varies by the edge of the device as detailed below.

### Left and Right Edges

The left and right sides of the PIC have fully functional elements supporting DDR, DDR2, and DDR3 memory interfaces. One of every 12 PIOs supports the dedicated DQS pins with the DQS control logic block. Figure 2-35 shows the DQS bus spanning 11 I/O pins. Two of every 12 PIOs support the dedicated DQS and DQS# pins with the DQS control logic block.

### Bottom Edge

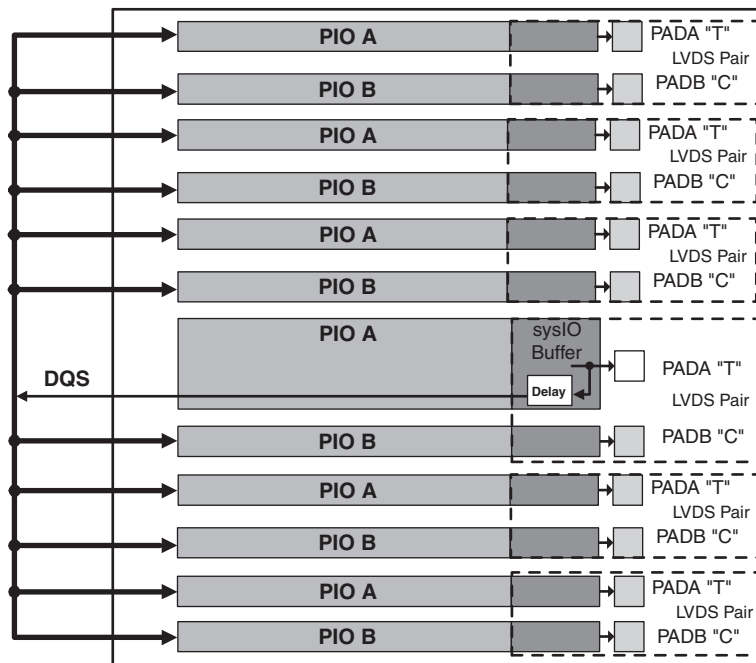
PICs on the bottom edge of the device do not support DDR memory and Generic DDR interfaces.

### Top Edge

PICs on the top side are similar to the PIO elements on the left and right sides but do not support gearing on the output registers. Hence, the modes to support output/tristate DDR3 memory are removed on the top side.

The exact DQS pins are shown in a dual function in the Logic Signal Connections table in this data sheet. Additional detail is provided in the Signal Descriptions table. The DQS signal from the bus is used to strobe the DDR data from the memory into input register blocks. Interfaces on the left, right and top edges are designed for DDR memories that support 10 bits of data.

**Figure 2-35. DQS Grouping on the Left, Right and Top Edges**



## DLL Calibrated DQS Delay Block

Source synchronous interfaces generally require the input clock to be adjusted in order to correctly capture data at the input register. For most interfaces, a PLL is used for this adjustment. However, in DDR memories the clock

(referred to as DQS) is not free-running so this approach cannot be used. The DQS Delay block provides the required clock alignment for DDR memory interfaces.

The delay required for the DQS signal is generated by two dedicated DLLs (DDR DLL) on opposite side of the device. Each DLL creates DQS delays in its half of the device as shown in Figure 2-36. The DDR DLL on the left side will generate delays for all the DQS Strobe pins on Banks 0, 7 and 6 and DDR DLL on the right will generate delays for all the DQS pins on Banks 1, 2 and 3. The DDR DLL loop compensates for temperature, voltage and process variations by using the system clock and DLL feedback loop. DDR DLL communicates the required delay to the DQS delay block using a 7-bit calibration bus (DCNTL[6:0])

The DQS signal (selected PIOs only, as shown in Figure 2-35) feeds from the PAD through a DQS control logic block to a dedicated DQS routing resource. The DQS control logic block consists of DQS Read Control logic block that generates control signals for the read side and DQS Write Control logic that generates the control signals required for the write side. A more detailed DQS control diagram is shown in Figure 2-37, which shows how the DQS control blocks interact with the data paths.

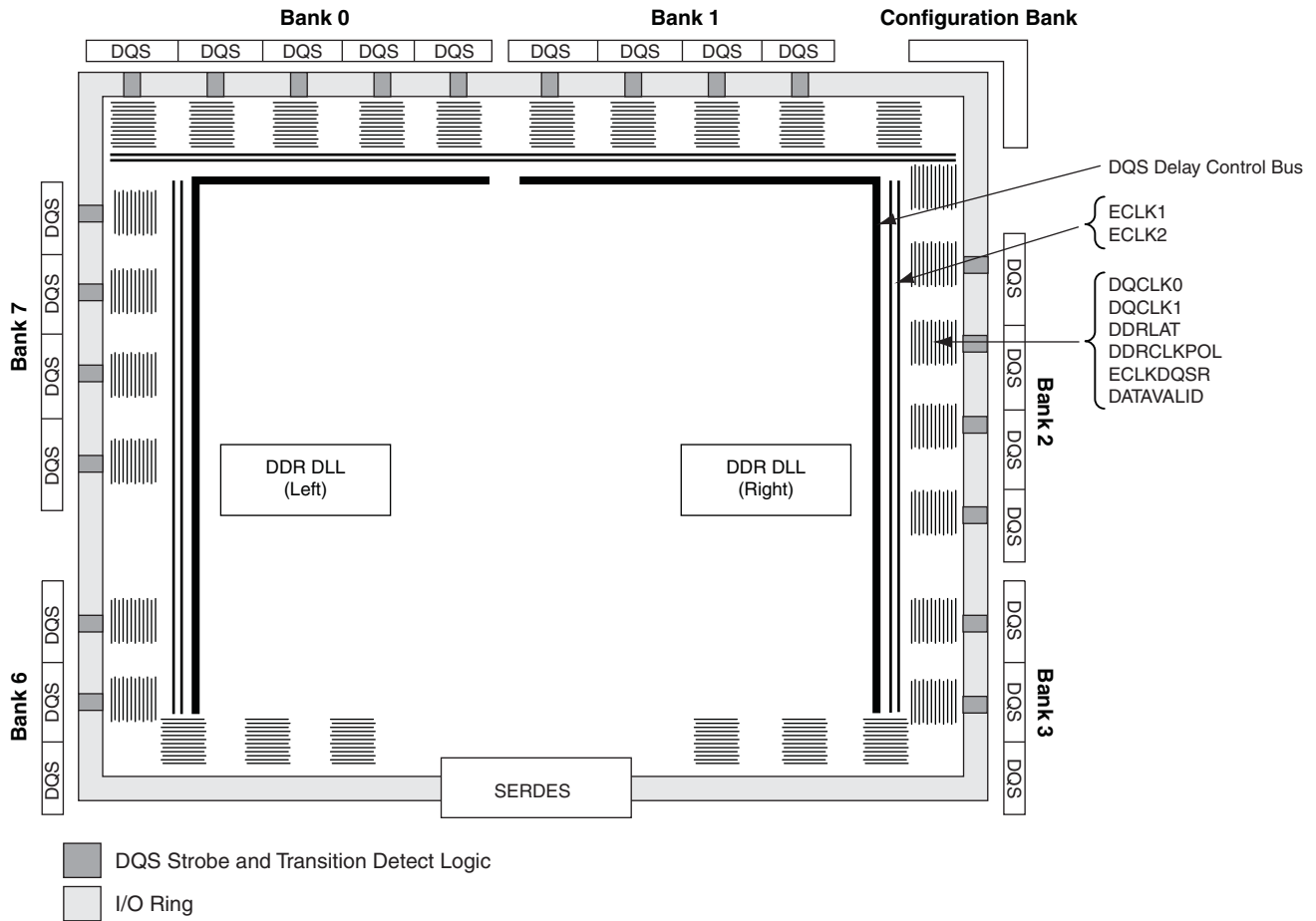
The DQS Read control logic receives the delay generated by the DDR DLL on its side and delays the incoming DQS signal by 90 degrees. This delayed ECLKDQSR is routed to 10 or 11 DQ pads covered by that DQS signal. This block also contains a polarity control logic that generates a DDRCLKPOL signal, which controls the polarity of the clock to the sync registers in the input register blocks. The DQS Read control logic also generates a DDRLAT signal that is in the input register block to transfer data from the first set of DDR register to the second set of DDR registers when using the DDRX2 gearbox mode for DDR3 memory interface.

The DQS Write control logic block generates the DQCLK0 and DQCLK1 clocks used to control the output gearing in the Output register block which generates the DDR data output and the DQS output. They are also used to control the generation of the DQS output through the DQS output register block. In addition to the DCNTL [6:0] input from the DDR DLL, the DQS Write control block also uses a Dynamic Delay DYN DEL [7:0] attribute which is used to further delay the DQS to accomplish the write leveling found in DDR3 memory. Write leveling is controlled by the DDR memory controller implementation. The DYN DELAY can set 128 possible delay step settings. In addition, the most significant bit will invert the clock for a 180-degree shift of the incoming clock. This will generate the DQSW signal used to generate the DQS output in the DQS output register block.

Figure 2-36 and Figure 2-37 show how the DQS transition signals that are routed to the PIOs.

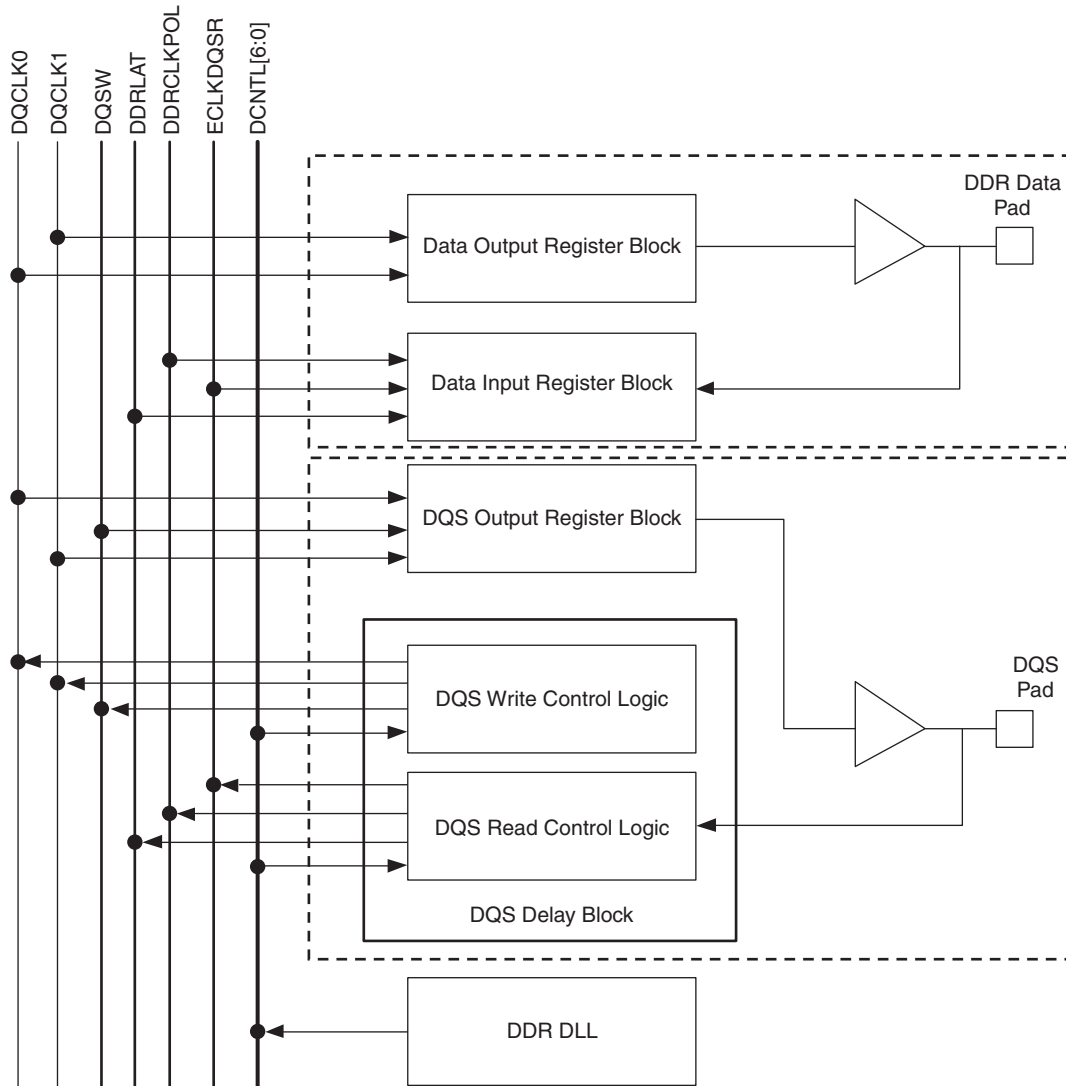
Please see TN1180, [LatticeECP3 High-Speed I/O Interface](#) for more information on this topic.

**Figure 2-36. Edge Clock, DLL Calibration and DQS Local Bus Distribution**



\*Includes shared configuration I/Os and dedicated configuration I/Os.

**Figure 2-37. DQS Local Bus**



### Polarity Control Logic

In a typical DDR Memory interface design, the phase relationship between the incoming delayed DQS strobe and the internal system clock (during the READ cycle) is unknown. The LatticeECP3 family contains dedicated circuits to transfer data between these domains. A clock polarity selector is used to prevent set-up and hold violations at the domain transfer between DQS (delayed) and the system clock. This changes the edge on which the data is registered in the synchronizing registers in the input register block. This requires evaluation at the start of each READ cycle for the correct clock polarity.

Prior to the READ operation in DDR memories, DQS is in tristate (pulled by termination). The DDR memory device drives DQS low at the start of the preamble state. A dedicated circuit detects the first DQS rising edge after the preamble state. This signal is used to control the polarity of the clock to the synchronizing registers.

### DDR3 Memory Support

LatticeECP3 supports the read and write leveling required for DDR3 memory interfaces.

Read leveling is supported by the use of the DDRCLKPOL and the DDRLAT signals generated in the DQS Read Control logic block. These signals dynamically control the capture of the data with respect to the DQS at the input register block.

To accomplish write leveling in DDR3, each DQS group has a slightly different delay that is set by DYN DELAY[7:0] in the DQS Write Control logic block. The DYN DELAY can set 128 possible delay step settings. In addition, the most significant bit will invert the clock for a 180-degree shift of the incoming clock.

LatticeECP3 input and output registers can also support DDR gearing that is used to receive and transmit the high speed DDR data from and to the DDR3 Memory.

LatticeECP3 supports the 1.5V SSTL I/O standard required for the DDR3 memory interface. For more information, refer to the sysIO section of this data sheet.

Please see TN1180, [LatticeECP3 High-Speed I/O Interface](#) for more information on DDR Memory interface implementation in LatticeECP3.

## sysI/O Buffer

Each I/O is associated with a flexible buffer referred to as a sysI/O buffer. These buffers are arranged around the periphery of the device in groups referred to as banks. The sysI/O buffers allow users to implement the wide variety of standards that are found in today's systems including LVDS, BLVDS, HSTL, SSTL Class I & II, LVCMOS, LVTTL, LVPECL, PCI.

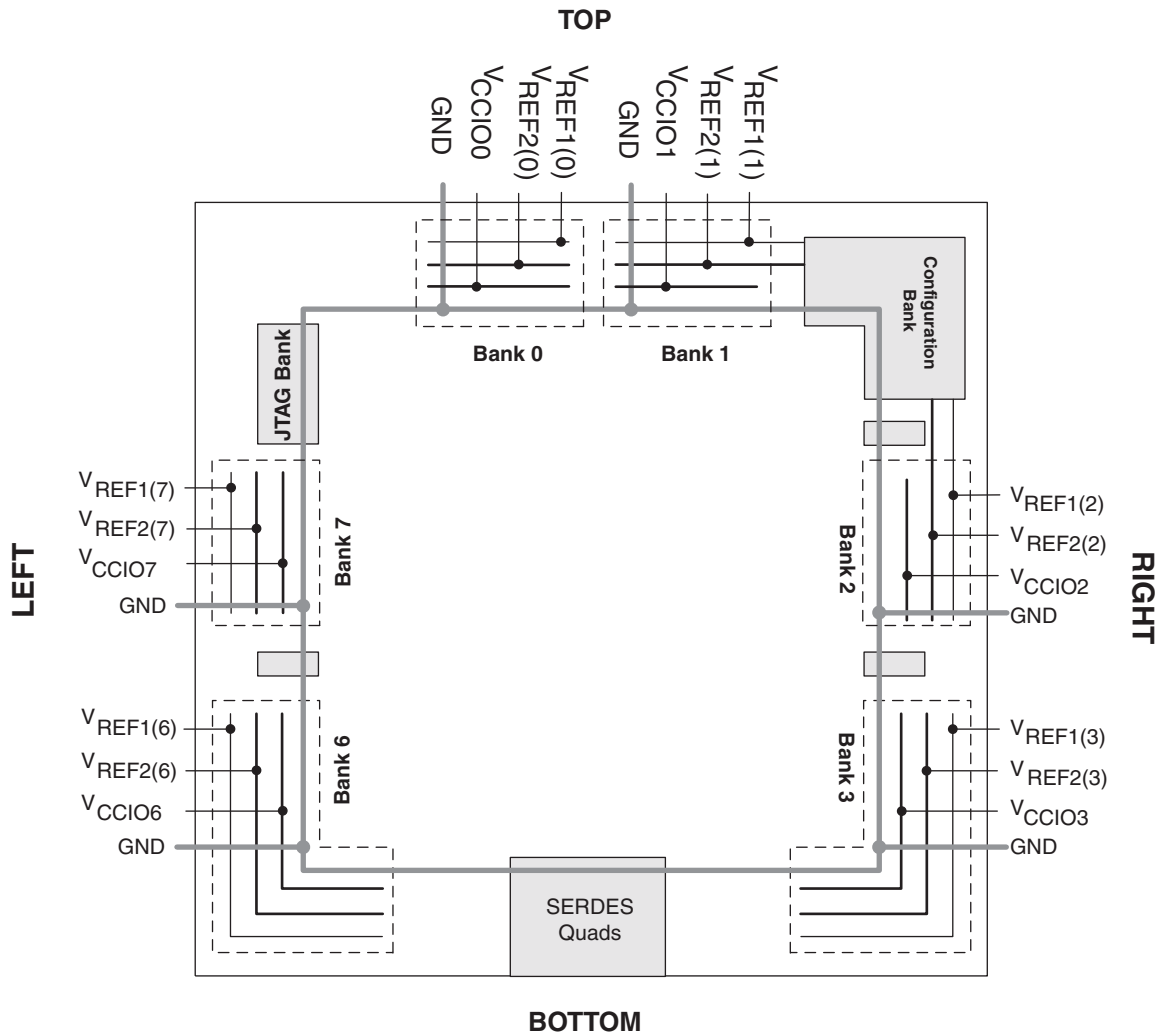
### sysI/O Buffer Banks

LatticeECP3 devices have six sysI/O buffer banks: six banks for user I/Os arranged two per side. The banks on the bottom side are wraparounds of the banks on the lower right and left sides. The seventh sysI/O buffer bank (Configuration Bank) is located adjacent to Bank 2 and has dedicated/shared I/Os for configuration. When a shared pin is not used for configuration it is available as a user I/O. Each bank is capable of supporting multiple I/O standards. Each sysI/O bank has its own I/O supply voltage ( $V_{CCIO}$ ). In addition, each bank, except the Configuration Bank, has voltage references,  $V_{REF1}$  and  $V_{REF2}$ , which allow it to be completely independent from the others. Figure 2-38 shows the seven banks and their associated supplies.

In LatticeECP3 devices, single-ended output buffers and ratioed input buffers (LVTTL, LVCMOS and PCI) are powered using  $V_{CCIO}$ . LVTTL, LVCMOS33, LVCMOS25 and LVCMOS12 can also be set as fixed threshold inputs independent of  $V_{CCIO}$ .

Each bank can support up to two separate  $V_{REF}$  voltages,  $V_{REF1}$  and  $V_{REF2}$ , that set the threshold for the referenced input buffers. Some dedicated I/O pins in a bank can be configured to be a reference voltage supply pin. Each I/O is individually configurable based on the bank's supply and reference voltages.

Figure 2-38. LatticeECP3 Banks



LatticeECP3 devices contain two types of sysI/O buffer pairs.

**1. Top (Bank 0 and Bank 1) and Bottom sysI/O Buffer Pairs (Single-Ended Outputs Only)**

The sysI/O buffer pairs in the top banks of the device consist of two single-ended output drivers and two sets of single-ended input buffers (both ratioed and referenced). One of the referenced input buffers can also be configured as a differential input. Only the top edge buffers have a programmable PCI clamp.

The two pads in the pair are described as “true” and “comp”, where the true pad is associated with the positive side of the differential input buffer and the comp (complementary) pad is associated with the negative side of the differential input buffer.

The top and bottom sides are ideal for general purpose I/O, PCI, and inputs for LVDS (LVDS outputs are only allowed on the left and right sides). The top side can be used for the DDR3 ADDR/CMD signals.

The I/O pins located on the top and bottom sides of the device (labeled PTxxA/B or PBxxA/B) are fully hot socketable. Note that the pads in Banks 3, 6 and 8 are wrapped around the corner of the device. In these banks, only the pads located on the top or bottom of the device are hot socketable. The top and bottom side pads can be identified by the Lattice Diamond tool.



## 2. Left and Right (Banks 2, 3, 6 and 7) sysI/O Buffer Pairs (50% Differential and 100% Single-Ended Outputs)

The sysI/O buffer pairs in the left and right banks of the device consist of two single-ended output drivers, two sets of single-ended input buffers (both ratioed and referenced) and one differential output driver. One of the referenced input buffers can also be configured as a differential input. In these banks the two pads in the pair are described as “true” and “comp”, where the true pad is associated with the positive side of the differential I/O, and the comp (complementary) pad is associated with the negative side of the differential I/O.

In addition, programmable on-chip input termination (parallel or differential, static or dynamic) is supported on these sides, which is required for DDR3 interface. However, there is no support for hot-socketing for the I/O pins located on the left and right side of the device as the PCI clamp is always enabled on these pins.

LVDS, RSDS, PPLVDS and Mini-LVDS differential output drivers are available on 50% of the buffer pairs on the left and right banks.

## 3. Configuration Bank sysI/O Buffer Pairs (Single-Ended Outputs, Only on Shared Pins When Not Used by Configuration)

The sysI/O buffers in the Configuration Bank consist of ratioed single-ended output drivers and single-ended input buffers. This bank does not support PCI clamp like the other banks on the top, left, and right sides.

The two pads in the pair are described as “true” and “comp”, where the true pad is associated with the positive side of the differential input buffer and the comp (complementary) pad is associated with the negative side of the differential input buffer.

Programmable PCI clamps are only available on the top banks. PCI clamps are used primarily on inputs and bi-directional pads to reduce ringing on the receiving end.

## Typical sysI/O I/O Behavior During Power-up

The internal power-on-reset (POR) signal is deactivated when  $V_{CC}$ ,  $V_{CCIO8}$  and  $V_{CCAUX}$  have reached satisfactory levels. After the POR signal is deactivated, the FPGA core logic becomes active. It is the user's responsibility to ensure that all other  $V_{CCIO}$  banks are active with valid input logic levels to properly control the output logic states of all the I/O banks that are critical to the application. For more information about controlling the output logic state with valid input logic levels during power-up in LatticeECP3 devices, see the list of technical documentation at the end of this data sheet.

The  $V_{CC}$  and  $V_{CCAUX}$  supply the power to the FPGA core fabric, whereas the  $V_{CCIO}$  supplies power to the I/O buffers. In order to simplify system design while providing consistent and predictable I/O behavior, it is recommended that the I/O buffers be powered-up prior to the FPGA core fabric.  $V_{CCIO}$  supplies should be powered-up before or together with the  $V_{CC}$  and  $V_{CCAUX}$  supplies.

## Supported sysI/O Standards

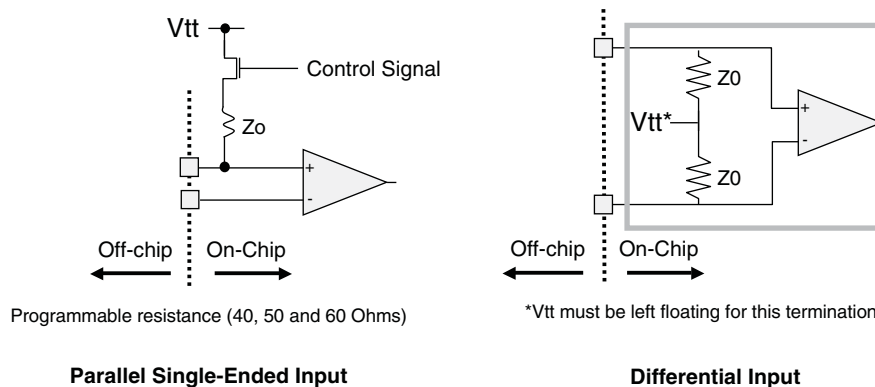
The LatticeECP3 sysI/O buffer supports both single-ended and differential standards. Single-ended standards can be further subdivided into LVCMOS, LVTTTL and other standards. The buffers support the LVTTTL, LVCMOS 1.2V, 1.5V, 1.8V, 2.5V and 3.3V standards. In the LVCMOS and LVTTTL modes, the buffer has individual configuration options for drive strength, slew rates, bus maintenance (weak pull-up, weak pull-down, or a bus-keeper latch) and open drain. Other single-ended standards supported include SSTL and HSTL. Differential standards supported include LVDS, BLVDS, LVPECL, MLVDS, RSDS, Mini-LVDS, PPLVDS (point-to-point LVDS), TRLVDS (Transition Reduced LVDS), differential SSTL and differential HSTL. For further information on utilizing the sysI/O buffer to support a variety of standards please see TN1177, [LatticeECP3 sysI/O Usage Guide](#).

## On-Chip Programmable Termination

The LatticeECP3 supports a variety of programmable on-chip terminations options, including:

- Dynamically switchable Single-Ended Termination with programmable resistor values of 40, 50, or 60 ohms. External termination to Vtt should be used for DDR2 and DDR3 memory controller implementation.
- Common mode termination of 80, 100, 120 ohms for differential inputs

**Figure 2-39. On-Chip Termination**



See Table 2-12 for termination options for input modes.

**Table 2-12. On-Chip Termination Options for Input Modes**

IO_TYPE	TERMINATE to VTT <sup>1,2</sup>	DIFFERENTIAL TERMINATION RESISTOR <sup>1</sup>
LVDS25	␣	80, 100, 120
BLVDS25	␣	80, 100, 120
MLVDS	␣	80, 100, 120
HSTL18_I	40, 50, 60	␣
HSTL18_II	40, 50, 60	␣
HSTL18D_I	40, 50, 60	␣
HSTL18D_II	40, 50, 60	␣
HSTL15_I	40, 50, 60	␣
HSTL15D_I	40, 50, 60	␣
SSTL25_I	40, 50, 60	␣
SSTL25_II	40, 50, 60	␣
SSTL25D_I	40, 50, 60	␣
SSTL25D_II	40, 50, 60	␣
SSTL18_I	40, 50, 60	␣
SSTL18_II	40, 50, 60	␣
SSTL18D_I	40, 50, 60	␣
SSTL18D_II	40, 50, 60	␣
SSTL15	40, 50, 60	␣
SSTL15D	40, 50, 60	␣

1. TERMINATE to VTT and DIFFERENTIAL TERMINATION RESISTOR when turned on can only have one setting per bank. Only left and right banks have this feature. Use of TERMINATE to VTT and DIFFERENTIAL TERMINATION RESISTOR are mutually exclusive in an I/O bank. On-chip termination tolerance +/- 20%
2. External termination to VTT should be used when implementing DDR2 and DDR3 memory controller.

Please see TN1177, [LatticeECP3 sysIO Usage Guide](#) for on-chip termination usage and value ranges.

## Equalization Filter

Equalization filtering is available for single-ended inputs on both true and complementary I/Os, and for differential inputs on the true I/Os on the left, right, and top sides. Equalization is required to compensate for the difficulty of sampling alternating logic transitions with a relatively slow slew rate. It is considered the most useful for the Input DDRX2 modes, used in DDR3 memory, LVDS, or TRLVDS signaling. Equalization filter acts as a tunable filter with settings to determine the level of correction. In the LatticeECP3 devices, there are four settings available: 0 (none), 1, 2 and 3. The default setting is 0. The equalization logic resides in the sysI/O buffers, the two bits of setting is set uniquely in each input IOLOGIC block. Therefore, each sysI/O can have a unique equalization setting within a DQS-12 group.

## Hot Socketing

LatticeECP3 devices have been carefully designed to ensure predictable behavior during power-up and power-down. During power-up and power-down sequences, the I/Os remain in tri-state until the power supply voltage is high enough to ensure reliable operation. In addition, leakage into I/O pins is controlled within specified limits. Please refer to the Hot Socketing Specifications in the DC and Switching Characteristics in this data sheet.

## SERDES and PCS (Physical Coding Sublayer)

LatticeECP3 devices feature up to 16 channels of embedded SERDES/PCS arranged in quads at the bottom of the devices supporting up to 3.2Gbps data rate. Figure 2-40 shows the position of the quad blocks for the LatticeECP3-150 devices. Table 2-14 shows the location of available SERDES Quads for all devices.

The LatticeECP3 SERDES/PCS supports a range of popular serial protocols, including:

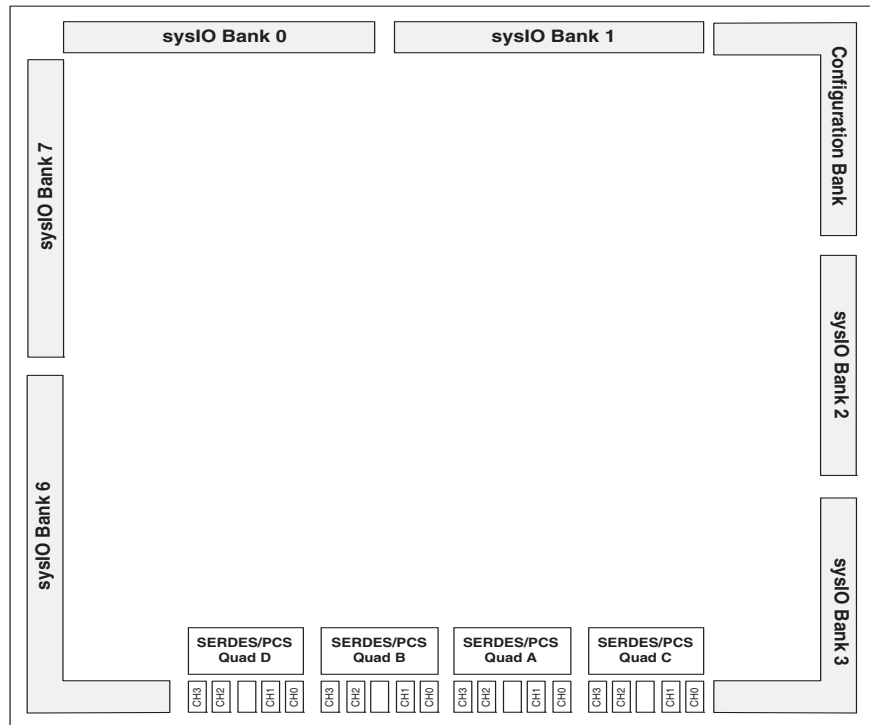
- PCI Express 1.1
- Ethernet (XAUI, GbE - 1000 Base CS/SX/LX and SGMII)
- Serial RapidIO
- SMPTE SDI (3G, HD, SD)
- CPRI
- SONET/SDH (STS-3, STS-12, STS-48)

Each quad contains four dedicated SERDES for high speed, full duplex serial data transfer. Each quad also has a PCS block that interfaces to the SERDES channels and contains protocol specific digital logic to support the standards listed above. The PCS block also contains interface logic to the FPGA fabric. All PCS logic for dedicated protocol support can also be bypassed to allow raw 8-bit or 10-bit interfaces to the FPGA fabric.

Even though the SERDES/PCS blocks are arranged in quads, multiple baud rates can be supported within a quad with the use of dedicated, per channel  $\div 1$ ,  $\div 2$  and  $\div 11$  rate dividers. Additionally, multiple quads can be arranged together to form larger data pipes.

For information on how to use the SERDES/PCS blocks to support specific protocols, as well on how to combine multiple protocols and baud rates within a device, please refer to TN1176, [LatticeECP3 SERDES/PCS Usage Guide](#).

**Figure 2-40. SERDES/PCS Quads (LatticeECP3-150)**



**Table 2-13. LatticeECP3 SERDES Standard Support**

Standard	Data Rate (Mbps)	Number of General/Link Width	Encoding Style
PCI Express 1.1	2500	x1, x2, x4	8b10b
Gigabit Ethernet	1250, 2500	x1	8b10b
SGMII	1250	x1	8b10b
XAUI	3125	x4	8b10b
Serial RapidIO Type I, Serial RapidIO Type II, Serial RapidIO Type III	1250, 2500, 3125	x1, x4	8b10b
CPRI-1, CPRI-2, CPRI-3, CPRI-4	614.4, 1228.8, 2457.6, 3072.0	x1	8b10b
SD-SDI (259M, 344M)	143 <sup>1</sup> , 177 <sup>1</sup> , 270, 360, 540	x1	NRZI/Scrambled
HD-SDI (292M)	1483.5, 1485	x1	NRZI/Scrambled
3G-SDI (424M)	2967, 2970	x1	NRZI/Scrambled
SONET-STS-3 <sup>2</sup>	155.52	x1	N/A
SONET-STS-12 <sup>2</sup>	622.08	x1	N/A
SONET-STS-48 <sup>2</sup>	2488	x1	N/A

1. For slower rates, the SERDES are bypassed and CML signals are directly connected to the FPGA routing.

2. The SONET protocol is supported in 8-bit SERDES mode. See TN1176 [Lattice ECP3 SERDES/PCS Usage Guide](#) for more information.

**Table 2-14. Available SERDES Quads per LatticeECP3 Devices**

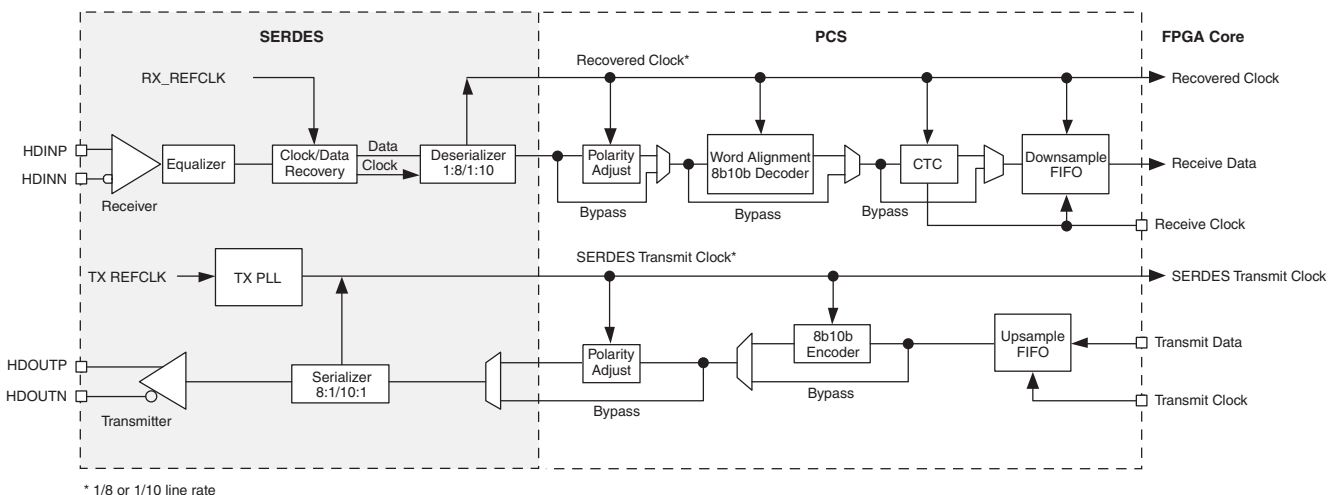
Package	ECP3-17	ECP3-35	ECP3-70	ECP3-95	ECP3-150
256 ftBGA	1	1	—	—	—
328 csBGA	2 channels	—	—	—	—
484 fpBGA	1	1	1	1	
672 fpBGA	—	1	2	2	2
1156 fpBGA	—	—	3	3	4

## SERDES Block

A SERDES receiver channel may receive the serial differential data stream, equalize the signal, perform Clock and Data Recovery (CDR) and de-serialize the data stream before passing the 8- or 10-bit data to the PCS logic. The SERDES transmitter channel may receive the parallel 8- or 10-bit data, serialize the data and transmit the serial bit stream through the differential drivers. Figure 2-41 shows a single-channel SERDES/PCS block. Each SERDES channel provides a recovered clock and a SERDES transmit clock to the PCS block and to the FPGA core logic.

Each transmit channel, receiver channel, and SERDES PLL shares the same power supply (VCCA). The output and input buffers of each channel have their own independent power supplies (VCCOB and VCCIB).

**Figure 2-41. Simplified Channel Block Diagram for SERDES/PCS Block**



## PCS

As shown in Figure 2-41, the PCS receives the parallel digital data from the deserializer and selects the polarity, performs word alignment, decodes (8b/10b), provides Clock Tolerance Compensation and transfers the clock domain from the recovered clock to the FPGA clock via the Down Sample FIFO.

For the transmit channel, the PCS block receives the parallel data from the FPGA core, encodes it with 8b/10b, selects the polarity and passes the 8/10 bit data to the transmit SERDES channel.

The PCS also provides bypass modes that allow a direct 8-bit or 10-bit interface from the SERDES to the FPGA logic. The PCS interface to the FPGA can also be programmed to run at 1/2 speed for a 16-bit or 20-bit interface to the FPGA logic.

## SCI (SERDES Client Interface) Bus

The SERDES Client Interface (SCI) is an IP interface that allows the SERDES/PCS Quad block to be controlled by registers rather than the configuration memory cells. It is a simple register configuration interface that allows SERDES/PCS configuration without power cycling the device.

The Diamond and ispLEVER design tools support all modes of the PCS. Most modes are dedicated to applications associated with a specific industry standard data protocol. Other more general purpose modes allow users to define their own operation. With these tools, the user can define the mode for each quad in a design.

Popular standards such as 10Gb Ethernet, x4 PCI Express and 4x Serial RapidIO can be implemented using IP (available through Lattice), a single quad (Four SERDES channels and PCS) and some additional logic from the core.

The LatticeECP3 family also supports a wide range of primary and secondary protocols. Within the same quad, the LatticeECP3 family can support mixed protocols with semi-independent clocking as long as the required clock frequencies are integer x1, x2, or x11 multiples of each other. Table 2-15 lists the allowable combination of primary and secondary protocol combinations.

## Flexible Quad SERDES Architecture

The LatticeECP3 family SERDES architecture is a quad-based architecture. For most SERDES settings and standards, the whole quad (consisting of four SERDES) is treated as a unit. This helps in silicon area savings, better utilization and overall lower cost.

However, for some specific standards, the LatticeECP3 quad architecture provides flexibility; more than one standard can be supported within the same quad.

Table 2-15 shows the standards can be mixed and matched within the same quad. In general, the SERDES standards whose nominal data rates are either the same or a defined subset of each other, can be supported within the same quad. In Table 2-15, the Primary Protocol column refers to the standard that determines the reference clock and PLL settings. The Secondary Protocol column shows the other standard that can be supported within the same quad.

Furthermore, Table 2-15 also implies that more than two standards in the same quad can be supported, as long as they conform to the data rate and reference clock requirements. For example, a quad may contain PCI Express 1.1, SGMII, Serial RapidIO Type I and Serial RapidIO Type II, all in the same quad.

**Table 2-15. LatticeECP3 Primary and Secondary Protocol Support**

Primary Protocol	Secondary Protocol
PCI Express 1.1	SGMII
PCI Express 1.1	Gigabit Ethernet
PCI Express 1.1	Serial RapidIO Type I
PCI Express 1.1	Serial RapidIO Type II
Serial RapidIO Type I	SGMII
Serial RapidIO Type I	Gigabit Ethernet
Serial RapidIO Type II	SGMII
Serial RapidIO Type II	Gigabit Ethernet
Serial RapidIO Type II	Serial RapidIO Type I
CPRI-3	CPRI-2 and CPRI-1
3G-SDI	HD-SDI and SD-SDI

There are some restrictions to be aware of when using spread spectrum. When a quad shares a PCI Express x1 channel with a non-PCI Express channel, ensure that the reference clock for the quad is compatible with all protocols within the quad. For example, a PCI Express spread spectrum reference clock is not compatible with most Gigabit Ethernet applications because of tight CTC ppm requirements.

While the LatticeECP3 architecture will allow the mixing of a PCI Express channel and a Gigabit Ethernet, Serial RapidIO or SGMII channel within the same quad, using a PCI Express spread spectrum clocking as the transmit

reference clock will cause a violation of the Gigabit Ethernet, Serial RapidIO and SGMII transmit jitter specifications.

For further information on SERDES, please see TN1176, [LatticeECP3 SERDES/PCS Usage Guide](#).

## IEEE 1149.1-Compliant Boundary Scan Testability

All LatticeECP3 devices have boundary scan cells that are accessed through an IEEE 1149.1 compliant Test Access Port (TAP). This allows functional testing of the circuit board on which the device is mounted through a serial scan path that can access all critical logic nodes. Internal registers are linked internally, allowing test data to be shifted in and loaded directly onto test nodes, or test data to be captured and shifted out for verification. The test access port consists of dedicated I/Os: TDI, TDO, TCK and TMS. The test access port has its own supply voltage  $V_{CCJ}$  and can operate with LVCMOS3.3, 2.5, 1.8, 1.5 and 1.2 standards.

For more information, please see TN1169, [LatticeECP3 sysCONFIG Usage Guide](#).

## Device Configuration

All LatticeECP3 devices contain two ports that can be used for device configuration. The Test Access Port (TAP), which supports bit-wide configuration, and the sysCONFIG port, support dual-byte, byte and serial configuration. The TAP supports both the IEEE Standard 1149.1 Boundary Scan specification and the IEEE Standard 1532 In-System Configuration specification. The sysCONFIG port includes seven I/Os used as dedicated pins with the remaining pins used as dual-use pins. See TN1169, [LatticeECP3 sysCONFIG Usage Guide](#) for more information about using the dual-use pins as general purpose I/Os.

There are various ways to configure a LatticeECP3 device:

1. JTAG
2. Standard Serial Peripheral Interface (SPI and SPI<sub>m</sub> modes) - interface to boot PROM memory
3. System microprocessor to drive a x8 CPU port (PCM mode)
4. System microprocessor to drive a serial slave SPI port (SSPI mode)
5. Generic byte wide flash with a MachXO™ device, providing control and addressing

On power-up, the FPGA SRAM is ready to be configured using the selected sysCONFIG port. Once a configuration port is selected, it will remain active throughout that configuration cycle. The IEEE 1149.1 port can be activated any time after power-up by sending the appropriate command through the TAP port.

LatticeECP3 devices also support the Slave SPI Interface. In this mode, the FPGA behaves like a SPI Flash device (slave mode) with the SPI port of the FPGA to perform read-write operations.

## Enhanced Configuration Options

LatticeECP3 devices have enhanced configuration features such as: decryption support, TransFR™ I/O and dual-boot image support.

### 1. TransFR (Transparent Field Reconfiguration)

TransFR I/O (TFR) is a unique Lattice technology that allows users to update their logic in the field without interrupting system operation using a single ispVM command. TransFR I/O allows I/O states to be frozen during device configuration. This allows the device to be field updated with a minimum of system disruption and downtime. See TN1087, [Minimizing System Interruption During Configuration Using TransFR Technology](#) for details.

### 2. Dual-Boot Image Support

Dual-boot images are supported for applications requiring reliable remote updates of configuration data for the



system FPGA. After the system is running with a basic configuration, a new boot image can be downloaded remotely and stored in a separate location in the configuration storage device. Any time after the update the LatticeECP3 can be re-booted from this new configuration file. If there is a problem, such as corrupt data during download or incorrect version number with this new boot image, the LatticeECP3 device can revert back to the original backup golden configuration and try again. This all can be done without power cycling the system. For more information, please see TN1169, [LatticeECP3 sysCONFIG Usage Guide](#).

### Soft Error Detect (SED) Support

LatticeECP3 devices have dedicated logic to perform Cycle Redundancy Code (CRC) checks. During configuration, the configuration data bitstream can be checked with the CRC logic block. In addition, the LatticeECP3 device can also be programmed to utilize a Soft Error Detect (SED) mode that checks for soft errors in configuration SRAM. The SED operation can be run in the background during user mode. If a soft error occurs, during user mode (normal operation) the device can be programmed to generate an error signal.

For further information on SED support, please see TN1184, [LatticeECP3 Soft Error Detection \(SED\) Usage Guide](#).

### External Resistor

LatticeECP3 devices require a single external, 10K ohm  $\pm 1\%$  value between the XRES pin and ground. Device configuration will not be completed if this resistor is missing. There is no boundary scan register on the external resistor pad.

### On-Chip Oscillator

Every LatticeECP3 device has an internal CMOS oscillator which is used to derive a Master Clock (MCCLK) for configuration. The oscillator and the MCCLK run continuously and are available to user logic after configuration is completed. The software default value of the MCCLK is nominally 2.5MHz. Table 2-16 lists all the available MCCLK frequencies. When a different Master Clock is selected during the design process, the following sequence takes place:

1. Device powers up with a nominal Master Clock frequency of 3.1MHz.
2. During configuration, users select a different master clock frequency.
3. The Master Clock frequency changes to the selected frequency once the clock configuration bits are received.
4. If the user does not select a master clock frequency, then the configuration bitstream defaults to the MCCLK frequency of 2.5MHz.

This internal 130 MHz  $\pm 15\%$  CMOS oscillator is available to the user by routing it as an input clock to the clock tree. For further information on the use of this oscillator for configuration or user mode, please see TN1169, [LatticeECP3 sysCONFIG Usage Guide](#).

**Table 2-16. Selectable Master Clock (MCCLK) Frequencies During Configuration (Nominal)**

MCCLK (MHz)	MCCLK (MHz)
	10
2.5 <sup>1</sup>	13
4.3	15 <sup>2</sup>
5.4	20
6.9	26
8.1	33 <sup>3</sup>
9.2	

1. Software default MCCLK frequency. Hardware default is 3.1MHz.
2. Maximum MCCLK with encryption enabled.
3. Maximum MCCLK without encryption.



## Density Shifting

The LatticeECP3 family is designed to ensure that different density devices in the same family and in the same package have the same pinout. Furthermore, the architecture ensures a high success rate when performing design migration from lower density devices to higher density devices. In many cases, it is also possible to shift a lower utilization design targeted for a high-density device to a lower density device. However, the exact details of the final resource utilization will impact the likelihood of success in each case. An example is that some user I/Os may become No Connects in smaller devices in the same package. Refer to the [LatticeECP3 Pin Migration Tables](#) and Diamond software for specific restrictions and limitations.

### Absolute Maximum Ratings<sup>1, 2, 3</sup>

Supply Voltage $V_{CC}$ . . . . .	-0.5 to 1.32V
Supply Voltage $V_{CCAUX}$ . . . . .	-0.5 to 3.75V
Supply Voltage $V_{CCJ}$ . . . . .	-0.5 to 3.75V
Output Supply Voltage $V_{CCIO}$ . . . . .	-0.5 to 3.75V
Input or I/O Tristate Voltage Applied <sup>4</sup> . . . . .	-0.5 to 3.75V
Storage Temperature (Ambient) . . . . .	-65 to 150°C
Junction Temperature ( $T_J$ ) . . . . .	+125°C

1. Stress above those listed under the "Absolute Maximum Ratings" may cause permanent damage to the device. Functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied.
2. Compliance with the Lattice [Thermal Management](#) document is required.
3. All voltages referenced to GND.
4. Overshoot and undershoot of -2V to ( $V_{IHMAX} + 2$ ) volts is permitted for a duration of <20ns.

### Recommended Operating Conditions<sup>1</sup>

Symbol	Parameter	Min.	Max.	Units
$V_{CC}^2$	Core Supply Voltage	1.14	1.26	V
$V_{CCAUX}^{2,4}$	Auxiliary Supply Voltage, Terminating Resistor Switching Power Supply (SERDES)	3.135	3.465	V
$V_{CCPLL}$	PLL Supply Voltage	3.135	3.465	V
$V_{CCIO}^{2,3}$	I/O Driver Supply Voltage	1.14	3.465	V
$V_{CCJ}^2$	Supply Voltage for IEEE 1149.1 Test Access Port	1.14	3.465	V
$V_{REF1}$ and $V_{REF2}$	Input Reference Voltage	0.5	1.7	V
$V_{TT}^5$	Termination Voltage	0.5	1.3125	V
$t_{JCOM}$	Junction Temperature, Commercial Operation	0	85	°C
$t_{JIND}$	Junction Temperature, Industrial Operation	-40	100	°C
<b>SERDES External Power Supply<sup>6</sup></b>				
$V_{CCIB}$	Input Buffer Power Supply (1.2V)	1.14	1.26	V
	Input Buffer Power Supply (1.5V)	1.425	1.575	V
$V_{CCOB}$	Output Buffer Power Supply (1.2V)	1.14	1.26	V
	Output Buffer Power Supply (1.5V)	1.425	1.575	V
$V_{CCA}$	Transmit, Receive, PLL and Reference Clock Buffer Power Supply	1.14	1.26	V

1. For correct operation, all supplies except  $V_{REF}$  and  $V_{TT}$  must be held in their valid operation range. This is true independent of feature usage.
2. If  $V_{CCIO}$  or  $V_{CCJ}$  is set to 1.2V, they must be connected to the same power supply as  $V_{CC}$ . If  $V_{CCIO}$  or  $V_{CCJ}$  is set to 3.3V, they must be connected to the same power supply as  $V_{CCAUX}$ .
3. See recommended voltages by I/O standard in subsequent table.
4.  $V_{CCAUX}$  ramp rate must not exceed 30mV/ $\mu$ s during power-up when transitioning between 0V and 3.3V.
5. If not used,  $V_{TT}$  should be left floating.
6. See TN1176, [LatticeECP3 SERDES/PCS Usage Guide](#) for information on board considerations for SERDES power supplies.

## Hot Socketing Specifications<sup>1, 2, 3</sup>

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
IDK_HS <sup>4</sup>	Input or I/O Leakage Current	$0 \leq V_{IN} \leq V_{IH} \text{ (Max.)}$	—	—	+/-1	mA
IDK <sup>5</sup>	Input or I/O Leakage Current	$0 \leq V_{IN} < V_{CCIO}$	—	—	+/-1	mA
		$V_{CCIO} \leq V_{IN} \leq V_{CCIO} + 0.5V$	—	18	—	mA

1.  $V_{CC}$ ,  $V_{CCAUX}$  and  $V_{CCIO}$  should rise/fall monotonically.
2.  $I_{DK}$  is additive to  $I_{PU}$ ,  $I_{PW}$  or  $I_{BH}$ .
3. LVCMOS and LVTTTL only.
4. Applicable to general purpose I/O pins located on the top and bottom sides of the device.
5. Applicable to general purpose I/O pins located on the left and right sides of the device.

## Hot Socketing Requirements<sup>1, 2</sup>

Description	Min.	Typ.	Max.	Units
Input current per SERDES I/O pin when device is powered down and inputs driven.	—	—	8	mA

1. Assumes the device is powered down, all supplies grounded, both P and N inputs driven by CML driver with maximum allowed  $V_{CCOB}$  (1.575V), 8b10b data, internal AC coupling.
2. Each P and N input must have less than the specified maximum input current. For a 16-channel device, the total input current would be  $8mA * 16 \text{ channels} * 2 \text{ input pins per channel} = 256mA$

## ESD Performance

Please refer to the [LatticeECP3 Product Family Qualification Summary](#) for complete qualification data, including ESD performance.

## DC Electrical Characteristics

### Over Recommended Operating Conditions

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
$I_{IL}, I_{IH}^{1,4}$	Input or I/O Low Leakage	$0 \leq V_{IN} \leq (V_{CCIO} - 0.2V)$	—	—	10	$\mu A$
$I_{IH}^{1,3}$	Input or I/O High Leakage	$(V_{CCIO} - 0.2V) < V_{IN} \leq 3.6V$	—	—	150	$\mu A$
$I_{PU}$	I/O Active Pull-up Current	$0 \leq V_{IN} \leq 0.7 V_{CCIO}$	-30	—	-210	$\mu A$
$I_{PD}$	I/O Active Pull-down Current	$V_{IL} (MAX) \leq V_{IN} \leq V_{CCIO}$	30	—	210	$\mu A$
$I_{BHLS}$	Bus Hold Low Sustaining Current	$V_{IN} = V_{IL} (MAX)$	30	—	—	$\mu A$
$I_{BHHS}$	Bus Hold High Sustaining Current	$V_{IN} = 0.7 V_{CCIO}$	-30	—	—	$\mu A$
$I_{BHLO}$	Bus Hold Low Overdrive Current	$0 \leq V_{IN} \leq V_{CCIO}$	—	—	210	$\mu A$
$I_{BHHO}$	Bus Hold High Overdrive Current	$0 \leq V_{IN} \leq V_{CCIO}$	—	—	-210	$\mu A$
$V_{BHT}$	Bus Hold Trip Points	$0 \leq V_{IN} \leq V_{IH} (MAX)$	$V_{IL} (MAX)$	—	$V_{IH} (MIN)$	V
C1	I/O Capacitance <sup>2</sup>	$V_{CCIO} = 3.3V, 2.5V, 1.8V, 1.5V, 1.2V,$ $V_{CC} = 1.2V, V_{IO} = 0 \text{ to } V_{IH} (MAX)$	—	5	8	pf
C2	Dedicated Input Capacitance <sup>2</sup>	$V_{CCIO} = 3.3V, 2.5V, 1.8V, 1.5V, 1.2V,$ $V_{CC} = 1.2V, V_{IO} = 0 \text{ to } V_{IH} (MAX)$	—	5	7	pf

1. Input or I/O leakage current is measured with the pin configured as an input or as an I/O with the output driver tri-stated. It is not measured with the output driver active. Bus maintenance circuits are disabled.

2.  $T_A$  25°C,  $f = 1.0MHz$ .

3. Applicable to general purpose I/Os in top and bottom banks.

4. When used as  $V_{REF}$  maximum leakage = 25 $\mu A$ .

**LatticeECP3 Supply Current (Standby)<sup>1, 2, 3, 4, 5, 6</sup>**

Over Recommended Operating Conditions

Symbol	Parameter	Device	Typical		Units
			-6L, -7L, -8L	-6, -7, -8, -9	
I <sub>CC</sub>	Core Power Supply Current	ECP-17EA	29.8	49.4	mA
		ECP3-35EA	53.7	89.4	mA
		ECP3-70EA	137.3	230.7	mA
		ECP3-95EA	137.3	230.7	mA
		ECP3-150EA	219.5	370.9	mA
I <sub>CCAUX</sub>	Auxiliary Power Supply Current	ECP-17EA	18.3	19.4	mA
		ECP3-35EA	19.6	23.1	mA
		ECP3-70EA	26.5	32.4	mA
		ECP3-95EA	26.5	32.4	mA
		ECP3-150EA	37.0	45.7	mA
I <sub>CCPLL</sub>	PLL Power Supply Current (Per PLL)	ECP-17EA	0.0	0.0	mA
		ECP3-35EA	0.1	0.1	mA
		ECP3-70EA	0.1	0.1	mA
		ECP3-95EA	0.1	0.1	mA
		ECP3-150EA	0.1	0.1	mA
I <sub>CCIO</sub>	Bank Power Supply Current (Per Bank)	ECP-17EA	1.3	1.4	mA
		ECP3-35EA	1.3	1.4	mA
		ECP3-70EA	1.4	1.5	mA
		ECP3-95EA	1.4	1.5	mA
		ECP3-150EA	1.4	1.5	mA
I <sub>CCJ</sub>	JTAG Power Supply Current	All Devices	2.5	2.5	mA
I <sub>CCA</sub>	Transmit, Receive, PLL and Reference Clock Buffer Power Supply	ECP-17EA	6.1	6.1	mA
		ECP3-35EA	6.1	6.1	mA
		ECP3-70EA	18.3	18.3	mA
		ECP3-95EA	18.3	18.3	mA
		ECP3-150EA	24.4	24.4	mA

1. For further information on supply current, please see the list of technical documentation at the end of this data sheet.
2. Assumes all outputs are tristated, all inputs are configured as LVCMOS and held at the V<sub>CCIO</sub> or GND.
3. Frequency 0 MHz.
4. Pattern represents a “blank” configuration data file.
5. T<sub>J</sub> = 85°C, power supplies at nominal voltage.
6. To determine the LatticeECP3 peak start-up current data, use the Power Calculator tool.

**SERDES Power Supply Requirements<sup>1, 2, 3</sup>**
**Over Recommended Operating Conditions**

Symbol	Description	Typ.	Max.	Units
<b>Standby (Power Down)</b>				
I <sub>CCA-SB</sub>	V <sub>CCA</sub> current (per channel)	3	5	mA
I <sub>CCIB-SB</sub>	Input buffer current (per channel)	—	—	mA
I <sub>CCOB-SB</sub>	Output buffer current (per channel)	—	—	mA
<b>Operating (Data Rate = 3.2 Gbps)</b>				
I <sub>CCA-OP</sub>	V <sub>CCA</sub> current (per channel)	68	77	mA
I <sub>CCIB-OP</sub>	Input buffer current (per channel)	5	7	mA
I <sub>CCOB-OP</sub>	Output buffer current (per channel)	19	25	mA
<b>Operating (Data Rate = 2.5 Gbps)</b>				
I <sub>CCA-OP</sub>	V <sub>CCA</sub> current (per channel)	66	76	mA
I <sub>CCIB-OP</sub>	Input buffer current (per channel)	4	5	mA
I <sub>CCOB-OP</sub>	Output buffer current (per channel)	15	18	mA
<b>Operating (Data Rate = 1.25 Gbps)</b>				
I <sub>CCA-OP</sub>	V <sub>CCA</sub> current (per channel)	62	72	mA
I <sub>CCIB-OP</sub>	Input buffer current (per channel)	4	5	mA
I <sub>CCOB-OP</sub>	Output buffer current (per channel)	15	18	mA
<b>Operating (Data Rate = 250 Mbps)</b>				
I <sub>CCA-OP</sub>	V <sub>CCA</sub> current (per channel)	55	65	mA
I <sub>CCIB-OP</sub>	Input buffer current (per channel)	4	5	mA
I <sub>CCOB-OP</sub>	Output buffer current (per channel)	14	17	mA
<b>Operating (Data Rate = 150 Mbps)</b>				
I <sub>CCA-OP</sub>	V <sub>CCA</sub> current (per channel)	55	65	mA
I <sub>CCIB-OP</sub>	Input buffer current (per channel)	4	5	mA
I <sub>CCOB-OP</sub>	Output buffer current (per channel)	14	17	mA

1. Equalization enabled, pre-emphasis disabled.

2. One quarter of the total quad power (includes contribution from common circuits, all channels in the quad operating, pre-emphasis disabled, equalization enabled).

3. Pre-emphasis adds 20mA to I<sub>CCA-OP</sub> data.

**sysI/O Recommended Operating Conditions**

Standard	V <sub>CCIO</sub>			V <sub>REF</sub> (V)		
	Min.	Typ.	Max.	Min.	Typ.	Max.
LVCMOS33 <sup>2</sup>	3.135	3.3	3.465	—	—	—
LVCMOS33D	3.135	3.3	3.465	—	—	—
LVCMOS25 <sup>2</sup>	2.375	2.5	2.625	—	—	—
LVCMOS18	1.71	1.8	1.89	—	—	—
LVCMOS15	1.425	1.5	1.575	—	—	—
LVCMOS12 <sup>2</sup>	1.14	1.2	1.26	—	—	—
LVTTTL33 <sup>2</sup>	3.135	3.3	3.465	—	—	—
PCI33	3.135	3.3	3.465	—	—	—
SSTL15 <sup>3</sup>	1.43	1.5	1.57	0.68	0.75	0.9
SSTL18_I, II <sup>2</sup>	1.71	1.8	1.89	0.833	0.9	0.969
SSTL25_I, II <sup>2</sup>	2.375	2.5	2.625	1.15	1.25	1.35
SSTL33_I, II <sup>2</sup>	3.135	3.3	3.465	1.3	1.5	1.7
HSTL15_I <sup>2</sup>	1.425	1.5	1.575	0.68	0.75	0.9
HSTL18_I, II <sup>2</sup>	1.71	1.8	1.89	0.816	0.9	1.08
LVDS25 <sup>2</sup>	2.375	2.5	2.625	—	—	—
LVDS25E	2.375	2.5	2.625	—	—	—
MLVDS <sup>1</sup>	2.375	2.5	2.625	—	—	—
LVPECL33 <sup>1, 2</sup>	3.135	3.3	3.465	—	—	—
Mini LVDS	2.375	2.5	2.625	—	—	—
BLVDS25 <sup>1, 2</sup>	2.375	2.5	2.625	—	—	—
RSDS <sup>2</sup>	2.375	2.5	2.625	—	—	—
RSDSE <sup>1, 2</sup>	2.375	2.5	2.625	—	—	—
TRLVDS	3.14	3.3	3.47	—	—	—
PPLVDS	3.14/2.25	3.3/2.5	3.47/2.75	—	—	—
SSTL15D <sup>3</sup>	1.43	1.5	1.57	—	—	—
SSTL18D_I <sup>2, 3</sup> , II <sup>2, 3</sup>	1.71	1.8	1.89	—	—	—
SSTL25D_I <sup>2</sup> , II <sup>2</sup>	2.375	2.5	2.625	—	—	—
SSTL33D_I <sup>2</sup> , II <sup>2</sup>	3.135	3.3	3.465	—	—	—
HSTL15D_I <sup>2</sup>	1.425	1.5	1.575	—	—	—
HSTL18D_I <sup>2</sup> , II <sup>2</sup>	1.71	1.8	1.89	—	—	—

1. Inputs on chip. Outputs are implemented with the addition of external resistors.
2. For input voltage compatibility, see TN1177, [LatticeECP3 sysIO Usage Guide](#).
3. VREF is required when using Differential SSTL to interface to DDR memory.

**sysI/O Single-Ended DC Electrical Characteristics**

Input/Output Standard	$V_{IL}$		$V_{IH}$		$V_{OL}$ Max. (V)	$V_{OH}$ Min. (V)	$I_{OL}^1$ (mA)	$I_{OH}^1$ (mA)
	Min. (V)	Max. (V)	Min. (V)	Max. (V)				
LVCMOS33	-0.3	0.8	2.0	3.6	0.4	$V_{CCIO} - 0.4$	20, 16, 12, 8, 4	-20, -16, -12, -8, -4
					0.2	$V_{CCIO} - 0.2$	0.1	-0.1
LVCMOS25	-0.3	0.7	1.7	3.6	0.4	$V_{CCIO} - 0.4$	20, 16, 12, 8, 4	-20, -16, -12, -8, -4
					0.2	$V_{CCIO} - 0.2$	0.1	-0.1
LVCMOS18	-0.3	$0.35 V_{CCIO}$	$0.65 V_{CCIO}$	3.6	0.4	$V_{CCIO} - 0.4$	16, 12, 8, 4	-16, -12, -8, -4
					0.2	$V_{CCIO} - 0.2$	0.1	-0.1
LVCMOS15	-0.3	$0.35 V_{CCIO}$	$0.65 V_{CCIO}$	3.6	0.4	$V_{CCIO} - 0.4$	8, 4	-8, -4
					0.2	$V_{CCIO} - 0.2$	0.1	-0.1
LVCMOS12	-0.3	$0.35 V_{CC}$	$0.65 V_{CC}$	3.6	0.4	$V_{CCIO} - 0.4$	6, 2	-6, -2
					0.2	$V_{CCIO} - 0.2$	0.1	-0.1
LVTTTL33	-0.3	0.8	2.0	3.6	0.4	$V_{CCIO} - 0.4$	20, 16, 12, 8, 4	-20, -16, -12, -8, -4
					0.2	$V_{CCIO} - 0.2$	0.1	-0.1
PCI33	-0.3	$0.3 V_{CCIO}$	$0.5 V_{CCIO}$	3.6	$0.1 V_{CCIO}$	$0.9 V_{CCIO}$	1.5	-0.5
SSTL18_I	-0.3	$V_{REF} - 0.125$	$V_{REF} + 0.125$	3.6	0.4	$V_{CCIO} - 0.4$	6.7	-6.7
SSTL18_II (DDR2 Memory)	-0.3	$V_{REF} - 0.125$	$V_{REF} + 0.125$	3.6	0.28	$V_{CCIO} - 0.28$	8	-8
							11	-11
SSTL2_I	-0.3	$V_{REF} - 0.18$	$V_{REF} + 0.18$	3.6	0.54	$V_{CCIO} - 0.62$	7.6	-7.6
							12	-12
SSTL2_II (DDR Memory)	-0.3	$V_{REF} - 0.18$	$V_{REF} + 0.18$	3.6	0.35	$V_{CCIO} - 0.43$	15.2	-15.2
							20	-20
SSTL3_I	-0.3	$V_{REF} - 0.2$	$V_{REF} + 0.2$	3.6	0.7	$V_{CCIO} - 1.1$	8	-8
SSTL3_II	-0.3	$V_{REF} - 0.2$	$V_{REF} + 0.2$	3.6	0.5	$V_{CCIO} - 0.9$	16	-16
SSTL15 (DDR3 Memory)	-0.3	$V_{REF} - 0.1$	$V_{REF} + 0.1$	3.6	0.3	$V_{CCIO} - 0.3$	7.5	-7.5
						$V_{CCIO} * 0.8$	9	-9
HSTL15_I	-0.3	$V_{REF} - 0.1$	$V_{REF} + 0.1$	3.6	0.4	$V_{CCIO} - 0.4$	4	-4
							8	-8
HSTL18_I	-0.3	$V_{REF} - 0.1$	$V_{REF} + 0.1$	3.6	0.4	$V_{CCIO} - 0.4$	8	-8
							12	-12
HSTL18_II	-0.3	$V_{REF} - 0.1$	$V_{REF} + 0.1$	3.6	0.4	$V_{CCIO} - 0.4$	16	-16

1. For electromigration, the average DC current drawn by I/O pads between two consecutive  $V_{CCIO}$  or GND pad connections, or between the last  $V_{CCIO}$  or GND in an I/O bank and the end of an I/O bank, as shown in the Logic Signal Connections table (also shown as I/O grouping) shall not exceed  $n * 8\text{mA}$ , where  $n$  is the number of I/O pads between the two consecutive bank  $V_{CCIO}$  or GND connections or between the last  $V_{CCIO}$  and GND in a bank and the end of a bank.



## sysI/O Differential Electrical Characteristics

### LVDS25

#### Over Recommended Operating Conditions

Parameter	Description	Test Conditions	Min.	Typ.	Max.	Units
$V_{INP}^1, V_{INM}^1$	Input Voltage		0	—	2.4	V
$V_{CM}^1$	Input Common Mode Voltage	Half the Sum of the Two Inputs	0.05	—	2.35	V
$V_{THD}$	Differential Input Threshold	Difference Between the Two Inputs	+/-100	—	—	mV
$I_{IN}$	Input Current	Power On or Power Off	—	—	+/-10	$\mu$ A
$V_{OH}$	Output High Voltage for $V_{OP}$ or $V_{OM}$	$R_T = 100$ Ohm	—	1.38	1.60	V
$V_{OL}$	Output Low Voltage for $V_{OP}$ or $V_{OM}$	$R_T = 100$ Ohm	0.9V	1.03	—	V
$V_{OD}$	Output Voltage Differential	$(V_{OP} - V_{OM}), R_T = 100$ Ohm	250	350	450	mV
$\Delta V_{OD}$	Change in $V_{OD}$ Between High and Low		—	—	50	mV
$V_{OS}$	Output Voltage Offset	$(V_{OP} + V_{OM})/2, R_T = 100$ Ohm	1.125	1.20	1.375	V
$\Delta V_{OS}$	Change in $V_{OS}$ Between H and L		—	—	50	mV
$I_{SAB}$	Output Short Circuit Current	$V_{OD} = 0V$ Driver Outputs Shorted to Each Other	—	—	12	mA

1, On the left and right sides of the device, this specification is valid only for  $V_{CCIO} = 2.5V$  or  $3.3V$ .

### Differential HSTL and SSTL

Differential HSTL and SSTL outputs are implemented as a pair of complementary single-ended outputs. All allowable single-ended output classes (class I and class II) are supported in this mode.

### LVDS25E

The top and bottom sides of LatticeECP3 devices support LVDS outputs via emulated complementary LVCMOS outputs in conjunction with a parallel resistor across the driver outputs. The scheme shown in Figure 3-1 is one possible solution for point-to-point signals.

Figure 3-1. LVDS25E Output Termination Example

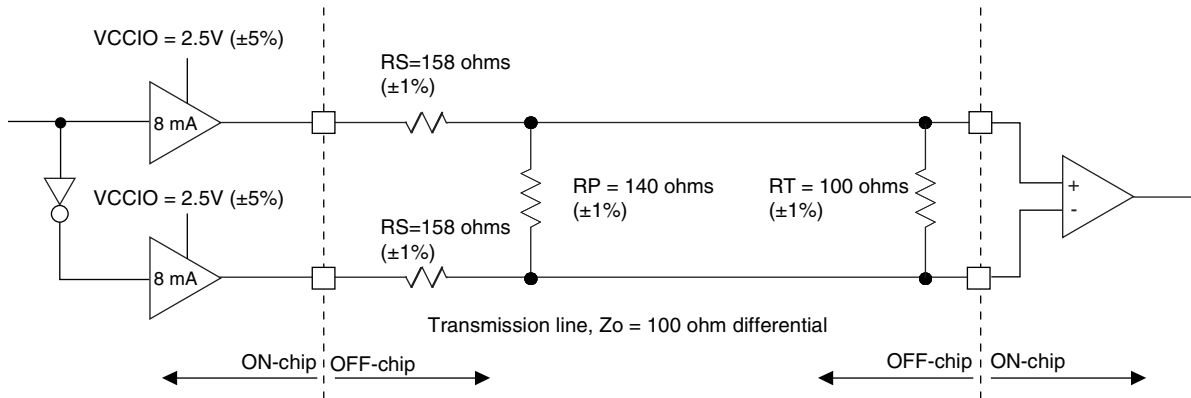


Table 3-1. LVDS25E DC Conditions

Parameter	Description	Typical	Units
V <sub>CCIO</sub>	Output Driver Supply (+/-5%)	2.50	V
Z <sub>OUT</sub>	Driver Impedance	20	Ω
R <sub>S</sub>	Driver Series Resistor (+/-1%)	158	Ω
R <sub>P</sub>	Driver Parallel Resistor (+/-1%)	140	Ω
R <sub>T</sub>	Receiver Termination (+/-1%)	100	Ω
V <sub>OH</sub>	Output High Voltage	1.43	V
V <sub>OL</sub>	Output Low Voltage	1.07	V
V <sub>OD</sub>	Output Differential Voltage	0.35	V
V <sub>CM</sub>	Output Common Mode Voltage	1.25	V
Z <sub>BACK</sub>	Back Impedance	100.5	Ω
I <sub>DC</sub>	DC Output Current	6.03	mA

### LVCMOS33D

All I/O banks support emulated differential I/O using the LVCMOS33D I/O type. This option, along with the external resistor network, provides the system designer the flexibility to place differential outputs on an I/O bank with 3.3V V<sub>CCIO</sub>. The default drive current for LVCMOS33D output is 12mA with the option to change the device strength to 4mA, 8mA, 16mA or 20mA. Follow the LVCMOS33 specifications for the DC characteristics of the LVCMOS33D.

### BLVDS25

The LatticeECP3 devices support the BLVDS standard. This standard is emulated using complementary LVCMOS outputs in conjunction with a parallel external resistor across the driver outputs. BLVDS is intended for use when multi-drop and bi-directional multi-point differential signaling is required. The scheme shown in Figure 3-2 is one possible solution for bi-directional multi-point differential signals.

Figure 3-2. BLVDS25 Multi-point Output Example

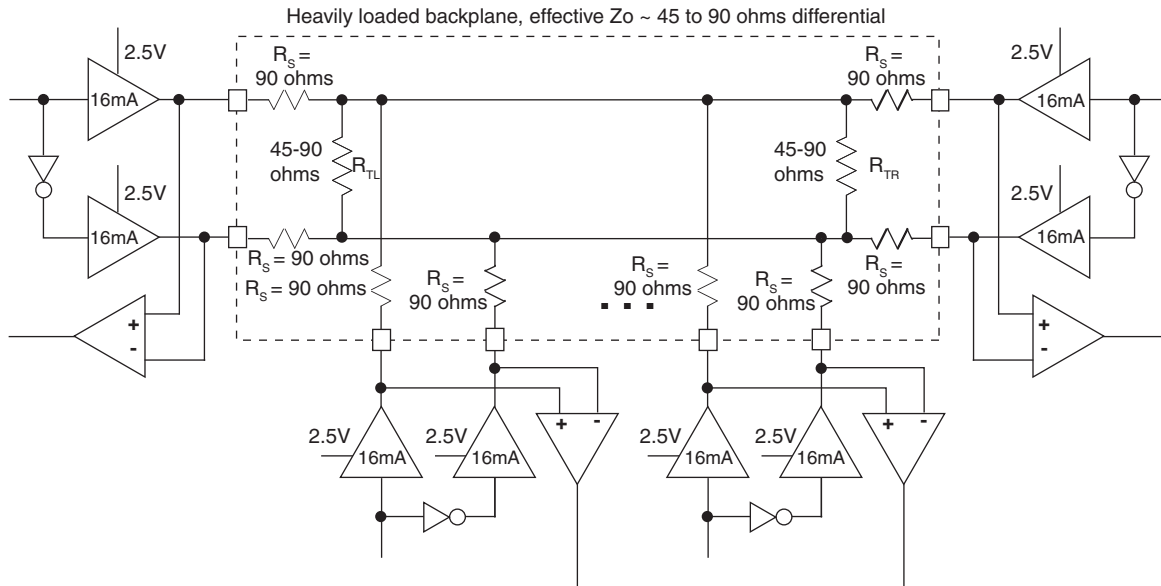


Table 3-2. BLVDS25 DC Conditions<sup>1</sup>

#### Over Recommended Operating Conditions

Parameter	Description	Typical		Units
		Zo = 45Ω	Zo = 90Ω	
V <sub>CCIO</sub>	Output Driver Supply (+/- 5%)	2.50	2.50	V
Z <sub>OUT</sub>	Driver Impedance	10.00	10.00	Ω
R <sub>S</sub>	Driver Series Resistor (+/- 1%)	90.00	90.00	Ω
R <sub>TL</sub>	Driver Parallel Resistor (+/- 1%)	45.00	90.00	Ω
R <sub>TR</sub>	Receiver Termination (+/- 1%)	45.00	90.00	Ω
V <sub>OH</sub>	Output High Voltage	1.38	1.48	V
V <sub>OL</sub>	Output Low Voltage	1.12	1.02	V
V <sub>OD</sub>	Output Differential Voltage	0.25	0.46	V
V <sub>CM</sub>	Output Common Mode Voltage	1.25	1.25	V
I <sub>DC</sub>	DC Output Current	11.24	10.20	mA

1. For input buffer, see LVDS table.

### LVPECL33

The LatticeECP3 devices support the differential LVPECL standard. This standard is emulated using complementary LVCMOS outputs in conjunction with a parallel resistor across the driver outputs. The LVPECL input standard is supported by the LVDS differential input buffer. The scheme shown in Figure 3-3 is one possible solution for point-to-point signals.

Figure 3-3. Differential LVPECL33

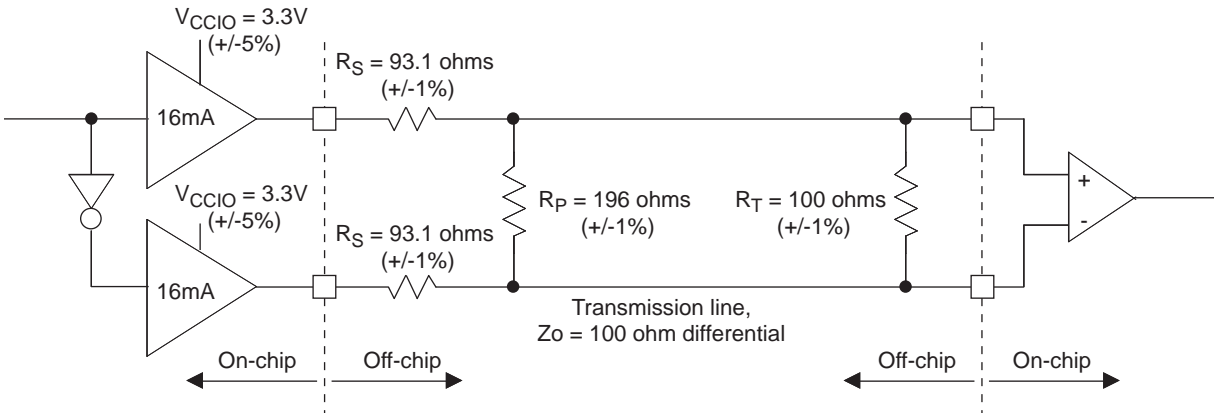


Table 3-3. LVPECL33 DC Conditions<sup>1</sup>

#### Over Recommended Operating Conditions

Parameter	Description	Typical	Units
$V_{CCIO}$	Output Driver Supply ( $\pm 5\%$ )	3.30	V
$Z_{OUT}$	Driver Impedance	10	$\Omega$
$R_S$	Driver Series Resistor ( $\pm 1\%$ )	93	$\Omega$
$R_P$	Driver Parallel Resistor ( $\pm 1\%$ )	196	$\Omega$
$R_T$	Receiver Termination ( $\pm 1\%$ )	100	$\Omega$
$V_{OH}$	Output High Voltage	2.05	V
$V_{OL}$	Output Low Voltage	1.25	V
$V_{OD}$	Output Differential Voltage	0.80	V
$V_{CM}$	Output Common Mode Voltage	1.65	V
$Z_{BACK}$	Back Impedance	100.5	$\Omega$
$I_{DC}$	DC Output Current	12.11	mA

1. For input buffer, see LVDS table.

### RSDS25E

The LatticeECP3 devices support differential RSDS and RSDSE standards. This standard is emulated using complementary LVCMOS outputs in conjunction with a parallel resistor across the driver outputs. The RSDS input standard is supported by the LVDS differential input buffer. The scheme shown in Figure 3-4 is one possible solution for RSDS standard implementation. Resistor values in Figure 3-4 are industry standard values for 1% resistors.

Figure 3-4. RSDS25E (Reduced Swing Differential Signaling)

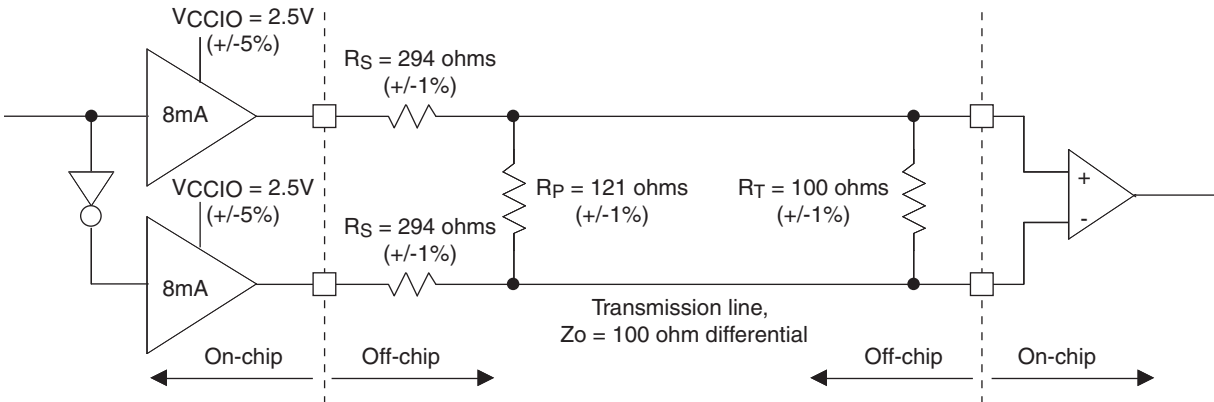


Table 3-4. RSDS25E DC Conditions<sup>1</sup>

#### Over Recommended Operating Conditions

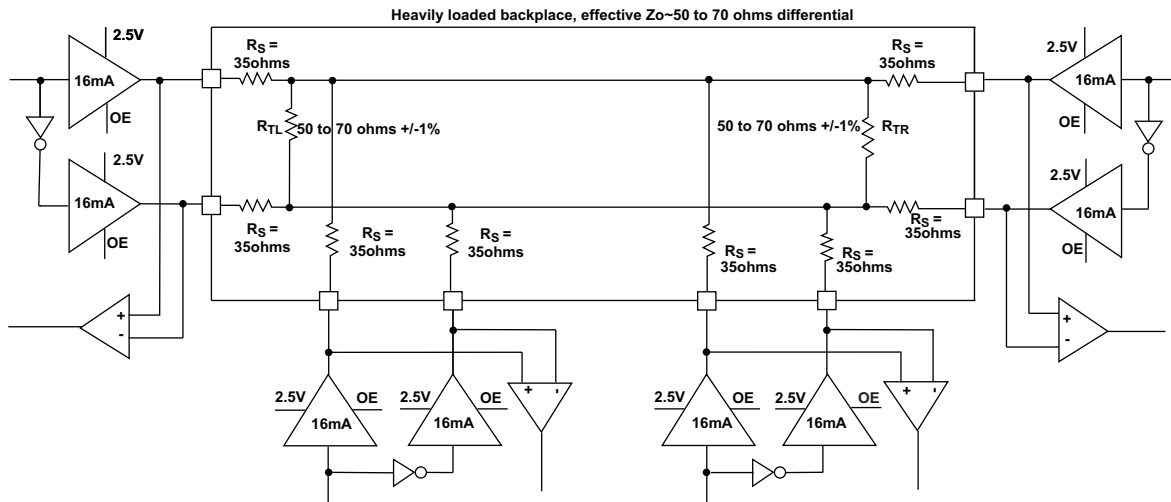
Parameter	Description	Typical	Units
V <sub>CCIO</sub>	Output Driver Supply (+/-5%)	2.50	V
Z <sub>OUT</sub>	Driver Impedance	20	Ω
R <sub>S</sub>	Driver Series Resistor (+/-1%)	294	Ω
R <sub>P</sub>	Driver Parallel Resistor (+/-1%)	121	Ω
R <sub>T</sub>	Receiver Termination (+/-1%)	100	Ω
V <sub>OH</sub>	Output High Voltage	1.35	V
V <sub>OL</sub>	Output Low Voltage	1.15	V
V <sub>OD</sub>	Output Differential Voltage	0.20	V
V <sub>CM</sub>	Output Common Mode Voltage	1.25	V
Z <sub>BACK</sub>	Back Impedance	101.5	Ω
I <sub>DC</sub>	DC Output Current	3.66	mA

1. For input buffer, see LVDS table.

**MLVDS25**

The LatticeECP3 devices support the differential MLVDS standard. This standard is emulated using complementary LVCMOS outputs in conjunction with a parallel resistor across the driver outputs. The MLVDS input standard is supported by the LVDS differential input buffer. The scheme shown in Figure 3-5 is one possible solution for MLVDS standard implementation. Resistor values in Figure 3-5 are industry standard values for 1% resistors.

**Figure 3-5. MLVDS25 (Multipoint Low Voltage Differential Signaling)**



**Table 3-5. MLVDS25 DC Conditions<sup>1</sup>**

Parameter	Description	Typical		Units
		Zo=50Ω	Zo=70Ω	
V <sub>CCIO</sub>	Output Driver Supply (+/-5%)	2.50	2.50	V
Z <sub>OUT</sub>	Driver Impedance	10.00	10.00	Ω
R <sub>S</sub>	Driver Series Resistor (+/-1%)	35.00	35.00	Ω
R <sub>TL</sub>	Driver Parallel Resistor (+/-1%)	50.00	70.00	Ω
R <sub>TR</sub>	Receiver Termination (+/-1%)	50.00	70.00	Ω
V <sub>OH</sub>	Output High Voltage	1.52	1.60	V
V <sub>OL</sub>	Output Low Voltage	0.98	0.90	V
V <sub>OD</sub>	Output Differential Voltage	0.54	0.70	V
V <sub>CM</sub>	Output Common Mode Voltage	1.25	1.25	V
I <sub>DC</sub>	DC Output Current	21.74	20.00	mA

1. For input buffer, see LVDS table.

## Typical Building Block Function Performance

### Pin-to-Pin Performance (LVCMOS25 12mA Drive)<sup>1, 2, 3</sup>

Function	-9 Timing	Units
<b>Basic Functions</b>		
16-bit Decoder	3.8	ns
32-bit Decoder	3.8	ns
64-bit Decoder	5.4	ns
4:1 MUX	4.0	ns
8:1 MUX	4.2	ns
16:1 MUX	4.5	ns
32:1 MUX	4.7	ns

1. These functions were generated using the ispLEVER design tool. Exact performance may vary with device and tool version. The tool uses internal parameters that have been characterized but are not tested on every device.
2. Commercial timing numbers are shown. Industrial numbers are typically slower and can be extracted from the Diamond or ispLEVER software.

### Register-to-Register Performance<sup>1, 2, 3</sup>

Function	-9 Timing	Units
<b>Basic Functions</b>		
16-bit Decoder	500	MHz
32-bit Decoder	500	MHz
64-bit Decoder	500	MHz
4:1 MUX	500	MHz
8:1 MUX	500	MHz
16:1 MUX	500	MHz
32:1 MUX	487	MHz
8-bit adder	500	MHz
16-bit adder	500	MHz
64-bit adder	336	MHz
16-bit counter	500	MHz
32-bit counter	500	MHz
64-bit counter	350	MHz
64-bit accumulator	326	MHz
<b>Embedded Memory Functions</b>		
512x36 Single Port RAM, EBR Output Registers	358	MHz
1024x18 True-Dual Port RAM (Write Through or Normal, EBR Output Registers)	358	MHz
1024x18 True-Dual Port RAM (Read-Before-Write, EBR Output Registers)	136	MHz
1024x18 True-Dual Port RAM (Write Through or Normal, PLC Output Registers)	260	MHz
<b>Distributed Memory Functions</b>		
16x4 Pseudo-Dual Port RAM (One PFU)	500	MHz
32x4 Pseudo-Dual Port RAM	500	MHz
64x8 Pseudo-Dual Port RAM	442	MHz
<b>DSP Function</b>		
18x18 Multiplier (All Registers)	420	MHz
9x9 Multiplier (All Registers)	420	MHz
36x36 Multiply (All Registers)	281	MHz

---

**Register-to-Register Performance<sup>1, 2, 3</sup>**

Function	-9 Timing	Units
18x18 Multiply/Accumulate (Input & Output Registers)	229	MHz
18x18 Multiply-Add/Sub (All Registers)	420	MHz

1. These timing numbers were generated using ispLEVER tool. Exact performance may vary with device and tool version. The tool uses internal parameters that have been characterized but are not tested on every device.
2. Commercial timing numbers are shown. Industrial numbers are typically slower and can be extracted from the Diamond or ispLEVER software.
3. LatticeECP3-17EA is not available for the -9 speed grade. The data may vary slightly from -9. The fastest speed grade for LatticeECP3-17EA is -8.

**Derating Timing Tables**

Logic timing provided in the following sections of this data sheet and the Diamond and ispLEVER design tools are worst case numbers in the operating range. Actual delays at nominal temperature and voltage for best case process, can be much better than the values given in the tables. The Diamond and ispLEVER design tools can provide logic timing numbers at a particular temperature and voltage.



## LatticeECP3 External Switching Characteristics <sup>1, 2</sup>

Over Recommended Commercial Operating Conditions

Parameter	Description	Device	-9		-8		-7		-6		Units
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
<b>Clocks</b>											
<b>Primary Clock<sup>6</sup></b>											
f <sub>MAX_PRI</sub>	Frequency for Primary Clock Tree	ECP3-150EA	—	500	—	500	—	420	—	375	MHz
t <sub>W_PRI</sub>	Clock Pulse Width for Primary Clock	ECP3-150EA	0.8	—	0.8	—	0.9	—	1.0	—	ns
t <sub>SKEW_PRI</sub>	Primary Clock Skew Within a Device	ECP3-150EA	—	300	—	300	—	330	—	360	ps
t <sub>SKEW_PRI_B</sub>	Primary Clock Skew Within a Bank	ECP3-150EA	—	250	—	250	—	280	—	300	ps
f <sub>MAX_PRI</sub>	Frequency for Primary Clock Tree	ECP3-70EA/95EA	—	500	—	500	—	420	—	375	MHz
t <sub>W_PRI</sub>	Pulse Width for Primary Clock	ECP3-70EA/95EA	0.8	—	0.8	—	0.9	—	1.0	—	ns
t <sub>SKEW_PRI</sub>	Primary Clock Skew Within a Device	ECP3-70EA/95EA	—	360	—	360	—	370	—	380	ps
t <sub>SKEW_PRI_B</sub>	Primary Clock Skew Within a Bank	ECP3-70EA/95EA	—	310	—	310	—	320	—	330	ps
f <sub>MAX_PRI</sub>	Frequency for Primary Clock Tree	ECP3-35EA	—	500	—	500	—	420	—	375	MHz
t <sub>W_PRI</sub>	Pulse Width for Primary Clock	ECP3-35EA	0.8	—	0.8	—	0.9	—	1.0	—	ns
t <sub>SKEW_PRI</sub>	Primary Clock Skew Within a Device	ECP3-35EA	—	300	—	300	—	330	—	360	ps
t <sub>SKEW_PRI_B</sub>	Primary Clock Skew Within a Bank	ECP3-35EA	—	250	—	250	—	280	—	300	ps
f <sub>MAX_PRI</sub>	Frequency for Primary Clock Tree	ECP3-17EA	—	—	—	500	—	420	—	375	MHz
t <sub>W_PRI</sub>	Pulse Width for Primary Clock	ECP3-17EA	—	—	0.8	—	0.9	—	1.0	—	ns
t <sub>SKEW_PRI</sub>	Primary Clock Skew Within a Device	ECP3-17EA	—	—	—	310	—	340	—	370	ps
t <sub>SKEW_PRI_B</sub>	Primary Clock Skew Within a Bank	ECP3-17EA	—	—	—	220	—	230	—	240	ps
<b>Edge Clock<sup>6</sup></b>											
f <sub>MAX_EDGE</sub>	Frequency for Edge Clock	ECP3-150EA	—	500	—	500	—	420	—	375	MHz
t <sub>W_EDGE</sub>	Clock Pulse Width for Edge Clock	ECP3-150EA	0.9	—	0.9	—	1.0	—	1.2	—	ns
t <sub>SKEW_EDGE_DQS</sub>	Edge Clock Skew Within an Edge of the Device	ECP3-150EA	—	200	—	200	—	210	—	220	ps
f <sub>MAX_EDGE</sub>	Frequency for Edge Clock	ECP3-70EA/95EA	—	500	—	500	—	420	—	375	MHz
t <sub>W_EDGE</sub>	Clock Pulse Width for Edge Clock	ECP3-70EA/95EA	0.9	—	0.9	—	1.0	—	1.2	—	ns
t <sub>SKEW_EDGE_DQS</sub>	Edge Clock Skew Within an Edge of the Device	ECP3-70EA/95EA	—	200	—	200	—	210	—	220	ps
f <sub>MAX_EDGE</sub>	Frequency for Edge Clock	ECP3-35EA	—	500	—	500	—	420	—	375	MHz
t <sub>W_EDGE</sub>	Clock Pulse Width for Edge Clock	ECP3-35EA	0.9	—	0.9	—	1.0	—	1.2	—	ns
t <sub>SKEW_EDGE_DQS</sub>	Edge Clock Skew Within an Edge of the Device	ECP3-35EA	—	200	—	200	—	210	—	220	ps
f <sub>MAX_EDGE</sub>	Frequency for Edge Clock	ECP3-17EA	—	—	—	500	—	420	—	375	MHz
t <sub>W_EDGE</sub>	Clock Pulse Width for Edge Clock	ECP3-17EA	—	—	0.9	—	1.0	—	1.2	—	ns
t <sub>SKEW_EDGE_DQS</sub>	Edge Clock Skew Within an Edge of the Device	ECP3-17EA	—	—	—	200	—	210	—	220	ps
<b>Generic SDR</b>											
<b>General I/O Pin Parameters Using Dedicated Clock Input Primary Clock Without PLL<sup>2</sup></b>											
t <sub>CO</sub>	Clock to Output - PIO Output Register	ECP3-150EA	—	3.9	—	3.9	—	4.3	—	4.7	ns
t <sub>SU</sub>	Clock to Data Setup - PIO Input Register	ECP3-150EA	0.0	—	0.0	—	0.0	—	0.0	—	ns
t <sub>H</sub>	Clock to Data Hold - PIO Input Register	ECP3-150EA	1.5	—	1.5	—	1.7	—	2.0	—	ns
t <sub>SU_DEL</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP3-150EA	1.3	—	1.3	—	1.5	—	1.7	—	ns

## LatticeECP3 External Switching Characteristics (Continued)<sup>1, 2</sup>

Over Recommended Commercial Operating Conditions

Parameter	Description	Device	-9		-8		-7		-6		Units
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
t <sub>H_DEL</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP3-150EA	0.0	—	0.0	—	0.0	—	0.0	—	ns
f <sub>MAX_IO</sub>	Clock Frequency of I/O and PFU Register	ECP3-150EA	—	500	—	500	—	420	—	375	MHz
t <sub>CO</sub>	Clock to Output - PIO Output Register	ECP3-70EA/95EA	—	3.8	—	3.8	—	4.2	—	4.6	ns
t <sub>SU</sub>	Clock to Data Setup - PIO Input Register	ECP3-70EA/95EA	0.0	—	0.0	—	0.0	—	0.0	—	ns
t <sub>H</sub>	Clock to Data Hold - PIO Input Register	ECP3-70EA/95EA	1.4	—	1.4	—	1.6	—	1.8	—	ns
t <sub>SU_DEL</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP3-70EA/95EA	1.3	—	1.3	—	1.5	—	1.7	—	ns
t <sub>H_DEL</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP3-70EA/95EA	0.0	—	0.0	—	0.0	—	0.0	—	ns
f <sub>MAX_IO</sub>	Clock Frequency of I/O and PFU Register	ECP3-70EA/95EA	—	500	—	500	—	420	—	375	MHz
t <sub>CO</sub>	Clock to Output - PIO Output Register	ECP3-35EA	—	3.7	—	3.7	—	4.1	—	4.5	ns
t <sub>SU</sub>	Clock to Data Setup - PIO Input Register	ECP3-35EA	0.0	—	0.0	—	0.0	—	0.0	—	ns
t <sub>H</sub>	Clock to Data Hold - PIO Input Register	ECP3-35EA	1.2	—	1.2	—	1.4	—	1.6	—	ns
t <sub>SU_DEL</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP3-35EA	1.3	—	1.3	—	1.4	—	1.5	—	ns
t <sub>H_DEL</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP3-35EA	0.0	—	0.0	—	0.0	—	0.0	—	ns
f <sub>MAX_IO</sub>	Clock Frequency of I/O and PFU Register	ECP3-35EA	—	500	—	500	—	420	—	375	MHz
t <sub>CO</sub>	Clock to Output - PIO Output Register	ECP3-17EA	—	—	—	3.5	—	3.9	—	4.3	ns
t <sub>SU</sub>	Clock to Data Setup - PIO Input Register	ECP3-17EA	—	—	0.0	—	0.0	—	0.0	—	ns
t <sub>H</sub>	Clock to Data Hold - PIO Input Register	ECP3-17EA	—	—	1.3	—	1.5	—	1.6	—	ns
t <sub>SU_DEL</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP3-17EA	—	—	1.3	—	1.4	—	1.5	—	ns
t <sub>H_DEL</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP3-17EA	—	—	0.0	—	0.0	—	0.0	—	ns
f <sub>MAX_IO</sub>	Clock Frequency of I/O and PFU Register	ECP3-17EA	—	—	—	500	—	420	—	375	MHz
<b>General I/O Pin Parameters Using Dedicated Clock Input Primary Clock with PLL with Clock Injection Removal Setting<sup>2</sup></b>											
t <sub>COPLL</sub>	Clock to Output - PIO Output Register	ECP3-150EA	—	3.3	—	3.3	—	3.6	—	39	ns
t <sub>SUPLL</sub>	Clock to Data Setup - PIO Input Register	ECP3-150EA	0.7	—	0.7	—	0.8	—	0.9	—	ns
t <sub>HPLL</sub>	Clock to Data Hold - PIO Input Register	ECP3-150EA	0.8	—	0.8	—	0.9	—	1.0	—	ns
t <sub>SU_DELPLL</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP3-150EA	1.6	—	1.6	—	1.8	—	2.0	—	ns
t <sub>H_DELPLL</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP3-150EA	—	0.0	—	0.0	—	0.0	—	0.0	ns
t <sub>COPLL</sub>	Clock to Output - PIO Output Register	ECP3-70EA/95EA	—	3.3	—	3.3	—	3.5	—	3.8	ns
t <sub>SUPLL</sub>	Clock to Data Setup - PIO Input Register	ECP3-70EA/95EA	0.7	—	0.7	—	0.8	—	0.9	—	ns

## LatticeECP3 External Switching Characteristics (Continued)<sup>1, 2</sup>

Over Recommended Commercial Operating Conditions

Parameter	Description	Device	-9		-8		-7		-6		Units
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
t <sub>HPLL</sub>	Clock to Data Hold - PIO Input Register	ECP3-70EA/95EA	0.7	—	0.7	—	0.7	—	0.8	—	ns
t <sub>SU_DELPLL</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP3-70EA/95EA	1.6	—	1.6	—	1.8	—	2.0	—	ns
t <sub>H_DELPLL</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP3-70EA/95EA	0.0	—	0.0	—	0.0	—	0.0	—	ns
t <sub>COPLL</sub>	Clock to Output - PIO Output Register	ECP3-35EA	—	3.2	—	3.2	—	3.4	—	3.6	ns
t <sub>SUPLL</sub>	Clock to Data Setup - PIO Input Register	ECP3-35EA	0.6	—	0.6	—	0.7	—	0.8	—	ns
t <sub>HPLL</sub>	Clock to Data Hold - PIO Input Register	ECP3-35EA	0.3	—	0.3	—	0.3	—	0.4	—	ns
t <sub>SU_DELPLL</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP3-35EA	1.6	—	1.6	—	1.7	—	1.8	—	ns
t <sub>H_DELPLL</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP3-35EA	0.0	—	0.0	—	0.0	—	0.0	—	ns
t <sub>COPLL</sub>	Clock to Output - PIO Output Register	ECP3-17EA	—	—	—	3.0	—	3.3	—	3.5	ns
t <sub>SUPLL</sub>	Clock to Data Setup - PIO Input Register	ECP3-17EA	—	—	0.6	—	0.7	—	0.8	—	ns
t <sub>HPLL</sub>	Clock to Data Hold - PIO Input Register	ECP3-17EA	—	—	0.3	—	0.3	—	0.4	—	ns
t <sub>SU_DELPLL</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP3-17EA	—	—	1.6	—	1.7	—	1.8	—	ns
t <sub>H_DELPLL</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP3-17EA	—	—	0.0	—	0.0	—	0.0	—	ns
<b>Generic DDR<sup>12</sup></b>											
<b>Generic DDRX1 Inputs with Clock and Data (&gt;10 Bits Wide) Centered at Pin (GDDR1_RX.SCLK.Centered) Using PCLK Pin for Clock Input</b>											
t <sub>SUGDDR</sub>	Data Setup Before CLK	All ECP3EA Devices	480	—	480	—	480	—	480	—	ps
t <sub>HOGDDR</sub>	Data Hold After CLK	All ECP3EA Devices	480	—	480	—	480	—	480	—	ps
f <sub>MAX_GDDR</sub>	DDR1 Clock Frequency	All ECP3EA Devices	—	250	—	250	—	250	—	250	MHz
<b>Generic DDRX1 Inputs with Clock and Data (&gt;10 Bits Wide) Aligned at Pin (GDDR1_RX.SCLK.PLL.Aligned) Using PLLCLKIN Pin for Clock Input</b>											
<b>Data Left, Right, and Top Sides and Clock Left and Right Sides</b>											
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	All ECP3EA Devices	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLKDDR</sub>	Data Hold After CLK	All ECP3EA Devices	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR1 Clock Frequency	All ECP3EA Devices	—	250	—	250	—	250	—	250	MHz
<b>Generic DDRX1 Inputs with Clock and Data (&gt;10 Bits Wide) Aligned at Pin (GDDR1_RX.SCLK.Aligned) Using DLL - CLKIN Pin for Clock Input</b>											
<b>Data Left, Right and Top Sides and Clock Left and Right Sides</b>											
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	All ECP3EA Devices	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLKDDR</sub>	Data Hold After CLK	All ECP3EA Devices	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR1 Clock Frequency	All ECP3EA Devices	—	250	—	250	—	250	—	250	MHz
<b>Generic DDRX1 Inputs with Clock and Data (&lt;10 Bits Wide) Centered at Pin (GDDR1_RX.DQS.Centered) Using DQS Pin for Clock Input</b>											
t <sub>SUGDDR</sub>	Data Setup After CLK	All ECP3EA Devices	535	—	535	—	535	—	535	—	ps
t <sub>HOGDDR</sub>	Data Hold After CLK	All ECP3EA Devices	535	—	535	—	535	—	535	—	ps
f <sub>MAX_GDDR</sub>	DDR1 Clock Frequency	All ECP3EA Devices	—	250	—	250	—	250	—	250	MHz
<b>Generic DDRX1 Inputs with Clock and Data (&lt;10bits wide) Aligned at Pin (GDDR1_RX.DQS.Aligned) Using DQS Pin for Clock Input</b>											
<b>Data and Clock Left and Right Sides</b>											
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	All ECP3EA Devices	—	0.225	—	0.225	—	0.225	—	0.225	UI

## LatticeECP3 External Switching Characteristics (Continued)<sup>1, 2</sup>

Over Recommended Commercial Operating Conditions

Parameter	Description	Device	-9		-8		-7		-6		Units
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
t <sub>DVECLGDDR</sub>	Data Hold After CLK	All ECP3EA Devices	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR1 Clock Frequency	All ECP3EA Devices	—	250	—	250	—	250	—	250	MHz
<b>Generic DDR2 Inputs with Clock and Data (&gt;10 Bits Wide) Centered at Pin (GDDR2_RX.ECLK.Centered) Using PCLK Pin for Clock Input</b>											
<b>Left and Right Sides</b>											
t <sub>SUGDDR</sub>	Data Setup Before CLK	ECP3-150EA	321	—	321	—	403	—	471	—	ps
t <sub>HOGDDR</sub>	Data Hold After CLK	ECP3-150EA	321	—	321	—	403	—	471	—	ps
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-150EA	—	405	—	405	—	325	—	280	MHz
t <sub>SUGDDR</sub>	Data Setup Before CLK	ECP3-70EA/95EA	321	—	321	—	403	—	535	—	ps
t <sub>HOGDDR</sub>	Data Hold After CLK	ECP3-70EA/95EA	321	—	321	—	403	—	535	—	ps
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-70EA/95EA	—	405	—	405	—	325	—	250	MHz
t <sub>SUGDDR</sub>	Data Setup Before CLK	ECP3-35EA	335	—	335	—	425	—	535	—	ps
t <sub>HOGDDR</sub>	Data Hold After CLK	ECP3-35EA	335	—	335	—	425	—	535	—	ps
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-35EA	—	405	—	405	—	325	—	250	MHz
t <sub>SUGDDR</sub>	Data Setup Before CLK	ECP3-17EA	—	—	335	—	425	—	535	—	ps
t <sub>HOGDDR</sub>	Data Hold After CLK	ECP3-17EA	—	—	335	—	425	—	535	—	ps
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-17EA	—	—	—	405	—	325	—	250	MHz
<b>Generic DDR2 Inputs with Clock and Data (&gt;10 Bits Wide) Aligned at Pin (GDDR2_RX.ECLK.Aligned)</b>											
<b>Left and Right Side Using DLLCLKIN Pin for Clock Input</b>											
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	ECP3-150EA	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLGDDR</sub>	Data Hold After CLK	ECP3-150EA	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-150EA	—	460	—	460	—	385	—	345	MHz
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	ECP3-70EA/95EA	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLGDDR</sub>	Data Hold After CLK	ECP3-70EA/95EA	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-70EA/95EA	—	460	—	460	—	385	—	311	MHz
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	ECP3-35EA	—	0.210	—	0.210	—	0.210	—	0.210	UI
t <sub>DVECLGDDR</sub>	Data Hold After CLK	ECP3-35EA	0.790	—	0.790	—	0.790	—	0.790	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-35EA	—	460	—	460	—	385	—	311	MHz
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	ECP3-17EA	—	—	—	0.210	—	0.210	—	0.210	UI
t <sub>DVECLGDDR</sub>	Data Hold After CLK	ECP3-17EA	—	—	0.790	—	0.790	—	0.790	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-17EA	—	—	—	460	—	385	—	311	MHz
<b>Top Side Using PCLK Pin for Clock Input</b>											
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	ECP3-150EA	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLGDDR</sub>	Data Hold After CLK	ECP3-150EA	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-150EA	—	235	—	235	—	170	—	130	MHz
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	ECP3-70EA/95EA	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLGDDR</sub>	Data Hold After CLK	ECP3-70EA/95EA	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-70EA/95EA	—	235	—	235	—	170	—	130	MHz
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	ECP3-35EA	—	0.210	—	0.210	—	0.210	—	0.210	UI
t <sub>DVECLGDDR</sub>	Data Hold After CLK	ECP3-35EA	0.790	—	0.790	—	0.790	—	0.790	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-35EA	—	235	—	235	—	170	—	130	MHz
t <sub>DVACLKDDR</sub>	Data Setup Before CLK	ECP3-17EA	—	—	—	0.210	—	0.210	—	0.210	UI
t <sub>DVECLGDDR</sub>	Data Hold After CLK	ECP3-17EA	—	—	0.790	—	0.790	—	0.790	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-17EA	—	—	—	235	—	170	—	130	MHz

## LatticeECP3 External Switching Characteristics (Continued)<sup>1, 2</sup>

Over Recommended Commercial Operating Conditions

Parameter	Description	Device	-9		-8		-7		-6		Units
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
<b>Generic DDRX2 Inputs with Clock and Data (&gt;10bits wide) are Aligned at Pin (GDDR2_RX.ECLK.Aligned) (No CLKDIV)</b>											
<b>Left and Right Sides Using DLLCLKPIN for Clock Input</b>											
t <sub>DVACLK</sub> GDDR	Data Setup Before CLK	ECP3-150EA	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLK</sub> GDDR	Data Hold After CLK	ECP3-150EA	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-150EA	—	460	—	460	—	385	—	345	MHz
t <sub>DVACLK</sub> GDDR	Data Setup Before CLK	ECP3-70EA/95EA	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLK</sub> GDDR	Data Hold After CLK	ECP3-70EA/95EA	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-70EA/95EA	—	460	—	460	—	385	—	311	MHz
t <sub>DVACLK</sub> GDDR	Data Setup Before CLK	ECP3-35EA	—	0.210	—	0.210	—	0.210	—	0.210	UI
t <sub>DVECLK</sub> GDDR	Data Hold After CLK	ECP3-35EA	0.790	—	0.790	—	0.790	—	0.790	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-35EA	—	460	—	460	—	385	—	311	MHz
t <sub>DVACLK</sub> GDDR	Data Setup Before CLK (Left and Right Sides)	ECP3-17EA	—	—	—	0.210	—	0.210	—	0.210	UI
t <sub>DVECLK</sub> GDDR	Data Hold After CLK	ECP3-17EA	—	—	0.790	—	0.790	—	0.790	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-17EA	—	—	—	460	—	385	—	311	MHz
<b>Top Side Using PCLK Pin for Clock Input</b>											
t <sub>DVACLK</sub> GDDR	Data Setup Before CLK	ECP3-150EA	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLK</sub> GDDR	Data Hold After CLK	ECP3-150EA	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-150EA	—	235	—	235	—	170	—	130	MHz
t <sub>DVACLK</sub> GDDR	Data Setup Before CLK	ECP3-70EA/95EA	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLK</sub> GDDR	Data Hold After CLK	ECP3-70EA/95EA	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-70EA/95EA	—	235	—	235	—	170	—	130	MHz
t <sub>DVACLK</sub> GDDR	Data Setup Before CLK	ECP3-35EA	—	0.210	—	0.210	—	0.210	—	0.210	UI
t <sub>DVECLK</sub> GDDR	Data Hold After CLK	ECP3-35EA	0.790	—	0.790	—	0.790	—	0.790	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-35EA	—	235	—	235	—	170	—	130	MHz
t <sub>DVACLK</sub> GDDR	Data Setup Before CLK	ECP3-17EA	—	—	—	0.210	—	0.210	—	0.210	UI
t <sub>DVECLK</sub> GDDR	Data Hold After CLK	ECP3-17EA	—	—	0.790	—	0.790	—	0.790	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	ECP3-17EA	—	—	—	235	—	170	—	130	MHz
<b>Generic DDRX2 Inputs with Clock and Data (&lt;10 Bits Wide) Centered at Pin (GDDR2_RX.DQS.Centered) Using DQS Pin for Clock Input</b>											
<b>Left and Right Sides</b>											
t <sub>SUGDDR</sub>	Data Setup Before CLK	All ECP3EA Devices	330	—	330	—	330	—	352	—	ps
t <sub>HOGDDR</sub>	Data Hold After CLK	All ECP3EA Devices	330	—	330	—	330	—	352	—	ps
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	All ECP3EA Devices	—	400	—	400	—	400	—	375	MHz
<b>Generic DDRX2 Inputs with Clock and Data (&lt;10 Bits Wide) Aligned at Pin (GDDR2_RX.DQS.Aligned) Using DQS Pin for Clock Input</b>											
<b>Left and Right Sides</b>											
t <sub>DVACLK</sub> GDDR	Data Setup Before CLK	All ECP3EA Devices	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVECLK</sub> GDDR	Data Hold After CLK	All ECP3EA Devices	0.775	—	0.775	—	0.775	—	0.775	—	UI
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	All ECP3EA Devices	—	400	—	400	—	400	—	375	MHz
<b>Generic DDRX1 Output with Clock and Data (&gt;10 Bits Wide) Centered at Pin (GDDR1_TX.SCLK.Centered)<sup>10</sup></b>											
t <sub>DVBGDDR</sub>	Data Valid Before CLK	ECP3-150EA	670	—	670	—	670	—	670	—	ps
t <sub>DVAGDDR</sub>	Data Valid After CLK	ECP3-150EA	670	—	670	—	670	—	670	—	ps
f <sub>MAX_GDDR</sub>	DDR1 Clock Frequency	ECP3-150EA	—	250	—	250	—	250	—	250	MHz
t <sub>DVBGDDR</sub>	Data Valid Before CLK	ECP3-70EA/95EA	666	—	666	—	665	—	664	—	ps
t <sub>DVAGDDR</sub>	Data Valid After CLK	ECP3-70EA/95EA	666	—	666	—	665	—	664	—	ps

## LatticeECP3 External Switching Characteristics (Continued)<sup>1, 2</sup>

Over Recommended Commercial Operating Conditions

Parameter	Description	Device	-9		-8		-7		-6		Units
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-70EA/95EA	—	250	—	250	—	250	—	250	MHz
t <sub>DVBGDDR</sub>	Data Valid Before CLK	ECP3-35EA	683	—	683	—	688	—	690	—	ps
t <sub>DVAGDDR</sub>	Data Valid After CLK	ECP3-35EA	683	—	683	—	688	—	690	—	ps
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-35EA	—	250	—	250	—	250	—	250	MHz
t <sub>DVBGDDR</sub>	Data Valid Before CLK	ECP3-17EA	—	—	683	—	688	—	690	—	ps
t <sub>DVAGDDR</sub>	Data Valid After CLK	ECP3-17EA	—	—	683	—	688	—	690	—	ps
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-17EA	—	—	—	250	—	250	—	250	MHz
<b>Generic DDRX1 Output with Clock and Data Aligned at Pin (GDDR1_TX.SCLK.Aligned)<sup>10</sup></b>											
t <sub>DIBGDDR</sub>	Data Invalid Before Clock	ECP3-150EA	—	335	—	335	—	338	—	341	ps
t <sub>DIAGDDR</sub>	Data Invalid After Clock	ECP3-150EA	—	335	—	335	—	338	—	341	ps
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-150EA	—	250	—	250	—	250	—	250	MHz
t <sub>DIBGDDR</sub>	Data Invalid Before Clock	ECP3-70EA/95EA	—	339	—	339	—	343	—	347	ps
t <sub>DIAGDDR</sub>	Data Invalid After Clock	ECP3-70EA/95EA	—	339	—	339	—	343	—	347	ps
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-70EA/95EA	—	250	—	250	—	250	—	250	MHz
t <sub>DIBGDDR</sub>	Data Invalid Before Clock	ECP3-35EA	—	322	—	322	—	320	—	321	ps
t <sub>DIAGDDR</sub>	Data Invalid After Clock	ECP3-35EA	—	322	—	322	—	320	—	321	ps
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-35EA	—	250	—	250	—	250	—	250	MHz
t <sub>DIBGDDR</sub>	Data Invalid Before Clock	ECP3-17EA	—	—	—	322	—	320	—	321	ps
t <sub>DIAGDDR</sub>	Data Invalid After Clock	ECP3-17EA	—	—	—	322	—	320	—	321	ps
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-17EA	—	—	—	250	—	250	—	250	MHz
<b>Generic DDRX1 Output with Clock and Data (&lt;10 Bits Wide) Centered at Pin (GDDR1_TX.DQS.Centered)<sup>10</sup></b>											
<b>Left and Right Sides</b>											
t <sub>DVBGDDR</sub>	Data Valid Before CLK	ECP3-150EA	670	—	670	—	670	—	670	—	ps
t <sub>DVAGDDR</sub>	Data Valid After CLK	ECP3-150EA	670	—	670	—	670	—	670	—	ps
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-150EA	—	250	—	250	—	250	—	250	MHz
t <sub>DVBGDDR</sub>	Data Valid Before CLK	ECP3-70EA/95EA	657	—	657	—	652	—	650	—	ps
t <sub>DVAGDDR</sub>	Data Valid After CLK	ECP3-70EA/95EA	657	—	657	—	652	—	650	—	ps
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-70EA/95EA	—	250	—	250	—	250	—	250	MHz
t <sub>DVBGDDR</sub>	Data Valid Before CLK	ECP3-35EA	670	—	670	—	675	—	676	—	ps
t <sub>DVAGDDR</sub>	Data Valid After CLK	ECP3-35EA	670	—	670	—	675	—	676	—	ps
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-35EA	—	250	—	250	—	250	—	250	MHz
t <sub>DVBGDDR</sub>	Data Valid Before CLK	ECP3-17EA	—	—	670	—	670	—	670	—	ps
t <sub>DVAGDDR</sub>	Data Valid After CLK	ECP3-17EA	—	—	670	—	670	—	670	—	ps
f <sub>MAX_GDDR</sub>	DDRX1 Clock Frequency	ECP3-17EA	—	—	—	250	—	250	—	250	MHz
<b>Generic DDRX2 Output with Clock and Data (&gt;10 Bits Wide) Aligned at Pin (GDDR2_TX.Aligned)</b>											
<b>Left and Right Sides</b>											
t <sub>DIBGDDR</sub>	Data Invalid Before Clock	All ECP3EA Devices	—	200	—	200	—	210	—	220	ps
t <sub>DIAGDDR</sub>	Data Invalid After Clock	All ECP3EA Devices	—	200	—	200	—	210	—	220	ps
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	All ECP3EA Devices	—	500	—	500	—	420	—	375	MHz
<b>Generic DDRX2 Output with Clock and Data (&gt;10 Bits Wide) Centered at Pin Using DQSDLL (GDDR2_TX.DQSDLL.Centered)<sup>11</sup></b>											
<b>Left and Right Sides</b>											
t <sub>DVBGDDR</sub>	Data Valid Before CLK	All ECP3EA Devices	400	—	400	—	400	—	431	—	ps
t <sub>DVAGDDR</sub>	Data Valid After CLK	All ECP3EA Devices	400	—	400	—	400	—	432	—	ps
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	All ECP3EA Devices	—	400	—	400	—	400	—	375	MHz



## LatticeECP3 External Switching Characteristics (Continued)<sup>1, 2</sup>

Over Recommended Commercial Operating Conditions

Parameter	Description	Device	-9		-8		-7		-6		Units
			Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
<b>Generic DDRX2 Output with Clock and Data (&gt;10 Bits Wide) Centered at Pin Using PLL (GDDR2_TX.PLL.Centered)<sup>10</sup></b>											
<b>Left and Right Sides</b>											
t <sub>DVBGDDR</sub>	Data Valid Before CLK	All ECP3EA Devices	285	—	285	—	370	—	431	—	ps
t <sub>DVAGDDR</sub>	Data Valid After CLK	All ECP3EA Devices	285	—	285	—	370	—	432	—	ps
f <sub>MAX_GDDR</sub>	DDR2 Clock Frequency	All ECP3EA Devices	—	500	—	500	—	420	—	375	MHz
<b>Memory Interface</b>											
<b>DDR/DDR2 I/O Pin Parameters (Input Data are Strobe Edge Aligned, Output Strobe Edge is Data Centered)<sup>4</sup></b>											
t <sub>DVADQ</sub>	Data Valid After DQS (DDR Read)	All ECP3 Devices	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVEDQ</sub>	Data Hold After DQS (DDR Read)	All ECP3 Devices	0.64	—	0.64	—	0.64	—	0.64	—	UI
t <sub>DQVBS</sub>	Data Valid Before DQS	All ECP3 Devices	0.25	—	0.25	—	0.25	—	0.25	—	UI
t <sub>DQVAS</sub>	Data Valid After DQS	All ECP3 Devices	0.25	—	0.25	—	0.25	—	0.25	—	UI
f <sub>MAX_DDR</sub>	DDR Clock Frequency	All ECP3 Devices	95	200	95	200	95	200	95	166	MHz
f <sub>MAX_DDR2</sub>	DDR2 clock frequency	All ECP3 Devices	125	266	125	266	125	200	125	166	MHz
<b>DDR3 (Using PLL for SCLK) I/O Pin Parameters</b>											
t <sub>DVADQ</sub>	Data Valid After DQS (DDR Read)	All ECP3 Devices	—	0.225	—	0.225	—	0.225	—	0.225	UI
t <sub>DVEDQ</sub>	Data Hold After DQS (DDR Read)	All ECP3 Devices	0.64	—	0.64	—	0.64	—	0.64	—	UI
t <sub>DQVBS</sub>	Data Valid Before DQS	All ECP3 Devices	0.25	—	0.25	—	0.25	—	0.25	—	UI
t <sub>DQVAS</sub>	Data Valid After DQS	All ECP3 Devices	0.25	—	0.25	—	0.25	—	0.25	—	UI
f <sub>MAX_DDR3</sub>	DDR3 clock frequency	All ECP3 Devices	300	400	300	400	266	333	266	300	MHz
<b>DDR3 Clock Timing</b>											
t <sub>CH</sub> (avg) <sup>9</sup>	Average High Pulse Width	All ECP3 Devices	0.47	0.53	0.47	0.53	0.47	0.53	0.47	0.53	UI
t <sub>CL</sub> (avg) <sup>9</sup>	Average Low Pulse Width	All ECP3 Devices	0.47	0.53	0.47	0.53	0.47	0.53	0.47	0.53	UI
t <sub>JIT</sub> (per, lck) <sup>9</sup>	Output Clock Period Jitter During DLL Locking Period	All ECP3 Devices	-90	90	-90	90	-90	90	-90	90	ps
t <sub>JIT</sub> (cc, lck) <sup>9</sup>	Output Cycle-to-Cycle Period Jitter During DLL Locking Period	All ECP3 Devices	—	180	—	180	—	180	—	180	ps

- Commercial timing numbers are shown. Industrial numbers are typically slower and can be extracted from the Diamond or ispLEVER software.
- General I/O timing numbers based on LVCMOS 2.5, 12mA, Fast Slew Rate, 0pf load.
- Generic DDR timing numbers based on LVDS I/O.
- DDR timing numbers based on SSTL25. DDR2 timing numbers based on SSTL18.
- DDR3 timing numbers based on SSTL15.
- Uses LVDS I/O standard.
- The current version of software does not support per bank skew numbers; this will be supported in a future release.
- Maximum clock frequencies are tested under best case conditions. System performance may vary upon the user environment.
- Using settings generated by IPexpress.
- These numbers are generated using best case PLL located in the center of the device.
- Uses SSTL25 Class II Differential I/O Standard.
- All numbers are generated with ispLEVER 8.1 software.

Figure 3-6. Generic DDRX1/DDR2 (With Clock and Data Edges Aligned)

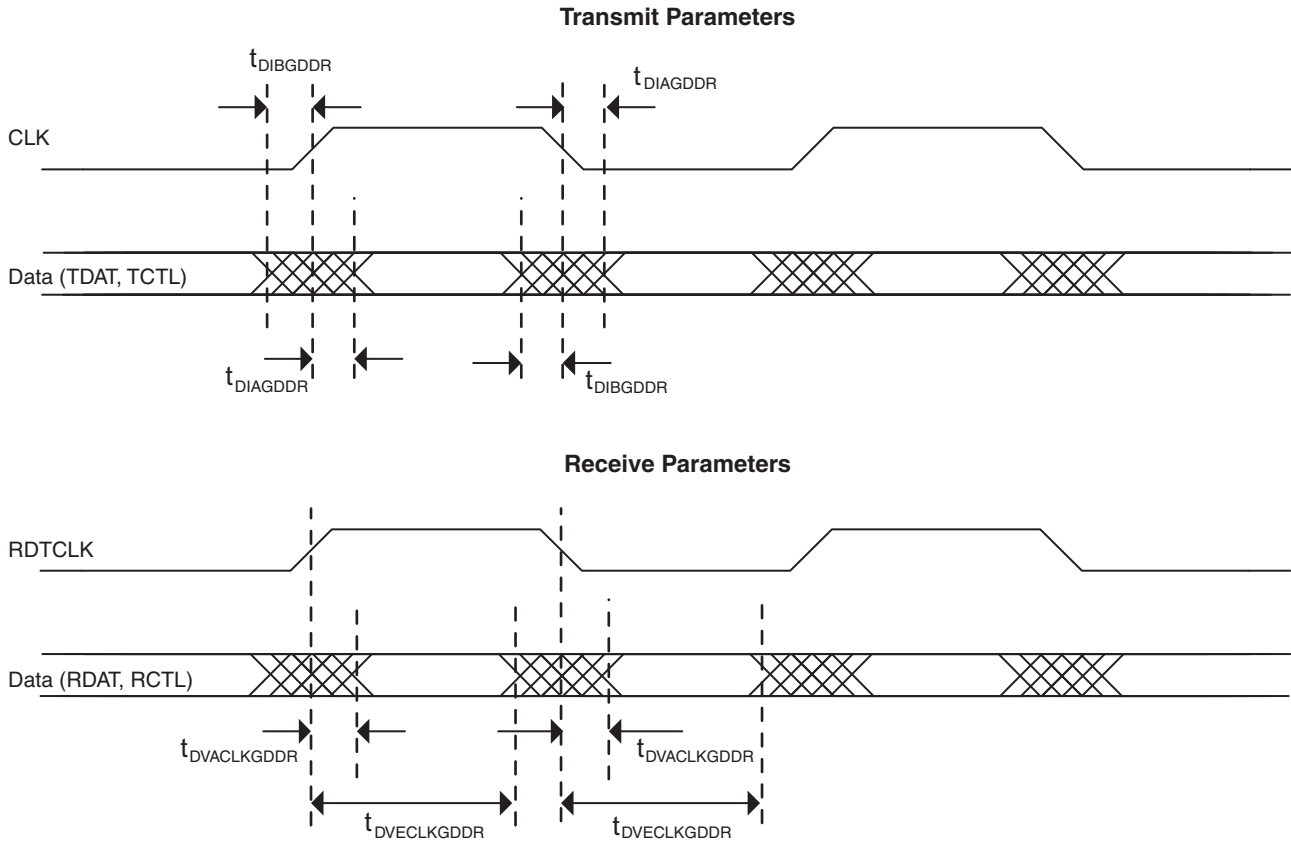


Figure 3-7. DDR/DDR2/DDR3 Parameters

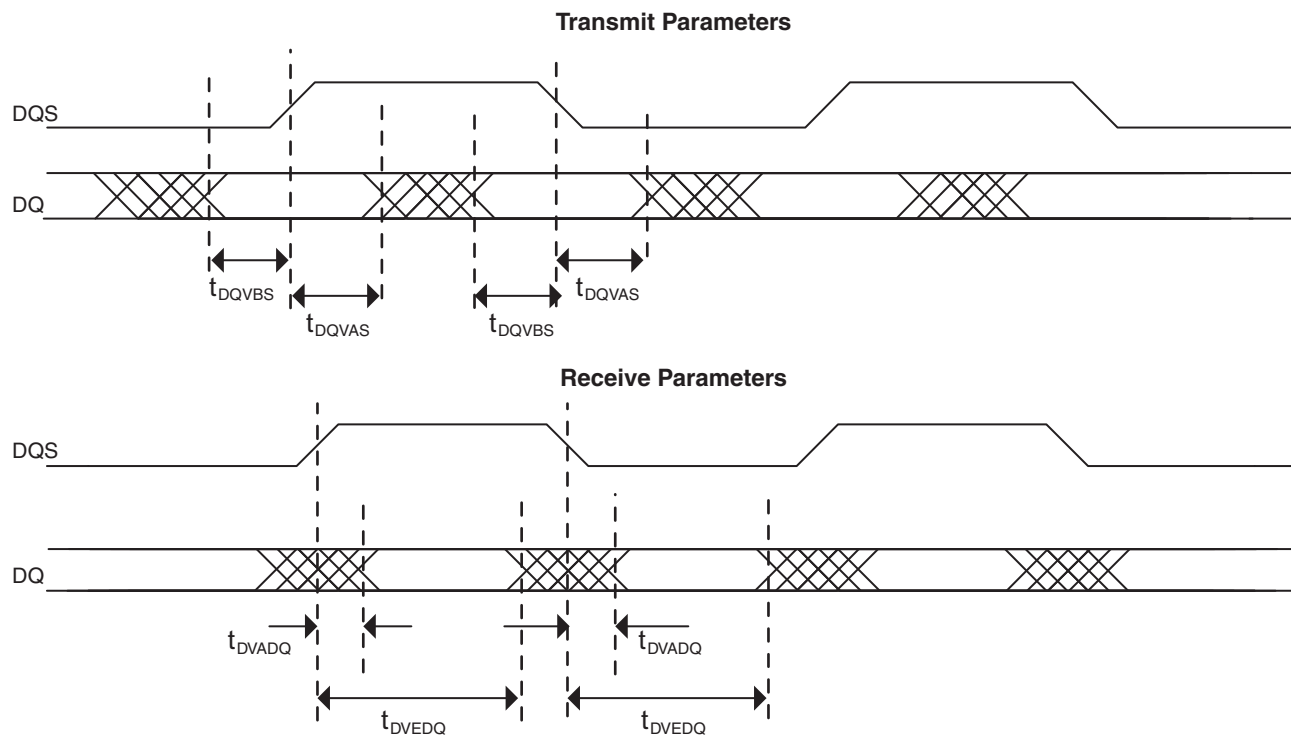
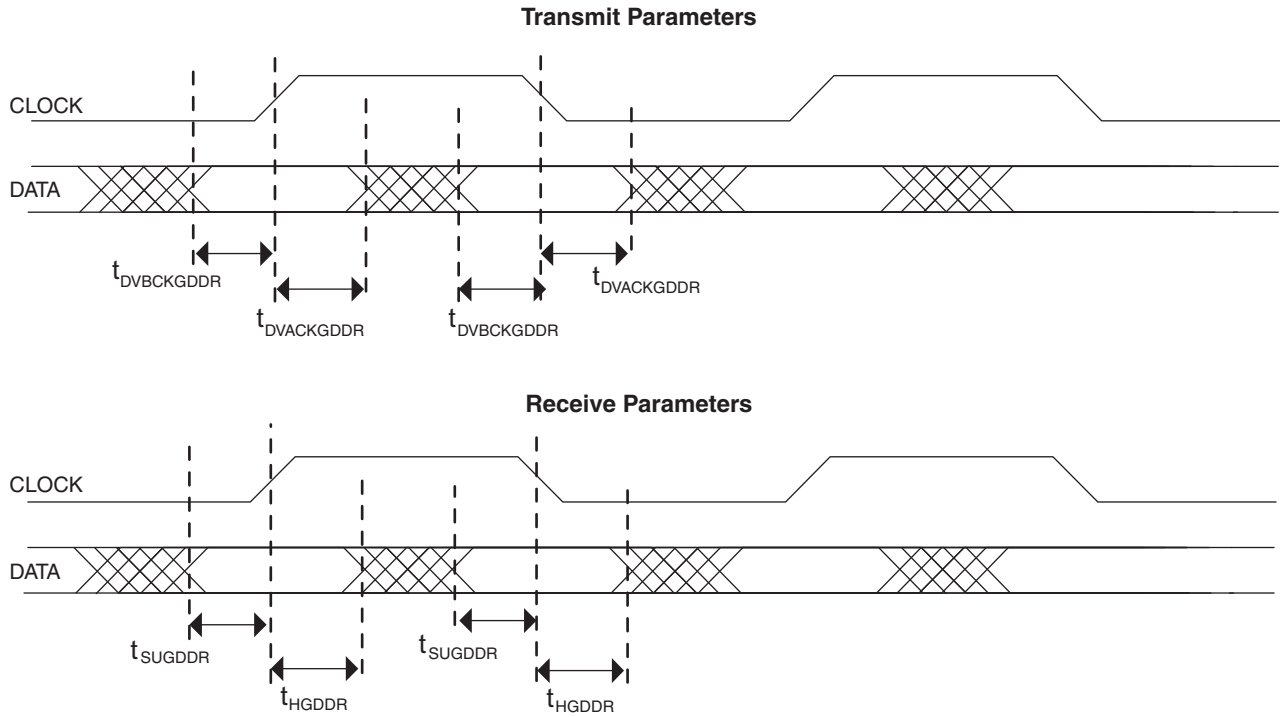




Figure 3-8. Generic DDRX1/DDR2 (With Clock Center on Data Window)



## LatticeECP3 Internal Switching Characteristics<sup>1, 2</sup>

Over Recommended Commercial Operating Conditions

Parameter	Description	-9		-8		-7		-6		Units.
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
<b>PFU/PFF Logic Mode Timing</b>										
t <sub>LUT4_PFU</sub>	LUT4 delay (A to D inputs to F output)	—	0.137	—	0.147	—	0.163	—	0.179	ns
t <sub>LUT6_PFU</sub>	LUT6 delay (A to D inputs to OFX output)	—	0.252	—	0.281	—	0.335	—	0.379	ns
t <sub>LSR_PFU</sub>	Set/Reset to output of PFU (Asynchronous)	—	0.543	—	0.593	—	0.674	—	0.756	ns
t <sub>LSRREC_PFU</sub>	Asynchronous Set/Reset recovery time for PFU Logic		0.270		0.298		0.345		0.391	ns
t <sub>SUM_PFU</sub>	Clock to Mux (M0,M1) Input Setup Time	0.128	—	0.134	—	0.144	—	0.153	—	ns
t <sub>HM_PFU</sub>	Clock to Mux (M0,M1) Input Hold Time	-0.093	—	-0.097	—	-0.103	—	-0.109	—	ns
t <sub>SUD_PFU</sub>	Clock to D input setup time	0.057	—	0.061	—	0.068	—	0.075	—	ns
t <sub>HD_PFU</sub>	Clock to D input hold time	0.028	—	0.019	—	0.013	—	0.015	—	ns
t <sub>CK2Q_PFU</sub>	Clock to Q delay, (D-type Register Configuration)	—	0.225	—	0.243	—	0.273	—	0.303	ns
<b>PFU Dual Port Memory Mode Timing</b>										
t <sub>CORAM_PFU</sub>	Clock to Output (F Port)	—	0.730	—	0.710	—	0.803	—	0.897	ns
t <sub>SUDATA_PFU</sub>	Data Setup Time	-0.125	—	-0.137	—	-0.155	—	-0.174	—	ns
t <sub>HDATA_PFU</sub>	Data Hold Time	0.180	—	0.188	—	0.217	—	0.246	—	ns
t <sub>SUADDR_PFU</sub>	Address Setup Time	-0.209	—	-0.227	—	-0.257	—	-0.286	—	ns
t <sub>HADDR_PFU</sub>	Address Hold Time	0.228	—	0.240	—	0.275	—	0.310	—	ns
t <sub>SUWREN_PFU</sub>	Write/Read Enable Setup Time	-0.055	—	-0.055	—	-0.055	—	-0.063	—	ns
t <sub>HWREN_PFU</sub>	Write/Read Enable Hold Time	0.059	—	0.059	—	0.059	—	0.071	—	ns
<b>PIC Timing</b>										
<b>PIO Input/Output Buffer Timing</b>										
t <sub>IN_PIO</sub>	Input Buffer Delay (LVCMOS25)	—	0.423	—	0.423	—	0.466	—	0.508	ns
t <sub>OUT_PIO</sub>	Output Buffer Delay (LVCMOS25)	—	1.241	—	1.241	—	1.301	—	1.361	ns
<b>IOLOGIC Input/Output Timing</b>										
t <sub>SUI_PIO</sub>	Input Register Setup Time (Data Before Clock)	0.956	—	0.956	—	1.124	—	1.293	—	ns
t <sub>HI_PIO</sub>	Input Register Hold Time (Data after Clock)	0.225	—	0.225	—	0.184	—	0.240	—	ns
t <sub>COO_PIO</sub>	Output Register Clock to Output Delay <sup>4</sup>	—	1.09	-	1.09	-	1.16	-	1.23	ns
t <sub>SUCE_PIO</sub>	Input Register Clock Enable Setup Time	0.220	—	0.220	—	0.185	—	0.150	—	ns
t <sub>HCE_PIO</sub>	Input Register Clock Enable Hold Time	-0.085	—	-0.085	—	-0.072	—	-0.058	—	ns
t <sub>SULSR_PIO</sub>	Set/Reset Setup Time	0.117	—	0.117	—	0.103	—	0.088	—	ns
t <sub>HLSR_PIO</sub>	Set/Reset Hold Time	-0.107	—	-0.107	—	-0.094	—	-0.081	—	ns
<b>EBR Timing</b>										
t <sub>CO_EBR</sub>	Clock (Read) to output from Address or Data	—	2.65	—	2.78	—	2.89	—	2.99	ns
t <sub>COO_EBR</sub>	Clock (Write) to output from EBR output Register	—	0.29	—	0.31	—	0.32	—	0.33	ns
t <sub>SUDATA_EBR</sub>	Setup Data to EBR Memory	-0.207	—	-0.218	—	-0.227	—	-0.237	—	ns
t <sub>HDATA_EBR</sub>	Hold Data to EBR Memory	0.237	—	0.249	—	0.257	—	0.265	—	ns
t <sub>SUADDR_EBR</sub>	Setup Address to EBR Memory	-0.067	—	-0.071	—	-0.070	—	-0.068	—	ns
t <sub>HADDR_EBR</sub>	Hold Address to EBR Memory	0.112	—	0.118	—	0.098	—	0.077	—	ns
t <sub>SUWREN_EBR</sub>	Setup Write/Read Enable to EBR Memory	-0.102	—	-0.107	—	-0.106	—	-0.106	—	ns

## LatticeECP3 Internal Switching Characteristics<sup>1, 2</sup> (Continued)

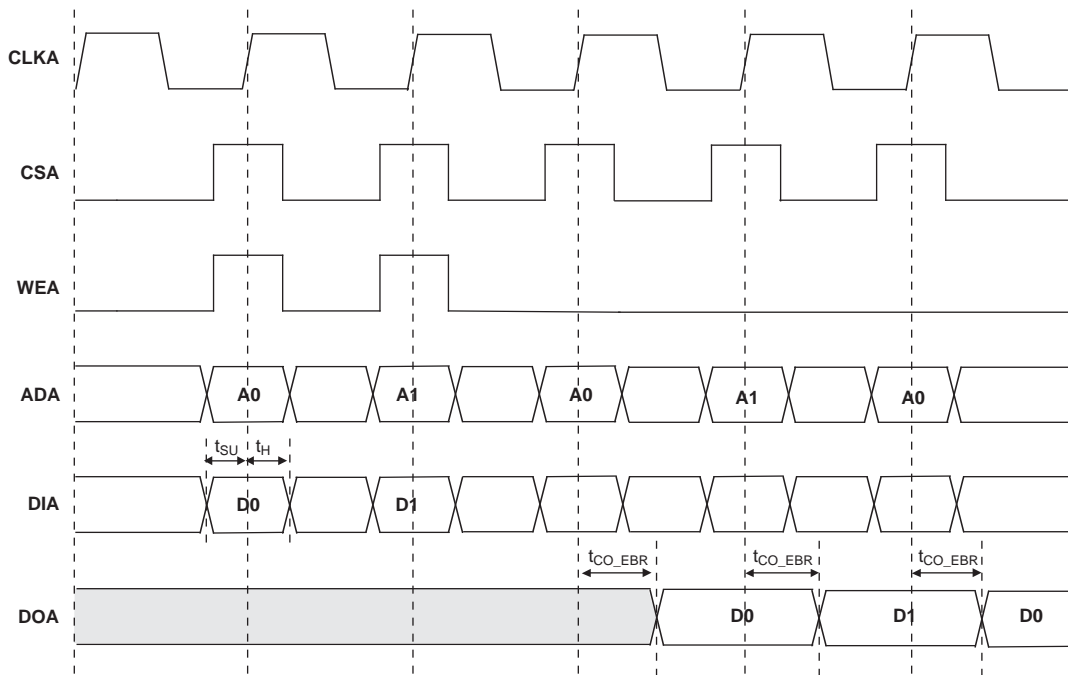
Over Recommended Commercial Operating Conditions

Parameter	Description	-9		-8		-7		-6		Units.
		Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
t <sub>HWREN_EBR</sub>	Hold Write/Read Enable to EBR Memory	0.134	—	0.141	—	0.145	—	0.149	—	ns
t <sub>SUCE_EBR</sub>	Clock Enable Setup Time to EBR Output Register	0.083	—	0.087	—	0.096	—	0.104	—	ns
t <sub>HCE_EBR</sub>	Clock Enable Hold Time to EBR Output Register	-0.063	—	-0.066	—	-0.080	—	-0.094	—	ns
t <sub>SUBE_EBR</sub>	Byte Enable Set-Up Time to EBR Output Register	-0.067	—	-0.071	—	-0.070	—	-0.068	—	ns
t <sub>HBE_EBR</sub>	Byte Enable Hold Time to EBR Output Register	0.112	—	0.118	—	0.098	—	0.077	—	ns
<b>DSP Block Timing<sup>3</sup></b>										
t <sub>SUI_DSP</sub>	Input Register Setup Time	0.30	—	0.32	—	0.36	—	0.39	—	ns
t <sub>HI_DSP</sub>	Input Register Hold Time	-0.16	—	-0.17	—	-0.19	—	-0.21	—	ns
t <sub>SUP_DSP</sub>	Pipeline Register Setup Time	2.19	—	2.23	—	2.30	—	2.37	—	ns
t <sub>HP_DSP</sub>	Pipeline Register Hold Time	-0.98	—	-1.02	—	-1.09	—	-1.15	—	ns
t <sub>SUO_DSP</sub>	Output Register Setup Time	3.01	—	3.09	—	3.22	—	3.34	—	ns
t <sub>HO_DSP</sub>	Output Register Hold Time	-1.62	—	-1.67	—	-1.76	—	-1.84	—	ns
t <sub>COI_DSP</sub>	Input Register Clock to Output Time	—	2.82	—	3.05	—	3.35	—	3.73	ns
t <sub>COP_DSP</sub>	Pipeline Register Clock to Output Time	—	1.20	—	1.30	—	1.47	—	1.64	ns
t <sub>COO_DSP</sub>	Output Register Clock to Output Time	—	0.57	—	0.58	—	0.60	—	0.62	ns
t <sub>SUOPT_DSP</sub>	Opcode Register Setup Time	0.29	—	0.31	—	0.35	—	0.39	—	ns
t <sub>HOPT_DSP</sub>	Opcode Register Hold Time	-0.18	—	-0.20	—	-0.24	—	-0.27	—	ns
t <sub>SUDATA_DSP</sub>	Cascade_data through ALU to Output Register Setup Time	1.56	—	1.69	—	1.94	—	2.14	—	ns
t <sub>HPDATA_DSP</sub>	Cascade_data through ALU to Output Register Hold Time	-0.47	—	-0.58	—	-0.80	—	-0.97	—	ns

1. Internal parameters are characterized but not tested on every device.
2. Commercial timing numbers are shown. Industrial timing numbers are typically slower and can be extracted from the Diamond or ispLEVER software.
3. DSP slice is configured in Multiply Add/Sub 18x18 mode.
4. The output register is in Flip-flop mode.

## Timing Diagrams

Figure 3-9. Read/Write Mode (Normal)



Note: Input data and address are registered at the positive edge of the clock and output data appears after the positive edge of the clock.

Figure 3-10. Read/Write Mode with Input and Output Registers

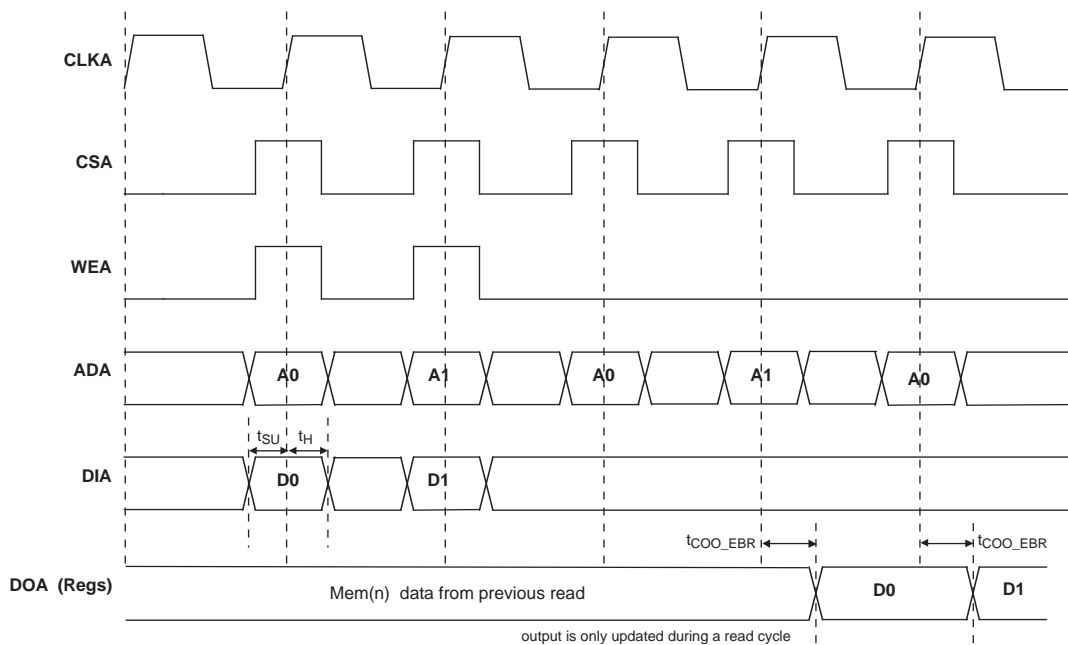
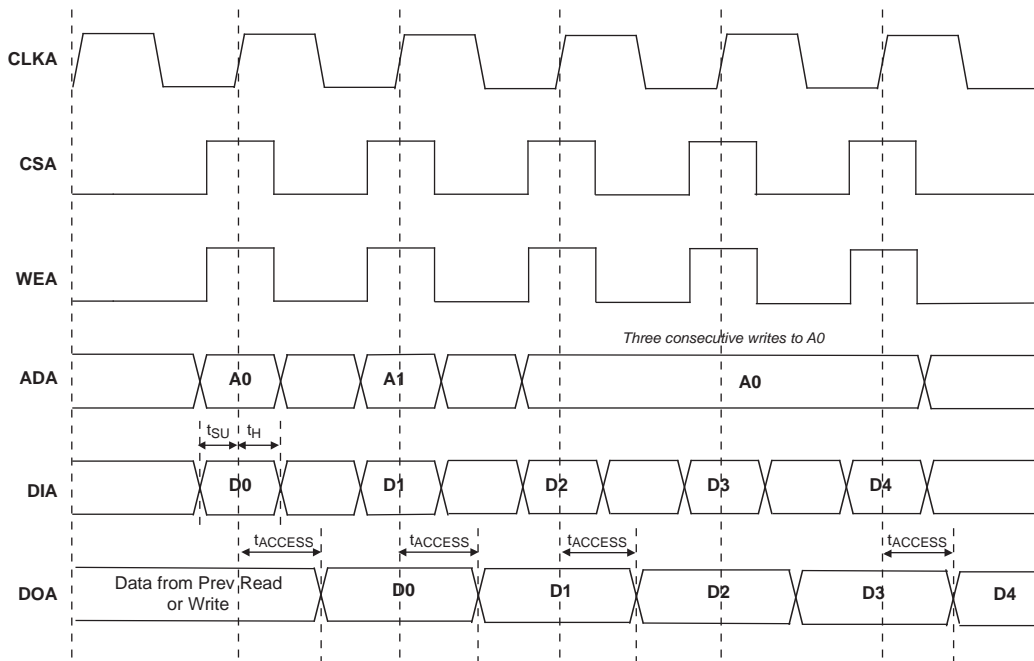


Figure 3-11. Write Through (SP Read/Write on Port A, Input Registers Only)



Note: Input data and address are registered at the positive edge of the clock and output data appears after the positive edge of the clock.

**LatticeECP3 Family Timing Adders<sup>1, 2, 3, 4, 5</sup>**
**Over Recommended Commercial Operating Conditions**

Buffer Type	Description	-9 <sup>6</sup>	-8	-7	-6	Units
<b>Input Adjusters</b>						
LVDS25E	LVDS, Emulated, VCCIO = 2.5V	0.03	0.03	-0.01	-0.03	ns
LVDS25	LVDS, VCCIO = 2.5V	0.03	0.03	0.00	-0.04	ns
BLVDS25	BLVDS, Emulated, VCCIO = 2.5V	0.03	0.03	0.00	-0.04	ns
MLVDS25	MLVDS, Emulated, VCCIO = 2.5V	0.03	0.03	0.00	-0.04	ns
RSDS25	RSDS, VCCIO = 2.5V	0.03	0.03	-0.01	-0.03	ns
PPLVDS	Point-to-Point LVDS	0.03	0.03	-0.01	-0.03	ns
TRLVDS	Transition-Reduced LVDS	0.03	0.03	0.00	-0.04	ns
Mini MLVDS	Mini LVDS	0.03	0.03	-0.01	-0.03	ns
LVPECL33	LVPECL, Emulated, VCCIO = 3.3V	0.17	0.17	0.23	0.28	ns
HSTL18_I	HSTL_18 class I, VCCIO = 1.8V	0.20	0.20	0.17	0.13	ns
HSTL18_II	HSTL_18 class II, VCCIO = 1.8V	0.20	0.20	0.17	0.13	ns
HSTL18D_I	Differential HSTL 18 class I	0.20	0.20	0.17	0.13	ns
HSTL18D_II	Differential HSTL 18 class II	0.20	0.20	0.17	0.13	ns
HSTL15_I	HSTL_15 class I, VCCIO = 1.5V	0.10	0.10	0.12	0.13	ns
HSTL15D_I	Differential HSTL 15 class I	0.10	0.10	0.12	0.13	ns
SSTL33_I	SSTL_3 class I, VCCIO = 3.3V	0.17	0.17	0.23	0.28	ns
SSTL33_II	SSTL_3 class II, VCCIO = 3.3V	0.17	0.17	0.23	0.28	ns
SSTL33D_I	Differential SSTL_3 class I	0.17	0.17	0.23	0.28	ns
SSTL33D_II	Differential SSTL_3 class II	0.17	0.17	0.23	0.28	ns
SSTL25_I	SSTL_2 class I, VCCIO = 2.5V	0.12	0.12	0.14	0.16	ns
SSTL25_II	SSTL_2 class II, VCCIO = 2.5V	0.12	0.12	0.14	0.16	ns
SSTL25D_I	Differential SSTL_2 class I	0.12	0.12	0.14	0.16	ns
SSTL25D_II	Differential SSTL_2 class II	0.12	0.12	0.14	0.16	ns
SSTL18_I	SSTL_18 class I, VCCIO = 1.8V	0.08	0.08	0.06	0.04	ns
SSTL18_II	SSTL_18 class II, VCCIO = 1.8V	0.08	0.08	0.06	0.04	ns
SSTL18D_I	Differential SSTL_18 class I	0.08	0.08	0.06	0.04	ns
SSTL18D_II	Differential SSTL_18 class II	0.08	0.08	0.06	0.04	ns
SSTL15	SSTL_15, VCCIO = 1.5V	0.087	0.087	0.059	0.032	ns
SSTL15D	Differential SSTL_15	0.087	0.087	0.059	0.032	ns
LVTTTL33	LVTTTL, VCCIO = 3.3V	0.07	0.07	0.07	0.07	ns
LVC MOS33	LVC MOS, VCCIO = 3.3V	0.07	0.07	0.07	0.07	ns
LVC MOS25	LVC MOS, VCCIO = 2.5V	0.00	0.00	0.00	0.00	ns
LVC MOS18	LVC MOS, VCCIO = 1.8V	-0.13	-0.13	-0.13	-0.13	ns
LVC MOS15	LVC MOS, VCCIO = 1.5V	-0.07	-0.07	-0.07	-0.07	ns
LVC MOS12	LVC MOS, VCCIO = 1.2V	-0.20	-0.20	-0.19	-0.19	ns
PCI33	PCI, VCCIO = 3.3V	0.07	0.07	0.07	0.07	ns
<b>Output Adjusters</b>						
LVDS25E	LVDS, Emulated, VCCIO = 2.5V	1.02	1.02	1.14	1.26	ns
LVDS25	LVDS, VCCIO = 2.5V	-0.11	-0.11	-0.07	-0.03	ns
BLVDS25	BLVDS, Emulated, VCCIO = 2.5V	1.01	1.01	1.13	1.25	ns
MLVDS25	MLVDS, Emulated, VCCIO = 2.5V	1.01	1.01	1.13	1.25	ns

**LatticeECP3 Family Timing Adders<sup>1, 2, 3, 4, 5</sup> (Continued)**
**Over Recommended Commercial Operating Conditions**

Buffer Type	Description	-9 <sup>6</sup>	-8	-7	-6	Units
RS2S25	RS2S, VCCIO = 2.5V	-0.07	-0.07	-0.04	-0.01	ns
PPLVDS	Point-to-Point LVDS, True LVDS, VCCIO = 2.5V or 3.3V	-0.22	-0.22	-0.19	-0.16	ns
LVPECL33	LVPECL, Emulated, VCCIO = 3.3V	0.67	0.67	0.76	0.86	ns
HSTL18_I	HSTL_18 class I 8mA drive, VCCIO = 1.8V	1.20	1.20	1.34	1.47	ns
HSTL18_II	HSTL_18 class II, VCCIO = 1.8V	0.89	0.89	1.00	1.11	ns
HSTL18D_I	Differential HSTL 18 class I 8mA drive	1.20	1.20	1.34	1.47	ns
HSTL18D_II	Differential HSTL 18 class II	0.89	0.89	1.00	1.11	ns
HSTL15_I	HSTL_15 class I 4mA drive, VCCIO = 1.5V	1.67	1.67	1.83	1.99	ns
HSTL15D_I	Differential HSTL 15 class I 4mA drive	1.67	1.67	1.83	1.99	ns
SSTL33_I	SSTL_3 class I, VCCIO = 3.3V	1.12	1.12	1.17	1.21	ns
SSTL33_II	SSTL_3 class II, VCCIO = 3.3V	1.08	1.08	1.12	1.15	ns
SSTL33D_I	Differential SSTL_3 class I	1.12	1.12	1.17	1.21	ns
SSTL33D_II	Differential SSTL_3 class II	1.08	1.08	1.12	1.15	ns
SSTL25_I	SSTL_2 class I 8mA drive, VCCIO = 2.5V	1.06	1.06	1.19	1.31	ns
SSTL25_II	SSTL_2 class II 16mA drive, VCCIO = 2.5V	1.04	1.04	1.17	1.31	ns
SSTL25D_I	Differential SSTL_2 class I 8mA drive	1.06	1.06	1.19	1.31	ns
SSTL25D_II	Differential SSTL_2 class II 16mA drive	1.04	1.04	1.17	1.31	ns
SSTL18_I	SSTL_1.8 class I, VCCIO = 1.8V	0.70	0.70	0.84	0.97	ns
SSTL18_II	SSTL_1.8 class II 8mA drive, VCCIO = 1.8V	0.70	0.70	0.84	0.97	ns
SSTL18D_I	Differential SSTL_1.8 class I	0.70	0.70	0.84	0.97	ns
SSTL18D_II	Differential SSTL_1.8 class II 8mA drive	0.70	0.70	0.84	0.97	ns
SSTL15	SSTL_1.5, VCCIO = 1.5V	1.22	1.22	1.35	1.48	ns
SSTL15D	Differential SSTL_15	1.22	1.22	1.35	1.48	ns
LVTTTL33_4mA	LVTTTL 4mA drive, VCCIO = 3.3V	0.25	0.25	0.24	0.23	ns
LVTTTL33_8mA	LVTTTL 8mA drive, VCCIO = 3.3V	-0.06	-0.06	-0.06	-0.07	ns
LVTTTL33_12mA	LVTTTL 12mA drive, VCCIO = 3.3V	-0.01	-0.01	-0.02	-0.02	ns
LVTTTL33_16mA	LVTTTL 16mA drive, VCCIO = 3.3V	-0.07	-0.07	-0.07	-0.08	ns
LVTTTL33_20mA	LVTTTL 20mA drive, VCCIO = 3.3V	-0.12	-0.12	-0.13	-0.14	ns
LVC MOS33_4mA	LVC MOS 3.3 4mA drive, fast slew rate	0.25	0.25	0.24	0.23	ns
LVC MOS33_8mA	LVC MOS 3.3 8mA drive, fast slew rate	-0.06	-0.06	-0.06	-0.07	ns
LVC MOS33_12mA	LVC MOS 3.3 12mA drive, fast slew rate	-0.01	-0.01	-0.02	-0.02	ns
LVC MOS33_16mA	LVC MOS 3.3 16mA drive, fast slew rate	-0.07	-0.07	-0.07	-0.08	ns
LVC MOS33_20mA	LVC MOS 3.3 20mA drive, fast slew rate	-0.12	-0.12	-0.13	-0.14	ns
LVC MOS25_4mA	LVC MOS 2.5 4mA drive, fast slew rate	0.12	0.12	0.10	0.09	ns
LVC MOS25_8mA	LVC MOS 2.5 8mA drive, fast slew rate	-0.05	-0.05	-0.06	-0.07	ns
LVC MOS25_12mA	LVC MOS 2.5 12mA drive, fast slew rate	0.00	0.00	0.00	0.00	ns
LVC MOS25_16mA	LVC MOS 2.5 16mA drive, fast slew rate	-0.12	-0.12	-0.13	-0.14	ns
LVC MOS25_20mA	LVC MOS 2.5 20mA drive, fast slew rate	-0.12	-0.12	-0.13	-0.14	ns
LVC MOS18_4mA	LVC MOS 1.8 4mA drive, fast slew rate	0.11	0.11	0.12	0.14	ns
LVC MOS18_8mA	LVC MOS 1.8 8mA drive, fast slew rate	0.11	0.11	0.12	0.14	ns
LVC MOS18_12mA	LVC MOS 1.8 12mA drive, fast slew rate	-0.04	-0.04	-0.03	-0.03	ns
LVC MOS18_16mA	LVC MOS 1.8 16mA drive, fast slew rate	-0.04	-0.04	-0.03	-0.03	ns

**LatticeECP3 Family Timing Adders<sup>1, 2, 3, 4, 5</sup> (Continued)**
**Over Recommended Commercial Operating Conditions**

Buffer Type	Description	-9 <sup>6</sup>	-8	-7	-6	Units
LVC MOS15_4mA	LVC MOS 1.5 4mA drive, fast slew rate	0.21	0.21	0.25	0.29	ns
LVC MOS15_8mA	LVC MOS 1.5 8mA drive, fast slew rate	0.05	0.05	0.07	0.09	ns
LVC MOS12_2mA	LVC MOS 1.2 2mA drive, fast slew rate	0.43	0.43	0.51	0.59	ns
LVC MOS12_6mA	LVC MOS 1.2 6mA drive, fast slew rate	0.23	0.23	0.28	0.33	ns
LVC MOS33_4mA	LVC MOS 3.3 4mA drive, slow slew rate	1.44	1.44	1.58	1.72	ns
LVC MOS33_8mA	LVC MOS 3.3 8mA drive, slow slew rate	0.98	0.98	1.10	1.22	ns
LVC MOS33_12mA	LVC MOS 3.3 12mA drive, slow slew rate	0.67	0.67	0.77	0.86	ns
LVC MOS33_16mA	LVC MOS 3.3 16mA drive, slow slew rate	0.97	0.97	1.09	1.21	ns
LVC MOS33_20mA	LVC MOS 3.3 20mA drive, slow slew rate	0.67	0.67	0.76	0.85	ns
LVC MOS25_4mA	LVC MOS 2.5 4mA drive, slow slew rate	1.48	1.48	1.63	1.78	ns
LVC MOS25_8mA	LVC MOS 2.5 8mA drive, slow slew rate	1.02	1.02	1.14	1.27	ns
LVC MOS25_12mA	LVC MOS 2.5 12mA drive, slow slew rate	0.74	0.74	0.84	0.94	ns
LVC MOS25_16mA	LVC MOS 2.5 16mA drive, slow slew rate	1.02	1.02	1.14	1.26	ns
LVC MOS25_20mA	LVC MOS 2.5 20mA drive, slow slew rate	0.74	0.74	0.83	0.93	ns
LVC MOS18_4mA	LVC MOS 1.8 4mA drive, slow slew rate	1.60	1.60	1.77	1.93	ns
LVC MOS18_8mA	LVC MOS 1.8 8mA drive, slow slew rate	1.11	1.11	1.25	1.38	ns
LVC MOS18_12mA	LVC MOS 1.8 12mA drive, slow slew rate	0.87	0.87	0.98	1.09	ns
LVC MOS18_16mA	LVC MOS 1.8 16mA drive, slow slew rate	0.86	0.86	0.97	1.07	ns
LVC MOS15_4mA	LVC MOS 1.5 4mA drive, slow slew rate	1.71	1.71	1.89	2.08	ns
LVC MOS15_8mA	LVC MOS 1.5 8mA drive, slow slew rate	1.20	1.20	1.34	1.48	ns
LVC MOS12_2mA	LVC MOS 1.2 2mA drive, slow slew rate	1.37	1.37	1.56	1.74	ns
LVC MOS12_6mA	LVC MOS 1.2 6mA drive, slow slew rate	1.11	1.11	1.27	1.43	ns
PCI33	PCI, VCCIO = 3.3V	-0.12	-0.12	-0.13	-0.14	ns

1. Timing adders are characterized but not tested on every device.

2. LVC MOS timing measured with the load specified in Switching Test Condition table.

3. All other standards tested according to the appropriate specifications.

4. Not all I/O standards and drive strengths are supported for all banks. See the Architecture section of this data sheet for details.

5. Commercial timing numbers are shown. Industrial numbers are typically slower and can be extracted from the Diamond or ispLEVER software.

6. This data does not apply to the LatticeECP3-17EA device.



**LatticeECP3 Maximum I/O Buffer Speed** <sup>1, 2, 3, 4, 5, 6</sup>
**Over Recommended Operating Conditions**

Buffer	Description	Max.	Units
<b>Maximum Input Frequency</b>			
LVDS25	LVDS, $V_{CCIO} = 2.5V$	400	MHz
MLVDS25	MLVDS, Emulated, $V_{CCIO} = 2.5V$	400	MHz
BLVDS25	BLVDS, Emulated, $V_{CCIO} = 2.5V$	400	MHz
PPLVDS	Point-to-Point LVDS	400	MHz
TRLVDS	Transition-Reduced LVDS	612	MHz
Mini LVDS	Mini LVDS	400	MHz
LVPECL33	LVPECL, Emulated, $V_{CCIO} = 3.3V$	400	MHz
HSTL18 (all supported classes)	HSTL_18 class I, II, $V_{CCIO} = 1.8V$	400	MHz
HSTL15	HSTL_15 class I, $V_{CCIO} = 1.5V$	400	MHz
SSTL33 (all supported classes)	SSTL_3 class I, II, $V_{CCIO} = 3.3V$	400	MHz
SSTL25 (all supported classes)	SSTL_2 class I, II, $V_{CCIO} = 2.5V$	400	MHz
SSTL18 (all supported classes)	SSTL_18 class I, II, $V_{CCIO} = 1.8V$	400	MHz
LVTTTL33	LVTTTL, $V_{CCIO} = 3.3V$	166	MHz
LVC MOS33	LVC MOS, $V_{CCIO} = 3.3V$	166	MHz
LVC MOS25	LVC MOS, $V_{CCIO} = 2.5V$	166	MHz
LVC MOS18	LVC MOS, $V_{CCIO} = 1.8V$	166	MHz
LVC MOS15	LVC MOS 1.5, $V_{CCIO} = 1.5V$	166	MHz
LVC MOS12	LVC MOS 1.2, $V_{CCIO} = 1.2V$	166	MHz
PCI33	PCI, $V_{CCIO} = 3.3V$	66	MHz
<b>Maximum Output Frequency</b>			
LVDS25E	LVDS, Emulated, $V_{CCIO} = 2.5V$	300	MHz
LVDS25	LVDS, $V_{CCIO} = 2.5V$	612	MHz
MLVDS25	MLVDS, Emulated, $V_{CCIO} = 2.5V$	300	MHz
RS DS25	RS DS, Emulated, $V_{CCIO} = 2.5V$	612	MHz
BLVDS25	BLVDS, Emulated, $V_{CCIO} = 2.5V$	300	MHz
PPLVDS	Point-to-point LVDS	612	MHz
LVPECL33	LVPECL, Emulated, $V_{CCIO} = 3.3V$	612	MHz
Mini-LVDS	Mini LVDS	612	MHz
HSTL18 (all supported classes)	HSTL_18 class I, II, $V_{CCIO} = 1.8V$	200	MHz
HSTL15 (all supported classes)	HSTL_15 class I, $V_{CCIO} = 1.5V$	200	MHz
SSTL33 (all supported classes)	SSTL_3 class I, II, $V_{CCIO} = 3.3V$	233	MHz
SSTL25 (all supported classes)	SSTL_2 class I, II, $V_{CCIO} = 2.5V$	233	MHz
SSTL18 (all supported classes)	SSTL_18 class I, II, $V_{CCIO} = 1.8V$	266	MHz
LVTTTL33	LVTTTL, $V_{CCIO} = 3.3V$	166	MHz
LVC MOS33 (For all drives)	LVC MOS, 3.3V	166	MHz
LVC MOS25 (For all drives)	LVC MOS, 2.5V	166	MHz
LVC MOS18 (For all drives)	LVC MOS, 1.8V	166	MHz
LVC MOS15 (For all drives)	LVC MOS, 1.5V	166	MHz
LVC MOS12 (For all drives except 2mA)	LVC MOS, $V_{CCIO} = 1.2V$	166	MHz
LVC MOS12 (2mA drive)	LVC MOS, $V_{CCIO} = 1.2V$	100	MHz

---

**LatticeECP3 Maximum I/O Buffer Speed (Continued)<sup>1, 2, 3, 4, 5, 6</sup>****Over Recommended Operating Conditions**

<b>Buffer</b>	<b>Description</b>	<b>Max.</b>	<b>Units</b>
PCI33	PCI, $V_{CCIO} = 3.3V$	66	MHz

1. These maximum speeds are characterized but not tested on every device.
2. Maximum I/O speed for differential output standards emulated with resistors depends on the layout.
3. LVCMOS timing is measured with the load specified in the Switching Test Conditions table of this document.
4. All speeds are measured at fast slew.
5. Actual system operation may vary depending on user logic implementation.
6. Maximum data rate equals 2 times the clock rate when utilizing DDR.

## sysCLOCK PLL Timing

### Over Recommended Operating Conditions

Parameter	Descriptions	Conditions	Clock	Min.	Typ.	Max.	Units	
f <sub>IN</sub>	Input clock frequency (CLKI, CLKFB)		Edge clock	2	—	500	MHz	
			Primary clock <sup>4</sup>	2	—	420	MHz	
f <sub>OUT</sub>	Output clock frequency (CLKOP, CLKOS)		Edge clock	4	—	500	MHz	
			Primary clock <sup>4</sup>	4	—	420	MHz	
f <sub>OUT1</sub>	K-Divider output frequency	CLKOK		0.03125	—	250	MHz	
f <sub>OUT2</sub>	K2-Divider output frequency	CLKOK2		0.667	—	166	MHz	
f <sub>VCO</sub>	PLL VCO frequency			500	—	1000	MHz	
f <sub>PDF</sub> <sup>3</sup>	Phase detector input frequency		Edge clock	2	—	500	MHz	
			Primary clock <sup>4</sup>	2	—	420	MHz	
<b>AC Characteristics</b>								
t <sub>PA</sub>	Programmable delay unit			65	130	260	ps	
t <sub>DT</sub>	Output clock duty cycle (CLKOS, at 50% setting)		Edge clock	45	50	55	%	
			f <sub>OUT</sub> ≤ 250 MHz	Primary clock	45	50	55	%
			f <sub>OUT</sub> > 250MHz	Primary clock	30	50	70	%
t <sub>CFA</sub>	Coarse phase shift error (CLKOS, at all settings)			-5	0	+5	% of period	
t <sub>OPW</sub>	Output clock pulse width high or low (CLKOS)			1.8	—	—	ns	
t <sub>OPJIT</sub> <sup>1</sup>	Output clock period jitter	f <sub>OUT</sub> ≥ 420MHz		—	—	200	ps	
		420MHz > f <sub>OUT</sub> ≥ 100MHz		—	—	250	ps	
		f <sub>OUT</sub> < 100MHz		—	—	0.025	UIPP	
t <sub>SK</sub>	Input clock to output clock skew when N/M = integer			—	—	500	ps	
t <sub>LOCK</sub> <sup>2</sup>	Lock time	2 to 25 MHz		—	—	200	us	
		25 to 500 MHz		—	—	50	us	
t <sub>UNLOCK</sub>	Reset to PLL unlock time to ensure fast reset			—	—	50	ns	
t <sub>HI</sub>	Input clock high time	90% to 90%		0.5	—	—	ns	
t <sub>LO</sub>	Input clock low time	10% to 10%		0.5	—	—	ns	
t <sub>IPJIT</sub>	Input clock period jitter			—	—	400	ps	
t <sub>RST</sub>	Reset signal pulse width high, RSTK			10	—	—	ns	
	Reset signal pulse width high, RST			500	—	—	ns	

1. Jitter sample is taken over 10,000 samples of the primary PLL output with clean reference clock with no additional I/O toggling.
2. Output clock is valid after t<sub>LOCK</sub> for PLL reset and dynamic delay adjustment.
3. Period jitter and cycle-to-cycle jitter numbers are guaranteed for f<sub>PDF</sub> > 4MHz. For f<sub>PDF</sub> < 4MHz, the jitter numbers may not be met in certain conditions. Please contact the factory for f<sub>PDF</sub> < 4MHz.
4. When using internal feedback, maximum can be up to 500 MHz.

## DLL Timing

### Over Recommended Operating Conditions

Parameter	Description	Condition	Min.	Typ.	Max.	Units
$f_{REF}$	Input reference clock frequency (on-chip or off-chip)		133	—	500	MHz
$f_{FB}$	Feedback clock frequency (on-chip or off-chip)		133	—	500	MHz
$f_{CLKOP}^1$	Output clock frequency, CLKOP		133	—	500	MHz
$f_{CLKOS}^2$	Output clock frequency, CLKOS		33.3	—	500	MHz
$t_{PJIT}$	Output clock period jitter (clean input)			—	200	ps p-p
$t_{DUTY}$	Output clock duty cycle (at 50% levels, 50% duty cycle input clock, 50% duty cycle circuit turned off, time reference delay mode)	Edge Clock	40		60	%
		Primary Clock	30		70	%
$t_{DUTYTRD}$	Output clock duty cycle (at 50% levels, arbitrary duty cycle input clock, 50% duty cycle circuit enabled, time reference delay mode)	Primary Clock < 250MHz	45		55	%
		Primary Clock $\geq$ 250MHz	30		70	%
		Edge Clock	45		55	%
$t_{DUTYCIR}$	Output clock duty cycle (at 50% levels, arbitrary duty cycle input clock, 50% duty cycle circuit enabled, clock injection removal mode) with DLL cascading	Primary Clock < 250MHz	40		60	%
		Primary Clock $\geq$ 250MHz	30		70	%
		Edge Clock	45		55	%
$t_{SKEW}^3$	Output clock to clock skew between two outputs with the same phase setting		—	—	100	ps
$t_{PHASE}$	Phase error measured at device pads between off-chip reference clock and feedback clocks		—	—	+/-400	ps
$t_{PWH}$	Input clock minimum pulse width high (at 80% level)		550	—	—	ps
$t_{PWL}$	Input clock minimum pulse width low (at 20% level)		550	—	—	ps
$t_{INSTB}$	Input clock period jitter		—	—	500	ps
$t_{LOCK}$	DLL lock time		8	—	8200	cycles
$t_{RSWD}$	Digital reset minimum pulse width (at 80% level)		3	—	—	ns
$t_{DEL}$	Delay step size		27	45	70	ps
$t_{RANGE1}$	Max. delay setting for single delay block (64 taps)		1.9	3.1	4.4	ns
$t_{RANGE4}$	Max. delay setting for four chained delay blocks		7.6	12.4	17.6	ns

1. CLKOP runs at the same frequency as the input clock.

2. CLKOS minimum frequency is obtained with divide by 4.

3. This is intended to be a “path-matching” design guideline and is not a measurable specification.

## SERDES High-Speed Data Transmitter<sup>1</sup>

Table 3-6. Serial Output Timing and Levels

Symbol	Description	Frequency	Min.	Typ.	Max.	Units
V <sub>TX-DIFF-P-P-1.44</sub>	Differential swing (1.44V setting) <sup>1, 2</sup>	0.15 to 3.125 Gbps	1150	1440	1730	mV, p-p
V <sub>TX-DIFF-P-P-1.35</sub>	Differential swing (1.35V setting) <sup>1, 2</sup>	0.15 to 3.125 Gbps	1080	1350	1620	mV, p-p
V <sub>TX-DIFF-P-P-1.26</sub>	Differential swing (1.26V setting) <sup>1, 2</sup>	0.15 to 3.125 Gbps	1000	1260	1510	mV, p-p
V <sub>TX-DIFF-P-P-1.13</sub>	Differential swing (1.13V setting) <sup>1, 2</sup>	0.15 to 3.125 Gbps	840	1130	1420	mV, p-p
V <sub>TX-DIFF-P-P-1.04</sub>	Differential swing (1.04V setting) <sup>1, 2</sup>	0.15 to 3.125 Gbps	780	1040	1300	mV, p-p
V <sub>TX-DIFF-P-P-0.92</sub>	Differential swing (0.92V setting) <sup>1, 2</sup>	0.15 to 3.125 Gbps	690	920	1150	mV, p-p
V <sub>TX-DIFF-P-P-0.87</sub>	Differential swing (0.87V setting) <sup>1, 2</sup>	0.15 to 3.125 Gbps	650	870	1090	mV, p-p
V <sub>TX-DIFF-P-P-0.78</sub>	Differential swing (0.78V setting) <sup>1, 2</sup>	0.15 to 3.125 Gbps	585	780	975	mV, p-p
V <sub>TX-DIFF-P-P-0.64</sub>	Differential swing (0.64V setting) <sup>1, 2</sup>	0.15 to 3.125 Gbps	480	640	800	mV, p-p
V <sub>OCM</sub>	Output common mode voltage	—	V <sub>CCOB</sub> -0.75	V <sub>CCOB</sub> -0.60	V <sub>CCOB</sub> -0.45	V
T <sub>TX-R</sub>	Rise time (20% to 80%)	—	145	185	265	ps
T <sub>TX-F</sub>	Fall time (80% to 20%)	—	145	185	265	ps
Z <sub>TX-OI-SE</sub>	Output Impedance 50/75/HiZ Ohms (single ended)	—	-20%	50/75/ Hi Z	+20%	Ohms
R <sub>LTX-RL</sub>	Return loss (with package)	—	10			dB
T <sub>TX-INTRASKEW</sub>	Lane-to-lane TX skew within a SERDES quad block (intra-quad)	—	—	—	200	ps
T <sub>TX-INTERSKEW</sub> <sup>3</sup>	Lane-to-lane skew between SERDES quad blocks (inter-quad)	—	—	—	1UI +200	ps

1. All measurements are with 50 ohm impedance.

2. See TN1176, [LatticeECP3 SERDES/PCS Usage Guide](#) for actual binary settings and the min-max range.

3. Inter-quad skew is between all SERDES channels on the device and requires the use of a low skew internal reference clock.

**Table 3-7. Channel Output Jitter**

Description	Frequency	Min.	Typ.	Max.	Units
Deterministic	3.125 Gbps	—	—	0.17	UI, p-p
Random	3.125 Gbps	—	—	0.25	UI, p-p
Total	3.125 Gbps	—	—	0.35	UI, p-p
Deterministic	2.5Gbps	—	—	0.17	UI, p-p
Random	2.5Gbps	—	—	0.20	UI, p-p
Total	2.5Gbps	—	—	0.35	UI, p-p
Deterministic	1.25 Gbps	—	—	0.10	UI, p-p
Random	1.25 Gbps	—	—	0.22	UI, p-p
Total	1.25 Gbps	—	—	0.24	UI, p-p
Deterministic	622 Mbps	—	—	0.10	UI, p-p
Random	622 Mbps	—	—	0.20	UI, p-p
Total	622 Mbps	—	—	0.24	UI, p-p
Deterministic	250 Mbps	—	—	0.10	UI, p-p
Random	250 Mbps	—	—	0.18	UI, p-p
Total	250 Mbps	—	—	0.24	UI, p-p
Deterministic	150 Mbps	—	—	0.10	UI, p-p
Random	150 Mbps	—	—	0.18	UI, p-p
Total	150 Mbps	—	—	0.24	UI, p-p

Note: Values are measured with PRBS 2<sup>7</sup>-1, all channels operating, FPGA logic active, I/Os around SERDES pins quiet, reference clock @ 10X mode.

## SERDES/PCS Block Latency

Table 3-8 describes the latency of each functional block in the transmitter and receiver. Latency is given in parallel clock cycles. Figure 3-12 shows the location of each block.

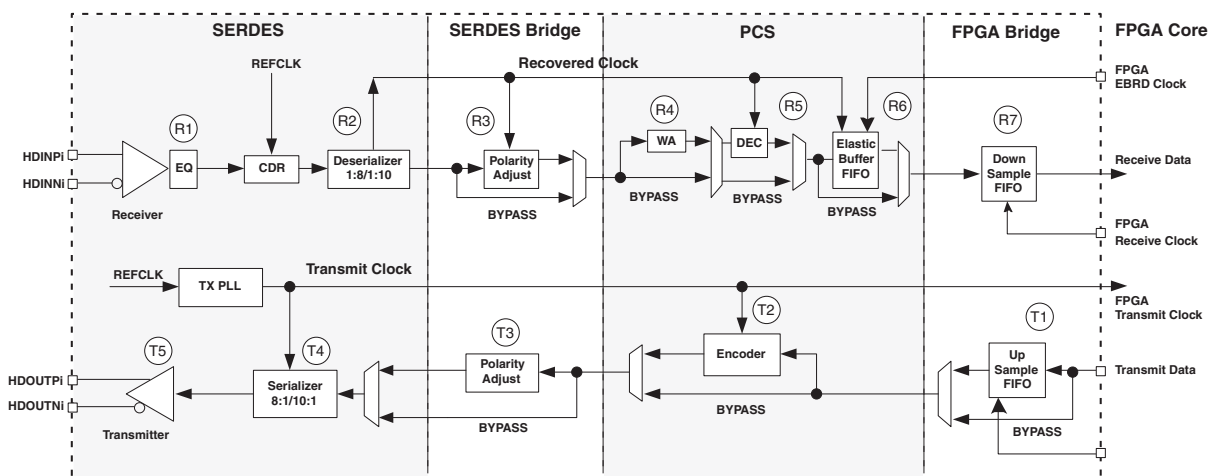
**Table 3-8. SERDES/PCS Latency Breakdown**

Item	Description	Min.	Avg.	Max.	Fixed	Bypass	Units
<b>Transmit Data Latency<sup>1</sup></b>							
T1	FPGA Bridge - Gearing disabled with different clocks	1	3	5	—	1	word clk
	FPGA Bridge - Gearing disabled with same clocks	—	—	—	3	1	word clk
	FPGA Bridge - Gearing enabled	1	3	5	—	—	word clk
T2	8b10b Encoder	—	—	—	2	1	word clk
T3	SERDES Bridge transmit	—	—	—	2	1	word clk
T4	Serializer: 8-bit mode	—	—	—	15 + Δ1	—	UI + ps
	Serializer: 10-bit mode	—	—	—	18 + Δ1	—	UI + ps
T5	Pre-emphasis ON	—	—	—	1 + Δ2	—	UI + ps
	Pre-emphasis OFF	—	—	—	0 + Δ3	—	UI + ps
<b>Receive Data Latency<sup>2</sup></b>							
R1	Equalization ON	—	—	—	Δ1	—	UI + ps
	Equalization OFF	—	—	—	Δ2	—	UI + ps
R2	Deserializer: 8-bit mode	—	—	—	10 + Δ3	—	UI + ps
	Deserializer: 10-bit mode	—	—	—	12 + Δ3	—	UI + ps
R3	SERDES Bridge receive	—	—	—	2	—	word clk
R4	Word alignment	3.1	—	4	—	—	word clk
R5	8b10b decoder	—	—	—	1	—	word clk
R6	Clock Tolerance Compensation	7	15	23	1	1	word clk
R7	FPGA Bridge - Gearing disabled with different clocks	1	3	5	—	1	word clk
	FPGA Bridge - Gearing disabled with same clocks	—	—	—	3	1	word clk
	FPGA Bridge - Gearing enabled	1	3	5	—	—	word clk

1. Δ1 = -245ps, Δ2 = +88ps, Δ3 = +112ps.

2. Δ1 = +118ps, Δ2 = +132ps, Δ3 = +700ps.

**Figure 3-12. Transmitter and Receiver Latency Block Diagram**



## SERDES High Speed Data Receiver

Table 3-9. Serial Input Data Specifications

Symbol	Description	Min.	Typ.	Max.	Units	
RX-CID <sub>S</sub>	Stream of nontransitions <sup>1</sup> (CID = Consecutive Identical Digits) @ 10 <sup>-12</sup> BER	3.125G	—	—	136	Bits
		2.5G	—	—	144	
		1.485G	—	—	160	
		622M	—	—	204	
		270M	—	—	228	
		150M	—	—	296	
V <sub>RX-DIFF-S</sub>	Differential input sensitivity	150	—	1760	mV, p-p	
V <sub>RX-IN</sub>	Input levels	0	—	V <sub>CCA</sub> +0.5 <sup>4</sup>	V	
V <sub>RX-CM-DC</sub>	Input common mode range (DC coupled)	0.6	—	V <sub>CCA</sub>	V	
V <sub>RX-CM-AC</sub>	Input common mode range (AC coupled) <sup>3</sup>	0.1	—	V <sub>CCA</sub> +0.2	V	
T <sub>RX-RELOCK</sub>	SCDR re-lock time <sup>2</sup>	—	1000	—	Bits	
Z <sub>RX-TERM</sub>	Input termination 50/75 Ohm/High Z	-20%	50/75/HiZ	+20%	Ohms	
RL <sub>RX-RL</sub>	Return loss (without package)	10	—	—	dB	

1. This is the number of bits allowed without a transition on the incoming data stream when using DC coupling.
2. This is the typical number of bit times to re-lock to a new phase or frequency within +/- 300 ppm, assuming 8b10b encoded data.
3. AC coupling is used to interface to LVPECL and LVDS. LVDS interfaces are found in laser drivers and Fibre Channel equipment. LVDS interfaces are generally found in 622 Mbps SERDES devices.
4. Up to 1.76V.

### Input Data Jitter Tolerance

A receiver's ability to tolerate incoming signal jitter is very dependent on jitter type. High speed serial interface standards have recognized the dependency on jitter type and have specifications to indicate tolerance levels for different jitter types as they relate to specific protocols. Sinusoidal jitter is considered to be a worst case jitter type.

Table 3-10. Receiver Total Jitter Tolerance Specification

Description	Frequency	Condition	Min.	Typ.	Max.	Units
Deterministic	3.125 Gbps	600 mV differential eye	—	—	0.47	UI, p-p
Random		600 mV differential eye	—	—	0.18	UI, p-p
Total		600 mV differential eye	—	—	0.65	UI, p-p
Deterministic	2.5 Gbps	600 mV differential eye	—	—	0.47	UI, p-p
Random		600 mV differential eye	—	—	0.18	UI, p-p
Total		600 mV differential eye	—	—	0.65	UI, p-p
Deterministic	1.25 Gbps	600 mV differential eye	—	—	0.47	UI, p-p
Random		600 mV differential eye	—	—	0.18	UI, p-p
Total		600 mV differential eye	—	—	0.65	UI, p-p
Deterministic	622 Mbps	600 mV differential eye	—	—	0.47	UI, p-p
Random		600 mV differential eye	—	—	0.18	UI, p-p
Total		600 mV differential eye	—	—	0.65	UI, p-p

Note: Values are measured with CJPAT, all channels operating, FPGA Logic active, I/Os around SERDES pins quiet, voltages are nominal, room temperature.



**Table 3-11. Periodic Receiver Jitter Tolerance Specification**

Description	Frequency	Condition	Min.	Typ.	Max.	Units
Periodic	2.97 Gbps	600 mV differential eye	—	—	0.24	UI, p-p
Periodic	2.5 Gbps	600 mV differential eye	—	—	0.22	UI, p-p
Periodic	1.485 Gbps	600 mV differential eye	—	—	0.24	UI, p-p
Periodic	622 Mbps	600 mV differential eye	—	—	0.15	UI, p-p
Periodic	150 Mbps	600 mV differential eye	—	—	0.5	UI, p-p

Note: Values are measured with PRBS  $2^7-1$ , all channels operating, FPGA Logic active, I/Os around SERDES pins quiet, voltages are nominal, room temperature.

### SERDES External Reference Clock

The external reference clock selection and its interface are a critical part of system applications for this product. Table 3-12 specifies reference clock requirements, over the full range of operating conditions.

**Table 3-12. External Reference Clock Specification (refclkp/refclkn)**

Symbol	Description	Min.	Typ.	Max.	Units
$F_{REF}$	Frequency range	15	—	320	MHz
$F_{REF-PPM}$	Frequency tolerance <sup>1</sup>	-1000	—	1000	ppm
$V_{REF-IN-SE}$	Input swing, single-ended clock <sup>2</sup>	200	—	$V_{CCA}$	mV, p-p
$V_{REF-IN-DIFF}$	Input swing, differential clock	200	—	$2 \cdot V_{CCA}$	mV, p-p differential
$V_{REF-IN}$	Input levels	0	—	$V_{CCA} + 0.3$	V
$D_{REF}$	Duty cycle <sup>3</sup>	40	—	60	%
$T_{REF-R}$	Rise time (20% to 80%)	200	500	1000	ps
$T_{REF-F}$	Fall time (80% to 20%)	200	500	1000	ps
$Z_{REF-IN-TERM-DIFF}$	Differential input termination	-20%	100/2K	+20%	Ohms
$C_{REF-IN-CAP}$	Input capacitance	—	—	7	pF

1. Depending on the application, the PLL\_LOL\_SET and CDR\_LOL\_SET control registers may be adjusted for other tolerance values as described in TN1176, [LatticeECP3 SERDES/PCS Usage Guide](#).
2. The signal swing for a single-ended input clock must be as large as the p-p differential swing of a differential input clock to get the same gain at the input receiver. Lower swings for the clock may be possible, but will tend to increase jitter.
3. Measured at 50% amplitude.

**Figure 3-13. SERDES External Reference Clock Waveforms**

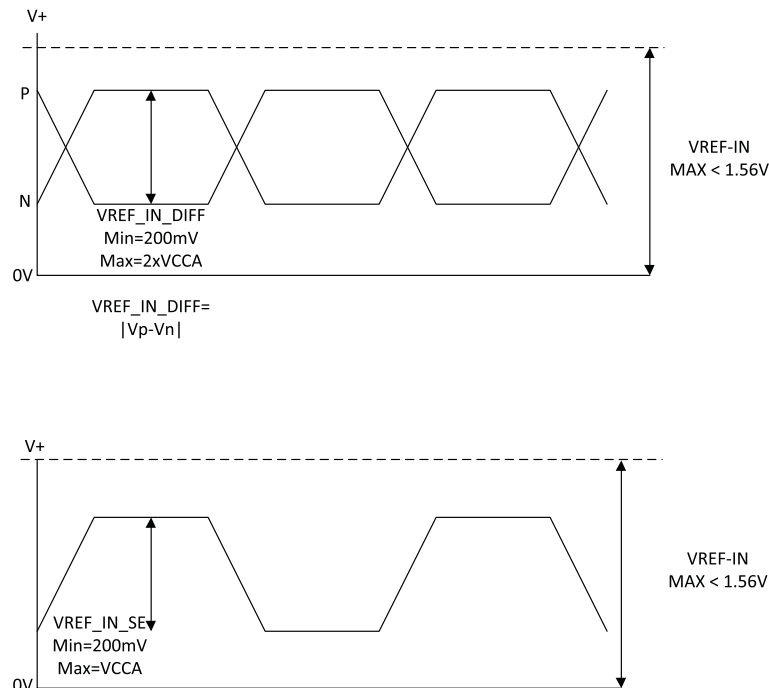


Figure 3-14. Jitter Transfer – 3.125 Gbps

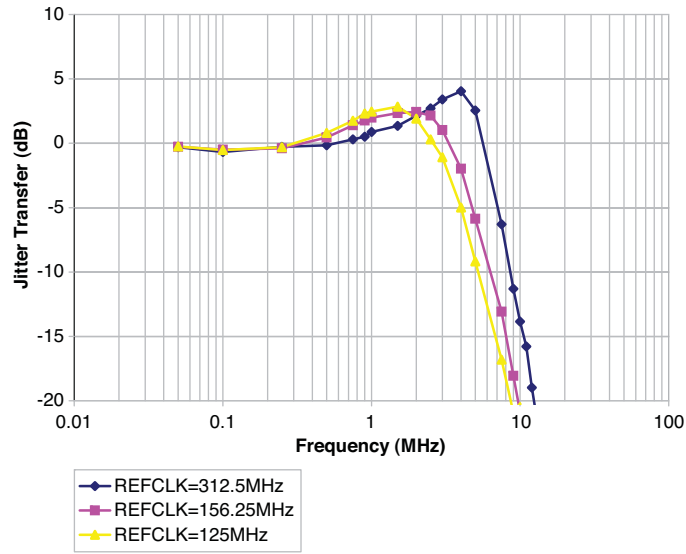


Figure 3-15. Jitter Transfer – 2.5 Gbps

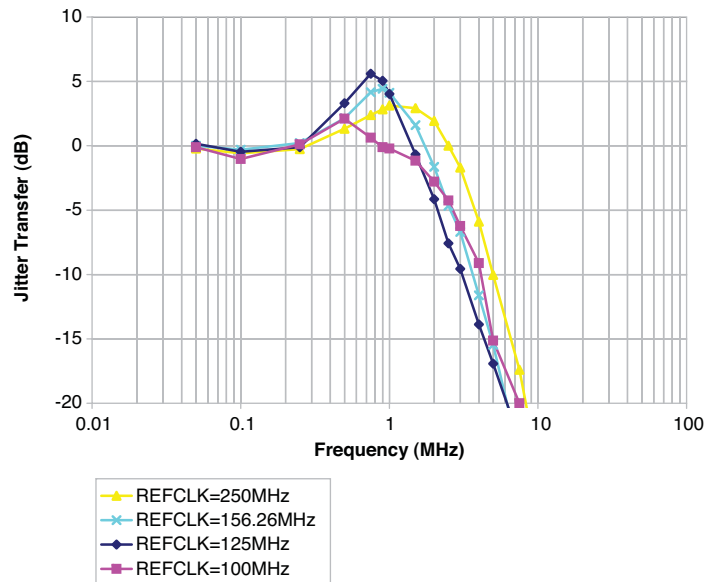


Figure 3-16. Jitter Transfer – 1.25 Gbps

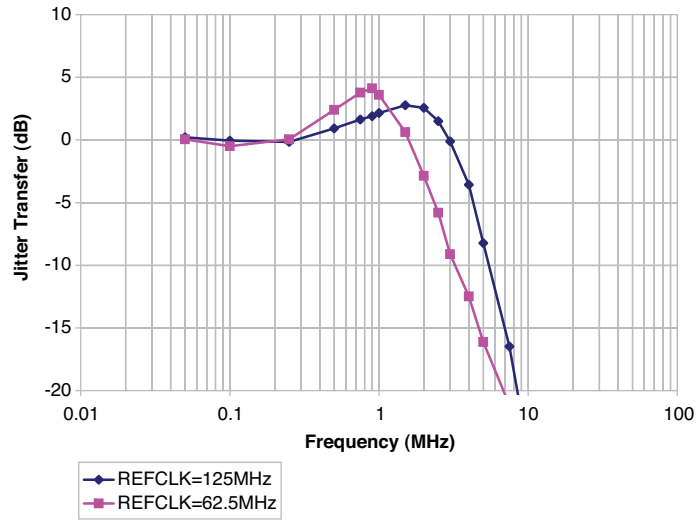
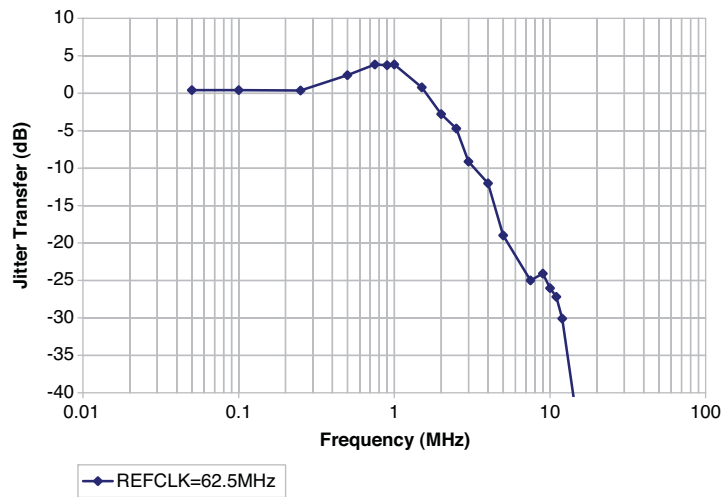


Figure 3-17. Jitter Transfer – 622 Mbps



## PCI Express Electrical and Timing Characteristics

### AC and DC Characteristics

#### Over Recommended Operating Conditions

Symbol	Description	Test Conditions	Min	Typ	Max	Units
<b>Transmit<sup>1</sup></b>						
UI	Unit interval		399.88	400	400.12	ps
V <sub>TX-DIFF_P-P</sub>	Differential peak-to-peak output voltage		0.8	1.0	1.2	V
V <sub>TX-DE-RATIO</sub>	De-emphasis differential output voltage ratio		-3	-3.5	-4	dB
V <sub>TX-CM-AC_P</sub>	RMS AC peak common-mode output voltage		—	—	20	mV
V <sub>TX-RCV-DETECT</sub>	Amount of voltage change allowed during receiver detection		—	—	600	mV
V <sub>TX-DC-CM</sub>	Tx DC common mode voltage		0	—	V <sub>CCOB</sub> + 5%	V
I <sub>TX-SHORT</sub>	Output short circuit current	V <sub>TX-D+=0.0V</sub> V <sub>TX-D-=0.0V</sub>	—	—	90	mA
Z <sub>TX-DIFF-DC</sub>	Differential output impedance		80	100	120	Ohms
RL <sub>TX-DIFF</sub>	Differential return loss		10	—	—	dB
RL <sub>TX-CM</sub>	Common mode return loss		6.0	—	—	dB
T <sub>TX-RISE</sub>	Tx output rise time	20 to 80%	0.125	—	—	UI
T <sub>TX-FALL</sub>	Tx output fall time	20 to 80%	0.125	—	—	UI
L <sub>TX-SKEW</sub>	Lane-to-lane static output skew for all lanes in port/link		—	—	1.3	ns
T <sub>TX-EYE</sub>	Transmitter eye width		0.75	—	—	UI
T <sub>TX-EYE-MEDIAN-TO-MAX-JITTER</sub>	Maximum time between jitter median and maximum deviation from median		—	—	0.125	UI
<b>Receive<sup>1,2</sup></b>						
UI	Unit Interval		399.88	400	400.12	ps
V <sub>RX-DIFF_P-P</sub>	Differential peak-to-peak input voltage		0.34 <sup>3</sup>	—	1.2	V
V <sub>RX-IDLE-DET-DIFF_P-P</sub>	Idle detect threshold voltage		65	—	340 <sup>3</sup>	mV
V <sub>RX-CM-AC_P</sub>	Receiver common mode voltage for AC coupling		—	—	150	mV
Z <sub>RX-DIFF-DC</sub>	DC differential input impedance		80	100	120	Ohms
Z <sub>RX-DC</sub>	DC input impedance		40	50	60	Ohms
Z <sub>RX-HIGH-IMP-DC</sub>	Power-down DC input impedance		200K	—	—	Ohms
RL <sub>RX-DIFF</sub>	Differential return loss		10	—	—	dB
RL <sub>RX-CM</sub>	Common mode return loss		6.0	—	—	dB
T <sub>RX-IDLE-DET-DIFF-ENTERTIME</sub>	Maximum time required for receiver to recognize and signal an unexpected idle on link		—	—	—	ms

1. Values are measured at 2.5 Gbps.

2. Measured with external AC-coupling on the receiver.

3. Not in compliance with PCI Express 1.1 standard.

## XAUI/Serial Rapid I/O Type 3/CPRI LV E.30 Electrical and Timing Characteristics

### AC and DC Characteristics

**Table 3-13. Transmit**

**Over Recommended Operating Conditions**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
$T_{RF}$	Differential rise/fall time	20%-80%	—	80	—	ps
$Z_{TX\_DIFF\_DC}$	Differential impedance		80	100	120	Ohms
$J_{TX\_DDJ}^{2,3,4}$	Output data deterministic jitter		—	—	0.17	UI
$J_{TX\_TJ}^{1,2,3,4}$	Total output data jitter		—	—	0.35	UI

1. Total jitter includes both deterministic jitter and random jitter.
2. Jitter values are measured with each CML output AC coupled into a 50-ohm impedance (100-ohm differential impedance).
3. Jitter and skew are specified between differential crossings of the 50% threshold of the reference signal.
4. Values are measured at 2.5 Gbps.

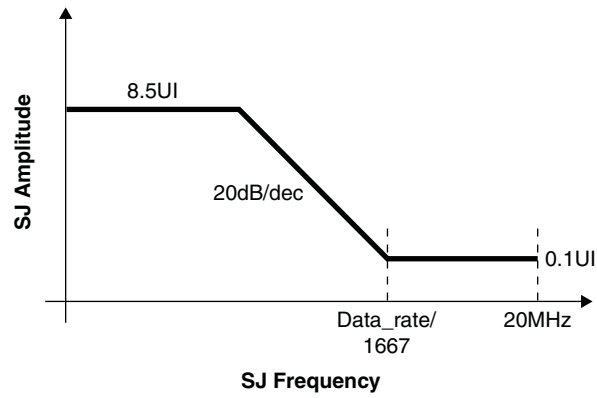
**Table 3-14. Receive and Jitter Tolerance**

**Over Recommended Operating Conditions**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
$RL_{RX\_DIFF}$	Differential return loss	From 100 MHz to 3.125 GHz	10	—	—	dB
$RL_{RX\_CM}$	Common mode return loss	From 100 MHz to 3.125 GHz	6	—	—	dB
$Z_{RX\_DIFF}$	Differential termination resistance		80	100	120	Ohms
$J_{RX\_DJ}^{1,2,3}$	Deterministic jitter tolerance (peak-to-peak)		—	—	0.37	UI
$J_{RX\_RJ}^{1,2,3}$	Random jitter tolerance (peak-to-peak)		—	—	0.18	UI
$J_{RX\_SJ}^{1,2,3}$	Sinusoidal jitter tolerance (peak-to-peak)		—	—	0.10	UI
$J_{RX\_TJ}^{1,2,3}$	Total jitter tolerance (peak-to-peak)		—	—	0.65	UI
$T_{RX\_EYE}$	Receiver eye opening		0.35	—	—	UI

1. Total jitter includes deterministic jitter, random jitter and sinusoidal jitter. The sinusoidal jitter tolerance mask is shown in Figure 3-18.
2. Jitter values are measured with each high-speed input AC coupled into a 50-ohm impedance.
3. Jitter and skew are specified between differential crossings of the 50% threshold of the reference signal.
4. Jitter tolerance parameters are characterized when Full Rx Equalization is enabled.
5. Values are measured at 2.5 Gbps.

Figure 3-18. XAUI Sinusoidal Jitter Tolerance Mask



Note: The sinusoidal jitter tolerance is measured with at least  $0.37UI_{pp}$  of Deterministic jitter ( $D_j$ ) and the sum of  $D_j$  and  $R_j$  (random jitter) is at least  $0.55UI_{pp}$ . Therefore, the sum of  $D_j$ ,  $R_j$  and  $S_j$  (sinusoidal jitter) is at least  $0.65UI_{pp}$  ( $D_j = 0.37$ ,  $R_j = 0.18$ ,  $S_j = 0.1$ ).

## Serial Rapid I/O Type 2/CPRI LV E.24 Electrical and Timing Characteristics

### AC and DC Characteristics

**Table 3-15. Transmit**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
$T_{RF}^1$	Differential rise/fall time	20%-80%	—	80	—	ps
$Z_{TX\_DIFF\_DC}$	Differential impedance		80	100	120	Ohms
$J_{TX\_DDJ}^{3,4,5}$	Output data deterministic jitter		—	—	0.17	UI
$J_{TX\_TJ}^{2,3,4,5}$	Total output data jitter		—	—	0.35	UI

1. Rise and Fall times measured with board trace, connector and approximately 2.5pf load.
2. Total jitter includes both deterministic jitter and random jitter. The random jitter is the total jitter minus the actual deterministic jitter.
3. Jitter values are measured with each CML output AC coupled into a 50-ohm impedance (100-ohm differential impedance).
4. Jitter and skew are specified between differential crossings of the 50% threshold of the reference signal.
5. Values are measured at 2.5 Gbps.

**Table 3-16. Receive and Jitter Tolerance**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
$RL_{RX\_DIFF}$	Differential return loss	From 100 MHz to 2.5 GHz	10	—	—	dB
$RL_{RX\_CM}$	Common mode return loss	From 100 MHz to 2.5 GHz	6	—	—	dB
$Z_{RX\_DIFF}$	Differential termination resistance		80	100	120	Ohms
$J_{RX\_DJ}^{2,3,4,5}$	Deterministic jitter tolerance (peak-to-peak)		—	—	0.37	UI
$J_{RX\_RJ}^{2,3,4,5}$	Random jitter tolerance (peak-to-peak)		—	—	0.18	UI
$J_{RX\_SJ}^{2,3,4,5}$	Sinusoidal jitter tolerance (peak-to-peak)		—	—	0.10	UI
$J_{RX\_TJ}^{1,2,3,4,5}$	Total jitter tolerance (peak-to-peak)		—	—	0.65	UI
$T_{RX\_EYE}$	Receiver eye opening		0.35	—	—	UI

1. Total jitter includes deterministic jitter, random jitter and sinusoidal jitter. The sinusoidal jitter tolerance mask is shown in Figure 3-18.
2. Jitter values are measured with each high-speed input AC coupled into a 50-ohm impedance.
3. Jitter and skew are specified between differential crossings of the 50% threshold of the reference signal.
4. Jitter tolerance, Differential Input Sensitivity and Receiver Eye Opening parameters are characterized when Full Rx Equalization is enabled.
5. Values are measured at 2.5 Gbps.



## Gigabit Ethernet/Serial Rapid I/O Type 1/SGMII/CPRI LV E.12 Electrical and Timing Characteristics

### AC and DC Characteristics

**Table 3-17. Transmit**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
$T_{RF}$	Differential rise/fall time	20%-80%	—	80	—	ps
$Z_{TX\_DIFF\_DC}$	Differential impedance		80	100	120	Ohms
$J_{TX\_DDJ}^{3,4,5}$	Output data deterministic jitter		—	—	0.10	UI
$J_{TX\_TJ}^{2,3,4,5}$	Total output data jitter		—	—	0.24	UI

1. Rise and fall times measured with board trace, connector and approximately 2.5pf load.
2. Total jitter includes both deterministic jitter and random jitter. The random jitter is the total jitter minus the actual deterministic jitter.
3. Jitter values are measured with each CML output AC coupled into a 50-ohm impedance (100-ohm differential impedance).
4. Jitter and skew are specified between differential crossings of the 50% threshold of the reference signal.
5. Values are measured at 1.25 Gbps.

**Table 3-18. Receive and Jitter Tolerance**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
$RL_{RX\_DIFF}$	Differential return loss	From 100 MHz to 1.25 GHz	10	—	—	dB
$RL_{RX\_CM}$	Common mode return loss	From 100 MHz to 1.25 GHz	6	—	—	dB
$Z_{RX\_DIFF}$	Differential termination resistance		80	100	120	Ohms
$J_{RX\_DJ}^{1,2,3,4,5}$	Deterministic jitter tolerance (peak-to-peak)		—	—	0.34	UI
$J_{RX\_RJ}^{1,2,3,4,5}$	Random jitter tolerance (peak-to-peak)		—	—	0.26	UI
$J_{RX\_SJ}^{1,2,3,4,5}$	Sinusoidal jitter tolerance (peak-to-peak)		—	—	0.11	UI
$J_{RX\_TJ}^{1,2,3,4,5}$	Total jitter tolerance (peak-to-peak)		—	—	0.71	UI
$T_{RX\_EYE}$	Receiver eye opening		0.29	—	—	UI

1. Total jitter includes deterministic jitter, random jitter and sinusoidal jitter. The sinusoidal jitter tolerance mask is shown in Figure 3-18.
2. Jitter values are measured with each high-speed input AC coupled into a 50-ohm impedance.
3. Jitter and skew are specified between differential crossings of the 50% threshold of the reference signal.
4. Jitter tolerance, Differential Input Sensitivity and Receiver Eye Opening parameters are characterized when Full Rx Equalization is enabled.
5. Values are measured at 1.25 Gbps.

## SMPTE SD/HD-SDI/3G-SDI (Serial Digital Interface) Electrical and Timing Characteristics

### AC and DC Characteristics

**Table 3-19. Transmit**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
BR <sub>SDO</sub>	Serial data rate		270	—	2975	Mbps
T <sub>JALIGNMENT</sub> <sup>2</sup>	Serial output jitter, alignment	270 Mbps	—	—	0.20	UI
T <sub>JALIGNMENT</sub> <sup>2</sup>	Serial output jitter, alignment	1485 Mbps	—	—	0.20	UI
T <sub>JALIGNMENT</sub> <sup>1,2</sup>	Serial output jitter, alignment	2970Mbps	—	—	0.30	UI
T <sub>JTIMING</sub>	Serial output jitter, timing	270 Mbps	—	—	0.20	UI
T <sub>JTIMING</sub>	Serial output jitter, timing	1485 Mbps	—	—	1.0	UI
T <sub>JTIMING</sub>	Serial output jitter, timing	2970 Mbps	—	—	2.0	UI

Notes:

- Timing jitter is measured in accordance with SMPTE RP 184-1996, SMPTE RP 192-1996 and the applicable serial data transmission standard, SMPTE 259M-1997 or SMPTE 292M (proposed). A color bar test pattern is used. The value of  $f_{SCLK}$  is 270 MHz or 360 MHz for SMPTE 259M, 540 MHz for SMPTE 344M or 1485 MHz for SMPTE 292M serial data rates. See the Timing Jitter Bandpass section.
- Jitter is defined in accordance with SMPTE RP1 184-1996 as: jitter at an equipment output in the absence of input jitter.
- All Tx jitter is measured at the output of an industry standard cable driver; connection to the cable driver is via a 50 ohm impedance differential signal from the Lattice SERDES device.
- The cable driver drives: RL=75 ohm, AC-coupled at 270, 1485, or 2970 Mbps, RREFLVL=RREFPRE=4.75kohm 1%.

**Table 3-20. Receive**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
BR <sub>SDI</sub>	Serial input data rate		270	—	2970	Mbps
CID	Stream of non-transitions (=Consecutive Identical Digits)		7(3G)/26(SMPTE Triple rates) @ 10-12 BER	—	—	Bits

**Table 3-21. Reference Clock**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
F <sub>VCLK</sub>	Video output clock frequency		27	—	74.25	MHz
DC <sub>V</sub>	Duty cycle, video clock		45	50	55	%

## HDMI (High-Definition Multimedia Interface) Electrical and Timing Characteristics

### AC and DC Characteristics

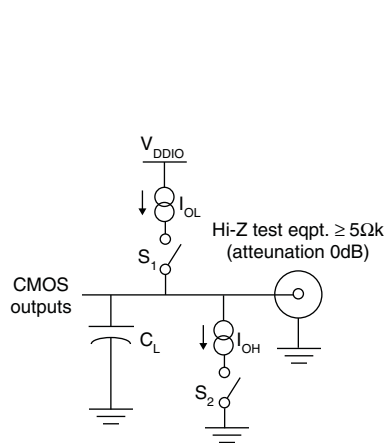
Table 3-22. Transmit and Receive<sup>1,2</sup>

Symbol	Description	Spec. Compliance		Units
		Min. Spec.	Max. Spec.	
<b>Transmit</b>				
Intra-pair Skew		—	75	ps
Inter-pair Skew		—	800	ps
TMDS Differential Clock Jitter		—	0.25	UI
<b>Receive</b>				
$R_T$	Termination Resistance	40	60	Ohms
$V_{ICM}$	Input AC Common Mode Voltage (50-ohm Setting)	—	50	mV
TMDS Clock Jitter	Clock Jitter Tolerance	—	0.25	UI

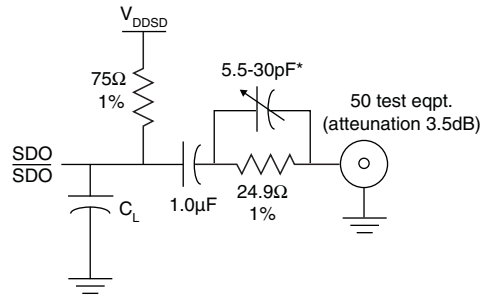
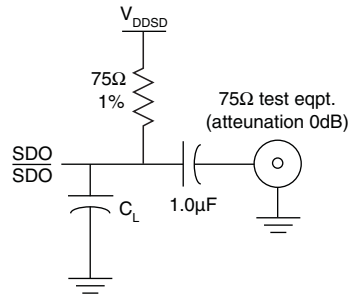
1. Output buffers must drive a translation device. Max. speed is 2Gbps. If translation device does not modify rise/fall time, the maximum speed is 1.5Gbps.
2. Input buffers must be AC coupled in order to support the 3.3V common mode. Generally, HDMI inputs are terminated by an external cable equalizer before data/clock is forwarded to the LatticeECP3 device.

**Figure 3-19. Test Loads**

**Test Loads**

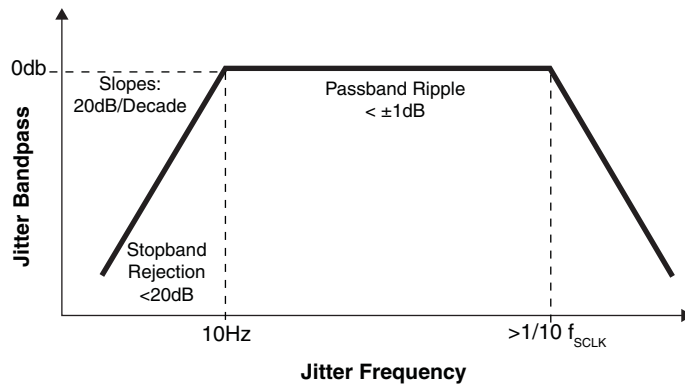


$C_L$  including probe and jig capacitance, 3pF max.  
 $S_1$  - open,  $S_2$  - closed for  $V_{OH}$  measurement.  
 $S_1$  - closed,  $S_2$  - open for  $V_{OL}$  measurement.



\*Risetime compensation.

**Timing Jitter Bandpass**



## LatticeECP3 sysCONFIG Port Timing Specifications

Over Recommended Operating Conditions

Parameter	Description	Min.	Max.	Units	
<b>POR, Configuration Initialization, and Wakeup</b>					
t <sub>ICFG</sub>	Time from the Application of V <sub>CC</sub> , V <sub>CCAUX</sub> or V <sub>CCIO8</sub> * (Whichever is the Last to Cross the POR Trip Point) to the Rising Edge of INITN	Master mode	—	23	ms
		Slave mode	—	6	ms
t <sub>VMC</sub>	Time from t <sub>ICFG</sub> to the Valid Master MCLK	—	5	μs	
t <sub>PRGM</sub>	PROGRAMN Low Time to Start Configuration	25	—	ns	
t <sub>PRGMRJ</sub>	PROGRAMN Pin Pulse Rejection	—	10	ns	
t <sub>DPPINIT</sub>	Delay Time from PROGRAMN Low to INITN Low	—	37	ns	
t <sub>DPPDONE</sub>	Delay Time from PROGRAMN Low to DONE Low	—	37	ns	
t <sub>DINIT</sub> <sup>1</sup>	PROGRAMN High to INITN High Delay	—	1	ms	
t <sub>MWC</sub>	Additional Wake Master Clock Signals After DONE Pin is High	100	500	cycles	
t <sub>CZ</sub>	MCLK From Active To Low To High-Z	—	300	ns	
<b>All Configuration Modes</b>					
t <sub>SUCDI</sub>	Data Setup Time to CCLK/MCLK	5	—	ns	
t <sub>HCDI</sub>	Data Hold Time to CCLK/MCLK	1	—	ns	
t <sub>CODO</sub>	CCLK/MCLK to DOUT in Flowthrough Mode	-0.2	12	ns	
<b>Slave Serial</b>					
t <sub>SSCH</sub>	CCLK Minimum High Pulse	5	—	ns	
t <sub>SSCL</sub>	CCLK Minimum Low Pulse	5	—	ns	
f <sub>CCLK</sub>	CCLK Frequency	Without encryption	—	33	MHz
		With encryption	—	20	MHz
<b>Master and Slave Parallel</b>					
t <sub>SUCS</sub>	CSN[1:0] Setup Time to CCLK/MCLK	7	—	ns	
t <sub>HCS</sub>	CSN[1:0] Hold Time to CCLK/MCLK	1	—	ns	
t <sub>SUWD</sub>	WRITEN Setup Time to CCLK/MCLK	7	—	ns	
t <sub>HWD</sub>	WRITEN Hold Time to CCLK/MCLK	1	—	ns	
t <sub>DCB</sub>	CCLK/MCLK to BUSY Delay Time	—	12	ns	
t <sub>CORD</sub>	CCLK to Out for Read Data	—	12	ns	
t <sub>BSCH</sub>	CCLK Minimum High Pulse	6	—	ns	
t <sub>BSCL</sub>	CCLK Minimum Low Pulse	6	—	ns	
t <sub>BSCYC</sub>	Byte Slave Cycle Time	30	—	ns	
f <sub>CCLK</sub>	CCLK/MCLK Frequency	Without encryption	—	33	MHz
		With encryption	—	20	MHz
<b>Master and Slave SPI</b>					
t <sub>CFGX</sub>	INITN High to MCLK Low	—	80	ns	
t <sub>CSSPI</sub>	INITN High to CSSPIN Low	0.2	2	μs	
t <sub>SOCDO</sub>	MCLK Low to Output Valid	—	15	ns	
t <sub>CSPID</sub>	CSSPIN[0:1] Low to First MCLK Edge Setup Time	0.3		μs	
f <sub>CCLK</sub>	CCLK Frequency	Without encryption	—	33	MHz
		With encryption	—	20	MHz
t <sub>SSCH</sub>	CCLK Minimum High Pulse	5	—	ns	
t <sub>SSCL</sub>	CCLK Minimum Low Pulse	5	—	ns	
t <sub>HLCH</sub>	HOLDN Low Setup Time (Relative to CCLK)	5	—	ns	

## LatticeECP3 sysCONFIG Port Timing Specifications (Continued)

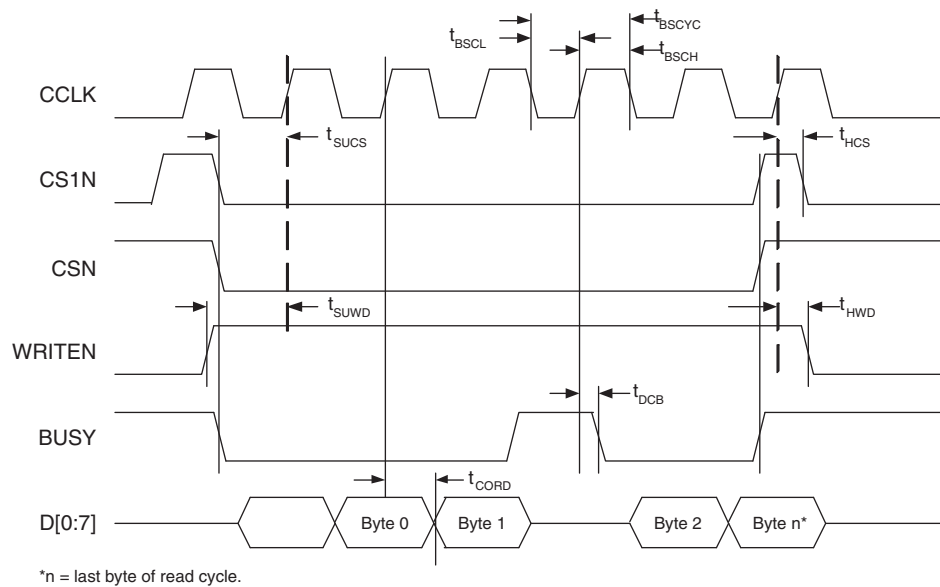
Over Recommended Operating Conditions

Parameter	Description	Min.	Max.	Units
$t_{CHHH}$	HOLDN Low Hold Time (Relative to CCLK)	5	—	ns
<b>Master and Slave SPI (Continued)</b>				
$t_{CHHL}$	HOLDN High Hold Time (Relative to CCLK)	5	—	ns
$t_{HHCH}$	HOLDN High Setup Time (Relative to CCLK)	5	—	ns
$t_{HLQZ}$	HOLDN to Output High-Z	—	9	ns
$t_{HHQX}$	HOLDN to Output Low-Z	—	9	ns

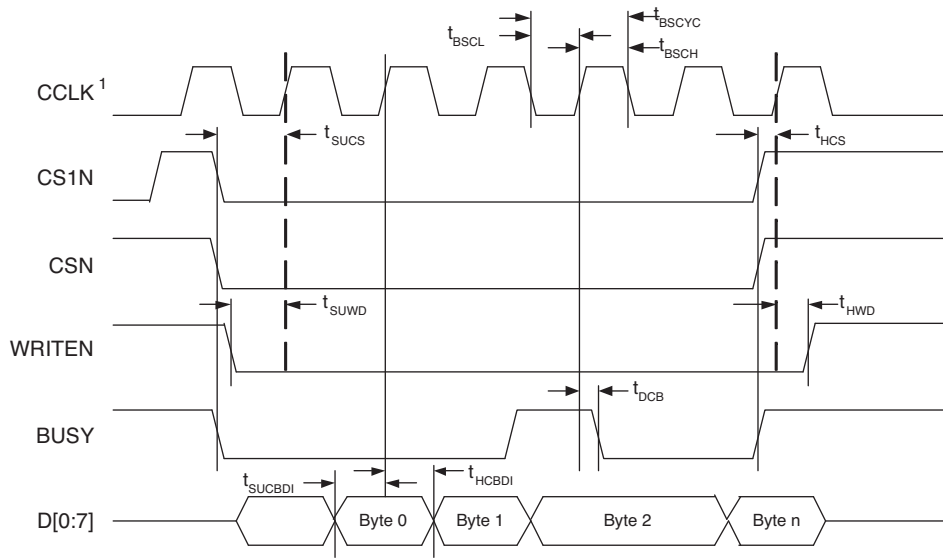
1. Re-toggling the PROGRAMN pin is not permitted until the INITN pin is high. Avoid consecutive toggling of the PROGRAMN.

Parameter	Min.	Max.	Units
Master Clock Frequency	Selected value - 15%	Selected value + 15%	MHz
Duty Cycle	40	60	%

Figure 3-20. sysCONFIG Parallel Port Read Cycle

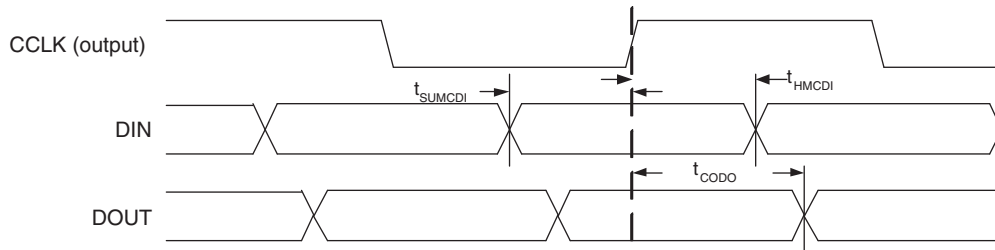


**Figure 3-21. sysCONFIG Parallel Port Write Cycle**

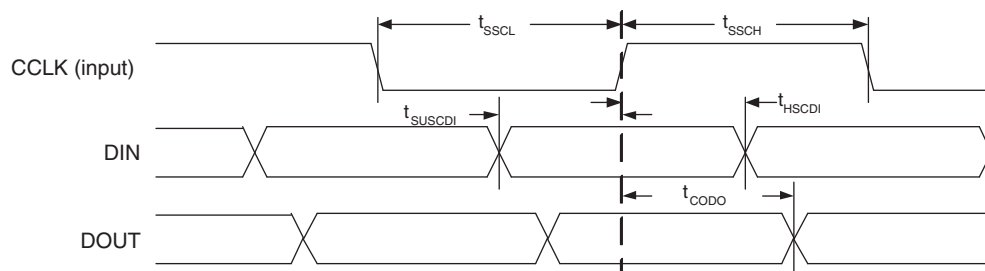


1. In Master Parallel Mode the FPGA provides CCLK (MCLK). In Slave Parallel Mode the external device provides CCLK.

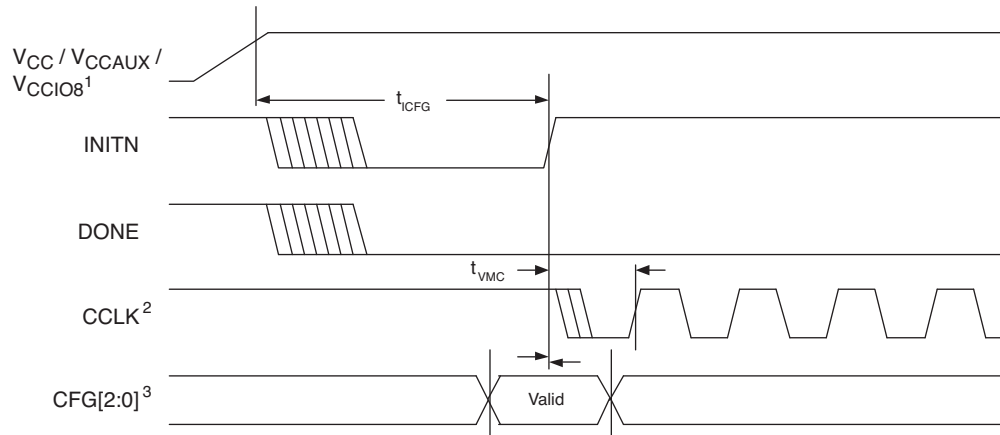
**Figure 3-22. sysCONFIG Master Serial Port Timing**



**Figure 3-23. sysCONFIG Slave Serial Port Timing**

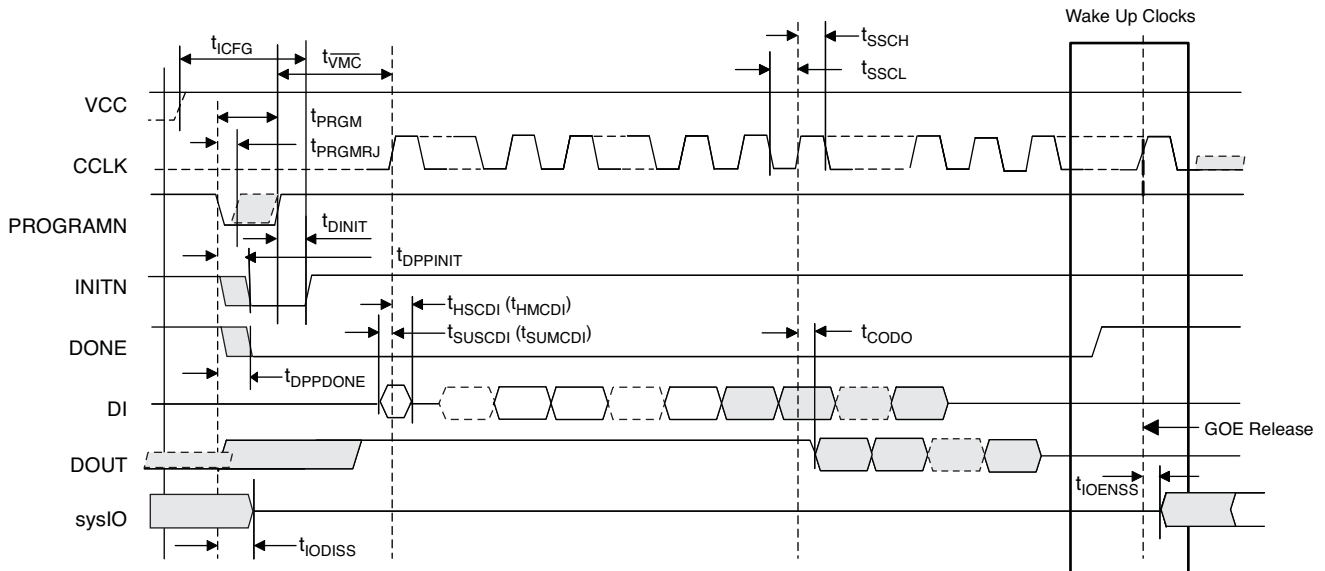


**Figure 3-24. Power-On-Reset (POR) Timing**



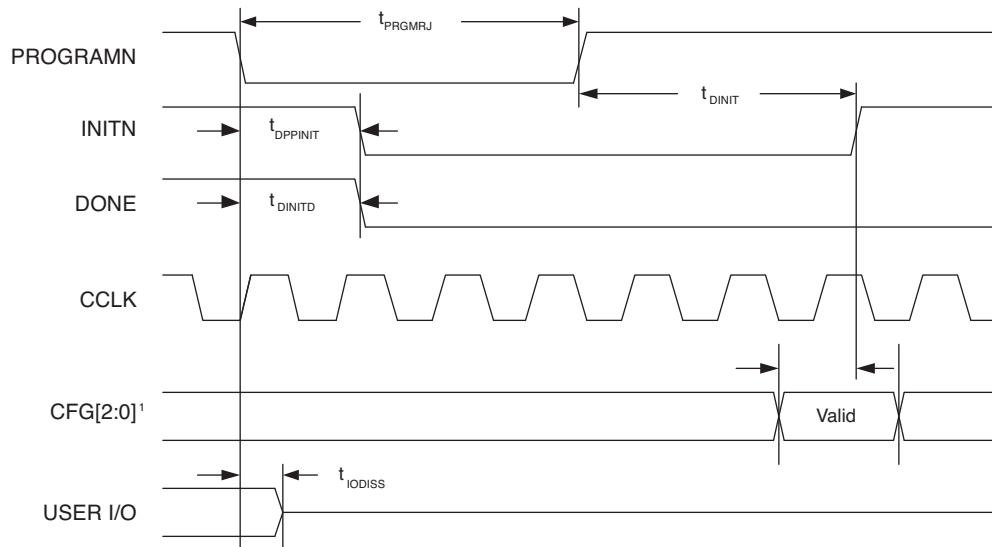
1. Time taken from VCC, VCCAUX or VCCIO8, whichever is the last to cross the POR trip point.
2. Device is in a Master Mode (SPI, SPI<sub>m</sub>).
3. The CFG pins are normally static (hard wired).

**Figure 3-25. sysCONFIG Port Timing**



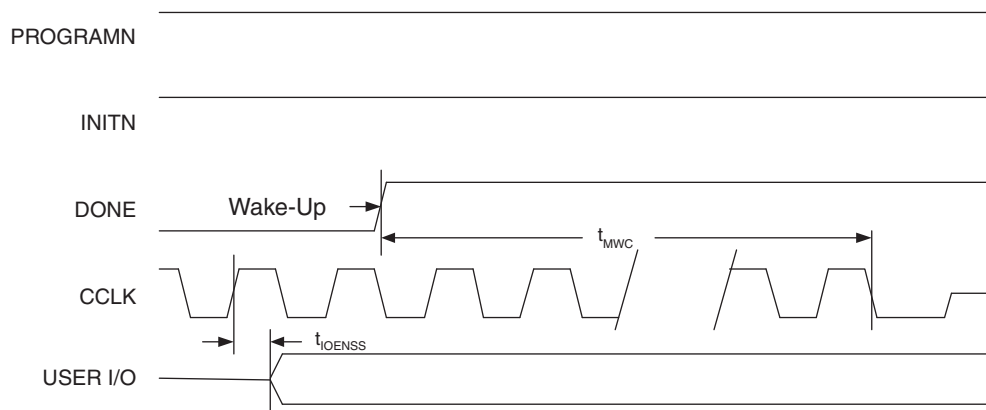


**Figure 3-26. Configuration from PROGRAMN Timing**

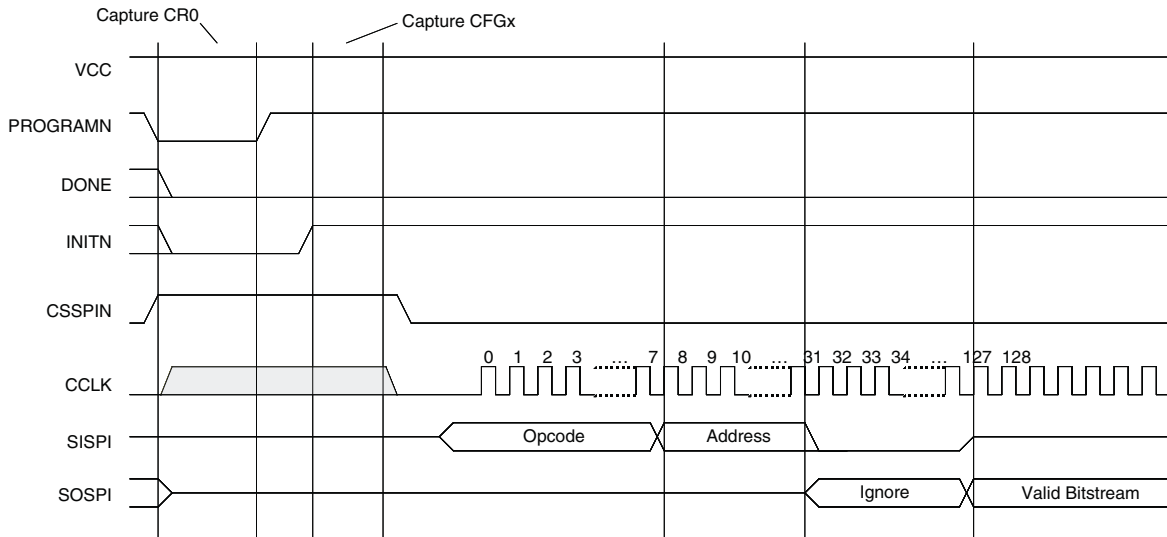


1. The CFG pins are normally static (hard wired)

**Figure 3-27. Wake-Up Timing**



**Figure 3-28. Master SPI Configuration Waveforms**

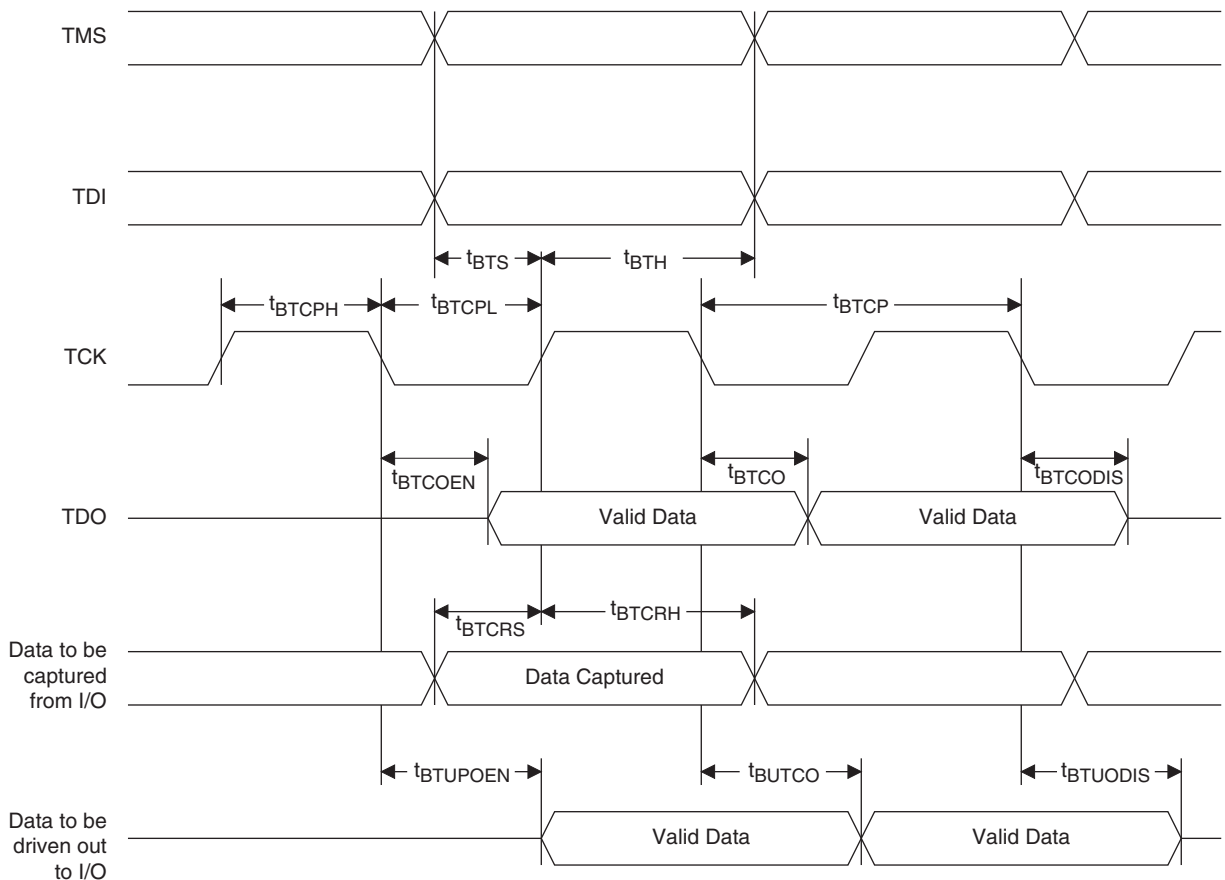


## JTAG Port Timing Specifications

Over Recommended Operating Conditions

Symbol	Parameter	Min	Max	Units
$f_{MAX}$	TCK clock frequency	—	25	MHz
$t_{BTCP}$	TCK [BSCAN] clock pulse width	40	—	ns
$t_{BTCPH}$	TCK [BSCAN] clock pulse width high	20	—	ns
$t_{BTCPL}$	TCK [BSCAN] clock pulse width low	20	—	ns
$t_{BTS}$	TCK [BSCAN] setup time	10	—	ns
$t_{BTH}$	TCK [BSCAN] hold time	8	—	ns
$t_{BTRF}$	TCK [BSCAN] rise/fall time	50	—	mV/ns
$t_{BTCO}$	TAP controller falling edge of clock to valid output	—	10	ns
$t_{BTCODIS}$	TAP controller falling edge of clock to valid disable	—	10	ns
$t_{BTCOEN}$	TAP controller falling edge of clock to valid enable	—	10	ns
$t_{BTCRS}$	BSCAN test capture register setup time	8	—	ns
$t_{BTCRH}$	BSCAN test capture register hold time	25	—	ns
$t_{BUTCO}$	BSCAN test update register, falling edge of clock to valid output	—	25	ns
$t_{BTUODIS}$	BSCAN test update register, falling edge of clock to valid disable	—	25	ns
$t_{BTUPOEN}$	BSCAN test update register, falling edge of clock to valid enable	—	25	ns

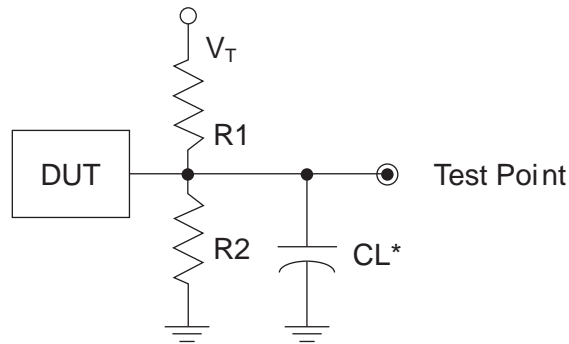
Figure 3-29. JTAG Port Timing Waveforms



## Switching Test Conditions

Figure 3-30 shows the output test load that is used for AC testing. The specific values for resistance, capacitance, voltage, and other test conditions are shown in Table 3-23.

**Figure 3-30. Output Test Load, LVTTTL and LVCMOS Standards**



\*CL Includes Test Fixture and Probe Capacitance

**Table 3-23. Test Fixture Required Components, Non-Terminated Interfaces**

Test Condition	R <sub>1</sub>	R <sub>2</sub>	C <sub>L</sub>	Timing Ref.	V <sub>T</sub>
LVTTTL and other LVCMOS settings (L -> H, H -> L)	∞	∞	0pF	LVCMOS 3.3 = 1.5V	—
				LVCMOS 2.5 = V <sub>CCIO</sub> /2	—
				LVCMOS 1.8 = V <sub>CCIO</sub> /2	—
				LVCMOS 1.5 = V <sub>CCIO</sub> /2	—
				LVCMOS 1.2 = V <sub>CCIO</sub> /2	—
LVCMOS 2.5 I/O (Z -> H)	∞	1MΩ	0pF	V <sub>CCIO</sub> /2	—
LVCMOS 2.5 I/O (Z -> L)	1MΩ	∞	0pF	V <sub>CCIO</sub> /2	V <sub>CCIO</sub>
LVCMOS 2.5 I/O (H -> Z)	∞	100	0pF	V <sub>OH</sub> - 0.10	—
LVCMOS 2.5 I/O (L -> Z)	100	∞	0pF	V <sub>OL</sub> + 0.10	V <sub>CCIO</sub>

Note: Output test conditions for all other interfaces are determined by the respective standards.

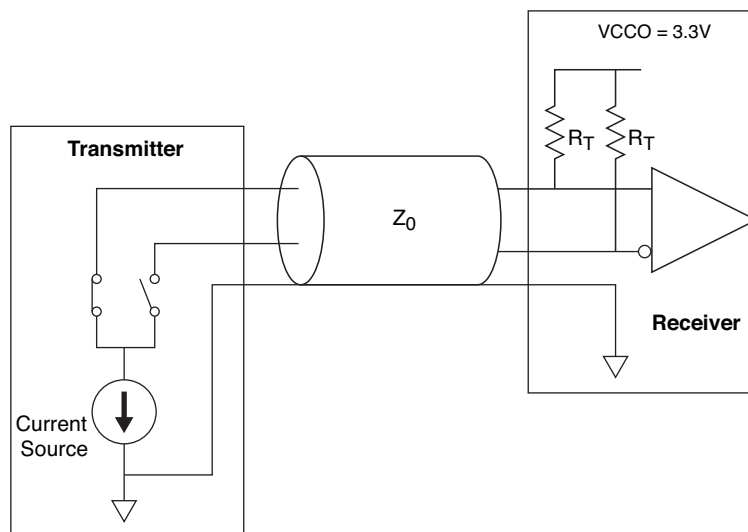
## sys/I/O Differential Electrical Characteristics

### Transition Reduced LVDS (TRLVDS DC Specification)

Over Recommended Operating Conditions

Symbol	Description	Min.	Nom.	Max.	Units
$V_{CCO}$	Driver supply voltage (+/- 5%)	3.14	3.3	3.47	V
$V_{ID}$	Input differential voltage	150	—	1200	mV
$V_{ICM}$	Input common mode voltage	3	—	3.265	V
$V_{CCO}$	Termination supply voltage	3.14	3.3	3.47	V
$R_T$	Termination resistance (off-chip)	45	50	55	Ohms

Note: LatticeECP3 only supports the TRLVDS receiver.



## Mini LVDS

Over Recommended Operating Conditions

Parameter Symbol	Description	Min.	Typ.	Max.	Units
$Z_O$	Single-ended PCB trace impedance	30	50	75	ohms
$R_T$	Differential termination resistance	50	100	150	ohms
$V_{OD}$	Output voltage, differential, $ V_{OP} - V_{OM} $	300	—	600	mV
$V_{OS}$	Output voltage, common mode, $ V_{OP} + V_{OM} /2$	1	1.2	1.4	V
$\Delta V_{OD}$	Change in $V_{OD}$ , between H and L	—	—	50	mV
$\Delta V_{ID}$	Change in $V_{OS}$ , between H and L	—	—	50	mV
$V_{THD}$	Input voltage, differential, $ V_{INP} - V_{INM} $	200	—	600	mV
$V_{CM}$	Input voltage, common mode, $ V_{INP} + V_{INM} /2$	$0.3+(V_{THD}/2)$	—	$2.1-(V_{THD}/2)$	
$T_R, T_F$	Output rise and fall times, 20% to 80%	—	—	550	ps
$T_{ODUTY}$	Output clock duty cycle	40	—	60	%

Note: Data is for 6mA differential current drive. Other differential driver current options are available.

**Point-to-Point LVDS (PPLVDS)**
**Over Recommended Operating Conditions**

Description	Min.	Typ.	Max.	Units
Output driver supply (+/- 5%)	3.14	3.3	3.47	V
	2.25	2.5	2.75	V
Input differential voltage	100	—	400	mV
Input common mode voltage	0.2	—	2.3	V
Output differential voltage	130	—	400	mV
Output common mode voltage	0.5	0.8	1.4	V

**RSDS**
**Over Recommended Operating Conditions**

Parameter Symbol	Description	Min.	Typ.	Max.	Units
$V_{OD}$	Output voltage, differential, $R_T = 100$ ohms	100	200	600	mV
$V_{OS}$	Output voltage, common mode	0.5	1.2	1.5	V
$I_{RSDS}$	Differential driver output current	1	2	6	mA
$V_{THD}$	Input voltage differential	100	—	—	mV
$V_{CM}$	Input common mode voltage	0.3	—	1.5	V
$T_R, T_F$	Output rise and fall times, 20% to 80%	—	500	—	ps
$T_{ODUTY}$	Output clock duty cycle	35	50	65	%

Note: Data is for 2mA drive. Other differential driver current options are available.

### Signal Descriptions

Signal Name	I/O	Description
<b>General Purpose</b>		
P[Edge] [Row/Column Number]_[A/B]	I/O	<p>[Edge] indicates the edge of the device on which the pad is located. Valid edge designations are L (Left), B (Bottom), R (Right), T (Top).</p> <p>[Row/Column Number] indicates the PFU row or the column of the device on which the PIC exists. When Edge is T (Top) or B (Bottom), only need to specify Column Number. When Edge is L (Left) or R (Right), only need to specify Row Number.</p> <p>[A/B] indicates the PIO within the PIC to which the pad is connected. Some of these user-programmable pins are shared with special function pins. These pins, when not used as special purpose pins, can be programmed as I/Os for user logic. During configuration the user-programmable I/Os are tri-stated with an internal pull-up resistor enabled. If any pin is not used (or not bonded to a package pin), it is also tri-stated with an internal pull-up resistor enabled after configuration.</p>
P[Edge][Row Number]E_[A/B/C/D]	I	These general purpose signals are input-only pins and are located near the PLLs.
GSRN	I	Global RESET signal (active low). Any I/O pin can be GSRN.
NC	—	No connect.
RESERVED	—	This pin is reserved and should not be connected to anything on the board.
GND	—	Ground. Dedicated pins.
V <sub>CC</sub>	—	Power supply pins for core logic. Dedicated pins.
V <sub>CCAUX</sub>	—	Auxiliary power supply pin. This dedicated pin powers all the differential and referenced input buffers.
V <sub>CCIOx</sub>	—	Dedicated power supply pins for I/O bank x.
V <sub>CCA</sub>	—	SERDES, transmit, receive, PLL and reference clock buffer power supply. All V <sub>CCA</sub> supply pins must always be powered to the recommended operating voltage range. If no SERDES channels are used, connect V <sub>CCA</sub> to V <sub>CC</sub> .
V <sub>CCPLL</sub> _[LOC]	—	General purpose PLL supply pins where LOC=L (left) or R (right).
V <sub>REF1_x</sub> , V <sub>REF2_x</sub>	—	Reference supply pins for I/O bank x. Pre-determined pins in each bank are assigned as V <sub>REF</sub> inputs. When not used, they may be used as I/O pins.
VTTx	—	Power supply for on-chip termination of I/Os.
XRES <sup>1</sup>	—	10K ohm +/-1% resistor must be connected between this pad and ground.
<b>PLL, DLL and Clock Functions</b>		
[LOC][num]_GPLL[T, C]_IN_[index]	I	General Purpose PLL (GPLL) input pads: LUM, LLM, RUM, RLM, num = row from center, T = true and C = complement, index A,B,C...at each side.
[LOC][num]_GPLL[T, C]_FB_[index]	I	Optional feedback GPLL input pads: LUM, LLM, RUM, RLM, num = row from center, T = true and C = complement, index A,B,C...at each side.
[LOC]0_GDLLT_IN_[index] <sup>2</sup>	I/O	General Purpose DLL (GDLL) input pads where LOC=RUM or LUM, T is True Complement, index is A or B.
[LOC]0_GDLLT_FB_[index] <sup>2</sup>	I/O	Optional feedback GDLL input pads where LOC=RUM or LUM, T is True Complement, index is A or B.
PCLK[T, C][n:0]_[3:0] <sup>2</sup>	I/O	Primary Clock pads, T = true and C = complement, n per side, indexed by bank and 0, 1, 2, 3 within bank.

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at [www.latticesemi.com/legal](http://www.latticesemi.com/legal). All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

**Signal Descriptions (Cont.)**

Signal Name	I/O	Description
[LOC]DQS[num]	I/O	DQ input/output pads: T (top), R (right), B (bottom), L (left), DQS, num = ball function number.
[LOC]DQ[num]	I/O	DQ input/output pads: T (top), R (right), B (bottom), L (left), DQ, associated DQS number.
<b>Test and Programming (Dedicated Pins)</b>		
TMS	I	Test Mode Select input, used to control the 1149.1 state machine. Pull-up is enabled during configuration.
TCK	I	Test Clock input pin, used to clock the 1149.1 state machine. No pull-up enabled.
TDI	I	Test Data in pin. Used to load data into device using 1149.1 state machine. After power-up, this TAP port can be activated for configuration by sending appropriate command. (Note: once a configuration port is selected it is locked. Another configuration port cannot be selected until the power-up sequence). Pull-up is enabled during configuration.
TDO	O	Output pin. Test Data Out pin used to shift data out of a device using 1149.1.
VCCJ	—	Power supply pin for JTAG Test Access Port.
<b>Configuration Pads (Used During sysCONFIG)</b>		
CFG[2:0]	I	Mode pins used to specify configuration mode values latched on rising edge of INITN. During configuration, a pull-up is enabled. These are dedicated pins.
INITN	I/O	Open Drain pin. Indicates the FPGA is ready to be configured. During configuration, a pull-up is enabled. It is a dedicated pin.
PROGRAMN	I	Initiates configuration sequence when asserted low. This pin always has an active pull-up. It is a dedicated pin.
DONE	I/O	Open Drain pin. Indicates that the configuration sequence is complete, and the startup sequence is in progress. It is a dedicated pin.
CCLK	I	Input Configuration Clock for configuring an FPGA in Slave SPI, Serial, and CPU modes. It is a dedicated pin.
MCLK	I/O	Output Configuration Clock for configuring an FPGA in SPI, SPIm, and Master configuration modes.
BUSY/SISPI	O	Parallel configuration mode busy indicator. SPI/SPIm mode data output.
CSN/SN/OEN	I/O	Parallel configuration mode active-low chip select. Slave SPI chip select. Parallel burst Flash output enable.
CS1N/HOLDN/RDY	I	Parallel configuration mode active-low chip select. Slave SPI hold input.
WRITEN	I	Write enable for parallel configuration modes.
DOUT/CSN/CSSPI1N	O	Serial data output. Chip select output. SPI/SPIm mode chip select.
D[0]/SPIFASTN	I/O	sysCONFIG Port Data I/O for Parallel mode. Open drain during configuration. sysCONFIG Port Data I/O for SPI or SPIm. When using the SPI or SPIm mode, this pin should either be tied high or low, must not be left floating. Open drain during configuration.
D1	I/O	Parallel configuration I/O. Open drain during configuration.
D2	I/O	Parallel configuration I/O. Open drain during configuration.
D3/SI	I/O	Parallel configuration I/O. Slave SPI data input. Open drain during configuration.
D4/SO	I/O	Parallel configuration I/O. Slave SPI data output. Open drain during configuration.
D5	I/O	Parallel configuration I/O. Open drain during configuration.
D6/SPID1	I/O	Parallel configuration I/O. SPI/SPIm data input. Open drain during configuration.



## Signal Descriptions (Cont.)

Signal Name	I/O	Description
D7/SPID0	I/O	Parallel configuration I/O. SPI/SPI <sub>m</sub> data input. Open drain during configuration.
DI/CSSPI0N/CEN	I/O	Serial data input for slave serial mode. SPI/SPI <sub>m</sub> mode chip select.
<b>Dedicated SERDES Signals<sup>3</sup></b>		
PCS[Index]_HDINN <sub>m</sub>	I	High-speed input, negative channel <i>m</i>
PCS[Index]_HDOUTN <sub>m</sub>	O	High-speed output, negative channel <i>m</i>
PCS[Index]_REFCLKN	I	Negative Reference Clock Input
PCS[Index]_HDINP <sub>m</sub>	I	High-speed input, positive channel <i>m</i>
PCS[Index]_HDOUTP <sub>m</sub>	O	High-speed output, positive channel <i>m</i>
PCS[Index]_REFCLKP	I	Positive Reference Clock Input
PCS[Index]_VCCOB <sub>m</sub>	—	Output buffer power supply, channel <i>m</i> (1.2V/1.5)
PCS[Index]_VCCIB <sub>m</sub>	—	Input buffer power supply, channel <i>m</i> (1.2V/1.5V)

1. When placing switching I/Os around these critical pins that are designed to supply the device with the proper reference or supply voltage, care must be given.
2. These pins are dedicated inputs or can be used as general purpose I/O.
3. *m* defines the associated channel in the quad.

**PICs and DDR Data (DQ) Pins Associated with the DDR Strobe (DQS) Pin**

PICs Associated with DQS Strobe	PIO Within PIC	DDR Strobe (DQS) and Data (DQ) Pins
<b>For Left and Right Edges of the Device</b>		
P[Edge] [n-3]	A	DQ
	B	DQ
P[Edge] [n-2]	A	DQ
	B	DQ
P[Edge] [n-1]	A	DQ
	B	DQ
P[Edge] [n]	A	[Edge]DQSn
	B	DQ
P[Edge] [n+1]	A	DQ
	B	DQ
P[Edge] [n+2]	A	DQ
	B	DQ
<b>For Top Edge of the Device</b>		
P[Edge] [n-3]	A	DQ
	B	DQ
P[Edge] [n-2]	A	DQ
	B	DQ
P[Edge] [n-1]	A	DQ
	B	DQ
P[Edge] [n]	A	[Edge]DQSn
	B	DQ
P[Edge] [n+1]	A	DQ
	B	DQ
P[Edge] [n+2]	A	DQ
	B	DQ

Note: "n" is a row PIC number.

## Pin Information Summary

Pin Information Summary		ECP3-17EA			ECP3-35EA			ECP3-70EA		
Pin Type		256 ftBGA	328 csBGA	484 fpBGA	256 ftBGA	484 fpBGA	672 fpBGA	484 fpBGA	672 fpBGA	1156 fpBGA
General Purpose Inputs/Outputs per Bank	Bank 0	26	20	36	26	42	48	42	60	86
	Bank 1	14	10	24	14	36	36	36	48	78
	Bank 2	6	7	12	6	24	24	24	34	36
	Bank 3	18	12	44	16	54	59	54	59	86
	Bank 6	20	11	44	18	63	61	63	67	86
	Bank 7	19	26	32	19	36	42	36	48	54
	Bank 8	24	24	24	24	24	24	24	24	24
General Purpose Inputs per Bank	Bank 0	0	0	0	0	0	0	0	0	0
	Bank 1	0	0	0	0	0	0	0	0	0
	Bank 2	2	2	2	2	4	4	4	8	8
	Bank 3	0	0	0	2	4	4	4	12	12
	Bank 6	0	0	0	2	4	4	4	12	12
	Bank 7	4	4	4	4	4	4	4	8	8
	Bank 8	0	0	0	0	0	0	0	0	0
General Purpose Out- puts per Bank	Bank 0	0	0	0	0	0	0	0	0	0
	Bank 1	0	0	0	0	0	0	0	0	0
	Bank 2	0	0	0	0	0	0	0	0	0
	Bank 3	0	0	0	0	0	0	0	0	0
	Bank 6	0	0	0	0	0	0	0	0	0
	Bank 7	0	0	0	0	0	0	0	0	0
	Bank 8	0	0	0	0	0	0	0	0	0
Total Single-Ended User I/O		133	116	222	133	295	310	295	380	490
VCC		6	16	16	6	16	32	16	32	32
VCCAUX		4	5	8	4	8	12	8	12	16
VTT		4	7	4	4	4	4	4	4	8
VCCA		4	6	4	4	4	8	4	8	16
VCCPLL		2	2	4	2	4	4	4	4	4
VCCIO	Bank 0	2	3	2	2	2	4	2	4	4
	Bank 1	2	3	2	2	2	4	2	4	4
	Bank 2	2	2	2	2	2	4	2	4	4
	Bank 3	2	3	2	2	2	4	2	4	4
	Bank 6	2	3	2	2	2	4	2	4	4
	Bank 7	2	3	2	2	2	4	2	4	4
	Bank 8	1	2	2	1	2	2	2	2	2
VCCJ		1	1	1	1	1	1	1	1	1
TAP		4	4	4	4	4	4	4	4	4
GND, GNDIO		51	126	98	51	98	139	98	139	233
NC		0	0	73	0	0	96	0	0	238
Reserved <sup>1</sup>		0	0	2	0	2	2	2	2	2
SERDES		26	18	26	26	26	26	26	52	78
Miscellaneous Pins		8	8	8	8	8	8	8	8	8
Total Bonded Pins		256	328	484	256	484	672	484	672	1156

**Pin Information Summary (Cont.)**

Pin Information Summary		ECP3-17EA			ECP3-35EA		
Pin Type		256 ftBGA	328 csBGA	484 fpBGA	256 ftBGA	484 fpBGA	672 fpBGA
Emulated Differential I/O per Bank	Bank 0	13	10	18	13	21	24
	Bank 1	7	5	12	7	18	18
	Bank 2	2	2	4	1	8	8
	Bank 3	4	2	13	5	20	19
	Bank 6	5	1	13	6	22	20
	Bank 7	6	9	10	6	11	13
	Bank 8	12	12	12	12	12	12
Highspeed Differential I/O per Bank	Bank 0	0	0	0	0	0	0
	Bank 1	0	0	0	0	0	0
	Bank 2	2	2	3	3	6	6
	Bank 3	5	4	9	4	9	12
	Bank 6	5	4	9	4	11	12
	Bank 7	5	6	8	5	9	10
	Bank 8	0	0	0	0	0	0
Total Single Ended/ Total Differential I/O per Bank	Bank 0	26/13	20/10	36/18	26/13	42/21	48/24
	Bank 1	14/7	10/5	24/12	14/7	36/18	36/18
	Bank 2	8/4	9/4	14/7	8/4	28/14	28/14
	Bank 3	18/9	12/6	44/22	18/9	58/29	63/31
	Bank 6	20/10	11/5	44/22	20/10	67/33	65/32
	Bank 7	23/11	30/15	36/18	23/11	40/20	46/23
	Bank 8	24/12	24/12	24/12	24/12	24/12	24/12
DDR Groups Bonded per Bank <sup>2</sup>	Bank 0	2	1	3	2	3	4
	Bank 1	1	0	2	1	3	3
	Bank 2	0	0	1	0	2	2
	Bank 3	1	0	3	1	3	4
	Bank 6	1	0	3	1	4	4
	Bank 7	1	2	2	1	3	3
	Configuration Bank 8	0	0	0	0	0	0
SERDES Quads		1	1	1	1	1	1

1. These pins must remain floating on the board.
2. Some DQS groups may not support DQS-12. Refer to the device pinout (.csv) file.

**Pin Information Summary (Cont.)**

Pin Information Summary		ECP3-70EA		
Pin Type		484 fpBGA	672 fpBGA	1156 fpBGA
Emulated Differential I/O per Bank	Bank 0	21	30	43
	Bank 1	18	24	39
	Bank 2	8	12	13
	Bank 3	20	23	33
	Bank 6	22	25	33
	Bank 7	11	16	18
	Bank 8	12	12	12
High-Speed Differential I/O per Bank	Bank 0	0	0	0
	Bank 1	0	0	0
	Bank 2	6	9	9
	Bank 3	9	12	16
	Bank 6	11	14	16
	Bank 7	9	12	13
	Bank 8	0	0	0
Total Single-Ended/ Total Differential I/O per Bank	Bank 0	42/21	60/30	86/43
	Bank 1	36/18	48/24	78/39
	Bank 2	28/14	42/21	44/22
	Bank 3	58/29	71/35	98/49
	Bank 6	67/33	78/39	98/49
	Bank 7	40/20	56/28	62/31
	Bank 8	24/12	24/12	24/12
DDR Groups Bonded per Bank <sup>1</sup>	Bank 0	3	5	7
	Bank 1	3	4	7
	Bank 2	2	3	3
	Bank 3	3	4	5
	Bank 6	4	4	5
	Bank 7	3	4	4
	Configuration Bank 8	0	0	0
SERDES Quads		1	2	3

1. Some DQS groups may not support DQS-12. Refer to the device pinout (.csv) file.

**Pin Information Summary (Cont.)**

Pin Information Summary		ECP3-95EA			ECP3-150EA	
Pin Type		484 fpBGA	672 fpBGA	1156 fpBGA	672 fpBGA	1156 fpBGA
General Purpose Inputs/Outputs per bank	Bank 0	42	60	86	60	94
	Bank 1	36	48	78	48	86
	Bank 2	24	34	36	34	58
	Bank 3	54	59	86	59	104
	Bank 6	63	67	86	67	104
	Bank 7	36	48	54	48	76
	Bank 8	24	24	24	24	24
General Purpose Inputs per Bank	Bank 0	0	0	0	0	0
	Bank 1	0	0	0	0	0
	Bank 2	4	8	8	8	8
	Bank 3	4	12	12	12	12
	Bank 6	4	12	12	12	12
	Bank 7	4	8	8	8	8
	Bank 8	0	0	0	0	0
General Purpose Outputs per Bank	Bank 0	0	0	0	0	0
	Bank 1	0	0	0	0	0
	Bank 2	0	0	0	0	0
	Bank 3	0	0	0	0	0
	Bank 6	0	0	0	0	0
	Bank 7	0	0	0	0	0
	Bank 8	0	0	0	0	0
Total Single-Ended User I/O		295	380	490	380	586
VCC		16	32	32	32	32
VCCAUX		8	12	16	12	16
VTT		4	4	8	4	8
VCCA		4	8	16	8	16
VCCPLL		4	4	4	4	4
VCCIO	Bank 0	2	4	4	4	4
	Bank 1	2	4	4	4	4
	Bank 2	2	4	4	4	4
	Bank 3	2	4	4	4	4
	Bank 6	2	4	4	4	4
	Bank 7	2	4	4	4	4
	Bank 8	2	2	2	2	2
VCCJ		1	1	1	1	1
TAP		4	4	4	4	4
GND, GNDIO		98	139	233	139	233
NC		0	0	238	0	116
Reserved <sup>1</sup>		2	2	2	2	2
SERDES		26	52	78	52	104
Miscellaneous Pins		8	8	8	8	8
Total Bonded Pins		484	672	1156	672	1156

**Pin Information Summary (Cont.)**

Pin Information Summary		ECP3-95EA			ECP3-150EA	
Pin Type		484 fpBGA	672 fpBGA	1156 fpBGA	672 fpBGA	1156 fpBGA
Emulated Differential I/O per Bank	Bank 0	21	30	43	30	47
	Bank 1	18	24	39	24	43
	Bank 2	8	12	13	12	18
	Bank 3	20	23	33	23	37
	Bank 6	22	25	33	25	37
	Bank 7	11	16	18	16	24
	Bank 8	12	12	12	12	12
Highspeed Differential I/O per Bank	Bank 0	0	0	0	0	0
	Bank 1	0	0	0	0	0
	Bank 2	6	9	9	9	15
	Bank 3	9	12	16	12	21
	Bank 6	11	14	16	14	21
	Bank 7	9	12	13	12	18
	Bank 8	0	0	0	0	0
Total Single Ended/ Total Differential I/O per Bank	Bank 0	42/21	60/30	86/43	60/30	94/47
	Bank 1	36/18	48/24	78/39	48/24	86/43
	Bank 2	28/14	42/21	44/22	42/21	66/33
	Bank 3	58/29	71/35	98/49	71/35	116/58
	Bank 6	67/33	78/39	98/49	78/39	116/58
	Bank 7	40/20	56/28	62/31	56/28	84/42
	Bank 8	24/12	24/12	24/12	24/12	24/12
DDR Groups Bonded per Bank	Bank 0	3	5	7	5	7
	Bank 1	3	4	7	4	7
	Bank 2	2	3	3	3	4
	Bank 3	3	4	5	4	7
	Bank 6	4	4	5	4	7
	Bank 7	3	4	4	4	6
	Configuration Bank8	0	0	0	0	0
SERDES Quads		1	2	3	2	4

1. These pins must remain floating on the board.

## Package Pinout Information

Package pinout information can be found under “Data Sheets” on the LatticeECP3 product pages on the Lattice website at [www.latticesemi.com/products/fpga/ecp3](http://www.latticesemi.com/products/fpga/ecp3) and in the Diamond or ispLEVER software tools. To create pinout information from within ispLEVER Design Planner, select **Tools > Spreadsheet View**. Then select **Select File > Export** and choose a type of output file. To create a pin information file from within Diamond select **Tools > Spreadsheet View** or **Tools > Package View**; then, select **File > Export** and choose a type of output file. See Diamond or ispLEVER Help for more information.

## Thermal Management

Thermal management is recommended as part of any sound FPGA design methodology. To assess the thermal characteristics of a system, Lattice specifies a maximum allowable junction temperature in all device data sheets. Designers must complete a thermal analysis of their specific design to ensure that the device and package do not exceed the junction temperature limits. Refer to the Thermal Management document to find the device/package specific thermal values.

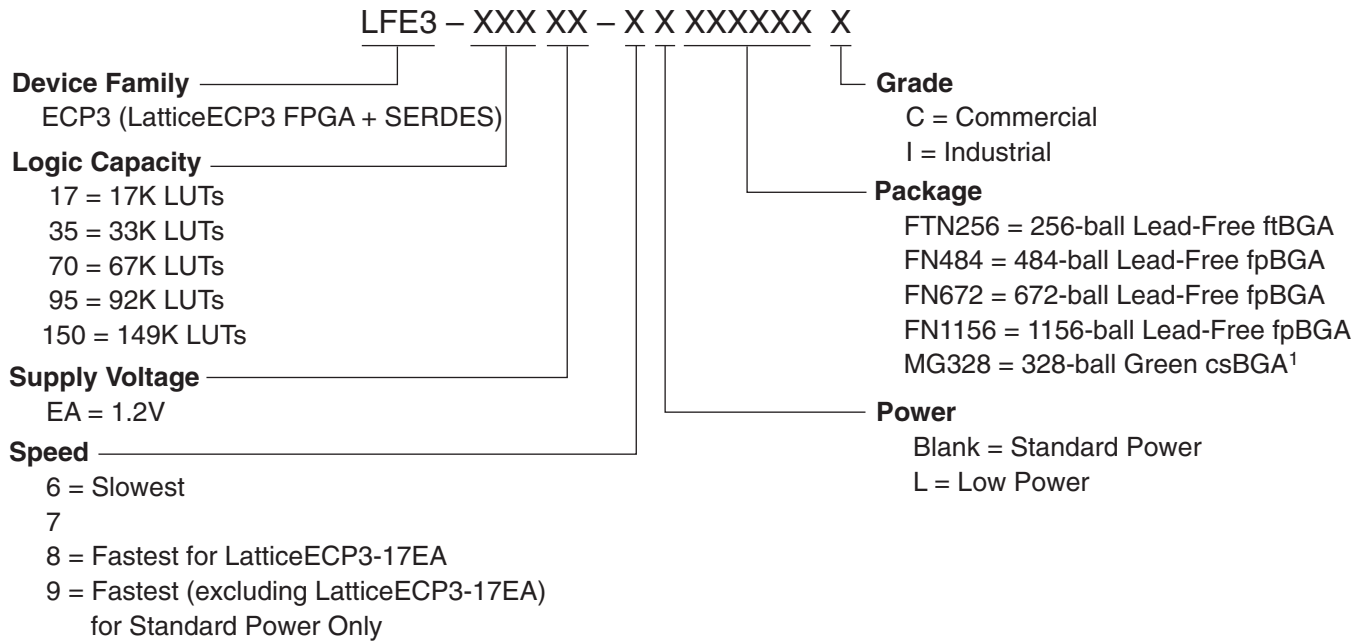
## For Further Information

For further information regarding Thermal Management, refer to the following:

- [Thermal Management](#) document
- TN1181, [Power Consumption and Management for LatticeECP3 Devices](#)
- Power Calculator tool included with the Diamond and ispLEVER design tools, or as a standalone download from [www.latticesemi.com/software](http://www.latticesemi.com/software)



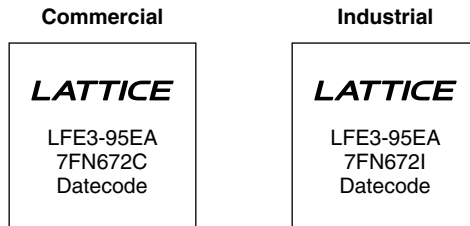
### LatticeECP3 Part Number Description



1. Green = Halogen free and lead free.

### Ordering Information

LatticeECP3 devices have top-side markings, for commercial and industrial grades, as shown below:



Note: See [PCN 05A-12](#) for information regarding a change to the top-side mark logo.

**LatticeECP3 Devices, Green and Lead-Free Packaging**

The following devices may have associated errata. Specific devices with associated errata will be notated with a footnote.

**Commercial**

Part Number	Voltage	Grade	Power	Package <sup>1</sup>	Pins	Temp.	LUTs (K)
LFE3-17EA-6FTN256C	1.2V	-6	STD	Lead-Free ftBGA	256	COM	17
LFE3-17EA-7FTN256C	1.2V	-7	STD	Lead-Free ftBGA	256	COM	17
LFE3-17EA-8FTN256C	1.2V	-8	STD	Lead-Free ftBGA	256	COM	17
LFE3-17EA-6LFTN256C	1.2V	-6	LOW	Lead-Free ftBGA	256	COM	17
LFE3-17EA-7LFTN256C	1.2V	-7	LOW	Lead-Free ftBGA	256	COM	17
LFE3-17EA-8LFTN256C	1.2V	-8	LOW	Lead-Free ftBGA	256	COM	17
LFE3-17EA-6MG328C	1.2V	-6	STD	Green csBGA	328	COM	17
LFE3-17EA-7MG328C	1.2V	-7	STD	Green csBGA	328	COM	17
LFE3-17EA-8MG328C	1.2V	-8	STD	Green csBGA	328	COM	17
LFE3-17EA-6LMG328C	1.2V	-6	LOW	Green csBGA	328	COM	17
LFE3-17EA-7LMG328C	1.2V	-7	LOW	Green csBGA	328	COM	17
LFE3-17EA-8LMG328C	1.2V	-8	LOW	Green csBGA	328	COM	17
LFE3-17EA-6FN484C	1.2V	-6	STD	Lead-Free fpBGA	484	COM	17
LFE3-17EA-7FN484C	1.2V	-7	STD	Lead-Free fpBGA	484	COM	17
LFE3-17EA-8FN484C	1.2V	-8	STD	Lead-Free fpBGA	484	COM	17
LFE3-17EA-6LFN484C	1.2V	-6	LOW	Lead-Free fpBGA	484	COM	17
LFE3-17EA-7LFN484C	1.2V	-7	LOW	Lead-Free fpBGA	484	COM	17
LFE3-17EA-8LFN484C	1.2V	-8	LOW	Lead-Free fpBGA	484	COM	17

1. Green = Halogen free and lead free.

Part Number	Voltage	Grade	Power	Package	Pins	Temp.	LUTs (K)
LFE3-35EA-6FTN256C	1.2V	-6	STD	Lead-Free ftBGA	256	COM	33
LFE3-35EA-7FTN256C	1.2V	-7	STD	Lead-Free ftBGA	256	COM	33
LFE3-35EA-8FTN256C	1.2V	-8	STD	Lead-Free ftBGA	256	COM	33
LFE3-35EA-9FTN256C	1.2V	-9	STD	Lead-Free ftBGA	256	COM	33
LFE3-35EA-6LFTN256C	1.2V	-6	LOW	Lead-Free ftBGA	256	COM	33
LFE3-35EA-7LFTN256C	1.2V	-7	LOW	Lead-Free ftBGA	256	COM	33
LFE3-35EA-8LFTN256C	1.2V	-8	LOW	Lead-Free ftBGA	256	COM	33
LFE3-35EA-6FN484C	1.2V	-6	STD	Lead-Free fpBGA	484	COM	33
LFE3-35EA-7FN484C	1.2V	-7	STD	Lead-Free fpBGA	484	COM	33
LFE3-35EA-8FN484C	1.2V	-8	STD	Lead-Free fpBGA	484	COM	33
LFE3-35EA-9FN484C	1.2V	-9	STD	Lead-Free fpBGA	484	COM	33
LFE3-35EA-6LFN484C	1.2V	-6	LOW	Lead-Free fpBGA	484	COM	33
LFE3-35EA-7LFN484C	1.2V	-7	LOW	Lead-Free fpBGA	484	COM	33
LFE3-35EA-8LFN484C	1.2V	-8	LOW	Lead-Free fpBGA	484	COM	33
LFE3-35EA-6FN672C	1.2V	-6	STD	Lead-Free fpBGA	672	COM	33
LFE3-35EA-7FN672C	1.2V	-7	STD	Lead-Free fpBGA	672	COM	33
LFE3-35EA-8FN672C	1.2V	-8	STD	Lead-Free fpBGA	672	COM	33
LFE3-35EA-9FN672C	1.2V	-9	STD	Lead-Free fpBGA	672	COM	33
LFE3-35EA-6LFN672C	1.2V	-6	LOW	Lead-Free fpBGA	672	COM	33
LFE3-35EA-7LFN672C	1.2V	-7	LOW	Lead-Free fpBGA	672	COM	33
LFE3-35EA-8LFN672C	1.2V	-8	LOW	Lead-Free fpBGA	672	COM	33

Part Number	Voltage	Grade	Power	Package	Pins	Temp.	LUTs (K)
LFE3-70EA-6FN484C	1.2V	-6	STD	Lead-Free fpBGA	484	COM	67
LFE3-70EA-7FN484C	1.2V	-7	STD	Lead-Free fpBGA	484	COM	67
LFE3-70EA-8FN484C	1.2V	-8	STD	Lead-Free fpBGA	484	COM	67
LFE3-70EA-9FN484C	1.2V	-9	STD	Lead-Free fpBGA	484	COM	67
LFE3-70EA-6LFN484C	1.2V	-6	LOW	Lead-Free fpBGA	484	COM	67
LFE3-70EA-7LFN484C	1.2V	-7	LOW	Lead-Free fpBGA	484	COM	67
LFE3-70EA-8LFN484C	1.2V	-8	LOW	Lead-Free fpBGA	484	COM	67
LFE3-70EA-6FN672C	1.2V	-6	STD	Lead-Free fpBGA	672	COM	67
LFE3-70EA-7FN672C	1.2V	-7	STD	Lead-Free fpBGA	672	COM	67
LFE3-70EA-8FN672C	1.2V	-8	STD	Lead-Free fpBGA	672	COM	67
LFE3-70EA-9FN672C	1.2V	-9	STD	Lead-Free fpBGA	672	COM	67
LFE3-70EA-6LFN672C	1.2V	-6	LOW	Lead-Free fpBGA	672	COM	67
LFE3-70EA-7LFN672C	1.2V	-7	LOW	Lead-Free fpBGA	672	COM	67
LFE3-70EA-8LFN672C	1.2V	-8	LOW	Lead-Free fpBGA	672	COM	67
LFE3-70EA-6FN1156C	1.2V	-6	STD	Lead-Free fpBGA	1156	COM	67
LFE3-70EA-7FN1156C	1.2V	-7	STD	Lead-Free fpBGA	1156	COM	67
LFE3-70EA-8FN1156C	1.2V	-8	STD	Lead-Free fpBGA	1156	COM	67
LFE3-70EA-9FN1156C	1.2V	-9	STD	Lead-Free fpBGA	1156	COM	67
LFE3-70EA-6LFN1156C	1.2V	-6	LOW	Lead-Free fpBGA	1156	COM	67
LFE3-70EA-7LFN1156C	1.2V	-7	LOW	Lead-Free fpBGA	1156	COM	67
LFE3-70EA-8LFN1156C	1.2V	-8	LOW	Lead-Free fpBGA	1156	COM	67

Part Number	Voltage	Grade	Power	Package	Pins	Temp.	LUTs (K)
LFE3-95EA-6FN484C	1.2V	-6	STD	Lead-Free fpBGA	484	COM	92
LFE3-95EA-7FN484C	1.2V	-7	STD	Lead-Free fpBGA	484	COM	92
LFE3-95EA-8FN484C	1.2V	-8	STD	Lead-Free fpBGA	484	COM	92
LFE3-95EA-9FN484C	1.2V	-9	STD	Lead-Free fpBGA	484	COM	92
LFE3-95EA-6LFN484C	1.2V	-6	LOW	Lead-Free fpBGA	484	COM	92
LFE3-95EA-7LFN484C	1.2V	-7	LOW	Lead-Free fpBGA	484	COM	92
LFE3-95EA-8LFN484C	1.2V	-8	LOW	Lead-Free fpBGA	484	COM	92
LFE3-95EA-6FN672C	1.2V	-6	STD	Lead-Free fpBGA	672	COM	92
LFE3-95EA-7FN672C	1.2V	-7	STD	Lead-Free fpBGA	672	COM	92
LFE3-95EA-8FN672C	1.2V	-8	STD	Lead-Free fpBGA	672	COM	92
LFE3-95EA-9FN672C	1.2V	-9	STD	Lead-Free fpBGA	672	COM	92
LFE3-95EA-6LFN672C	1.2V	-6	LOW	Lead-Free fpBGA	672	COM	92
LFE3-95EA-7LFN672C	1.2V	-7	LOW	Lead-Free fpBGA	672	COM	92
LFE3-95EA-8LFN672C	1.2V	-8	LOW	Lead-Free fpBGA	672	COM	92
LFE3-95EA-6FN1156C	1.2V	-6	STD	Lead-Free fpBGA	1156	COM	92
LFE3-95EA-7FN1156C	1.2V	-7	STD	Lead-Free fpBGA	1156	COM	92
LFE3-95EA-8FN1156C	1.2V	-8	STD	Lead-Free fpBGA	1156	COM	92
LFE3-95EA-9FN1156C	1.2V	-9	STD	Lead-Free fpBGA	1156	COM	92
LFE3-95EA-6LFN1156C	1.2V	-6	LOW	Lead-Free fpBGA	1156	COM	92
LFE3-95EA-7LFN1156C	1.2V	-7	LOW	Lead-Free fpBGA	1156	COM	92
LFE3-95EA-8LFN1156C	1.2V	-8	LOW	Lead-Free fpBGA	1156	COM	92

Part Number	Voltage	Grade	Power	Package	Pins	Temp.	LUTs (K)
LFE3-150EA-6FN672C	1.2V	-6	STD	Lead-Free fpBGA	672	COM	149
LFE3-150EA-7FN672C	1.2V	-7	STD	Lead-Free fpBGA	672	COM	149
LFE3-150EA-8FN672C	1.2V	-8	STD	Lead-Free fpBGA	672	COM	149
LFE3-150EA-9FN672C	1.2V	-9	STD	Lead-Free fpBGA	672	COM	149
LFE3-150EA-6LFN672C	1.2V	-6	LOW	Lead-Free fpBGA	672	COM	149
LFE3-150EA-7LFN672C	1.2V	-7	LOW	Lead-Free fpBGA	672	COM	149
LFE3-150EA-8LFN672C	1.2V	-8	LOW	Lead-Free fpBGA	672	COM	149
LFE3-150EA-6FN1156C	1.2V	-6	STD	Lead-Free fpBGA	1156	COM	149
LFE3-150EA-7FN1156C	1.2V	-7	STD	Lead-Free fpBGA	1156	COM	149
LFE3-150EA-8FN1156C	1.2V	-8	STD	Lead-Free fpBGA	1156	COM	149
LFE3-150EA-9FN1156C	1.2V	-9	STD	Lead-Free fpBGA	1156	COM	149
LFE3-150EA-6LFN1156C	1.2V	-6	LOW	Lead-Free fpBGA	1156	COM	149
LFE3-150EA-7LFN1156C	1.2V	-7	LOW	Lead-Free fpBGA	1156	COM	149
LFE3-150EA-8LFN1156C	1.2V	-8	LOW	Lead-Free fpBGA	1156	COM	149

Part Number	Voltage	Grade	Power	Package	Pins	Temp.	LUTs (K)
LFE3-150EA-6FN672CTW <sup>1</sup>	1.2V	-6	STD	Lead-Free fpBGA	672	COM	149
LFE3-150EA-7FN672CTW <sup>1</sup>	1.2V	-7	STD	Lead-Free fpBGA	672	COM	149
LFE3-150EA-8FN672CTW <sup>1</sup>	1.2V	-8	STD	Lead-Free fpBGA	672	COM	149
LFE3-150EA-6FN1156CTW <sup>1</sup>	1.2V	-6	STD	Lead-Free fpBGA	1156	COM	149
LFE3-150EA-7FN1156CTW <sup>1</sup>	1.2V	-7	STD	Lead-Free fpBGA	1156	COM	149
LFE3-150EA-8FN1156CTW <sup>1</sup>	1.2V	-8	STD	Lead-Free fpBGA	1156	COM	149

1. Note: Specifications for the LFE3-150EA-*spFNpkg*CTW and LFE3-150EA-*spFNpkg*lTW devices, (where *sp* is the speed and *pkg* is the package), are the same as the LFE3-150EA-*spFNpkg*C and LFE3-150EA-*spFNpkg*l devices respectively, except as specified below.

- The CTC (Clock Tolerance Circuit) inside the SERDES hard PCS in the TW device is not functional but it can be bypassed and implemented in soft IP.
- The SERDES XRES pin on the TW device passes CDM testing at 250V.

**Industrial**

The following devices may have associated errata. Specific devices with associated errata will be notated with a footnote.

Part Number	Voltage	Grade	Power	Package <sup>1</sup>	Pins	Temp.	LUTs (K)
LFE3-17EA-6FTN256I	1.2V	-6	STD	Lead-Free ftBGA	256	IND	17
LFE3-17EA-7FTN256I	1.2V	-7	STD	Lead-Free ftBGA	256	IND	17
LFE3-17EA-8FTN256I	1.2V	-8	STD	Lead-Free ftBGA	256	IND	17
LFE3-17EA-6LFTN256I	1.2V	-6	LOW	Lead-Free ftBGA	256	IND	17
LFE3-17EA-7LFTN256I	1.2V	-7	LOW	Lead-Free ftBGA	256	IND	17
LFE3-17EA-8LFTN256I	1.2V	-8	LOW	Lead-Free ftBGA	256	IND	17
LFE3-17EA-6MG328I	1.2V	-6	STD	Lead-Free csBGA	328	IND	17
LFE3-17EA-7MG328I	1.2V	-7	STD	Lead-Free csBGA	328	IND	17
LFE3-17EA-8MG328I	1.2V	-8	STD	Lead-Free csBGA	328	IND	17
LFE3-17EA-6LMG328I	1.2V	-6	LOW	Green csBGA	328	IND	17
LFE3-17EA-7LMG328I	1.2V	-7	LOW	Green csBGA	328	IND	17
LFE3-17EA-8LMG328I	1.2V	-8	LOW	Green csBGA	328	IND	17
LFE3-17EA-6FN484I	1.2V	-6	STD	Lead-Free fpBGA	484	IND	17
LFE3-17EA-7FN484I	1.2V	-7	STD	Lead-Free fpBGA	484	IND	17
LFE3-17EA-8FN484I	1.2V	-8	STD	Lead-Free fpBGA	484	IND	17
LFE3-17EA-6LFN484I	1.2V	-6	LOW	Lead-Free fpBGA	484	IND	17
LFE3-17EA-7LFN484I	1.2V	-7	LOW	Lead-Free fpBGA	484	IND	17
LFE3-17EA-8LFN484I	1.2V	-8	LOW	Lead-Free fpBGA	484	IND	17

1. Green = Halogen free and lead free.

Part Number	Voltage	Grade	Power	Package	Pins	Temp.	LUTs (K)
LFE3-35EA-6FTN256I	1.2V	-6	STD	Lead-Free ftBGA	256	IND	33
LFE3-35EA-7FTN256I	1.2V	-7	STD	Lead-Free ftBGA	256	IND	33
LFE3-35EA-8FTN256I	1.2V	-8	STD	Lead-Free ftBGA	256	IND	33
LFE3-35EA-9FTN256I	1.2V	-9	STD	Lead-Free ftBGA	256	IND	33
LFE3-35EA-6LFTN256I	1.2V	-6	LOW	Lead-Free ftBGA	256	IND	33
LFE3-35EA-7LFTN256I	1.2V	-7	LOW	Lead-Free ftBGA	256	IND	33
LFE3-35EA-8LFTN256I	1.2V	-8	LOW	Lead-Free ftBGA	256	IND	33
LFE3-35EA-6FN484I	1.2V	-6	STD	Lead-Free fpBGA	484	IND	33
LFE3-35EA-7FN484I	1.2V	-7	STD	Lead-Free fpBGA	484	IND	33
LFE3-35EA-8FN484I	1.2V	-8	STD	Lead-Free fpBGA	484	IND	33
LFE3-35EA-9FN484I	1.2V	-9	STD	Lead-Free fpBGA	484	IND	33
LFE3-35EA-6LFN484I	1.2V	-6	LOW	Lead-Free fpBGA	484	IND	33
LFE3-35EA-7LFN484I	1.2V	-7	LOW	Lead-Free fpBGA	484	IND	33
LFE3-35EA-8LFN484I	1.2V	-8	LOW	Lead-Free fpBGA	484	IND	33
LFE3-35EA-6FN672I	1.2V	-6	STD	Lead-Free fpBGA	672	IND	33
LFE3-35EA-7FN672I	1.2V	-7	STD	Lead-Free fpBGA	672	IND	33
LFE3-35EA-8FN672I	1.2V	-8	STD	Lead-Free fpBGA	672	IND	33
LFE3-35EA-9FN672I	1.2V	-9	STD	Lead-Free fpBGA	672	IND	33
LFE3-35EA-6LFN672I	1.2V	-6	LOW	Lead-Free fpBGA	672	IND	33
LFE3-35EA-7LFN672I	1.2V	-7	LOW	Lead-Free fpBGA	672	IND	33
LFE3-35EA-8LFN672I	1.2V	-8	LOW	Lead-Free fpBGA	672	IND	33



Part Number	Voltage	Grade	Power	Package	Pins	Temp.	LUTs (K)
LFE3-70EA-6FN484I	1.2V	-6	STD	Lead-Free fpBGA	484	IND	67
LFE3-70EA-7FN484I	1.2V	-7	STD	Lead-Free fpBGA	484	IND	67
LFE3-70EA-8FN484I	1.2V	-8	STD	Lead-Free fpBGA	484	IND	67
LFE3-70EA-9FN484I	1.2V	-9	STD	Lead-Free fpBGA	484	IND	67
LFE3-70EA-6LFN484I	1.2V	-6	LOW	Lead-Free fpBGA	484	IND	67
LFE3-70EA-7LFN484I	1.2V	-7	LOW	Lead-Free fpBGA	484	IND	67
LFE3-70EA-8LFN484I	1.2V	-8	LOW	Lead-Free fpBGA	484	IND	67
LFE3-70EA-6FN672I	1.2V	-6	STD	Lead-Free fpBGA	672	IND	67
LFE3-70EA-7FN672I	1.2V	-7	STD	Lead-Free fpBGA	672	IND	67
LFE3-70EA-8FN672I	1.2V	-8	STD	Lead-Free fpBGA	672	IND	67
LFE3-70EA-9FN672I	1.2V	-9	STD	Lead-Free fpBGA	672	IND	67
LFE3-70EA-6LFN672I	1.2V	-6	LOW	Lead-Free fpBGA	672	IND	67
LFE3-70EA-7LFN672I	1.2V	-7	LOW	Lead-Free fpBGA	672	IND	67
LFE3-70EA-8LFN672I	1.2V	-8	LOW	Lead-Free fpBGA	672	IND	67
LFE3-70EA-6FN1156I	1.2V	-6	STD	Lead-Free fpBGA	1156	IND	67
LFE3-70EA-7FN1156I	1.2V	-7	STD	Lead-Free fpBGA	1156	IND	67
LFE3-70EA-8FN1156I	1.2V	-8	STD	Lead-Free fpBGA	1156	IND	67
LFE3-70EA-9FN1156I	1.2V	-9	STD	Lead-Free fpBGA	1156	IND	67
LFE3-70EA-6LFN1156I	1.2V	-6	LOW	Lead-Free fpBGA	1156	IND	67
LFE3-70EA-7LFN1156I	1.2V	-7	LOW	Lead-Free fpBGA	1156	IND	67
LFE3-70EA-8LFN1156I	1.2V	-8	LOW	Lead-Free fpBGA	1156	IND	67

Part Number	Voltage	Grade	Power	Package	Pins	Temp.	LUTs (K)
LFE3-95EA-6FN484I	1.2V	-6	STD	Lead-Free fpBGA	484	IND	92
LFE3-95EA-7FN484I	1.2V	-7	STD	Lead-Free fpBGA	484	IND	92
LFE3-95EA-8FN484I	1.2V	-8	STD	Lead-Free fpBGA	484	IND	92
LFE3-95EA-9FN484I	1.2V	-9	STD	Lead-Free fpBGA	484	IND	92
LFE3-95EA-6LFN484I	1.2V	-6	LOW	Lead-Free fpBGA	484	IND	92
LFE3-95EA-7LFN484I	1.2V	-7	LOW	Lead-Free fpBGA	484	IND	92
LFE3-95EA-8LFN484I	1.2V	-8	LOW	Lead-Free fpBGA	484	IND	92
LFE3-95EA-6FN672I	1.2V	-6	STD	Lead-Free fpBGA	672	IND	92
LFE3-95EA-7FN672I	1.2V	-7	STD	Lead-Free fpBGA	672	IND	92
LFE3-95EA-8FN672I	1.2V	-8	STD	Lead-Free fpBGA	672	IND	92
LFE3-95EA-9FN672I	1.2V	-9	STD	Lead-Free fpBGA	672	IND	92
LFE3-95EA-6LFN672I	1.2V	-6	LOW	Lead-Free fpBGA	672	IND	92
LFE3-95EA-7LFN672I	1.2V	-7	LOW	Lead-Free fpBGA	672	IND	92
LFE3-95EA-8LFN672I	1.2V	-8	LOW	Lead-Free fpBGA	672	IND	92
LFE3-95EA-6FN1156I	1.2V	-6	STD	Lead-Free fpBGA	1156	IND	92
LFE3-95EA-7FN1156I	1.2V	-7	STD	Lead-Free fpBGA	1156	IND	92
LFE3-95EA-8FN1156I	1.2V	-8	STD	Lead-Free fpBGA	1156	IND	92
LFE3-95EA-9FN1156I	1.2V	-9	STD	Lead-Free fpBGA	1156	IND	92
LFE3-95EA-6LFN1156I	1.2V	-6	LOW	Lead-Free fpBGA	1156	IND	92
LFE3-95EA-7LFN1156I	1.2V	-7	LOW	Lead-Free fpBGA	1156	IND	92
LFE3-95EA-8LFN1156I	1.2V	-8	LOW	Lead-Free fpBGA	1156	IND	92

Part Number	Voltage	Grade	Power	Package	Pins	Temp.	LUTs (K)
LFE3-150EA-6FN672I	1.2V	-6	STD	Lead-Free fpBGA	672	IND	149
LFE3-150EA-7FN672I	1.2V	-7	STD	Lead-Free fpBGA	672	IND	149
LFE3-150EA-8FN672I	1.2V	-8	STD	Lead-Free fpBGA	672	IND	149
LFE3-150EA-9FN672I	1.2V	-9	STD	Lead-Free fpBGA	672	IND	149
LFE3-150EA-6LFN672I	1.2V	-6	LOW	Lead-Free fpBGA	672	IND	149
LFE3-150EA-7LFN672I	1.2V	-7	LOW	Lead-Free fpBGA	672	IND	149
LFE3-150EA-8LFN672I	1.2V	-8	LOW	Lead-Free fpBGA	672	IND	149
LFE3-150EA-6FN1156I	1.2V	-6	STD	Lead-Free fpBGA	1156	IND	149
LFE3-150EA-7FN1156I	1.2V	-7	STD	Lead-Free fpBGA	1156	IND	149
LFE3-150EA-8FN1156I	1.2V	-8	STD	Lead-Free fpBGA	1156	IND	149
LFE3-150EA-9FN1156I	1.2V	-9	STD	Lead-Free fpBGA	1156	IND	149
LFE3-150EA-6LFN1156I	1.2V	-6	LOW	Lead-Free fpBGA	1156	IND	149
LFE3-150EA-7LFN1156I	1.2V	-7	LOW	Lead-Free fpBGA	1156	IND	149
LFE3-150EA-8LFN1156I	1.2V	-8	LOW	Lead-Free fpBGA	1156	IND	149

Part Number	Voltage	Grade	Power	Package	Pins	Temp.	LUTs (K)
LFE3-150EA-6FN672ITW <sup>1</sup>	1.2V	-6	STD	Lead-Free fpBGA	672	IND	149
LFE3-150EA-7FN672ITW <sup>1</sup>	1.2V	-7	STD	Lead-Free fpBGA	672	IND	149
LFE3-150EA-8FN672ITW <sup>1</sup>	1.2V	-8	STD	Lead-Free fpBGA	672	IND	149
LFE3-150EA-6FN1156ITW <sup>1</sup>	1.2V	-6	STD	Lead-Free fpBGA	1156	IND	149
LFE3-150EA-7FN1156ITW <sup>1</sup>	1.2V	-7	STD	Lead-Free fpBGA	1156	IND	149
LFE3-150EA-8FN1156ITW <sup>1</sup>	1.2V	-8	STD	Lead-Free fpBGA	1156	IND	149

1. Specifications for the LFE3-150EA-*spFNpkg*CTW and LFE3-150EA-*spFNpkg*ITW devices, (where *sp* is the speed and *pkg* is the package), are the same as the LFE3-150EA-*spFNpkg*C and LFE3-150EA-*spFNpkg*I devices respectively, except as specified below.

- The CTC (Clock Tolerance Circuit) inside the SERDES hard PCS in the TW device is not functional but it can be bypassed and implemented in soft IP.
- The SERDES XRES pin on the TW device passes CDM testing at 250V.

## For Further Information

A variety of technical notes for the LatticeECP3 family are available on the Lattice website at [www.latticesemi.com](http://www.latticesemi.com).

- TN1169, [LatticeECP3 sysCONFIG Usage Guide](#)
- TN1176, [LatticeECP3 SERDES/PCS Usage Guide](#)
- TN1177, [LatticeECP3 sysIO Usage Guide](#)
- TN1178, [LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide](#)
- TN1179, [LatticeECP3 Memory Usage Guide](#)
- TN1180, [LatticeECP3 High-Speed I/O Interface](#)
- TN1181, [Power Consumption and Management for LatticeECP3 Devices](#)
- TN1182, [LatticeECP3 sysDSP Usage Guide](#)
- TN1184, [LatticeECP3 Soft Error Detection \(SED\) Usage Guide](#)
- TN1189, [LatticeECP3 Hardware Checklist](#)

For further information on interface standards refer to the following websites:

- JEDEC Standards (LVTTTL, LVCMOS, SSTL, HSTL): [www.jedec.org](http://www.jedec.org)
- PCI: [www.pcisig.com](http://www.pcisig.com)



# LatticeECP3 Family Data Sheet

## Revision History

June 2013

Data Sheet DS1021

Date	Version	Section	Change Summary
February 2009	01.0	—	Initial release.
May 2009	01.1	All	Removed references to Parallel burst mode Flash.
		Introduction	Features - Changed 250 Mbps to 230 Mbps in Embedded SERDES bulleted section and added a footnote to indicate 230 Mbps applies to 8b10b and 10b12b applications.
			Updated data for ECP3-17 in LatticeECP3 Family Selection Guide table.
			Changed embedded memory from 552 to 700 Kbits in LatticeECP3 Family Selection Guide table.
		Architecture	Updated description for CLKFB in General Purpose PLL Diagram.
			Corrected Primary Clock Sources text section.
			Corrected Secondary Clock/Control Sources text section.
			Corrected Secondary Clock Regions table.
			Corrected note below Detailed sysDSP Slice Diagram.
			Corrected Clock, Clock Enable, and Reset Resources text section.
			Corrected ECP3-17 EBR number in Embedded SRAM in the LatticeECP3 Family table.
			Added On-Chip Termination Options for Input Modes table.
			Updated Available SERDES Quads per LatticeECP3 Devices table.
			Updated Simplified Channel Block Diagram for SERDES/PCS Block diagram.
			Updated Device Configuration text section.
		Corrected software default value of MCCLK to be 2.5 MHz.	
		DC and Switching Characteristics	Updated VCCOB Min/Max data in Recommended Operating Conditions table.
			Corrected footnote 2 in sysIO Recommended Operating Conditions table.
			Added added footnote 7 for $t_{SKEW\_PRIB}$ to External Switching Characteristics table.
			Added 2-to-1 Gearing text section and table.
			Updated External Reference Clock Specification (refclkp/refclkn) table.
			LatticeECP3 sysCONFIG Port Timing Specifications - updated $t_{DINIT}$ information.
			Added sysCONFIG Port Timing waveform.
			Serial Input Data Specifications table, delete Typ data for $V_{RX-DIFF-S}$ .
			Added footnote 4 to sysCLOCK PLL Timing table for $t_{PFD}$ .
			Added SERDES/PCS Block Latency Breakdown table.
			External Reference Clock Specifications table, added footnote 4, add symbol name vREF-IN-DIFF.
Added SERDES External Reference Clock Waveforms.			
Updated Serial Output Timing and Levels table.			
Pin-to-pin performance table, changed "typically 3% slower" to "typically slower".			

Date	Version	Section	Change Summary
May 2009 (cont.)	01.1 (cont.)	DC and Switching Characteristics (cont.)	Updated timing information
			Updated SERDES minimum frequency.
			Added data to the following tables: External Switching Characteristics, Internal Switching Characteristics, Family Timing Adders, Maximum I/O Buffer Speed, DLL Timing, High Speed Data Transmitter, Channel Output Jitter, Typical Building Block Function Performance, Register-to-Register Performance, and Power Supply Requirements.
			Updated Serial Input Data Specifications table.
			Updated Transmit table, Serial Rapid I/O Type 2 Electrical and Timing Characteristics section.
		Pinout Information	Updated Signal Description tables. Updated Pin Information Summary tables and added footnote 1.
July 2009	01.2	Multiple	Changed references of “multi-boot” to “dual-boot” throughout the data sheet.
		Architecture	Updated On-Chip Programmable Termination bullets.
			Updated On-Chip Termination Options for Input Modes table.
			Updated On-Chip Termination figure.
		DC and Switching Characteristics	Changed min/max data for FREF_PPM and added footnote 4 in SERDES External Reference Clock Specification table.
			Updated SERDES minimum frequency.
		Pinout Information	Corrected MCLK to be I/O and CCLK to be I in Signal Descriptions table
August 2009	01.3	DC and Switching Characteristics	Corrected truncated numbers for $V_{CCIB}$ and $V_{CCOB}$ in Recommended Operating Conditions table.
September 2009	01.4	Architecture	Corrected link in sysMEM Memory Block section.
			Updated information for On-Chip Programmable Termination and modified corresponding figure.
			Added footnote 2 to On-Chip Programmable Termination Options for Input Modes table.
			Corrected Per Quadrant Primary Clock Selection figure.
		DC and Switching Characteristics	Modified -8 Timing data for 1024x18 True-Dual Port RAM (Read-Before-Write, EBR Output Registers)
			Added ESD Performance table.
			LatticeECP3 External Switching Characteristics table - updated data for $t_{DIBGDDR}$ , $t_{W\_PRI}$ , $t_{W\_EDGE}$ and $t_{SKEW\_EDGE\_DQS}$ .
			LatticeECP3 Internal Switching Characteristics table - updated data for $t_{COO\_PIO}$ and added footnote #4.
			sysCLOCK PLL Timing table - updated data for $f_{OUT}$ .
			External Reference Clock Specification (refclkp/refclkn) table - updated data for $V_{REF-IN-SE}$ and $V_{REF-IN-DIFF}$ .
			LatticeECP3 sysCONFIG Port Timing Specifications table - updated data for $t_{MWC}$ .
			Added TRLVDS DC Specification table and diagram.
			Updated Mini LVDS table.
			November 2009
Architecture	Updated Figure 2-4, General Purpose PLL Diagram.		
	Updated SONET/SDH to SERDES and PCS protocols.		

Date	Version	Section	Change Summary
November 2009 (cont.)	01.5 (cont.)	Architecture (cont.)	Updated Table 2-13, SERDES Standard Support to include SONET/SDH and updated footnote 2.
		DC and Switching Characteristics	Added footnote to ESD Performance table.
		Updated SERDES Power Supply Requirements table and footnotes.	
		Updated Maximum I/O Buffer Speed table.	
		Updated Pin-to-Pin Performance table.	
		Updated sysCLOCK PLL Timing table.	
		Updated DLL timing table.	
		Updated High-Speed Data Transmitter tables.	
		Updated High-Speed Data Receiver table.	
		Updated footnote for Receiver Total Jitter Tolerance Specification table.	
		Updated Periodic Receiver Jitter Tolerance Specification table.	
		Updated SERDES External Reference Clock Specification table.	
		Updated PCI Express Electrical and Timing AC and DC Characteristics.	
		Deleted Reference Clock table for PCI Express Electrical and Timing AC and DC Characteristics.	
		Updated SMPTE AC/DC Characteristics Transmit table.	
		Updated Mini LVDS table.	
		Updated RSDS table.	
		Added Supply Current (Standby) table for EA devices.	
		Updated Internal Switching Characteristics table.	
		Updated Register-to-Register Performance table.	
		Added HDMI Electrical and Timing Characteristics data.	
		Updated Family Timing Adders table.	
		Updated sysCONFIG Port Timing Specifications table.	
		Updated Recommended Operating Conditions table.	
		Updated Hot Socket Specifications table.	
		Updated Single-Ended DC table.	
		Updated TRLVDS table and figure.	
		Updated Serial Data Input Specifications table.	
Updated HDMI Transmit and Receive table.			
Ordering Information	Added LFE3-150EA "TW" devices and footnotes to the Commercial and Industrial tables.		
March 2010	01.6	Architecture	Added Read-Before-Write information.
		DC and Switching Characteristics	Added footnote #6 to Maximum I/O Buffer Speed table.
		Corrected minimum operating conditions for input and output differential voltages in the Point-to-Point LVDS table.	
		Pinout Information	Added pin information for the LatticeECP3-70EA and LatticeECP3-95EA devices.
Ordering Information	Added ordering part numbers for the LatticeECP3-70EA and LatticeECP3-95EA devices.		
Removed dual mark information.			
December 2010	01.7EA	Multiple	Data sheet made final. Removed "preliminary" headings.
			Removed data for 70E and 95E devices. A separate data sheet is available for these specific devices.
			Updated for Lattice Diamond design software.

Date	Version	Section	Change Summary
December 2010 (cont.)	01.7EA (cont.)	Introduction	Corrected number of user I/Os
		Architecture	Corrected the package type in Table 2-14 Available SERDES Quad per LatticeECP3 Devices.
			Updated description of General Purpose PLL
			Added additional information in the Flexible Quad SERDES Architecture section.
			Added footnotes and corrected the information in Table 2-16 Selectable master Clock (MCCLK) Frequencies During Configuration (Nominal).
			Updated Figure 2-16, Per Region Secondary Clock Selection.
			Updated description for On-Chip Programmable Termination.
			Added information about number of rows of DSP slices.
			Updated footnote 2 for Table 2-12, On-Chip Termination Options for Input Modes.
			Updated information for sysIO buffer pairs.
			Corrected minimum number of General Purpose PLLs (was 4, now 2).
		DC and Switching Characteristics	Regenerated sysCONFIG Port Timing figure.
			Added $t_{V}$ (clock pulse width) in External Switching Characteristics table.
			Corrected units, revised and added data, and corrected footnote 1 in External Switching Characteristics table.
			Added Jitter Transfer figures in SERDES External Reference Clock section.
			Corrected capacitance information in the DC Electrical Characteristics table.
			Corrected data in the Register-to-Register Performance table.
			Corrected GDDR Parameter name HOGDDR.
			Corrected RSDS25 -7 data in Family Timing Adders table.
			Added footnotes 10-12 to DDR data information in the External Switching Characteristics table.
			Corrected titles for Figures 3-7 (DDR/DDR2/DDR3 Parameters) and 3-8 (Generic DDR/DDR2 Parameters).
			Updated titles for Figures 3-5 (MLVDS25 (Multipoint Low Voltage Differential Signaling)) and 3-6 (Generic DDRX1/DDR2 (With Clock and Data Edges Aligned)).
			Updated Supply Current table.
Pinout Information	Added GDDR interface information to the External Switching and Characteristics table.		
	Added footnote to sysIO Recommended Operating Conditions table.		
	Added footnote to LVDS25 table.		
	Corrected DDR section footnotes and references.		
	Corrected Hot Socketing support from "top and bottom banks" to "top and bottom I/O pins".		
April 2011	01.8EA	Architecture	Updated Secondary Clock/Control Sources text section.



Date	Version	Section	Change Summary
April 2011 (cont.)	01.8EA (cont.)	DC and Switching Characteristics	Added data for 150 Mbps to SERDES Power Supply Requirements table.
			Updated Frequencies in Table 3-6 Serial Output Timing and Levels
			Added Data for 150 Mbps to Table 3-7 Channel Output Jitter
			Corrected External Switching Characteristics table, Description for DDR3 Clock Timing, $t_{JIT}$ .
			Corrected Internal Switching Characteristics table, Description for EBR Timing, $t_{SUWREN\_EBR}$ and $t_{HWREN\_EBR}$ .
			Added footnote 1 to sysConfig Port Timing Specifications table.
			Updated description for RX-CIDs to 150M in Table 3-9 Serial Input Data Specifications
			Updated Frequency to 150 Mbps in Table 3-11 Periodic Receiver Jitter Tolerance Specification
July 2011	01.9EA	DC and Switching Characteristics	Removed ESD Performance table and added reference to LatticeECP3 Product Family Qualification Summary document.
			sysCLOCK PLL Timing table, added footnote 4.
			External Reference Clock Specification table – removed reference to VREF-CM-AC and removed footnote for VREF-CM-AC.
		Pinout Information	Pin Information Summary table: Corrected VCCIO Bank8 data for LatticeECP3-17EA 256-ball ftBGA package and LatticeECP3-35EA 256-ball ftBGA package.
November 2011	02.0EA	Introduction	Added information for LatticeECP3-17EA, 328-ball csBGA package.
		Architecture	Added information for LatticeECP3-17EA, 328-ball csBGA package.
		DC and Switching Characteristics	Updated LatticeECP3 Supply Current table power numbers.
			Typical Building Block Function Performance table, LatticeECP3 External Switching Characteristics table, LatticeECP3 Internal Switching Characteristics table and LatticeECP3 Family Timing Adders: Added speed grade -9 and updated speed grade -8, -7 and -6 timing numbers.
		Pinout Information	Added information for LatticeECP3-17EA, 328-ball csBGA package.
		Ordering Information	Added information for LatticeECP3-17EA, 328-ball csBGA package.
			Added ordering information for low power devices and -9 speed grade devices.
February 2012	02.1EA	All	Updated document with new corporate logo.
April 2012	02.2EA	Architecture	Updated first paragraph of Output Register Block section.
			Updated the information about sysIO buffer pairs below Figure 2-38.
			Updated the information relating to migration between devices in the Density Shifting section.
		DC and Switching Characteristics	Corrected the Definitions in the sysCLOCK PLL Timing table for $t_{RST}$ .
Ordering Information	Updated topside marks with new logos in the Ordering Information section.		
June 2013	02.3EA	Architecture	sysI/O Buffer Banks text section – Updated description of “Top (Bank 0 and Bank 1) and Bottom sysIO Buffer Pairs (Single-Ended Outputs Only)” for hot socketing information.
			sysI/O Buffer Banks text section – Updated description of “Configuration Bank sysI/O Buffer Pairs (Single-Ended Outputs, Only on Shared Pins When Not Used by Configuration)” for PCI clamp information.
			On-Chip Oscillator section – clarified the speed of the internal CMOS oscillator (130 MHz +/- 15%).

Date	Version	Section	Change Summary
June 2013 (cont.)	02.3EA (cont.)	Architecture (cont.)	Architecture Overview section – Added information on the state of the register on power up and after configuration.
		DC and Switching Characteristics	<p>sys/I/O Recommended Operating Conditions table – Removed reference to footnote 1 from RSDS standard.</p> <p>sys/I/O Single-Ended DC Electrical Characteristics table – Modified footnote 1.</p> <p>Added Oscillator Output Frequency table.</p> <p>LatticeECP3 sysCONFIG Port Timing Specifications table – Updated min. column for <math>t_{CODO}</math> parameter.</p> <p>LatticeECP3 Family Timing Adders table – Description column, references to <math>V_{CCIO} = 3.0V</math> changed to 3.3V. For PPLVDS, description changed from emulated to True LVDS and <math>V_{CCIO} = 2.5V</math> changed to <math>V_{CCIO} = 2.5V</math> or 3.3V.</p> <p>LatticeECP3 Maximum I/O Buffer Speed table – Description column, references to <math>V_{CCIO} = 3.0V</math> changed to 3.3V.</p> <p>Updated SERDES External Reference Clock Waveforms.</p> <p>Transmitter and Receiver Latency Block Diagram – Updated sections of the diagram to match descriptions on the SERDES/PCS Latency Break-down table.</p>
		Pinout Information	<p>“Logic Signal Connections” section heading renamed “Package Pinout Information”. Software menu selections within this section have been updated.</p> <p>Signal Descriptions table – Updated description for <math>V_{CCA}</math> signal.</p>



## Section II. LatticeECP3 Family Technical Notes

---

## Introduction

The LatticeECP3™ FPGA family combines a high-performance FPGA fabric, high-performance I/Os and up to 16 channels of embedded SERDES with associated Physical Coding Sublayer (PCS) logic. The PCS logic can be configured to support numerous industry-standard, high-speed serial data transfer protocols.

Each channel of PCS logic contains dedicated transmit and receive SERDES for high-speed, full-duplex serial data transfer at data rates up to 3.2 Gbps. The PCS logic in each channel can be configured to support an array of popular data protocols including GbE, XAUI, SONET/SDH, PCI Express, SRIO, CPRI, OBSAI, SD-SDI, HD-SDI and 3G-SDI. In addition, the protocol-based logic can be fully or partially bypassed in a number of configurations to allow users flexibility in designing their own high-speed data interface.

The PCS also provides bypass modes that allow a direct 8-bit or 10-bit interface from the SERDES to the FPGA logic. Each SERDES pin can be independently DC-coupled and can allow for both high-speed and low-speed operation on the same SERDES pin for applications such as Serial Digital Video.

## Features

- **Up to 16 Channels of High-Speed SERDES**
  - 150 Mbps to 3.2 Gbps for Generic 8b10b, 10-bit SERDES and 8-bit SERDES modes. Refer to Table 8-1.
  - 230 Mbps to 3.2 Gbps per channel for all other protocols
  - 3.2 Gbps operation with low 110 mW power per channel
  - Receive equalization and transmit pre-emphasis for small form factor backplane operation
  - Supports PCI Express, Gigabit Ethernet (1GbE and SGMII) and XAUI, plus multiple other standards
  - Supports user-specified generic 8b10b mode
  - Out-of-band (OOB) signal interface for low-speed inputs (video application)
- **Multiple Clock Rate Support**
  - Separate reference clocks for each PCS quad allow easy handling of multiple protocol rates on a single device
- **Full-Function Embedded Physical Coding Sub-layer (PCS) Logic Supporting Industry Standard Protocols**
  - Up to 16 channels of full-duplex data supported per device
  - Multiple protocol support on one chip
  - Supports popular 8b10b-based packet protocols
  - SERDES Only mode allows direct 8-bit or 10-bit interface to FPGA logic
- **Multiple Protocol Compliant Clock Tolerance Compensation (CTC) Logic**
  - Compensates for frequency differential between reference clock and received data rate
  - Allows user-defined skip pattern of 1, 2, or 4 bytes in length
- **Integrated Loopback Modes for System Debugging**
  - Three loopback modes are provided for system debugging

## New Features Over LatticeECP2M™ SERDES/PCS

- Supports multiple protocols/standards within one quad of SERDES. The standards are required to have nominal frequencies either at the full rate or half rate of the supported standards listed in Table 8-1. Configuration flexibility should not be a barrier to supporting different mixes of protocols and standards. PCI Express, Gigabit Ethernet, SGMII and Serial RapidIO modes are supported in the multi-protocol grouping.
- Supports XAUI compliance features and extended SERDES maximum performance to 3.2 Gbps.

- Supports SONET/SDH OC-3/STM-1, OC-12/STM-4 and OC-48/STM-16 rates.
- Added support for per RX and TX DIV11 for SD-SDI, HD-SDI and 3G-SDI. Multi-rate SDI support.

## Using This Technical Note

The ispLEVER® design tools from Lattice support all modes of the PCS. Most modes are dedicated to applications for a specific industry standard data protocol. Other modes are more general purpose in nature and allow designers to define their own custom application settings. ispLEVER design tools allow the user to define the mode for each quad in their design. This technical note describes operation of the SERDES and PCS for all modes supported by ispLEVER. If you are using Lattice Diamond® design software, see Appendix D.

This document provides a thorough description of the complete functionality of the embedded SERDES and associated PCS logic. Electrical and timing characteristics of the embedded SERDES are provided in the [LatticeECP3 Family Data Sheet](#). Operation of the PCS logic is provided in the PCS section of this document. A table of all status and control registers associated with the SERDES and PCS logic which can be accessed via the SCI Bus is provided in the appendices. Package pinout information is provided in the Pinout Information section of the [LatticeECP3 Family Data Sheet](#).

## Standards Supported

The supported standards are listed in Table 8-1.

**Table 8-1. Standards Supported by the SERDES**

Standard	Data Rate (Mbps)	System Reference Clock (MHz)	FPGA Clock (MHz)	Number of General/Link Width	Encoding Style
PCI Express 1.1	2500	100	250	x1, x2, x4	8b10b
Gigabit Ethernet, SGMII	1250	125	125	x1	8b10b
	2500	125	250	x1	8b10b
	3125	156.25	156.25	x1	8b10b
XAUI	3125	156.25	156.25	x4	8b10b
Serial RapidIO Type I, Serial RapidIO Type II, Serial RapidIO Type III	1250, 2500, 3125	125, 125, 156.25	125, 250, 156.25	x1, x4	8b10b
OBSAI-1, OBSAI-2, OBSAI-3, OBSAI-4	768, 1536, 2304, 3072	76.8, 76.8, 153.6, 115.2, 153.6	76.8, 153.6, 230.4, 153.6	x1	8b10b
CPRI-1, CPRI-2, CPRI-3, CPRI-4	614.4, 1228.8, 2457.6, 3072.0	61.44, 61.44, 122.88, 122.88, 153.6	61.44, 122.88, 122.88, 153.6	x1	8b10b
SD-SDI (259M, 344M)	143 <sup>1</sup> , 177 <sup>1</sup> , 270, 360, 540	14.3 <sup>1</sup> , 17.7 <sup>1</sup> , 27, 36, 54	143, 177, 27, 36, 54	x1	NRZI/Scrambled
HD-SDI (292M)	1483.5, 1485	74.175, 148.35, 74.25, 148.50	74.175, 148.35, 74.25, 148.5	x1	NRZI/Scrambled
3G-SDI (424M)	2967, 2970	148.35, 148.5	148.35, 148.5	x1	NRZI/Scrambled
SONET STS-3 <sup>2</sup>	155.52	15.552	15.552	x1	N/A
SONET STS-12 <sup>2</sup>	622.08	62.208	62.208		
SONET STS-48 <sup>2</sup>	2488	248.8	248.8		
10-Bit SERDES	150 - 3125	15 - 312.5	15 - 312.5	x1, x2, x3, x4	N/A

**Table 8-1. Standards Supported by the SERDES (Continued)**

Standard	Data Rate (Mbps)	System Reference Clock (MHz)	FPGA Clock (MHz)	Number of General/Link Width	Encoding Style
8-Bit SERDES	150 - 3125	15 - 312.5	15 - 312.5	x1, x2, x3, x4	N/A
Generic 8b10b	150 - 3125	15 - 312.5	15 - 312.5	x1, x2, x3, x4	8b10b

1. For slower rates, the SERDES are bypassed and signals are directly fed to the FPGA core.
2. The SONET protocol is supported in 8-Bit SERDES mode. Refer to the SONET section of this document for detailed information.

## Architecture Overview

The SERDES/PCS block is arranged in quads containing logic for four independent full-duplex data channels.

Figure 8-1 shows the arrangement of SERDES/PCS quads on the LatticeECP3-150 FPGA (other devices have fewer quads).

**Figure 8-1. LatticeECP3-150 Block Diagram**

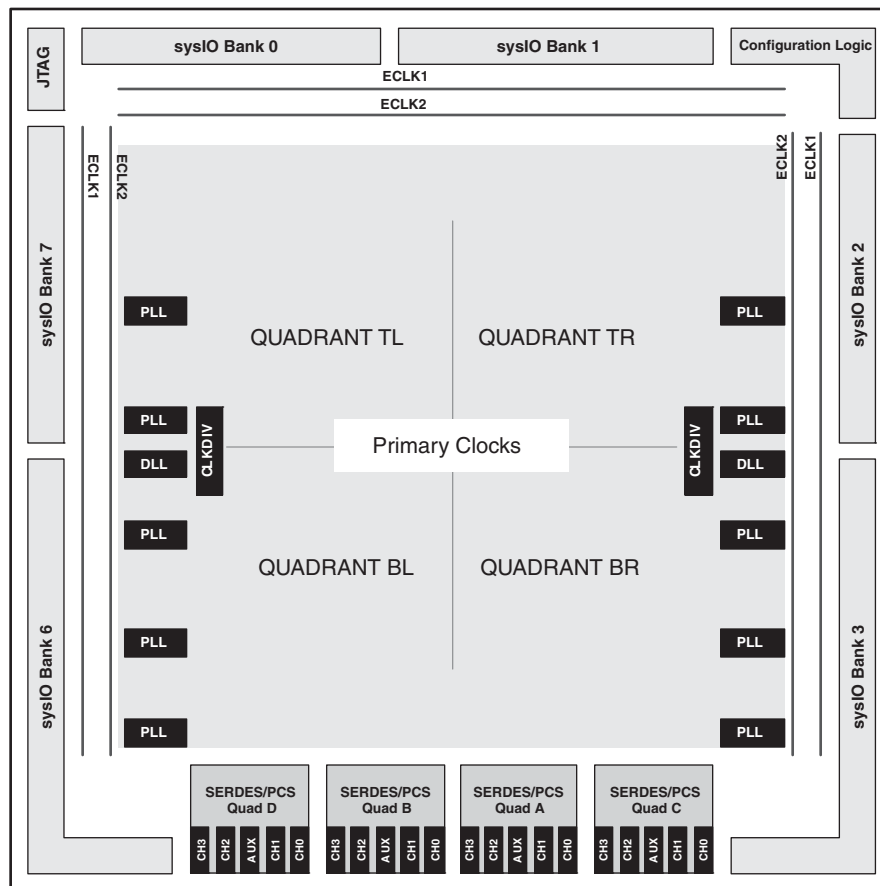


Table 8-2 shows the number of available SERDES/PCS quads for each device in the LatticeECP3 family.

**Table 8-2. Number of SERDES/PCS Quads per LatticeECP3 Device**

Package	ECP3-17	ECP3-35	ECP3-70	ECP3-95	ECP3-150
256-ball ftBGA	1	1	—	—	—
328-ball csBGA	2 channels <sup>1</sup>	—	—	—	—
484-ball ftBGA	1	1	1	1	
672-ball ftBGA	—	1	2	2	2
1156-ball ftBGA	—	—	3	3	4

1. Channels 0 and 3 are available.

Every quad can be programmed into one of several protocol-based modes. Each quad requires its own reference clock which can be sourced externally from package pins or internally from the FPGA logic.

Each quad can be programmed with select protocols that have nominal frequencies which can utilize the full and half-rate options per channel. For example, a PCI Express x1 at 2.5Gbps and a Gigabit Ethernet channel can be utilized in the same quad using the half-rate option on the Gigabit Ethernet channel. If a quad shares a PCI Express x1 channel with a non-PCI Express channel, ensure that the reference clock for the quad is compatible with all protocols within the quad. For example, a PCI Express spread spectrum reference clock is not compatible with most Gigabit Ethernet applications.

Since each quad has its own reference clock, different quads can support different standards on the same chip. This feature makes the LatticeECP3 family of devices ideal for bridging between different standards.

PCS quads are not dedicated solely to industry standard protocols. Each quad (and each channel within a quad) can be programmed for many user-defined data manipulation modes. For example, word alignment and clock tolerance compensation can be programmed for user-defined operation.

## PCS Quads and Channels

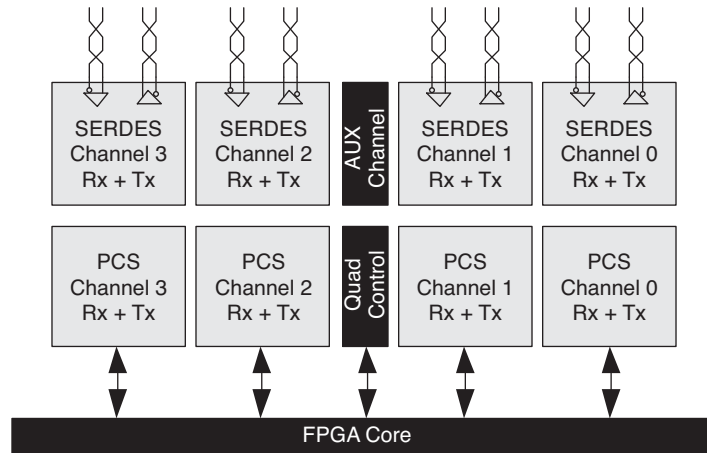
Each quad on a device supports up to four channels of full-duplex data. One to four channels in a quad can be utilized, depending on the application. Users can set many options for each channel independently within a given quad.

Figure 8-1 shows an example of a device with four PCS quads which contain a total of 16 PCS channels.

## Per Channel SERDES/PCS and FPGA Interface Ports

All PCS quads regardless of the mode used have the same external high-speed serial interface at the package pins. However, every PCS mode has its own unique list of input/output ports from/to the FPGA logic appropriate to the protocol chosen for the quad. A detailed description of the quad input/output signals for each mode is provided in this document. Figure 8-2 describes a simplified SERDES/PCS quad.

**Figure 8-2. SERDES/PCS Quad Block Diagram**

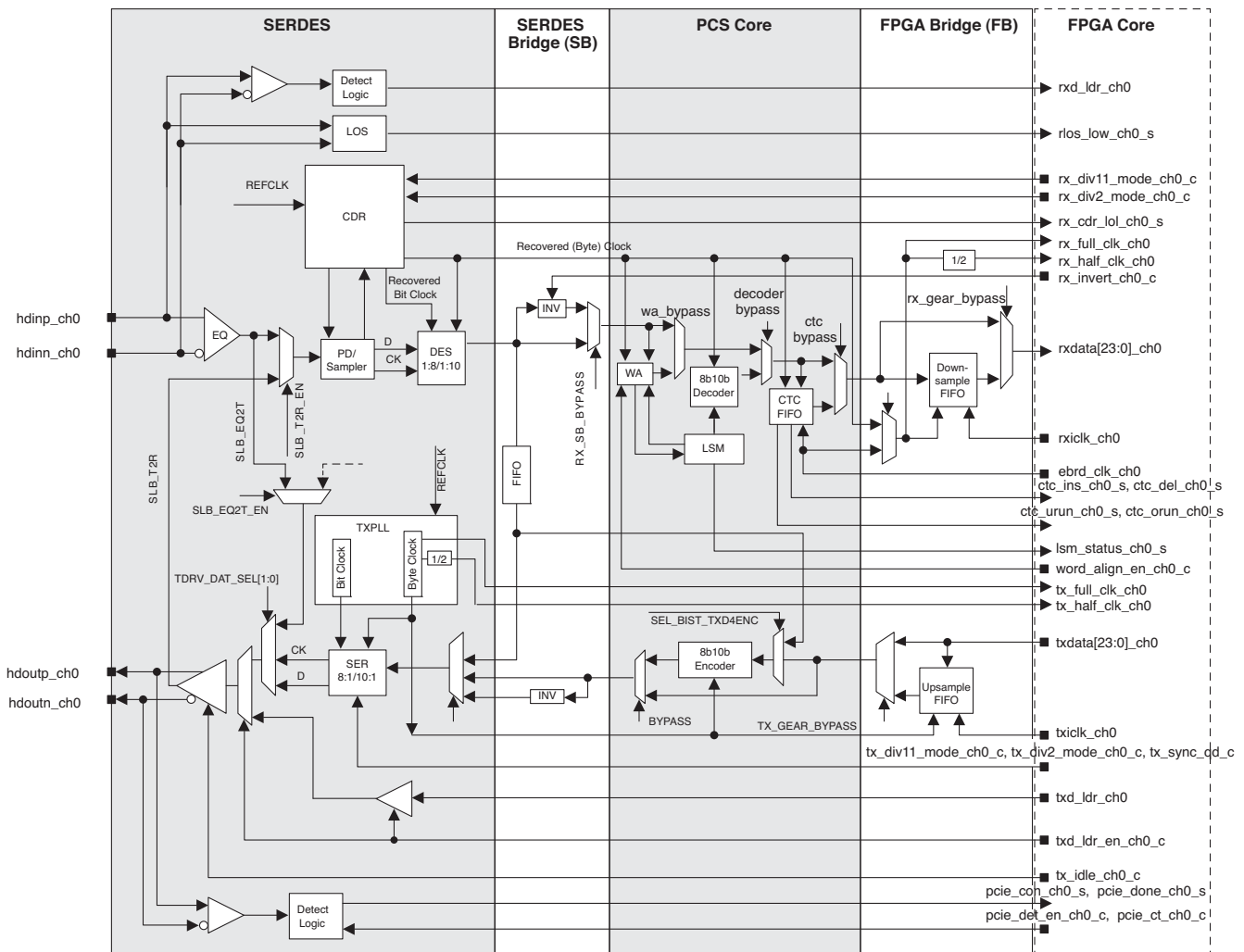


**Detailed Channel Block Diagram**

Figure 8-3 is a detailed block diagram representation of the major functionality in a single channel of the LatticeECP3 SERDES/PCS. This diagram shows all the major blocks and the majority of the control and status signals that are visible to the user logic in the FPGA. This diagram also shows the major sub-blocks in the channel – SERDES, SERDES Bridge, PCS Core and the FPGA Bridge.



**Figure 8-3. LatticeECP3 SERDES/PCS Detailed Channel Block Diagram**



### Clocks and Resets

A PCS quad supplies per-channel locked reference clocks and per-channel recovered receive clocks to the FPGA logic interface. Each PCS quad provides clocks on both primary and secondary FPGA clock routing. The PCS/FPGA interface also has ports for the transmit and receive clocks supplied from the FPGA fabric for all four channels in each quad.

Each quad has reset inputs to force reset of both the SERDES and PCS logic in a quad or just the SERDES. In addition, separate resets dedicated for the PCS logic are provided for each channel for both the transmit and receive directions.

### Transmit Data Bus

The signals for the transmit data path are from the FPGA to the FPGA Bridge in the PCS Block. The datapath can be geared 2:1 to the internal PCS data path, which is 8 bits wide (plus control/status signals). The highest speed of the interface for PCI Express x1 is 250 MHz in 1:1 geared mode. With 2:1 gearing (i.e. a 16-bit wide data path), a possible speed is 156.25 MHz (for XAUI 4x channel mode). The SERDES and PCS will support data rates up to 3.2 Gbps data that correspond to an interface speed of 160 MHz (with 2:1 gearing).

## Receive Data Bus

The signals for the receive path are from the FPGA Bridge in the PCS Block to the FPGA. The data path may be geared 2:1 to the internal PCS data path which is 8 bits wide. The data bus width at the FPGA interface is 16 bits wide. It is possible to disable the 2:1 gearing via a software register bit, in which case, the bus widths are halved (8 bits wide). When the data is geared 2:1, the lower bits (rxdata[9:0]) correspond to the word that has been received first and the higher bits (rxdata[19:10]) correspond to the word that has been received second. If the data is not geared 2:1, the lower bits ((rxdata[9:0]) are the active bits and the higher bits should not be used. Table 8-3 describes the use of the data bus for each protocol mode.

**Table 8-3. Data Bus Usage by Mode**

Data Bus PCS Cell Name <sup>4</sup>	G8B10B	CPRI	OBSAI	PCI Express	SRIO	Gigabit Ethernet	XAUI	8-Bit SERDES	10-Bit SERDES	SDI
FF_TX_D_0_0					txdata_ch0[0]					
FF_TX_D_0_1					txdata_ch0[1]					
FF_TX_D_0_2					txdata_ch0[2]					
FF_TX_D_0_3					txdata_ch0[3]					
FF_TX_D_0_4					txdata_ch0[4]					
FF_TX_D_0_5					txdata_ch0[5]					
FF_TX_D_0_6					txdata_ch0[6]					
FF_TX_D_0_7					txdata_ch0[7]					
FF_TX_D_0_8				tx_k_ch0[0]			txc_ch0[0]	GND		txdata_ch0[8]
FF_TX_D_0_9	tx_force_disp_ch0[0] <sup>1</sup>					GND				txdata_ch0[9]
FF_TX_D_0_10	tx_disp_sel_ch0[0] <sup>1</sup>				GND	xmit_ch0[0] <sup>2</sup>			GND	
FF_TX_D_0_11	GND			pci_ei_en_ch0[0]	GND	tx_disp_correct_ch0[0]			GND	
FF_TX_D_0_12					txdata_ch0[8]					txdata_ch0[10]
FF_TX_D_0_13					txdata_ch0[9]					txdata_ch0[11]
FF_TX_D_0_14					txdata_ch0[10]					txdata_ch0[12]
FF_TX_D_0_15					txdata_ch0[11]					txdata_ch0[13]
FF_TX_D_0_16					txdata_ch0[12]					txdata_ch0[14]
FF_TX_D_0_17					txdata_ch0[13]					txdata_ch0[15]
FF_TX_D_0_18					txdata_ch0[14]					txdata_ch0[16]
FF_TX_D_0_19					txdata_ch0[15]					txdata_ch0[17]
FF_TX_D_0_20				tx_k_ch0[1]			txc_ch0[1]	GND		txdata_ch0[18]
FF_TX_D_0_21	tx_force_disp_ch0[1] <sup>1</sup>					GND				txdata_ch0[19]
FF_TX_D_0_22	tx_disp_sel_ch0[1] <sup>1</sup>				GND	xmit_ch0[1] <sup>2</sup>			GND	
FF_TX_D_0_23	GND			pci_ei_en_ch0[1]	GND	tx_disp_correct_ch0[1]			GND	
FF_RX_D_0_0					rxdata_ch0[0]					
FF_RX_D_0_1					rxdata_ch0[1]					
FF_RX_D_0_2					rxdata_ch0[2]					
FF_RX_D_0_3					rxdata_ch0[3]					
FF_RX_D_0_4					rxdata_ch0[4]					
FF_RX_D_0_5					rxdata_ch0[5]					
FF_RX_D_0_6					rxdata_ch0[6]					
FF_RX_D_0_7					rxdata_ch0[7]					
FF_RX_D_0_8				rx_k_ch0[0]			rxk_ch0[0]	NC		rxdata_ch0[8]
FF_RX_D_0_9	rx_disp_err_ch0[0]			rxstatus0_ch0[0]		rx_disp_err_ch0[0]		NC		rxdata_ch0[9]
FF_RX_D_0_10	rx_cv_err_ch0[0] <sup>3</sup>			rxstatus0_ch0[1]		rx_cv_err_ch0[0] <sup>3</sup>				NC
FF_RX_D_0_11	NC			rxstatus0_ch0[2]				NC		
FF_RX_D_0_12					rxdata_ch0[8]					rxdata_ch0[10]
FF_RX_D_0_13					rxdata_ch0[9]					rxdata_ch0[11]
FF_RX_D_0_14					rxdata_ch0[10]					rxdata_ch0[12]
FF_RX_D_0_15					rxdata_ch0[11]					rxdata_ch0[13]
FF_RX_D_0_16					rxdata_ch0[12]					rxdata_ch0[14]
FF_RX_D_0_17					rxdata_ch0[13]					rxdata_ch0[15]

**Table 8-3. Data Bus Usage by Mode (Continued)**

Data Bus PCS Cell Name <sup>4</sup>	G8B10B	CPRI	OBSAI	PCI Express	SRIO	Gigabit Ethernet	XAUI	8-Bit SERDES	10-Bit SERDES	SDI
FF_RX_D_0_18	rxdata_ch0[14]								rxdata_ch0[16]	
FF_RX_D_0_19	rxdata_ch0[15]								rxdata_ch0[17]	
FF_RX_D_0_20	rx_k_ch0[1]						rx_c_ch0[1]	NC		rxdata_ch0[18]
FF_RX_D_0_21	rx_disp_err_ch0[1]		rxstatus1_ch0[0]		rx_disp_err_ch0[1]			NC		rxdata_ch0[19]
FF_RX_D_0_22	rx_cv_err_ch0[1] <sup>3</sup>		rxstatus1_ch0[1]		rx_cv_err_ch0[1] <sup>3</sup>			NC		NC
FF_RX_D_0_23	NC		rxstatus1_ch0[2]		NC					

1. The force\_disp signal will force the disparity for the associated data word on bits [7:0] to the column selected by the tx\_disp\_sel signal. If disp\_sel is a one, the 10-bit code is taken from the 'current RD+' column (positive disparity). If the tx\_disp\_sel is a zero, the 10-bit code is taken from the 'current RD-' (negative disparity) column.
2. The Lattice Gigabit Ethernet PCS IP core provides an auto-negotiation state machine that generates the signal xmit. It is used to interact with the Gigabit Ethernet Idle State Machine in the hard logic.
3. When there is a code violation, the packet PCS 8b10b decoder will replace the output from the decoder with hex EE and K asserted (K=1 and d=EE is not part of the 8b10b coding space).
4. FF\_TX\_D\_0\_0: FPGA Fabric Transmit Data Bus Channel 0 Bit 0.

## Mode-Specific Control/Status Signal Descriptions

Table 8-4 describes the mode-specific control/status signals.

**Table 8-4. Control Signals and their Functions**

Signal Name	Description
<b>Transmit Control Signals</b>	
tx_k_ch[3:0]	Per channel, active-high control character indicator.
tx_force_disp_ch[3:0]	Per channel, active-high signal which instructs the PCS to accept disparity value from the disp_sel_ch(0-3) FPGA interface input.
tx_disp_sel_ch[3:0]	Per channel, disparity value supplied from FPGA logic. Valid when force_disp_ch(0-3) is high.
tx_correct_disp_ch[3:0]	Corrects disparity identifier when asserted by adjusting the 8b10b encoder to begin in the negative disparity state.
<b>Receive Status Signals</b>	
rx_k_ch[3:0]	Per channel, active-high control character indicator.
rx_disp_err_ch[3:0]	Per channel, active-high signal driven by the PCS to indicate a disparity error was detected with the associated data.
rx_cv_err_ch[3:0]	Per channel, code violation signal to indicate an error was detected with the associated data.

### Control

Each mode has its own set of control signals which allows direct control of various PCS features from the FPGA logic. In general, each of these control inputs duplicates the effect of writing to a corresponding control register bit or bits.

{signal}\_c is the control signal from the FPGA core to the FPGA bridge. All of the control signals are used asynchronously inside the SERDES/PCS.

### Status

Each mode has its own set of status or alarm signals that can be monitored by the FPGA logic. In general, each of these status outputs corresponds to a specific status register bit or bits. The Diamond design tools give the user the option to bring these ports out to the PCS FPGA interface.

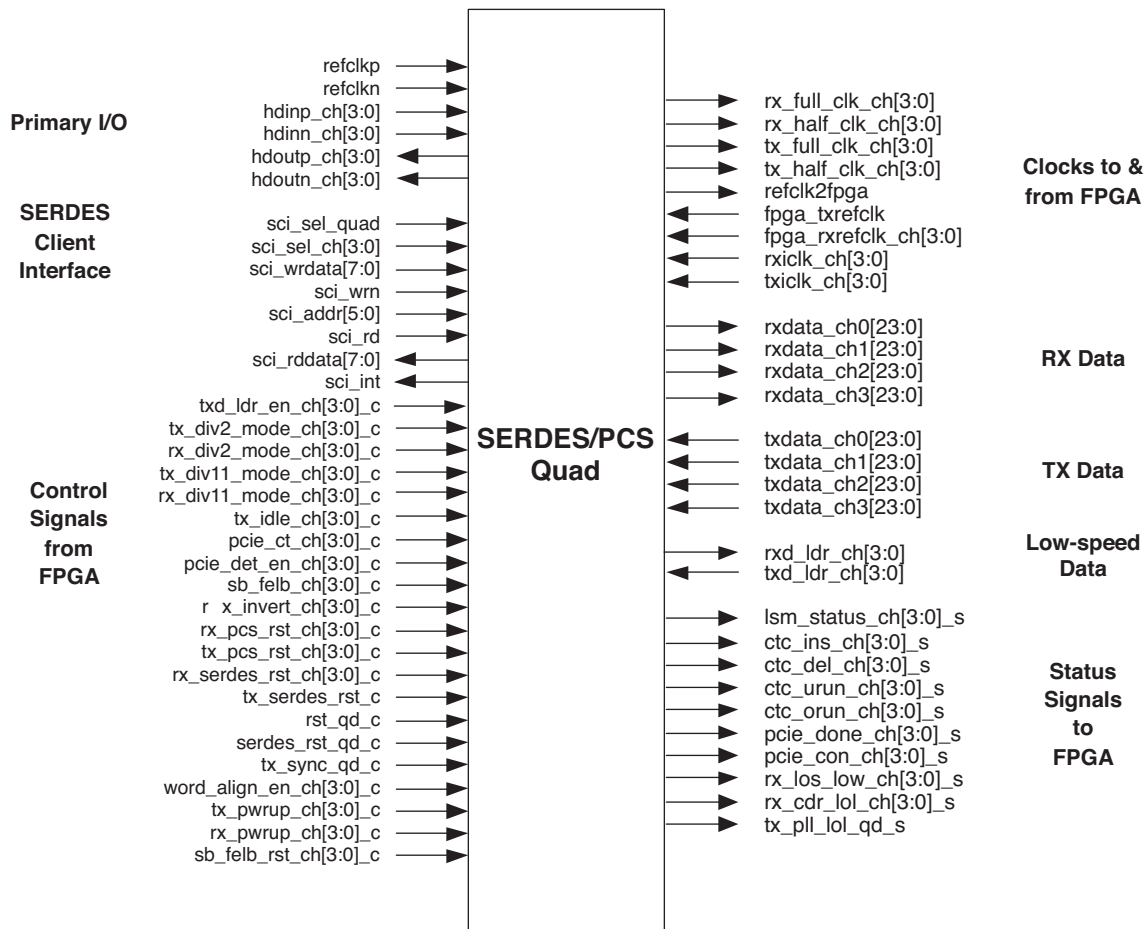
{signal}\_s is the status the signal to the FPGA core from the FPGA bridge. All of the status signals are asynchronous from the SERDES/PCS. These should be synchronized to a clock domain before they are used in the FPGA design.

Please refer to the Mode-Specific Control/Status Signals section of this document for detailed information about control and status signals.

## SERDES/PCS

The quad contains four channels with both RX and TX circuits, and an auxiliary channel that contains the TX PLL. The reference clock to the TX PLL can be provided either by the primary differential reference clock pins or by the FPGA core. The quad SERDES/PCS macro performs the serialization and de-serialization function for four lanes of data. In addition, the TxPLL within the SERDES/PCS block provides the system clock for the FPGA logic. The quad also supports both full-data-rate and half-data-rate modes of operation on each TX and RX circuit independently. The block-level diagram is shown in Figure 8-4.

**Figure 8-4. SERDES/PCS Block Signal Interface**



## I/O Descriptions

Table 8-5 lists all default and optional inputs and outputs to/from a PCS quad. Users can choose optional ports for a PCS quad using the IPexpress™ GUI.

**Table 8-5. SERDES\_PCS I/O Descriptions**

Signal Name	I/O	Type	Description
<b>Primary I/O, SERDES Quad</b>			
hdinp_ch0	I	Channel	High-speed CML input, positive, channel 0
hdinn_ch0	I	Channel	High-speed CML input, negative, channel 0
hdinp_ch1	I	Channel	High-speed CML input, positive, channel 1
hdinn_ch1	I	Channel	High-speed CML input, negative, channel 1
hdinp_ch2	I	Channel	High-speed CML input, positive, channel 2
hdinn_ch2	I	Channel	High-speed CML input, negative, channel 2
hdinp_ch3	I	Channel	High-speed CML input, positive, channel 3
hdinn_ch3	I	Channel	High-speed CML input, negative, channel 3
hdoutp_ch0	O	Channel	High-speed CML output, positive, channel 0
hdoutn_ch0	O	Channel	High-speed CML output, negative, channel 0
hdoutp_ch1	O	Channel	High-speed CML output, positive, channel 1
hdoutn_ch1	O	Channel	High-speed CML output, negative, channel 1
hdoutp_ch2	O	Channel	High-speed CML output, positive, channel 2
hdoutn_ch2	O	Channel	High-speed CML output, negative, channel 2
hdoutp_ch3	O	Channel	High-speed CML output, positive, channel 3
hdoutn_ch3	O	Channel	High-speed CML output, negative, channel 3
refclkp	I	Quad	Reference Clock input, positive, dedicated CML input
refclkn	I	Quad	Reference Clock input, negative, dedicated CML input
<b>Receive / Transmit Data Bus (See Tables 8-3 and 8-4 for Detailed Data Bus Usage)</b>			
rxdata_ch0[23:0]	O	Channel	Data signals for the channel 0 receive path
rxdata_ch1[23:0]	O	Channel	Data signals for the channel 1 receive path
rxdata_ch2[23:0]	O	Channel	Data signals for the channel 2 receive path
rxdata_ch3[23:0]	O	Channel	Data signals for the channel 3 receive path
txdata_ch0[23:0]	I	Channel	Data signals for the channel 0 transmit path
txdata_ch1[23:0]	I	Channel	Data signals for the channel 1 transmit path
txdata_ch2[23:0]	I	Channel	Data signals for the channel 2 transmit path
txdata_ch3[23:0]	I	Channel	Data signals for the channel 3 transmit path
<b>Control Signals</b>			
tx_idle_ch[3:0]_c	I	Channel	Controls transmission of electrical idle by SERDES transmitter. 1 = Force SERDES transmitter to output electrical idle 0 = Normal operation
pcie_det_en_ch[3:0]_c	I	Channel	FPGA logic (user logic) informs the SERDES block that it will be requesting for a PCI Express Receiver Detection operation. 1 = Enable PCI Express receiver detect, 0 = Normal operation
pcie_ct_ch[3:0]_c	I	Channel	1 = Request transmitter to do far-end receiver detection 0 = Normal data operation
rx_invert_ch[3:0]_c	I	Channel	Control the inversion of received data. 1 = Invert the data, 0 = Don't invert the data
word_align_en_ch[3:0]_c	I	Channel	Control comma aligner. 1 = Enable comma aligner, 0 = Lock comma aligner at current position
sb_felb_ch[3:0]_c	I	Channel	SERDES Bridge Parallel Loopback 1 = Enable loopback from RX to TX, 0 = Normal data operation

**Table 8-5. SERDES\_PCS I/O Descriptions (Continued)**

Signal Name	I/O	Type	Description
sb_felb_rst_ch[3:0]_c	I	Channel	SERDES Bridge Parallel Loopback FIFO Clear 1 = Reset Loopback FIFO, 0 = Normal Loopback operation
tx_sync_qd_c	I	Quad	Serializer Reset Transition = Reset, Level = Normal Operation
rx_div2_mode_ch[3:0]_c	I	Channel	Receiver Rate Mode Select (Full/Half Rate) 1 = Half Rate, 0 = Full Rate
tx_div2_mode_ch[3:0]_c	I	Channel	Transmitter Rate Mode Select (Full/Half Rate) 1 = Half Rate, 0 = Full Rate
rx_div11_mode_ch[3:0]_c	I	Channel	Receiver Rate Mode Select (Div11/Full Rate) 1 = Div11 Rate, 0 = Full Rate
tx_div11_mode_ch[3:0]_c	I	Channel	Transmitter Rate Mode Select (Div11/Full Rate) 1 = Div11 Rate, 0 = Full Rate
txd_ldr_en_ch[3:0]_c	I	Channel	Low Data Rate TX Serial Path Enable 1 = Enable, 0 = Disable
<b>Reset Signals</b>			
rx_pcs_rst_ch[3:0]_c	I	Channel	Active-high, asynchronous input. Resets individual receive channel logic only in PCS.
tx_pcs_rst_ch[3:0]_c	I	Channel	Active-high, asynchronous input. Resets individual transmit channel logic only in PCS.
rx_serdes_rst_ch[3:0]_c	I	Channel	Active-high. Resets selected digital logic in the SERDES receive channel.
tx_serdes_rst_c	I	Quad	Active-high. Resets selected digital logic in all SERDES transmit channels.
rst_qd_c	I	Quad	Active-high, asynchronous input. Resets all SERDES channels including the auxiliary channel and PCS.
serdes_rst_qd_c	I	Quad	Active-high, asynchronous input to SERDES quad. Resets all SERDES channels including the Quad channel but not PCS logic.
tx_pwrup_ch[3:0]_c	I	Channel	Active-high transmit channel power up. 0 = Transmit channel power-down.
rx_pwrup_ch[3:0]_c	I	Channel	Active-high receive channel power up. 0 = Receive channel power-down.
<b>Status Signals</b>			
pcie_done_ch[0:3]_s	O	Channel	1 = Far-end receiver detection complete 0 = Far-end receiver detection incomplete
pcie_con_ch[3:0]_s	O	Channel	Result of far-end receiver detection. 1 = Far-end receiver detected 0 = Far end receiver not detected.
rx_los_low_ch[3:0]_s	O	Channel	Loss of signal (LO THRESHOLD RANGE) detection for each channel.
lsm_status_ch[3:0]_s	O	Channel	1 = Lane is synchronous to commas 0 = Lane has not found comma
ctc_urrun_ch[3:0]_s	O	Channel	1 = Receive clock compensator FIFO underrun error 0 = No FFIFO errors
ctc_orun_ch[3:0]_s	O	Channel	1 = Receive clock compensator FIFO overrun error 0 = No FIFO errors
rx_cdr_lol_ch[3:0]_s	O	Channel	1 = Receive CDR loss of lock 0 = Lock maintained
tx_pll_lol_qd_s	O	Quad	1 = Transmit PLL loss of lock 0 = Lock maintained
ctc_ins_ch[3:0]_s	O	Channel	1 = SKIP Character Added by CTC
ctc_del_ch[3:0]_s	O	Channel	1 = SKIP Character Deleted by CTC

**Table 8-5. SERDES\_PCS I/O Descriptions (Continued)**

Signal Name	I/O	Type	Description
<b>FPGA Interface Clocks</b>			
rx_full_clk_ch[3:0]	O	Channel	Receive channel recovered clock. In user mode, the source is always the channel's recovered clock. For standards such as 10 GbE that support clock compensation, the source is the respective transmit channel's system clock. For PCS bypass modes, it is also the transmit system clock, thus requiring raw mode to actually be done using either 8b10b mode with the 8b10b decoder disabled (10-bit or 20-bit data path).
rx_half_clk_ch[3:0]	O	Channel	Receive channel recovered half clock. In 2:1 gearing mode, it is a divide-by-2 output.
tx_full_clk_ch[3:0]	O	Channel	TX PLL full rate clock. Only tx_full_clk_ch0 can drive the primary clock routing directly. All of the tx_full_clk_ch[3:0] signals can drive the secondary clock routing by applying a USE SECONDARY clocking preference. <sup>3</sup>
tx_half_clk_ch[3:0]	O	Channel	TX PLL half clock. Only tx_half_clk_ch0 can drive the primary clock routing directly. All of the tx_half_clk_ch[3:0] signals can drive the secondary clock routing by applying a USE SECONDARY clocking preference. <sup>3</sup>
refclk2fpga	O	Quad	Reference clock to FPGA core. If selected, this clock is always active as long as reference clock is active even when the quad is in power down mode.
fpga_rxrefclk_ch[3:0]	I	Quad	RX reference clock from FPGA logic, for CDR PLL
fpga_txrefclk	I	Quad	TX reference clock from FPGA logic, for TX SERDES PLL
ebrd_clk_ch[3:0] <sup>2</sup>	I	Channel	Receive channel clock input from FPGA for CTC FIFO read.
rxiclk_ch[3:0]	I	Channel	Receive channel clock input from FPGA. Used to clock the RX FPGA Interface FIFO with a clock synchronous to the reference and/or receive reference clock.
txiclk_ch[3:0]	I	Channel	Transmit channel clock input from FPGA. Per channel transmit clock inputs from FPGA. Used to clock the TX FPGA Interface FIFO with a clock synchronous to the reference clock. Also used to clock the RX FPGA Interface FIFO with a clock synchronous to the reference clock when CTC is used.
<b>Low-Speed Receive/Transmit Data and SERDES Client Interface Signals</b>			
rxd_ldr_ch[3:0]	O	Channel	Single-ended serial low data rate outputs (RX) to FPGA core
txd_ldr_ch[3:0]	I	Channel	Single-ended serial low data rate inputs (TX) from FPGA core
sci_wrdata[7:0]	I	—	Write data input
sci_wrn	I	—	Write input strobe
sci_sel_quad	I	—	Selects quad registers
sci_sel_ch[3:0]	I	—	Selects channel registers
sci_addr[5:0]	I	—	Address bus input
sci_rd	I	—	Read data select
sci_rddata[7:0]	O	—	Read data output
sci_int	O	—	Interrupt output

1. During configuration, both hdoutp and hdoutn are pulled high to VCCOB.
2. This clock is not provided in the wrapper module port list. Software automatically assigns the clock depending on the CTC mode. See the FPGA Interface Clocks section of this document for detailed information.
3. General routing is used to access the Secondary Clock Net. The user may get a PAR warning but the delay will be small enough to ignore in most applications. Use the Timing Preferences and review the Trace Report to make sure the timing is not violated.

---

## SERDES/PCS Functional Description

LatticeECP3 devices have from one to four quads of embedded SERDES/PCS logic. Each quad, in turn, supports four independent full-duplex data channels. A single channel can support a data link and each quad can support up to four such channels.

The embedded SERDES CDR PLLs and TX PLLs support data rates that cover a wide range of industry standard protocols.

Refer to Figure 8-3 for each of the items below.

- **SERDES**
  - Equalizer
  - CDR (Clock and Data Recovery)
  - Deserializer
  - PreEmphasis
  - Serializer
  - Two Serial Loopback modes, TX-to-RX or RX-to-TX
- **SERDES Bridge (SB)**
  - Inverter – Inverts receive data, required by PCI Express
  - SERDES Bridge Parallel Loopback
- **PCS Core**
  - Word Alignment
  - 8b10b Decoder
  - 8b10b Encoder
  - Link State Machine
  - Clock Tolerance Compensation
- **FPGA Bridge (FB)**
  - Downsample FIFO
  - Upsample FIFO

## SERDES

### Equalizer

As the data rate of digital transmission advances over Gbps, frequency-dependent attenuation results in severe intersymbol interference in the received signal and makes it mandatory to use an equalizer in the data transceiver to recover data correctly. Six pole positions are provided: Mid\_Low, Mid\_Med, Mid\_High, Long\_Low, Long\_Med, Long\_High frequency ranges.

A default selection of the pole position is included when a protocol standard is selected. This selection is based on most commonly used condition on user's system board. However, user may wish to optimize the signal by adjusting this setting. User has to perform bit-error-rate tests on all different pole settings to determine the optimal one.

### Pre-Emphasis

Pre-emphasis refers to a system process designed to increase the magnitude of some frequencies with respect to the magnitude of other frequencies. The goal is to improve the overall signal-to-noise ratio by minimizing the adverse effects of such phenomena as attenuation differences. Users can select up to 80% of pre-emphasis.

By default, pre-emphasis is not enabled. User can determine if the Tx drive signal has to route a long distance to the Rx side. If the routing trace length is long, he can try out different pre-emphasis settings to determine the best signal eye at the Rx-end of the trace.



## Reference Clocks

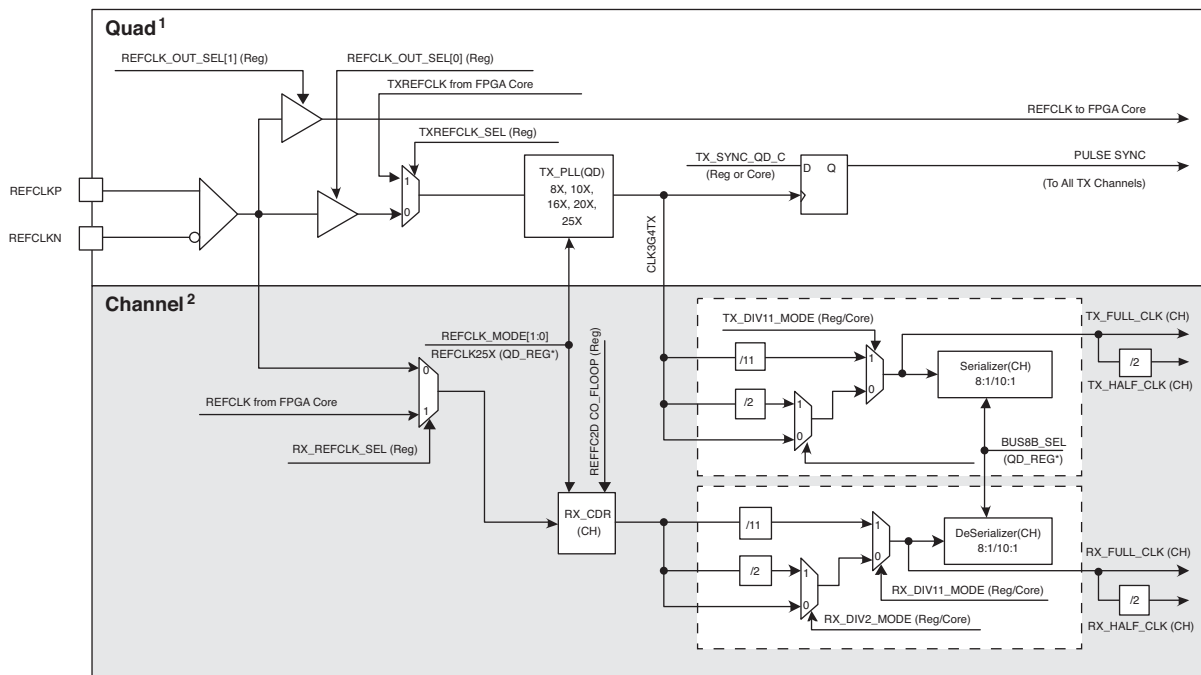
The SERDES quad contains four channels with both RX and TX circuits, and an auxiliary channel that contains the TX PLL. The reference clock to the TX PLL can be provided either by the primary differential reference clock pins, by the neighbor quad's reference clock or by the FPGA core. In addition, the PLL within the SERDES block provides an output clock that can be used as the system clock driving the FPGA fabric.

The reference clock to the RX can be provided either by the reference clock to the TX PLL or by the FPGA core. The reference clocks from the FPGA core to the TX PLL and to the RX may come from different sources.

## SERDES Clock Architecture

Figure 8-5 shows the overall SERDES clock architecture. The diagram is split into two halves, Quad and Channel. Only one per-channel section is shown for the sake of brevity. Also, the various control bits for the different blocks are shown. These could be quad-based control register bits or channel-based control register bits. In some cases, it can be channel control port based. Some are a combination of both register and control ports. Using both modes enables dynamic control of certain functional properties.

**Figure 8-5. SERDES Clock Architecture**



1. All control bits are quad based.
2. All control bits are channel based, except as indicated (\*).
3. These clocks are user-transparent.

The main components of the clock architecture include:

- Per RX and per TX Divider (DIV) modes – DIV2, DIV11
- Multi-quad REFCLK connections
- Multiple channel transmit synchronization using the tx\_sync\_qd\_c signal from the FPGA
- OOB low data rate application support

## Rate Modes

The TX in each channel can be independently programmed to run at one of the following rates:

- FULL\_RATE
- HALF\_RATE (DIV2)
- DIV11

The RX can also use a separate reference clock per channel, allowing the transmitter and receiver to run at completely different rates.

It is important to note that the PLL VCO is left untouched, supporting the highest rate for that protocol. All divided rates required by that protocol are supported by programming the divider mux selects. This enables very quick data rate changeover since the PLL does not need to be reprogrammed. This is valuable in many applications.

*Note: The LatticeECP3 PCS cannot handle a change in rates at runtime. Simply changing the refclk rate will not allow the SERDES to work at a new range. The SERDES must be reprogrammed and this can only be done with a new bitstream.*

The TX PLL and the four CDR PLLs generally run at the same frequency, which is a multiple of the reference clock frequency. Table 8-6 illustrates the various clock rate modes possible. The bit clock indicated is a multiple of the reference clock frequency.

**Table 8-6. TXPLL and RX CDRPLL Supported Modes**

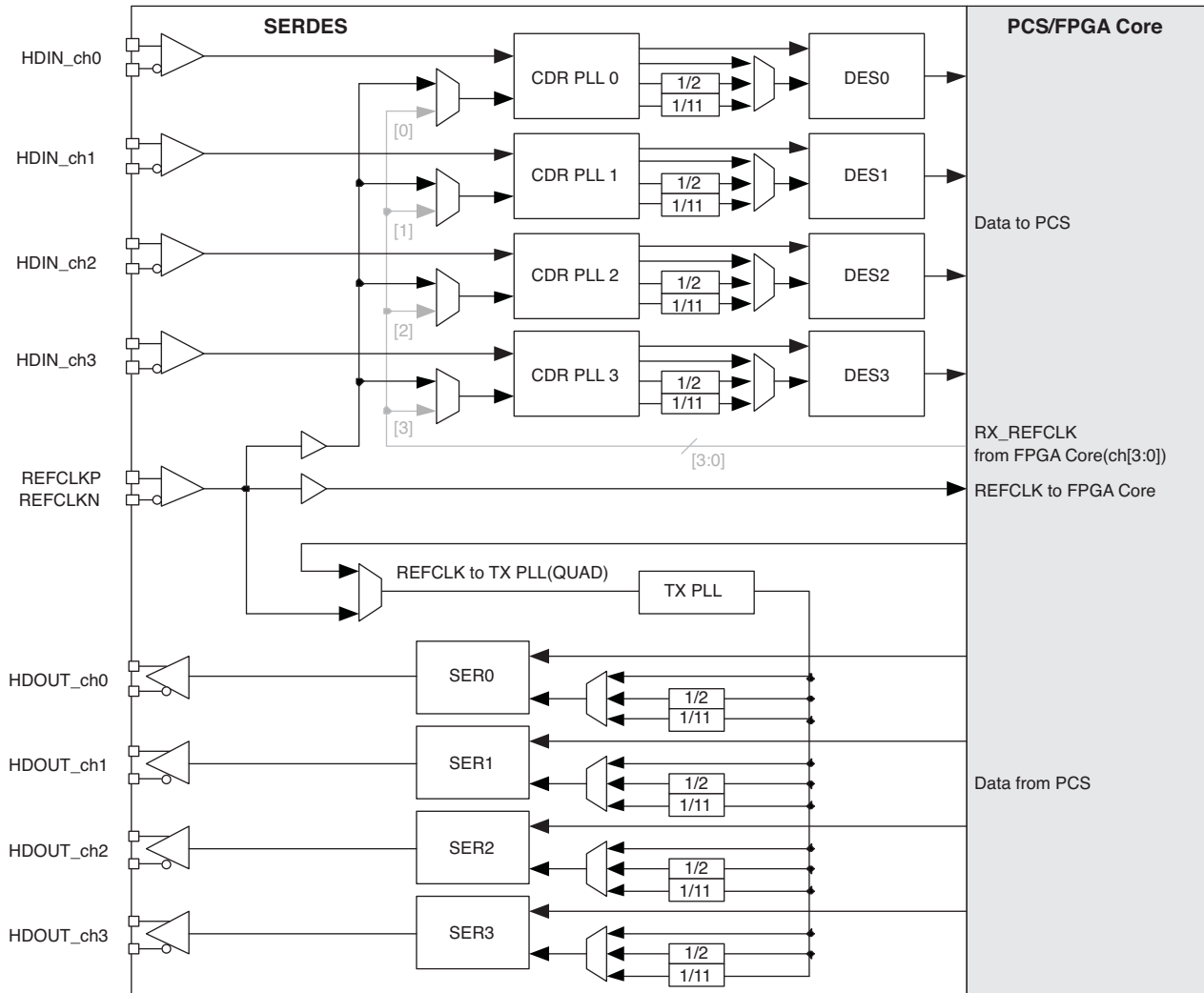
Reference Clock Mode	refclkPmode (Quad)	Bus_width	Bit Clock (Full Rate)	Bit Clock (div2, div11)
20x	0	10	Refclk x 20	Refclk x 10
16x	0	8	Refclk x 16	Refclk x 8
10x	1	10	Refclk x 10	Refclk x 5
8x	1	8	Refclk x 8	Refclk x 4
25x	—	8	Refclk x 25	Refclk x 12.5
25x	—	10	Refclk x 25	Refclk x 12.5
20x	0	10	Refclk x 20	Refclk x 20/11 <sup>1</sup>

1. DIV11 mode.

### Reference Clock from an FPGA Core

As described in Figure 8-5, the TX reference clock can be brought from the FPGA core. In this case, the extra jitter caused by the FPGA resources to route the clock to the SERDES will be passed onto the transmit data and may violate TX jitter specifications on a case-by-case basis. Caution should be used when using an FPGA-originated SERDES TX reference clock.

**Figure 8-6. Reference Clock Block Diagram**

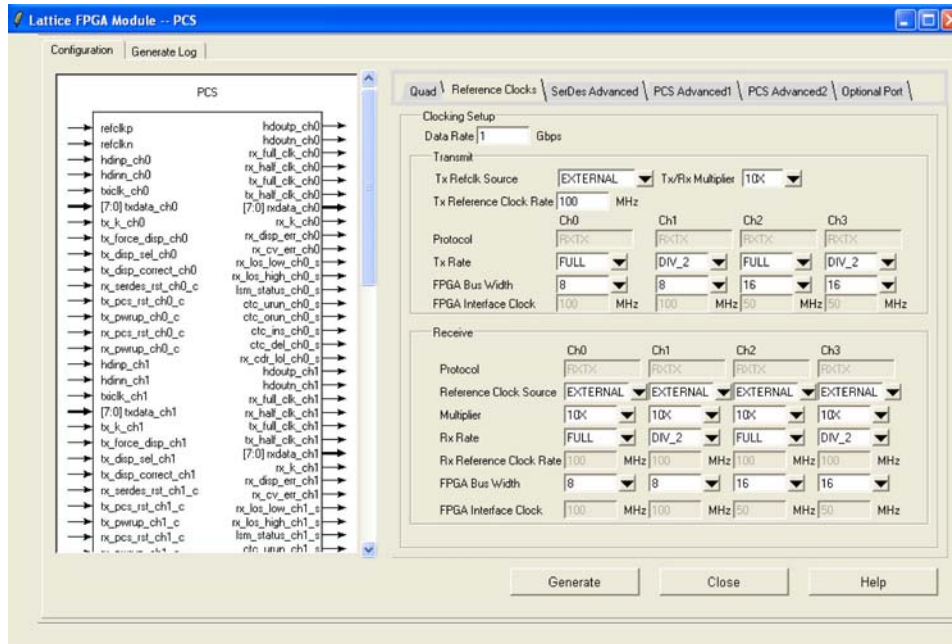


## Full Data, Div 2 and Div 11 Data Rates

Each TX Serializer and RX Deserializer can be split into full data rate and div2 rate or div11 rate depending on the protocol, allowing for different data rates in each direction and in each channel. Refer to Figure 8-6 for further information.

As shown in Figure 8-7, all four channels can be configured differently.

**Figure 8-7. Example of Full Data Rate and Half Data Rate in the IPexpress GUI**



The actual data rate and FPGA interface clock rate for this example are described in Table 8-7. The IPexpress GUI will be discussed in detail later in this document.

**Table 8-7. Clock Rate Example**

Channel	Data Rate	Reference Clock Multiplier	Data Rate Mode	Calculated Reference Clock Rate	FPGA Interface Data Bus Width	FPGA Interface Clock Rate	tx_full_clk	tx_half_clk
Channel 0	1 Gbps	10 x	FULL	100 MHz	8 (10) <sup>3</sup>	100 MHz	100 MHz	50 MHz
Channel 1	500 Mbps	10 x	DIV2	100 MHz	8 (10)	50 MHz	50 MHz	25 MHz
Channel 2	1 Gbps	10 x	FULL	100 MHz	16 (20)	50 MHz	100 MHz	50 MHz
Channel 3	500 Mbps	10 x	DIV2 <sup>2</sup>	100 MHz	16 (20)	25 MHz	50 MHz	25 MHz

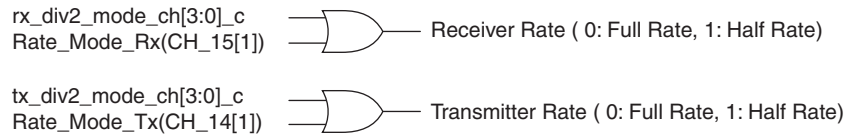
1. The clocks in the shaded cells are used as the FPGA interface clocks in each mode.
2. In DIV2 mode, the tx\_full\_clk is adjusted to half rate. tx\_half\_clk is used for the 16-bit bus interface only.
3. 10-bit SERDES only mode or SDI mode.

## Dynamic Switching Between Full Rate and Half Rate (DIV2)

This section describes how to switch between Full Rate and Half Rate (DIV) dynamically.

The two rate mode control signals are or'ed as shown in Figure 8-8.

**Figure 8-8. Rate Mode Control Signals**



tx\_div2\_mode\_chx\_c is an input control signal to the TX path from the FPGA fabric.

rx\_div2\_mode\_chx\_c is an input control signal to the RX path from the FPGA fabric.

Rate\_Mode\_Tx(CH\_14[1]) is the Control Register bit for the TX path.

Rate\_Mode\_Rx(CH\_15[1]) is the Control Register bit for the RX path.

In the rx lane, the pcs\_rst should be applied after switching.

In the tx lane, no reset is required for the new rate to take place.

## Reference Clock Sources

### refclkp, refclkn

Dedicated CML input. This is the first choice unless different clock sources for RX and TX are used. The clock signal may be CML, LVDS or LVPECL. Refer to TN1114, [Electrical Recommendations for Lattice SERDES](#), for example interface circuits.

### fpga\_txrefclk, fpga\_rxrefclk

Reference clock from FPGA logic. The Primary Clock pad (PCLK) should be used as the clock input pin to the FPGA. The clock signal may be CML, LVDS, LVPECL or single-ended.

### FPGA PLL

When an FPGA PLL is used as the reference clock, the reference clock to the PLL should be assigned to a dedicated PLL input pad. The FPGA PLL output jitter may not meet system specifications at higher data rates. Use of an FPGA PLL is not recommended in jitter-sensitive applications.

## Spread Spectrum Clocking (SSC) Support

The ports on the two ends of Link must transmit data at a rate that is within 600ppm of each other at all times. This is specified to allow bit-rate clock source with a +/- 300ppm tolerance. The minimum clock period cannot be violated. The preferred method is to adjust the spread technique to not allow for modulation above the nominal frequency. The data rate can be modulated from +0% to -0.5% of the nominal data rate frequency, at a modulation rate in the range not exceeding 30KHz to 33KHz. Along with the +/- 300ppm tolerance limit, both ports require the same bit rate clock when the data is modulated with an SSC.

In PCI Express applications, the root complex is responsible for spreading the reference clock. The endpoint will use the same clock to pass back the spectrum through the TX. Thus, there is no need for a separate RXREFCLK. The predominant application is an add-in card. Add-in cards are not required to use the REFCLK from the connector but must receive and transmit with the same SSC as the PCI Express connector REFCLK.

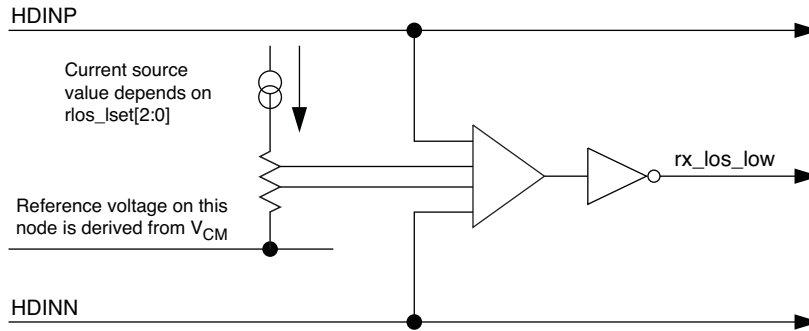
While the LatticeECP3 architecture will allow the mixing of a PCI Express channel and a Gigabit Ethernet, Serial RapidIO or SGMII channel within the same quad, using a PCI Express SSC as the transmit reference clock will cause a violation of the Gigabit Ethernet, Serial RapidIO and SGMII transmit jitter specifications.

## Loss of Signal

Each channel contains a programmable loss-of-signal detector as shown in Figure 8-9.

The loss-of-signal threshold depends on the value of the programmable current source. The current source value is chosen using the `rios_lset[2:0]` control bit. The result of threshold detection is indicated to the FPGA through the `rx_los_low` status signal.

**Figure 8-9. Loss of Signal Detector**



Note: `rx_los_low` shows that a signal has been detected for data rates above 1 Gbps with a maximum CID (Consecutive Identical Digits) of 7 bits (i.e., a minimum input signal transition density as is sent by 8b10b).

`rx_los_low` is supported with a default setting of `rios_lset[2:0] = 2`, except in PCI Express mode and SDI mode. In PCI Express mode, 2 and 3 are supported.

In SDI mode, it is recommended to use the carrier detect output signal (`/CD`) from the external SDI cable equalizer.

**Table 8-8. Response Time for Loss of Signal Detector**

Description	Min.	Typ.	Max.	Units
Time to detect that signal is lost ( <code>rx_los_low</code> 0 to 1)	—	8	10	ns
Time to detect that signal is present ( <code>rx_los_low</code> 1 to 0)	—	8	10	ns

## Loss Of Lock

Both the transmit PLL and the individual channel CDRs have digital counter-based, loss-of-lock detectors. If the transmit PLL loses lock, the loss-of-lock for the PLL is asserted and remains asserted until the PLL reacquires lock. If a CDR loses lock, the loss-of-lock for that channel is asserted and locking to the reference clock retrains the VCO in the CDR. When this is achieved, loss-of-lock for that channel is de-asserted and the CDR is switched back over to lock to the incoming data. The CDR will either remain locked to the data, or will go back out of lock again in which case the re-training cycle will repeat. For detailed information on CDR loss-of-lock, refer to the SERDES/PCS RESET section of this document.

**Table 8-9. Response Time for Loss-of-Lock Detector**

Description	Min.	Typ.	Max.	Units
Time to detect that loop is unlocked ( <code>tx_pll_lol</code> , <code>rx_cdr_lol</code> , 0 to 1)	—	200	500	us
Time to detect that loop is locked ( <code>tx_pll_lol</code> , <code>rx_cdr_lol</code> , 1 to 0)	—	200	500	us

## TX Lane-to-Lane Skew

A control signal, `tx_sync_qd_c`, resets all of the active TX channels to start serialization with bit 0. Most multi-channel protocol standards have requirements to ensure that the TX lane-to-lane skew is within a certain specification.

The reset to the TX serializers is generated either by toggling the `tx_sync_qd_c` signal or by a transition in PLL loss of lock.

---

## SERDES PCS Configuration Setup

The LatticeECP3 PCS can be configured for use in various applications. Setup is chosen with the IPexpress module generation tool which allows the user to select the mode and feature options for the PCS. Option selections are saved in an auto-configuration file which is subsequently used by the Bitstream Generator to write the user selections into the bitstream. To change PCS option selections it is recommended that the user rerun IPexpress to regenerate a PCS module and create a new auto-configuration file. Some options can be changed by manually editing the auto-configuration file before running the Bitstream Generator. After configuration, PCS options can be changed dynamically by writing to PCS registers via the optional SERDES Client Interface bus. The SERDES Client Interface allows the SERDES/PCS quad to be controlled by registers as opposed to configuration memory cells. A table of control and status registers accessible through the SCI is provided in Appendix A.

### Auto-Configuration File

Initial register setup for each PCS mode can be performed by using the auto-configuration feature in IPexpress. The module generator provides an auto-configuration file which contains the quad and channel register settings for the chosen mode. This file can be referred to for front-end simulation and also can be integrated into the bitstream. When an auto-configuration file is integrated into the bitstream all the quad and channel registers will be set to values defined in the auto-configuration file during configuration. The SCI must be included in a design if the user needs to change control registers or monitor status registers during operation.

### Transmit Data

The PCS quad transmit data path consists of an 8b10b encoder and serializer per channel.

#### 8b10b Encoder

This module implements an 8b10b encoder as described within the IEEE 802.3ae-2002 1000BASE-X specification. The encoder performs the 8-bit to 10-bit code conversion as described in the specification, along with maintaining the running disparity rules as specified. The 8b10b encoder can be bypassed on a per-channel basis by setting the attribute CHx\_8B10B to "BYPASS" where x is the channel number.

#### Serializer

The 8b10b encoded data undergoes parallel-to-serial conversion and is transmitted off-chip via the embedded SERDES.

### Receive Data

The PCS quad receive data path consists of the following sub-blocks per channel: Deserializer, Word Aligner, 8b10b Decoder, Optional Link State Machine, and Optional Receive Clock Tolerance Compensation (CTC) FIFO.

#### Deserializer

Data is brought on-chip to the embedded SERDES where it goes from serial to parallel.

#### Word Alignment (Byte Boundary Detect)

This module performs the comma codeword detection and alignment operation. The comma character is used by the receive logic to perform 10-bit word alignment upon the incoming data stream. The comma description can be found in section 36.2.4.9 of the 802.3.2002 1000BASE-X specification as well as section 48.2.6.3, Figure 48-7 of the 10GBASE-X specification.

A number of programmable options are supported within the word alignment module:

- Word alignment control from either embedded Link State Machine (LSM) or from FPGA control. Supported in 8-bit SERDES Only, 10-bit SERDES Only and SDI modes, in addition to 8b10b packet mode.

- Ability to set two programmable word alignment characters (typically one for positive and one for negative disparity) and a programmable per bit mask register for alignment comparison. Alignment characters and the mask register is set on a per quad basis. For many protocols, the word alignment characters can be set to “XX0000011” (jhgfi edcba bits for positive running disparity comma character matching code groups K28.1, K28.5, and K28.7) and “XX01111100” (jhgfi edcba bits for negative running disparity comma character matching code groups K28.1, K28.5, and K28.7). However, the user can define any 10-bit pattern.
- The first alignment character is defined by the 10-bit value assigned to attribute COMMA\_A. This value applies to all channels in a PCS quad.
- The second alignment character is defined by the 10-bit value assigned to attribute COMMA\_B. This value applies to all channels in a PCS quad.
- The mask register defines which word alignment bits to compare (a ‘1’ in a bit of the mask register means check the corresponding bit in the word alignment character register). The mask registers defined by the 10-bit value assigned to attribute COMMA\_M. This value applies to all channels in a PCS quad. When the attribute CHx\_RXWA (word alignment) is set to “ENABLED” and CHx\_ILSM (Internal Link State Machine) is set to “ENABLED”, one of the protocol-based Link State Machines will control word alignment. For more information on the operation of the protocol-based Link State Machines, see the Protocol-Specific Link State Machine section below.

### 8b10b Decoder

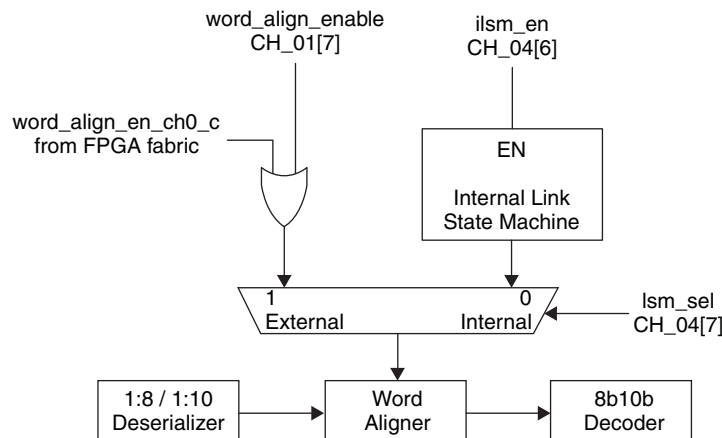
The 8b10b decoder implements an 8b10b decoder operation as described with the IEEE 802.3-2002 specification. The decoder performs the 10-bit to 8-bit code conversion along with verifying the running disparity. When a code violation is detected, the receive data, rxdata, is set to 0xEE with rx\_k\_chn set to ‘1’.

### External Link State Machine Option

When the attribute CHx\_ILSM (Internal Link State Machine) is set to DISABLED, and CHx\_RXWA (word alignment) is set to ENABLED, the control signal word\_align\_en\_ch(0-3)\_c is used to enable the word aligner. This signal should be sourced from a FPGA external Link State Machine implemented in the FPGA fabric. When the word\_align\_en\_ch(0-3)\_c is high, the word aligner will lock alignment and stay locked. It will stop comparing incoming data to the user-defined word alignment characters and will maintain current alignment on the first successful compare to either COMMA\_A or COMMA\_B. If re-alignment is required, pulse the word\_align\_en\_ch(0-3)\_c low to high. The word aligner will re-lock on the next match to one of the user-defined word alignment characters. If desired, word\_align\_en\_ch(0-3)\_c can be controlled by a Link State Machine implemented externally to the PCS quad to allow a change in word alignment only under specific conditions.

Figure 8-10 illustrates the Link State Machine options.

**Figure 8-10. PCS Word Aligner and Link State Machine Options**





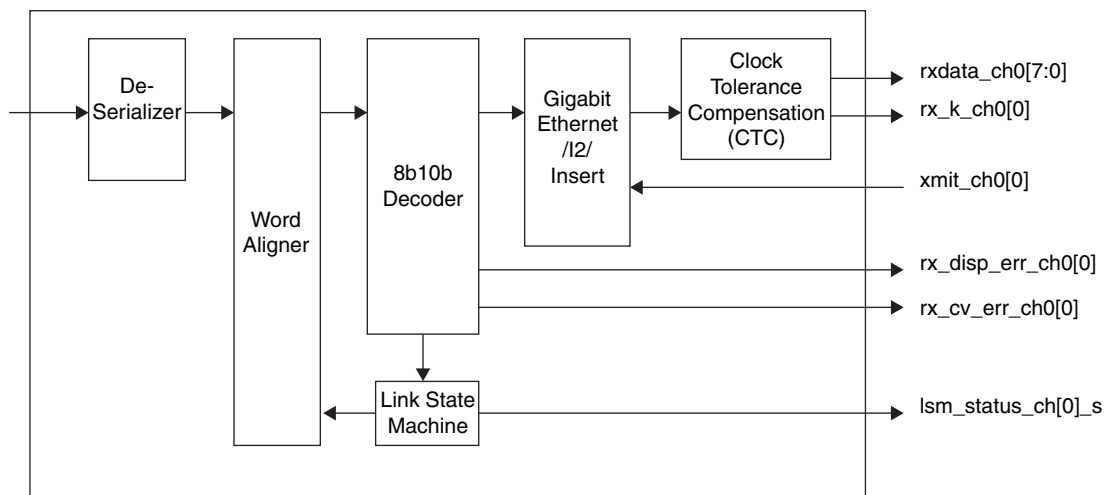
When a Link State Machine is selected and enabled for a particular channel, that channel's lsm\_status\_ch(0-3)\_s status signal will go high upon successful link synchronization.

Note that the lsm\_status\_ch(0:3)\_s status signal may have glitches. User should add deglitch logic on this output to ensure stable signal transition.

### Idle Insert for Gigabit Ethernet Mode

The PCS set to Gigabit Ethernet Mode provides for insertion of /I2/ symbols into the receive data stream for auto-negotiation. Gigabit Ethernet auto-negotiation is performed in soft logic. This function inserts a sequence of 8 /I2/ ordered sets every 2048 clock cycles. /I2/ insertion is controlled by the xmit\_ch(0-3) input to the PCS which is driven from the autonegotiation soft logic. Figure 8-11 shows one channel (channel 0 in this example) of receive logic when the PCS is set to Gigabit Ethernet Mode showing these control/status signals.

**Figure 8-11. PCS Receive Path for Gigabit Ethernet Mode (Channel 0 Example)**



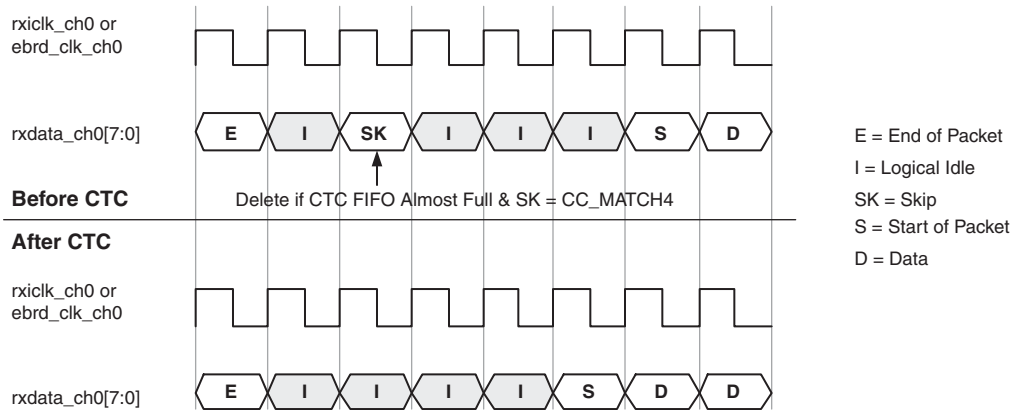
### Clock Tolerance Compensation

The Clock Tolerance Compensation (CTC) module performs clock rate adjustment between the recovered receive clocks and the locked reference clock. Clock compensation is performed by inserting or deleting bytes at pre-defined positions, without causing loss of packet data. A 16-byte CTC FIFO is used to transfer data between the two clock domains and will accommodate clock differences of up to the specified ppm tolerance for the LatticeECP3 SERDES (see DC and Switching Characteristics section of the [LatticeECP3 Family Data Sheet](#)).

A channel has the Clock Tolerance Compensation block enable when that channel's attribute CHx\_CTC is set to "ENABLED". The CTC is bypassed when that channel's attribute CHx\_CTC is set to "DISABLED".

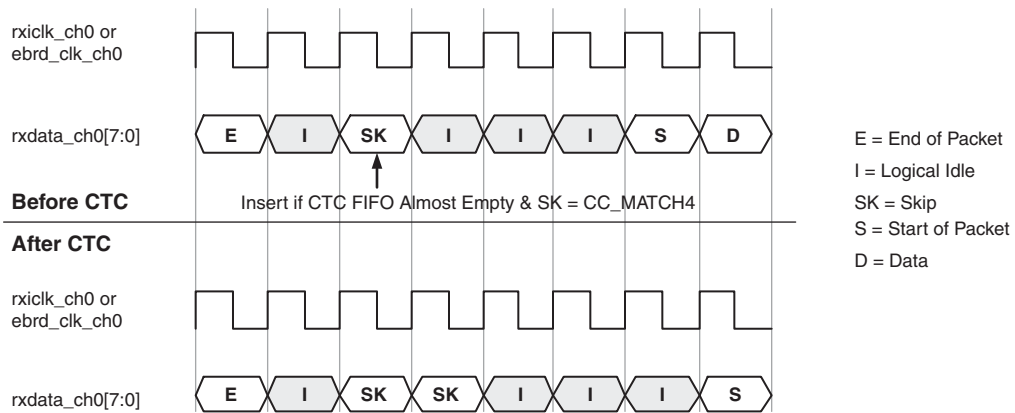
A diagram illustrating 1-byte deletion is shown in Figure 8-12.

**Figure 8-12. Clock Tolerance Compensation 1-Byte Deletion Example**



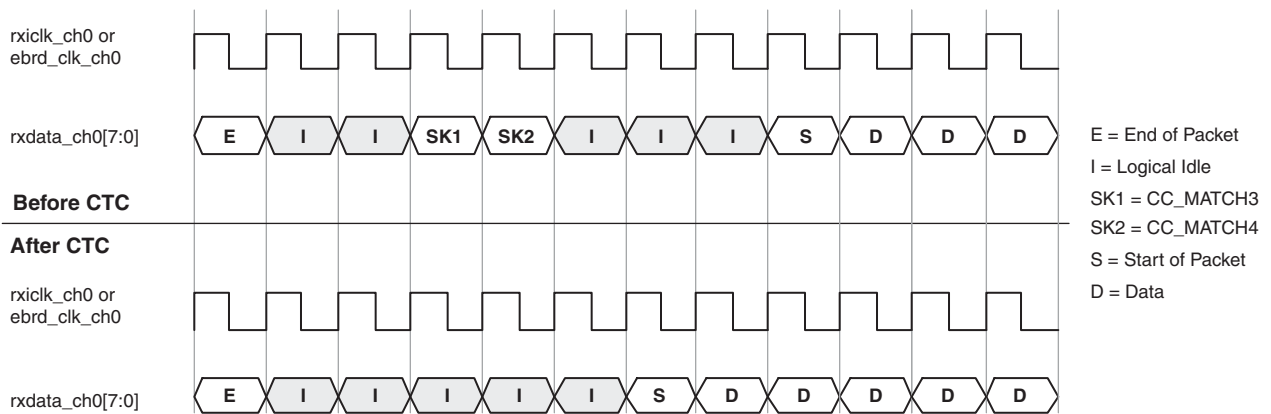
A diagram illustrating 1-byte insertion is shown in Figure 8-13.

**Figure 8-13. Clock Tolerance Compensation 1-Byte Insertion Example**



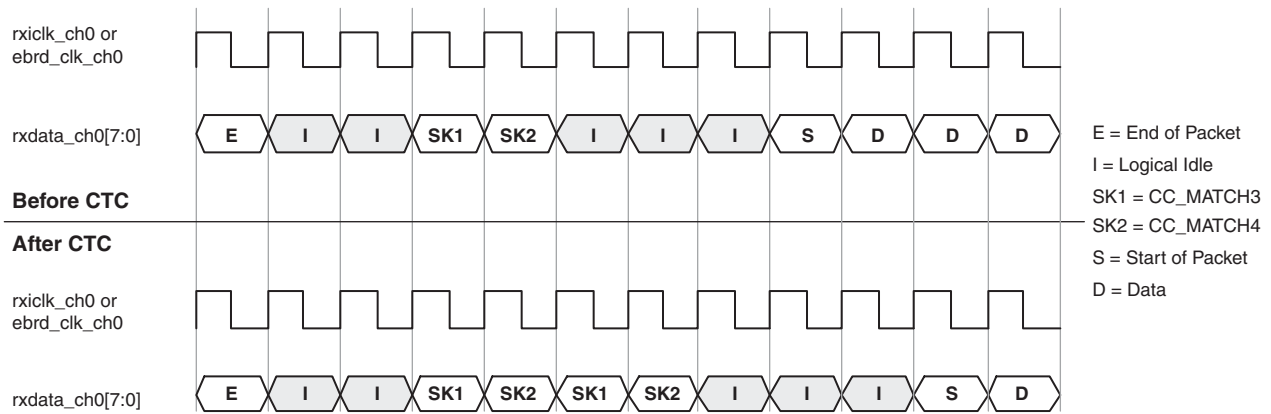
A diagram illustrating 2-byte deletion is shown in Figure 8-14.

**Figure 8-14. Clock Tolerance Compensation 2-Byte Deletion Example**



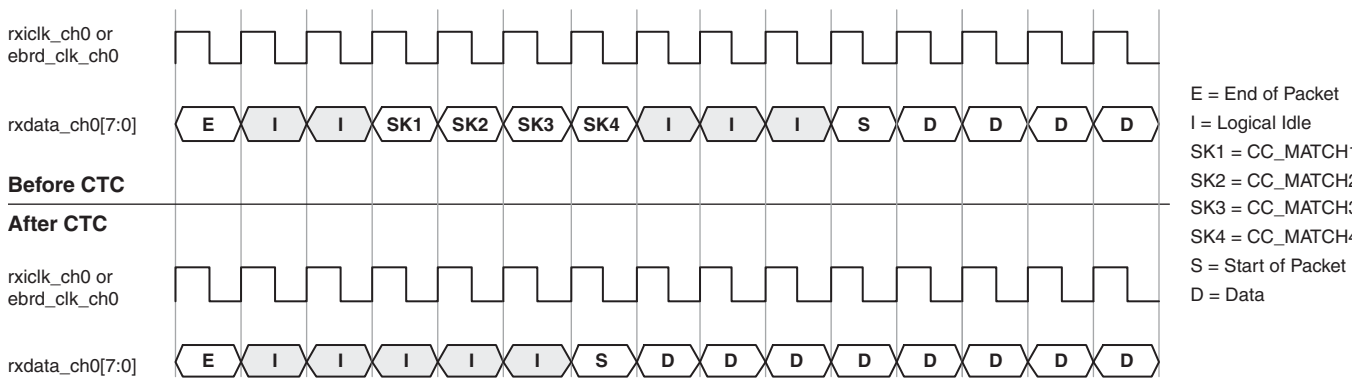
A diagram illustrating 2-byte insertion is shown in Figure 8-15.

**Figure 8-15. Clock Tolerance Compensation 2-Byte Insertion Example**



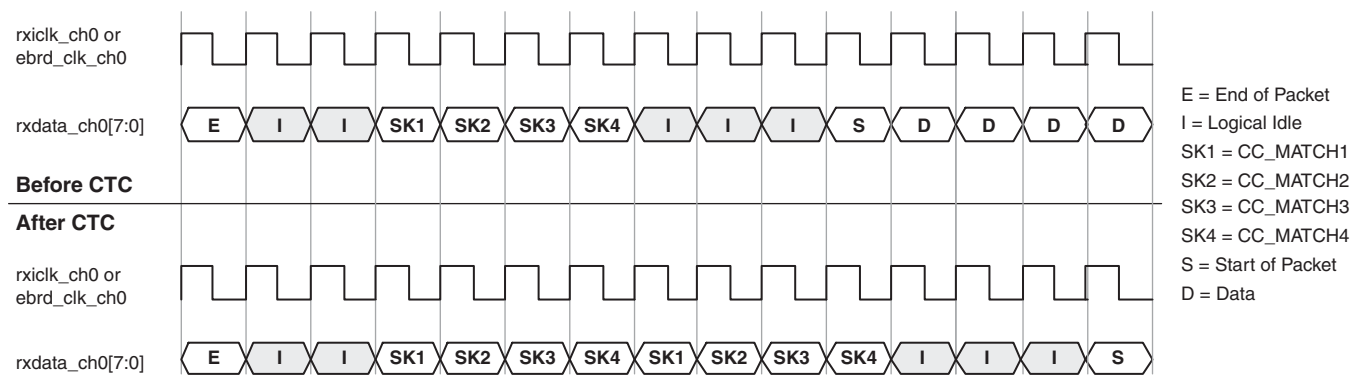
A diagram illustrating 4-byte deletion is shown in Figure 8-16.

**Figure 8-16. Clock Tolerance Compensation 4-Byte Deletion Example**



A diagram illustrating 4-byte insertion is shown in Figure 8-17.

**Figure 8-17. Clock Tolerance Compensation 4-Byte Insertion Example**



When the CTC is used, the following settings for clock compensation must be set, as appropriate, for the intended application:

- **Set the insertion/deletion pattern length using the CC\_MATCH\_MODE attribute.** This sets the number of skip bytes the CTC compares to before performing an insertion or deletion. Values for CC\_MATCH\_MODE are “1” (1-byte insertion/deletion), “2” (2-byte insertion/deletion), and “4” (4-byte insertion/deletion). The minimum interpacket gap must also be set as appropriate for the targeted application. The interpacket gap is set by assigning values to attribute CC\_MIN\_IPG. Allowed values for CC\_MIN\_IPG are “0”, “1”, “2”, and “3”. The minimum allowed interpacket gap after skip character deletion is performed based on these attribute settings is described in Table 8-10.
- **The skip byte or ordered set must be set corresponding to the CC\_MATCH\_MODE chosen.** For 4-byte insertion/deletion (CC\_MATCH\_MODE = “4”), the first byte must be assigned to attribute CC\_MATCH1, the second byte must be assigned to attribute CC\_MATCH2, the third byte must be assigned to attribute CC\_MATCH3, and the fourth byte must be assigned to attribute CC\_MATCH4. Values assigned are 10-bit binary values.

For example:

If a 4-byte skip ordered set is /K28.5/D21.4/D21.5/D21.5, then “CC\_MATCH1” should be “0110111100”, “CC\_MATCH2” = “0010010101”, “CC\_MATCH3” = “00101110101” and “CC\_MATCH4” = “00101110101”.

For a 2-byte insertion/deletion (CC\_MATCH\_MODE = “2”), the first byte must be assigned to attribute CC\_MATCH3, and the second byte must be assigned to attribute CC\_MATCH4.

For a 1-byte insertion/deletion (CC\_MATCH\_MODE = “1”), the skip byte must be assigned to attribute CC\_MATCH4.

- **The clock compensation FIFO high water and low water marks must be set to appropriate values for the targeted protocol.** Values can range from 0 to 15 and the high water mark must be set to a higher value than the low water mark (they should not be set to equal values). The high water mark is set by assigning a value to attribute CCHMARK. Allowed values for CCHMARK are hex values ranging from “0” to “F”. The low water mark is set by assigning a value to attribute CCLMARK. Allowed values for CCLMARK are hex values ranging from “0” to “F”.
- Clock compensation FIFO overrun can be monitored on a per-channel basis on the PCS/FPGA interface port labeled cc\_overrun\_ch(0-3) if “Error Status Ports” is selected when generating the PCS block with the module generator.
- Clock compensation FIFO underrun can be monitored on a per-channel basis on the PCS/FPGA interface port labeled cc\_underrun\_ch(0-3) if “Error Status Ports” is selected when generating the PCS block with the module generator.

## Calculating Minimum Interpacket Gap

Table 8-10 shows the relationship between the user-defined values for interpacket gap (defined by the CC\_MIN\_IPG attribute), and the guaranteed minimum number of bytes between packets after a skip character deletion from the PCS. The table shows the interpacket gap as a multiplier number. The minimum number of bytes between packets is equal to the number of bytes per insertion/deletion times the multiplier number shown in the table. For example, if the number of bytes per insertion/deletion is 4 (CC\_MATCH\_MODE is set to “4”), and the minimum interpacket gap attribute CC\_MIN\_IPG is set to “2”, then the minimum interpacket gap is equal to 4 (CC\_MATCH\_MODE = “4”) times 3 (Table 8-10 with CC\_MIN\_IPG = “2”) or 12 bytes. The PCS will not perform a skip character deletion until the minimum number of interpacket bytes have passed through the CTC.

**Table 8-10. Minimum Interpacket Gap Multiplier**

CC_MIN_IPG	Insert/Deletion Multiplier Factor
0	1x
1	2x
2	3x
3	4x

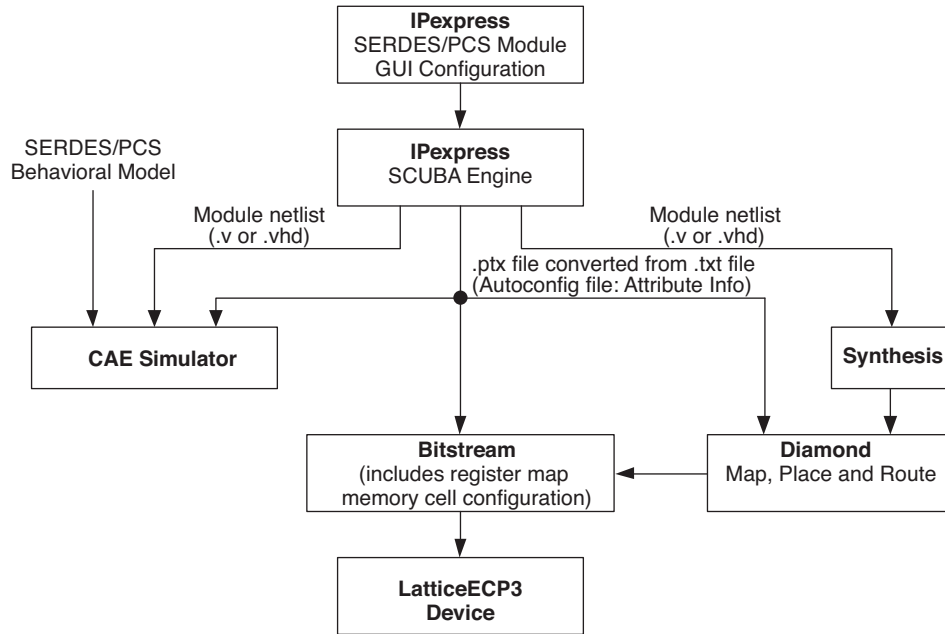
**Note on CTC support in the LatticeECP3-150EA device family with TW suffix:** For the initial release of LatticeECP3-150EA device with TW suffix, the CTC in the PCS is not supported. The CTC feature can be bypassed and implemented in soft IP. Many IP cores from Lattice implement the CTC logic in soft form.

## Using IPexpress with Diamond

IPexpress is used to create and configure SERDES and PCS blocks. Designers use the GUI to select the SERDES Protocol Standard for a particular quad or channel. IPexpress takes the input from this GUI and generates a configuration file (.txt file) and HDL netlist. The HDL model is used in the simulation and synthesis flow. The configuration file contains attribute-level map information. This file is input for simulation and the bitgen program. It is strongly recommended that designers make changes and updates in IPexpress and then regenerate the configuration file. In some exceptional situations, users can modify the configuration file.

Figure 8-18 shows the tools flow when using IPexpress to generate the SERDES/PCS block for the SERDES Protocol Standard.

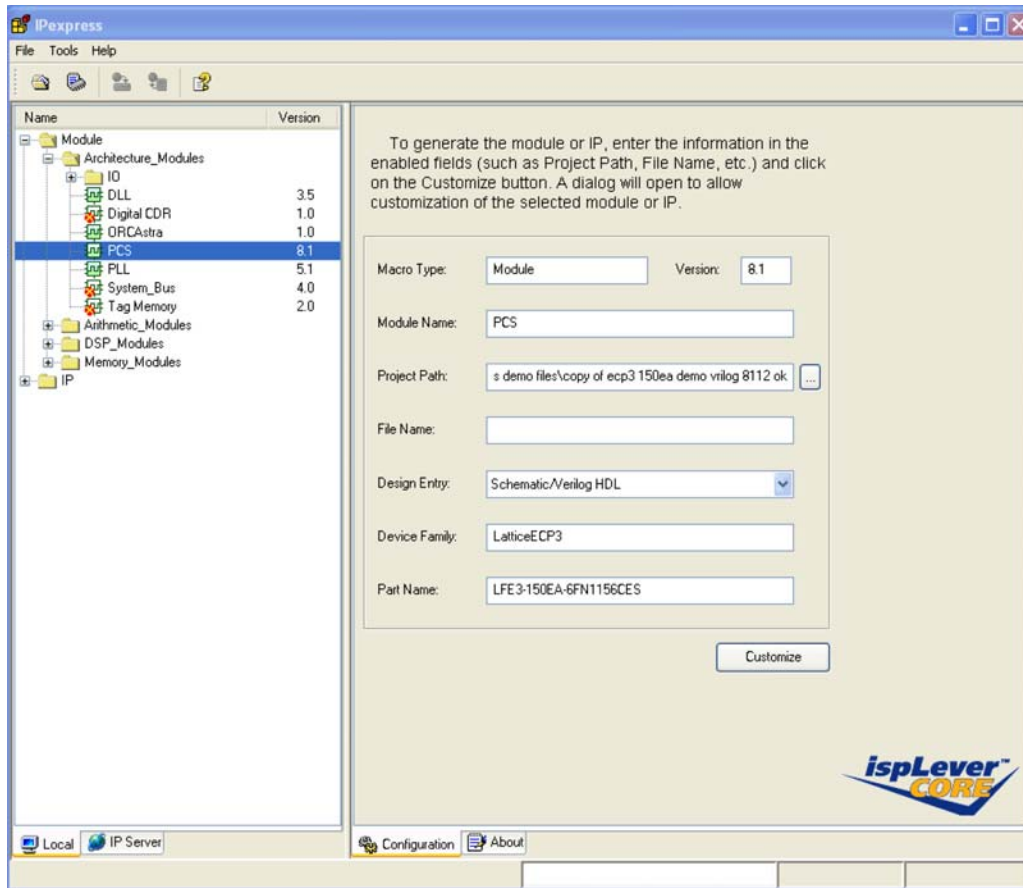
**Figure 8-18. SERDES\_PCS Diamond User Flow**



## PCS Module Generation in IPexpress

Figure 8-19 shows the main window when PCS is selected in the IPexpress GUI.

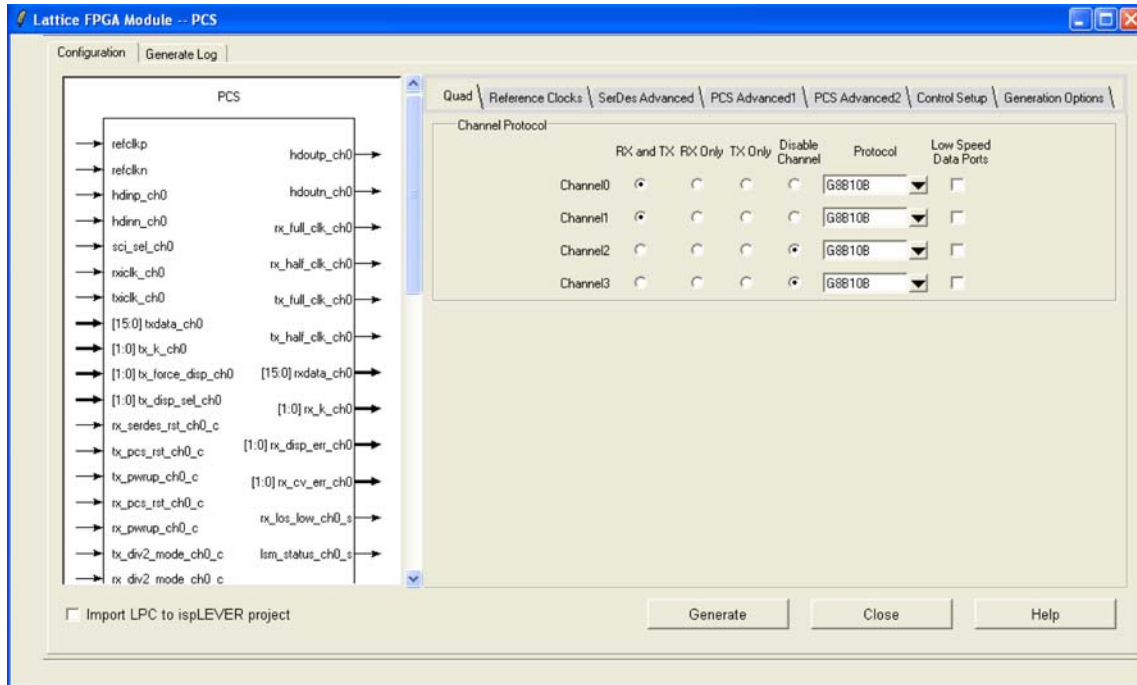
**Figure 8-19. IPexpress PCS Main Window**



### Quad Setup Tab

Figure 8-20 shows the Quad Setup Tab window when the file name is entered and the **Customize** button is checked in the main window. The first entry required in this window is to select a Protocol Mode for each channel. Each channel can be configured as 'RX and TX', 'RX Only', 'TX Only', 'Disabled' or 'Low Speed Data Port'.

**Figure 8-20. Configuration GUI - Quad Setup Tab**



**Table 8-11. SERDES\_PCS GUI Attributes - Quad Tab Setup**

GUI Text	Attribute Name	Range	Default Value
Channel Protocol	CHx_MODE	RX and TX, RX Only, TX Only	DISABLED
Disable Channel <sup>1</sup>	CHx_MODE	ENABLE, DISABLE	DISABLED
Protocol	CHx_PROTOCOL	GIGE, SGMII, XAUI, SRIO, PCIE, SDI, G8B10B, 10BSER, 8BSER, CPRI, OBSAI	G8B10B
Low Speed Data Port	CHx_LDR	RX and TX, RX Only, TX Only	DISABLED

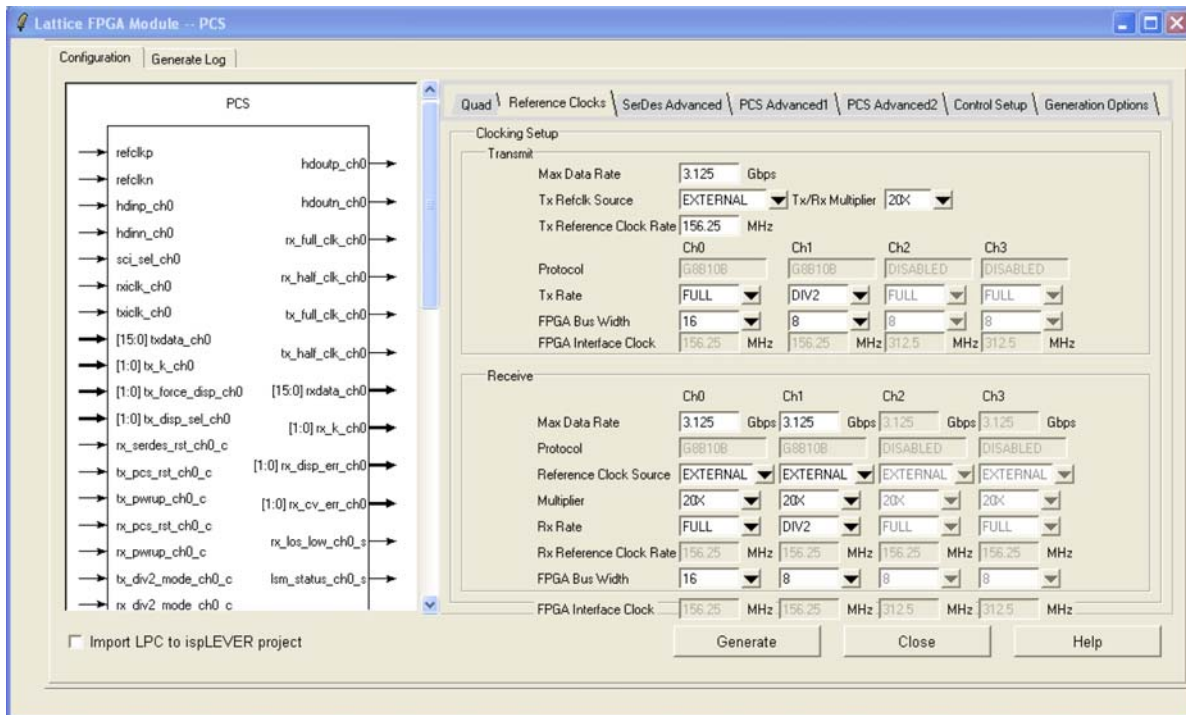
1. For the LatticeECP3-17EA device in the 328-ball csBGA package, only channels 0 and 3 are available.



### Reference Clock Setup Tab

In this tab, the attributes of the TX and RX reference clock sources are selected. Users can select either a EXTERNAL or INTERNAL reference clock. Further, there is a tool to provide the required clock rate and multiplier settings for a particular data rate. In addition, for a given data bus width the tool provides the required clock rate to interface the channel to the core.

**Figure 8-21. Configuration GUI - Reference Clocks Setup Tab**



**Table 8-12. SERDES\_PCS GUI Attributes - Reference Clocks Setup Tab**

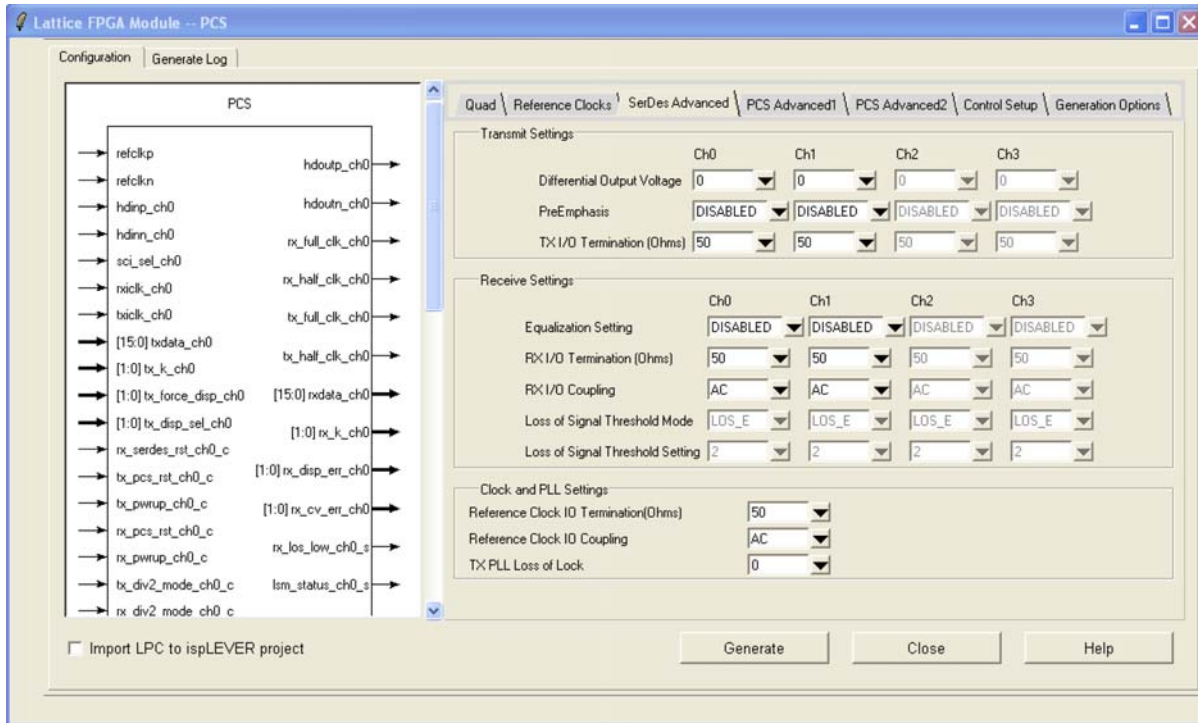
GUI Text	Attribute Name	Range	Default Value (GUI)	Default Value (Attribute)
<b>Transmit</b>				
Max. Data Rate <sup>1</sup>	N/A	0.23 to 3.2 Gbps	2.5 Gbps	N/A
TX Refclk Source	PLL_SRC	INTERNAL, EXTERNAL	INTERNAL	REFCLK_INT
TX/RX Multiplier	REFCK_MULT	8X, 10X, 16X, 20X, 25X	Protocol Dependent	
TX Reference clock Rate	#REFCLK_RATE <sup>2</sup>		Protocol Dependent	
Protocol	No user access. For information only.			
TX Rate	CHx_TX_DATA_RATE	FULL, DIV2, DIV11	FULL	FULL
FPGA Bus Width	CHs_TX_DATA_WIDTH	8, 10, 16, 20	Protocol Dependent	
FPGA Interface Clock	#CH0_TX_FICLK_RATE			
<b>Receive</b>				
Max. Data Rate <sup>1</sup>	N/A	0.23 to 3.2 Gbps	2.5 Gbps	N/A
Protocol	No user access. For information only.			
Refclk Source	CHx_CDR_SRC	INTERNAL, EXTERNAL	INTERNAL	REFCLK_INT
Multiplier	No user access. For information only			
RX Rate	CHx_RX_DATA_RATE	FULL, DIV2, DIV11	FULL	FULL
RX Reference Clock Rate	#CH0_RXREFCLK_RATE			
FPGA Bus Width	CHx_RX_DATA_WIDTH	8, 10, 16, 20	Protocol Dependent	
FPGA Interface Clock	#CH0_RX_FICLK_RATE			

1. Rate is not reflected in the autoconfig file. Instead, DATARATE RANGE is specified for a given data rate as: 150 Mbps ≤ LOWLOW ≤ 230 Mbps, 230 Mbps < LOW ≤ 450 Mbps, 450 Mbps < MEDLOW ≤ 0.9 Gbps, 0.9 Gbps < MED ≤ 1.8 Gbps, 1.8 Gbps < MEDHIGH ≤ 2.55 Gbps, 2.55 Gbps < HIGH ≤ 3.2Gbps.
2. Attributes preceded by '#' represent attributes that are for user information only. These attributes are also included in the auto-config file for reference.

**SERDES Advance Setup**

This tab is used to access the advanced attributes of the transmit and receive SERDES for all four channels. Transmit attributes such as PreEmphasis, Termination, Differential Output Voltage selection are selected. Receive attributes such as Equalization, Termination, I/O Coupling are selected. Attributes for Transmit SERDES Clock and PLL are also selected.

**Figure 8-22. Configuration GUI - SERDES Advanced Setup Tab**



**Table 8-13. SERDES\_PCS GUI Attributes – SERDES Advanced Setup Tab**

GUI Text	Attribute Name	Value	Default Value
Differential Output Voltage	CHx_TDRV <sup>8</sup>	-4 (640mV) <sup>5</sup> , -3 (780mV), -2 (870mV), -1 (920mV), 0 (1040mV:default), 1 (1130mV) <sup>6</sup> , 2 (1260mV) <sup>7</sup> , 3 (1350mV) <sup>7</sup> , 4 (1440mV) <sup>7</sup>	0
PreEmphasis	CHx_TX_PRE	Disabled, 0 (0%), 1 (5%), 2 (12%), 3 (18%), 4 (25%), 5 (33%), 6 (40%), 7 (48%)	DISABLED
TX I/O Termination (Ohms) <sup>3</sup>	CHx_RTERM_TX	50, 75, 5K	50
Equalization <sup>1</sup>	CHx_RX_EQ	Disabled, Mid_Low, Mid_Med, Mid_High, Long_Low, Long_Med, Long_High	DISABLED
RX I/O Termination (Ohms) <sup>3</sup>	CHx_RTERM_RX	50, 60, 75, High	50
RX I/O Coupling	CHx_RX_DCC	AC, DC	AC <sup>2</sup>
Loss of Signal Threshold	CHx_LOS_THRESHOLD_LO	2 (+15%),3 (+25%)	2 <sup>4</sup>
TX PLL Reference Clock I/O Termination (Ohms) <sup>3</sup>	PLL_TERM	50, 2K	50
TX PLL Reference Clock I/O Coupling	PLL_DCC	AC, DC	AC <sup>10</sup>
PLL Loss of Lock	PLL_LOL_SET	0: +/- 1350ppm x2 <sup>9</sup> 1: +/- 2400ppm x2 2: +/- 6800ppm 3: +/- 400ppm	0

1. Refer to Table 8-106 for details.
2. The typical capacitor value of the internal on-chip AC coupling is 5 pF.
3. Termination resistors and their usage:
  - RX I/O Termination:
    - 50: So far all of the protocols except SMTPE use a 50-Ohm termination resistor.
    - 60: Provided for flexibility.
    - 75: SMPTE uses a 75-Ohm termination resistor.
    - HIGH: Default value when Rx is not used.
  - TX I/O Termination:
    - 50: So far all of the protocols except SMTPE use a 50-Ohm termination resistor.
    - 75: SMPTE uses a 75-Ohm termination resistor.
    - 5K: Such as PCI Express electric idle and PCI Express RX detection. User does not set this termination value for RX detection. Refer to the PCI Express Receiver Detection section.
  - TX PLL Termination:
    - 50: If there is no 50-Ohm termination resistor on the PCB.
    - 2K: If there is a 50-Ohm termination resistor on the PCB.
4. PCS configuration, GUI supports only the value 2 for all protocols except PCI Express. For PCI Express, both values 2 and 3 are supported.
5. TDRV\_AMP\_BOOST(CH\_13[3]) is set to 1 to achieve this amplitude.
6. This setting is the PCI Express default setting. It is recommended to use this default setting with the PCI Express protocol. The IPexpress GUI TDRV drop-down window is grayed out for this reason. Other settings can still be used by editing the attribute CHn\_TDRV in the auto-config file(.txt file).
7. VCCOB must be 1.5V for these settings.
8. The values are typical numbers. There is about +/-20% of margin for the whole frequency range. Refer to Table 8-105, CHn\_TDRV row for details.
9. 'x2' indicates successful double count of ppm in the internal LOL counter.
10. AC-coupling is recommended for most applications. DC-coupling should be used only in conjunction with an external AC-coupling capacitor.

### Assignment of PCS Location

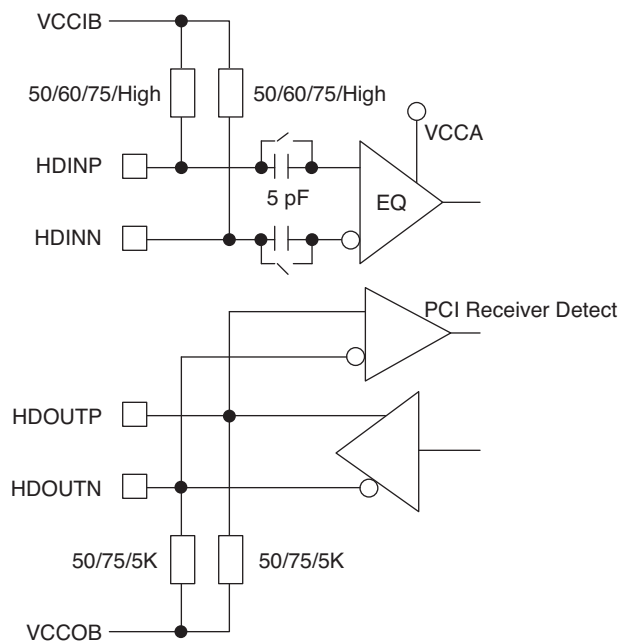
Users can specify their desired location of the PCS quad by writing in the constraint file (.lpf). The preference "locate" is used. An example of the syntax is shown below.

```
LOCATE COMP "pcs_inst_name" SITE "PCSB" ;
```

Quad Name	Site Name
Quad A	PCSA
Quad B	PCSB
Quad C	PCSC
Quad D	PCSD

High-speed I/O termination topology is shown in Figure 8-23.

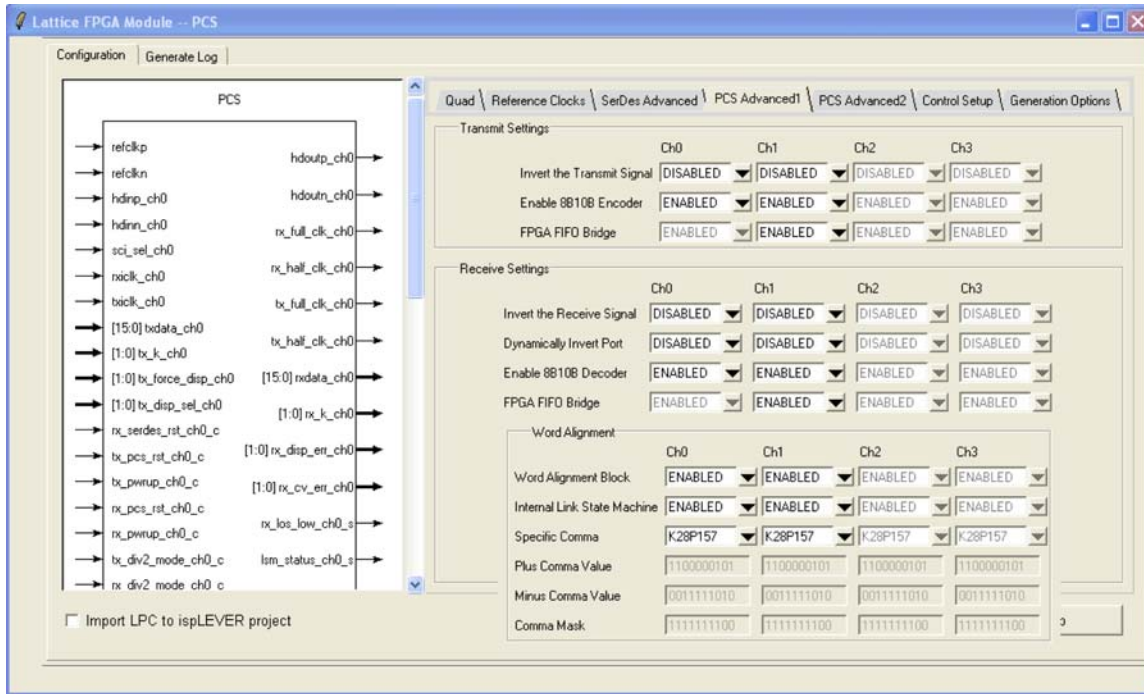
**Figure 8-23. High-Speed I/O Terminations**



**PCS Advanced1 Setup**

This tab is used to access the advanced attributes of the transmit and receive PCS for all four channels. The polarity and operating mode (e.g. 8b10b) of each individual TX and RX channel can be individually selected. In addition, word alignment values such as comma values, comma mask and comma align can be selected.

**Figure 8-24. Configuration GUI - PCS Advanced1 Setup Tab**



**Table 8-14. SERDES/PCS GUI – PCS Advanced1 Setup Tab**

	GUI Text	Attribute Name	Default
Transmitter	Invert the Transmit Signal	CHx_TX_SB	DISABLED
	Enable 8b10b Encoder	CHx_TX_8B10B	Protocol Dependent
	FPGA FIFO Bridge	CHx_TX_FIFO	Protocol Dependent
Receiver	Invert the Receive Signal	CHx_RX_SB	DISABLED
	Dynamically Invert Port	N/A	DISABLED
	Enable 8b10b Decoder	CHx_RX_8B10B	Protocol Dependent
	FPGA FIFO Bridge	CHx_RX_FIFO	Protocol Dependent
Word Alignment	Word Alignment Block	CHx_RXWA	Protocol Dependent
	Internal Link	CHx_ILSM	Protocol Dependent
	Specific Comma	#CHx_SCOMMA	Protocol Dependent
	Plus Comma Value	CHx_COMMA_A <sup>1</sup>	1100000101
	Minus Comma Value	CHx_COMMA_B	0011111010
	Comma Mask	CHx_COMMA_M	Protocol Dependent <sup>2</sup>

1. By definition, COMMA\_A and COMMA\_B are one set of 8b10b-encoded control characters with positive and negative running disparities. Users must provide the proper IDLE sequence per the protocol in order to get the link state machine synchronization. For example, 1 GbE protocol needs K28.5+D5.6 or D16.2 as IDLE (word alignment and sync state machine). The default values are in little endian format.
2. In most applications, K28.5 is used as the comma character. The default value of the mask is 111111111. In G8B10B mode, any comma can be used so the mask will be 1111111100 to detect any of the three comma characters, K28.1, 28.5, 28.7.

### PCS Advanced2 Setup

This tab is used for setting values for the Clock Tolerance Compensation block.

Figure 8-25. Configuration GUI – PCS Advanced2 Setup Tab

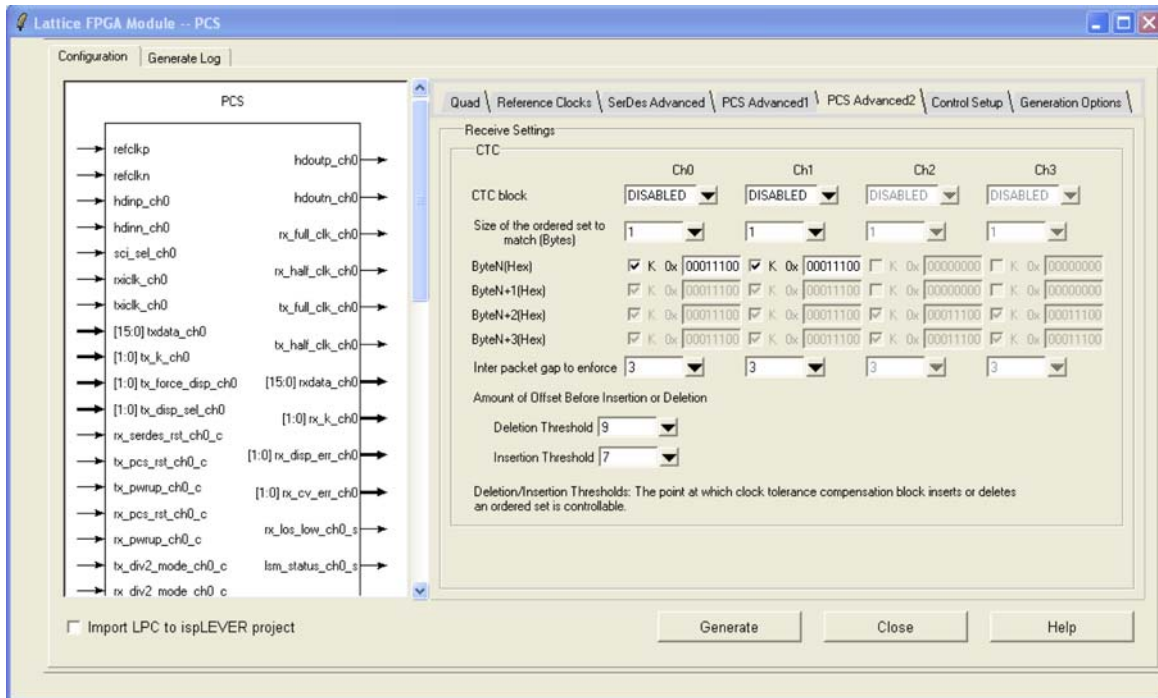


Table 8-15. SERDES/PCS GUI – PCS Advanced2 Setup Tab

GUI Text	Attribute Name	Default
CTC block	CHx_CTC	Protocol Dependent <sup>1</sup>
Size of ordered set	CHx_CC_MATCH_MODE	Protocol Dependent
Byte N	CHx_CC_MATCH1	Protocol Dependent
Byte N+1	CHx_CC_MATCH2	Protocol Dependent
Byte N+2	CHx_CC_MATCH3	Protocol Dependent
Byte N+3	CHx_CC_MATCH4	Protocol Dependent
Interpacket gap	CHx_CC_MIN_IPG	Protocol Dependent
Deletion threshold	CCHMARK	9
Insertion threshold	CCLMARK	7

1. Always disabled: XAUI, SDI, CPRI, OBSAI, Serial Rapid IO, 10-bit SERDES, 8-bit SERDES  
All other modes: Disabled by default. Most of the CTC features are provided in IP.



### Control Setup

This tab is used to select the SCI interface and other debug and control options. In addition, users can enable the SCI, error reporting, PLL quarter clock, and loopback capability.

Figure 8-26. Configuration GUI – Control Setup Tab

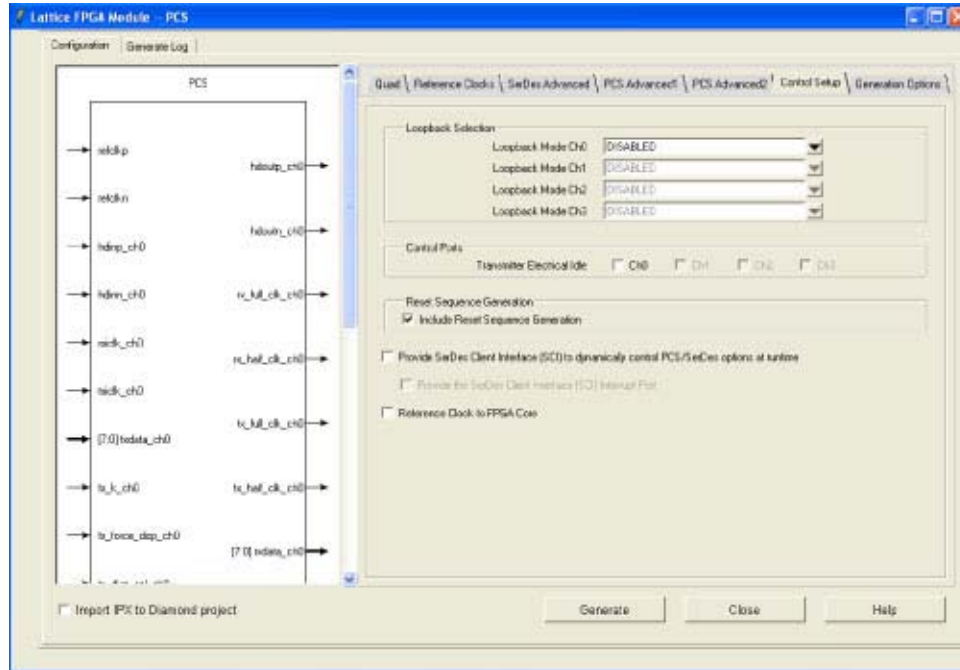


Table 8-16. Tab 5, SERDES\_PCS GUI Attributes – Control Setup Tab

GUI Text	Attribute Name	Default
Loopback Mode (Ch0, Ch1, Ch2, Ch3)	DISABLED Loopback serial data after equalizer Loopback serial data after transmit driver Loopback parallel data after de_serializer	DISABLED <sup>1</sup>
Transmitter Electrical Idle	signal tx_idle_ch0_c is provided	DISABLED
Include Reset Sequence Generation <sup>2</sup>	Include the TX and RX Reset Sequence	ENABLED
Provide SERDES Client Interface	N/A	
Provide the SERDES Client Interface Interrupt Port	INT_ALL	DISABLED
Reference Clock to FPGA core	QD_REFCK2CORE	DISABLED

1. When loopback mode is in default mode (DISABLED), the two SERDES Bridge Parallel loopback control signals (sb\_felb\_ch[3:0]\_c and sb\_felb\_rst\_ch[3:0]\_c) are available in the HDL module for users to turn on and off the loopback mode dynamically. These signals should be tied to ground if loopback mode is not used.
2. Reset Sequence Generation is described in the SERDES/PCS RESET section of this document.

### Reference Clock to FPGA Core and Reset Sequence

The reset sequence uses a reference clock in the Reset State Machine.

If “Internal” is selected as the Tx Refclk source, the Reset State Machine uses the internal reference clock.

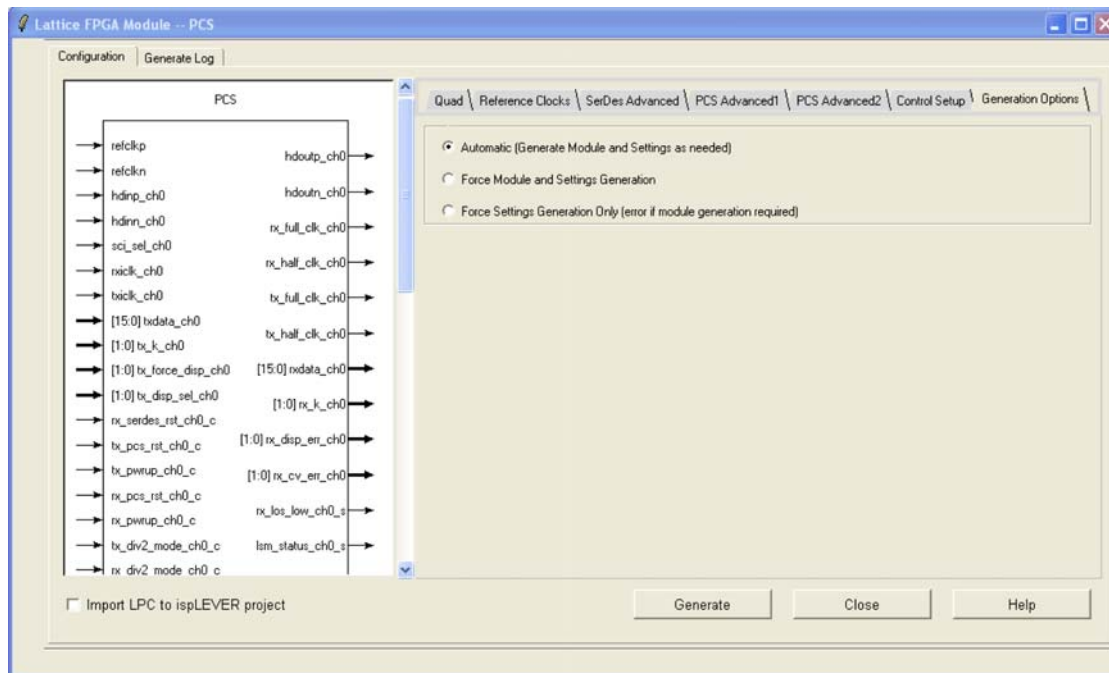
If “External” is selected as the Tx Refclk source, the Reset State Machine uses the Reference Clock to FPGA Core signal transparent to the user (Control Register bit, QD\_0A[1] is set). The REFCLK2FPGA signal will be included in the wrapper module only when “Reference Clock to FPGA Core” is selected.



## Generation Options

This tab provides options for users to select the PCS module generation output files of their choice.

**Figure 8-27. Configuration GUI – Generation Options Tab**



When migrating an older project created in a previous version of Diamond to the latest version for the first time, it is highly recommended to regenerate the PCS module in the latest version even if no configuration changes are required. This will ensure the most current set of primitive library elements is used. If this is not done, it is possible that the design may either fail in the flow or go through the flow but show unexpected behavior on the board.

When regenerating the PCS module in an existing project, often the HDL file remains the same and only the configuration file must be regenerated. In this case, users can run the “Generate Bitstream Data” and “Force One Level” option to save compile time.

- **Automatic** – When selected, IPexpress will generate only the necessary files. This can include both the HDL and TXT files or just the TXT file. This is the default setting.
- **Force Module and Settings Generation** – When selected, both the HDL and TXT files will be generated. This forces the Project Navigator processes to be reset back to synthesis.
- **Force Settings Generation Only** – When selected, only the TXT file will be generated. If HDL generation is necessary, an error message will be provided.
- **Flow Definitions** – The generation option works differently between the two module flows.
  - **HDL Source Flow: HDL File in Project Navigator**  
The existing LPC file can be opened from IPexpress for regeneration. In this case, the reset point set by the GUI will be the new starting point. So, when the user double-clicks a process or runs the “Force One Level” option, it will start at that reset point.
  - **LPC Source Flow: LPC File in Project Navigator**  
Opening the LPC file and regenerating the PCS module will reset the whole process whether or not the HDL module is regenerated.

In either case, the checkmarks in the Processes window will remain unchanged but will be updated as soon as the user starts the process.

### Configuration File Description

IPexpress generates this file which contains attribute-level map information. This file is used by the simulation model as well as the bitstream generation process to automatically initialize the PCS quad to the mode selected in IPexpress.

The configuration file uses ".txt" as the file type extension.

Below is an example of the configuration file.

```
# This file is used by the simulation model as well as the bitstream
# generation process to automatically initialize the PCS quad to the mode
# selected in the IPexpress. This file is expected to be modified by the
# end user to adjust the PCS quad to the final design requirements.
```

```
DEVICE_NAME "LFE3-95E"
CH0_MODE "RXTX"
CH1_MODE "DISABLED"
CH2_MODE "DISABLED"
CH3_MODE "DISABLED"
TX_DATARATE_RANGE "HIGH"
PLL_SRC "REFCLK_EXT"
REFCK_MULT "10X"
#REFCLK_RATE 250.0

CH0_PROTOCOL "G8B10B"
CH0_LDR "RXTX"
CH0_RX_DATARATE_RANGE "HIGH"
CH0_TX_DATA_RATE "FULL"
CH0_TX_DATA_WIDTH "8"
CH0_TX_FIFO "DISABLED"
CH0_CDR_SRC "REFCLK_EXT"
#CH0_TX_FICLK_RATE 250.0
CH0_RX_DATA_RATE "FULL"
CH0_RX_DATA_WIDTH "8"
CH0_RX_FIFO "DISABLED"
#CH0_RX_FICLK_RATE 250.0
CH0_TDRV "0"
CH0_TX_PRE "DISABLED"
CH0_RTERM_TX "50"
CH0_RX_EQ "DISABLED"
CH0_RTERM_RX "50"
CH0_RX_DCC "AC"
CH0_LOS_THRESHOLD_LO "2"
CH0_TX_SB "DISABLED"
CH0_TX_8B10B "ENABLED"
CH0_RX_SB "DISABLED"
CH0_RX_8B10B "ENABLED"
CH0_RXWA "ENABLED"
CH0_ILSM "ENABLED"
#CH0_SCOMMA "1111111111"
CH0_COMMA_A "1100000101"
CH0_COMMA_B "0011111010"
CH0_COMMA_M "1111111100"
CH0_CTC "ENABLED"
```

CH0_CC_MATCH_MODE	"2"
CH0_CC_MATCH1	"0000000000"
CH0_CC_MATCH2	"0000000000"
CH0_CC_MIN_IPG	"3"
CH0_SSLB	"DISABLED"
CH0_SPLBPORTS	"DISABLED"
CH0_PCSLBPORTS	"DISABLED"
PLL_TERM	"50"
PLL_DCC	"AC"
PLL_LOL_SET	"0"
CCHMARK	"9"
CCLMARK	"7"
INT_ALL	"DISABLED"
QD_REFCK2CORE	"ENABLED"

### 8-Bit and 10-Bit SERDES-Only Modes

This section describes the operation of the two modes of the SERDES/PCS block, 8-bit SERDES-Only and 10-bit SERDES-Only. These modes are intended for applications requiring access to a high-speed I/O interface without the protocol-based manipulation provided in the LatticeECP3 PCS logic.

#### Transmit Path

- Serializer: Conversion of 8-bit or 10-bit parallel data to serial data.

#### Receive Path

- Deserializer: Conversion of serial data to 8-bit or 10-bit parallel data.
- Optional word alignment to user-defined alignment pattern.

### Generic 8b10b Mode

The Generic 8b10b mode of the SERDES/PCS block is intended for applications requiring 8b10b encoding/decoding without the need for additional protocol-specific data manipulation. The LatticeECP3 SERDES/PCS block can support Generic 8b10b applications up to 3.2 Gbps per channel. In Generic 8b10b mode, the word aligner can be controlled from the embedded PCS Link State Machine (LSM).

When the embedded Link State Machine is selected and enabled, the `lsm_status_ch[3:0]_s` status signal will go high upon successful link synchronization.

To achieve link synchronization in this mode, the following conditions need to be satisfied on the 8b10b patterns received at the SERDES channel's receiver input (`hdinp_ch[0-3]/hdinn_ch[0-3]`):

- Periodic 8b10b encoded comma characters need to be present in the serial data. Periodicity is required to allow the LSM to re-synchronize to commas upon loss of sync. The comma characters should correspond to the "Specific Comma" value shown in Figure 8-24. For example, when the Specific Comma is set to K28P157, the comma value on the serial link can be the 8b10b encoded version of any of K28.1 (k=1, Data=0x3C), K28.5 (k=1, Data=0xBC), or K28.1 (k=1, Data=0xFC). Note though that K28.5 is most commonly used.
- A comma character has to be followed by a data character
- Two comma characters have to be an even number of word clock cycles apart

Additional information:

- It takes roughly four good comma/data pairs for the LSM to reach link synchronization
- It takes four consecutive errors (illegal 8b10b encoded characters, code violations, disparity errors, uneven number of clock cycles between commas) to cause the LSM to unlock
- A CDR loss of lock condition will cause the LSM to unlock by virtue of the many code violation and disparity errors that will result
- When the internal reset sequence state machine is used, a CDR loss of lock (rx\_cdr\_lof\_ch[3:0]\_s) or a loss of signal (rx\_los\_low\_ch[3:0]\_s) condition will cause the RX reset sequence state machine to reset the SERDES and cause the LSM to unlock

The two examples below illustrate the difference between a valid and an invalid even clock cycle boundary between COMMA occurrences (*Note: C = comma, D = data*).

Valid (even) comma boundary:

Word Clock Cycle	0	1	2	3	4	5	6	7	8	9	10	11
CHARACTER	C	D	C	D	C	D	D	D	D	D	C	D

Invalid (odd) comma boundary:

Word Clock Cycle	0	1	2	3	4	5	6	7	8	9	10	11	12
CHARACTER	C	D	C	D	C	D	D	D	D	C	D	C	D

In the invalid (odd) comma boundary case above, the comma occurrences on cycles 9 and 11 are invalid since they do not fall on an even boundary apart from the previous comma.

Alternatively, the LSM can be disabled, and the word aligner is controlled from the fabric word\_align\_en\_ch[3:0]\_c input pin. See [“External Link State Machine Option” on page 21](#) and [“PCS Advanced1 Setup” on page 35](#) for more information.

#### Transmit Path

- Serializer
- 8b10b encoder

#### Receive Path

- Deserializer
- Word alignment to a user-defined word alignment character or characters from embedded GbE Link State Machine
- 8b10b decoding
- Clock Tolerance Compensation (optional)

## LatticeECP3 PCS in Gigabit Ethernet and SGMII Modes

The Gigabit Ethernet mode of the LatticeECP3 SERDES/PCS block supports full compatibility, from the Serial I/O to the GMII/SGMII interface of the IEEE 802.3-2002 1000 BASE-X Gigabit Ethernet standard.

### Transmit Path

- Serializer
- 8b10b encoding

### Receive Path

- Deserializer
- Word alignment based on IEEE 802.3-2002 1000 BASE-X defined alignment characters.
- 8b10b decoding
- The Gigabit Ethernet Link State Machine is compliant to Figure 36-9 (Synchronization State Machine, 1000BASE-X) of IEEE 802.3-2002 with one exception. Figure 36-9 requires that four consecutive good code groups are received in order for the LSM to transition from one SYNC\_ACQUIRED\_{N} (N=2,3,4) to SYNC\_ACQUIRED\_{N-1}. Instead, the actual LSM implementation requires five consecutive good code groups to make the transition.
- Gigabit Ethernet Carrier Detection: Section 36.2.5.1.4 of IEEE 802.3-2002 (1000BASE-X) defines the carrier\_detect function. In Gigabit Ethernet mode, this feature is not included in the PCS and a carrier\_detect signal is not provided to the FPGA fabric.
- Clock Tolerance Compensation logic capable of accommodating clock domain differences.

### Gigabit Ethernet (1000BASE-X) Idle Insert

This is required for Clock Compensation and Auto-negotiation. Auto-negotiation is done in FPGA logic. The Lattice Gigabit Ethernet PCS IP core provides the auto-negotiation discussed below.

Idle pattern insertion is required for clock compensation and auto-negotiation. Auto-negotiation is done in FPGA logic. This module automatically inserts /I2/ symbols into the receive data stream during auto-negotiation. While auto-negotiating, the link partner will continuously transmit /C1/ and /C2/ ordered sets. The clock-compensator will not delete these ordered sets as it is configured to only insert/delete /I2/ ordered sets. In order to prevent overruns and underruns in the clock-compensator, /I2/ ordered sets must be periodically inserted to provide insertion/deletion opportunities for the clock compensator.

While performing auto-negotiation, this module will insert a sequence of eight /I2/ ordered sets (two bytes each) every 2048 clock cycles. As this module is after the 8b10b decoder, this operation will not introduce any running disparity errors. These /I2/ ordered sets will not be passed on to the FPGA receive interface as the GMII interface is driven to IDLE by the RX state machine during auto-negotiation. Once auto-negotiation is complete, /I2/ insertion is disabled to prevent any corruption of the received data.

Note that this state machine is active only during auto-negotiation. The auto-negotiation state machine and the GbE receive state machines are implemented in the soft logic. This state machine depends on the signal xmit\_ch[3:0] from the auto-negotiation state machine. This signal is provided on the TX data bus. Though this signal is relatively static (especially after auto-negotiation) it is included in the TX data bus.

**Table 8-17. GbE IDLE State Machine Control and Status Signals**

Module Signal	Direction	Description
xmit_ch[3:0]	In	From FPGA logic Auto-Negotiation State Machine

### Gigabit Ethernet Idle Insert and correct\_disp\_ch[3:0] Signals

The `correct_disp_ch[3:0]` signal is used on the transmit side of the PCS to ensure that an interpacket gap begins in the negative disparity state. Note that at the end of an Ethernet frame, the current disparity state of the transmitter can be either positive or negative, depending on the size and data content of the Ethernet frame.

However, from the FPGA soft-logic side of the PCS, the current disparity state of the PCS transmitter is unknown. This is where the `correct_disp_ch[3:0]` signal comes into play. If the `correct_disp_ch[3:0]` signal is asserted for one clock cycle upon entering an interpacket gap, it will force the PCS transmitter to insert an IDLE1 ordered-set into the transmit data stream if the current disparity is positive. However, if the current disparity is negative, then no change is made to the transmit data stream.

From the FPGA soft-logic side of the PCS, the interpacket gap is typically characterized by the continuous transmission of the IDLE2 ordered set which is as follows: `tx_k_ch=1, txdata= 0xBC tx_k_ch=0, txdata=0x50`.

Note that in the PCS channel, IDLE2s mean that current disparity is to be preserved. IDLE1s mean that the current disparity state should be flipped. Therefore, it is possible to ensure that the interpacket gap begins in a negative disparity state. If the disparity state before the interpacket gap is negative, then a continuous stream of IDLE2s are transmitted during the interpacket gap. If the disparity state before the interpacket gap is positive, then a single IDLE1 is transmitted followed by a continuous stream of IDLE2s.

In the FPGA soft-logic side of the PCS, the interpacket gap is always driven with IDLE2s into the PCS. The `correct_disp_ch[3:0]` signal is asserted for one clock cycle, `k_cntrl=0, data=0x50` when the interpacket gap first begins. If necessary, the PCS will convert this IDLE2 into an IDLE1. For the remainder of the interpacket gap, IDLE2s should be driven into the PCS and the `correct_disparity_chx` signal should remain deasserted.

For example, if a continuous stream of 512 bytes of Ethernet frames and 512 bytes of // are sent, it can be observed that:

- During the first interpacket gap, all negative disparity //2/s are seen (K28.5(-) D16.2(+))
- During the next interpacket gap, the period begins with positive disparity //1/ (K28.5 (+), D5.6 (+/- are the same)), then all remaining ordered sets are negative disparity //2/s
- During the next interpacket gap, all negative disparity //2/s are seen
- During the next interpacket gap, the period begins with positive disparity //1/ (K28.5 (+), D5.6 (+/- are the same)), then all remaining ordered sets are negative disparity //2/s

A number of programmable options are supported within the encoder module. These are:

- Ability to force negative or positive disparity on a per-word basis
- Ability to input data directly from the FIFO Bridge - external multiplexer
- Ability to replaced code words dependant upon running disparity (100BASE-X and FC)
- Software register controlled bypass mode

## **XAUI Mode**

With the Lattice XAUI IP Core, the XAUI mode of the SERDES/PCS block supports full compatibility from Serial I/O to the XGMII interface of the IEEE 802.3-2002 XAUI standard. XAUI Mode supports 10 Gigabit Ethernet.

### **Transmit Path**

- Serializer
- Transmit State Machine which performs translation of XGMII idles to proper ||A||, ||K||, ||R|| characters according to the IEEE 802.3ae-2002 specification
- 8b10b Encoding

---

**Receive Path**

- Deserializer
- Word alignment based on IEEE 802.3-2002 defined alignment characters.
- 8b10b Decoding
- The XAUI Link State Machine is compliant to Figure 48-7 - PCS synchronization state diagram of IEEE 802.3ae-2002 with one exception. Figure 48-7 requires that four consecutive good code groups be received in order for the LSM to transition from one SYNC\_ACQUIRED\_{N} (N=2,3,4) to SYNC\_ACQUIRED\_{N-1}. Instead, the actual LSM implementation requires five consecutive good code groups to make the transition.
- Clock Tolerance Compensation logic in PCS is disabled in XAUI mode. MCA (Multi-Channel Alignment) and CTC are done in the XAUI IP core.
- x4 multi-channel alignment should be done in FPGA core logic.

**LatticeECP3 PCS in PCI Express Revision 1.1 (2.5Gpbs) Mode**

The PCI Express mode of the SERDES/PCS block supports x1, x2, and x4 PCI Express applications.

**Transmit Path**

- Serializer
- 8b10b Encoding
- Receiver Detection
- Electrical Idle

**Receive Path**

- Deserializer
- Word alignment based on the Sync Code
- 8b10b Decoding
- Link Synchronization State Machine functions incorporating operations defined in the PCS Synchronization State Machine (Figure 48-7) of the IEEE 802.3ae-2002 10GBASE-X Specification.
- x2 or x4 PCI Express operation with one PCS quad set to PCI Express mode.
- Clock Tolerance Compensation logic capable of accommodating clock domain differences.
- x2 or x4 multi-channel alignment should be done in FPGA core logic.

Table 8-18 describes the PCI Express mode specific ports.

**Table 8-18. PCI Express Mode Specific Ports**

Signal	Direction	Class	Description
pcie_done_ch[3:0]_s	Out	Channel	1 = Far-end receiver detection complete 0 = Far-end receiver detection incomplete
pcie_con_ch[3:0]_s	Out	Channel	Result of far-end receiver detection 1 = Far-end receiver detected 0 = Far-end receiver not detected
pcie_det_en_ch[3:0]_c	In	Channel	FPGA logic (user logic) informs the SERDES block that it will request a PCI Express Receiver Detection operation. 1 = Enable PCI Express Receiver Detect 0 = Normal Operation
pcie_ct_ch[3:0]_c	In	Channel	1 = Request transmitter to do far-end receiver detection 0 = Normal data operation
rxstatus[2:0]	Out	Channel	Per-channel PCI Express receive status port. RxStatus# is an encoded status of the receive data path. 2 bits wide if in 16-bit data bus mode.

The status signal, rxstatus, is an encoded status of the receive data path. The encoding is as follows.

**Table 8-19. rxstatus Encoding**

rxstatus[2:0]			Description	Priority
0	0	0	Received data OK	8
0	0	1	1 byte inserted by CTC	7
0	1	0	1 byte deleted by CTC	6
0	1	1	Receiver detected (pcie_done, pcie_con)	1
1	0	0	8b10b decode error (code violation - rx_cv_err)	2
1	0	1	CTC FIFO overflow (ctc_orun)	3
1	1	0	CTC FIFO underflow (ctc_urun)	4
1	1	1	Receive disparity error (rx_disp_err)	5

### PCI Express Termination

At the electrical level, PCI Express utilizes two uni-directional low voltage differential signaling pairs at 2.5Gbps for each lane. Transmit and receive are separate differential pairs, for a total of four data wires per lane. An input receiver with programmable equalization and output transmitters with programmable pre-emphasis permits optimization of the link. The PCI Express specification requires that the differential line must be common mode terminated at the receiving end. Each link requires a termination resistor at the far (receiver) end. The nominal resistor values used are 100 ohms. This is accomplished by using the embedded termination features of the CML inputs as shown in Figure 8-28. The specification requires AC coupling capacitors (CTX) on the transmit side of the link. This eliminates potential common-mode bias mismatches between transmit and receive devices. The capacitors must be added external to the Lattice CML outputs.

### PCI Express L2 State

For the PCI Express L2 state, the rx\_pwrup\_c signal should not be de-asserted to power-down the rx channel. This will force the RX termination to high impedance and will not allow the far-end to detect the receiver. Rather, the rx\_pcs\_rst\_c signal should be used to hold the channel in reset to save power.



Figure 8-28. PCI Express Interface Diagram

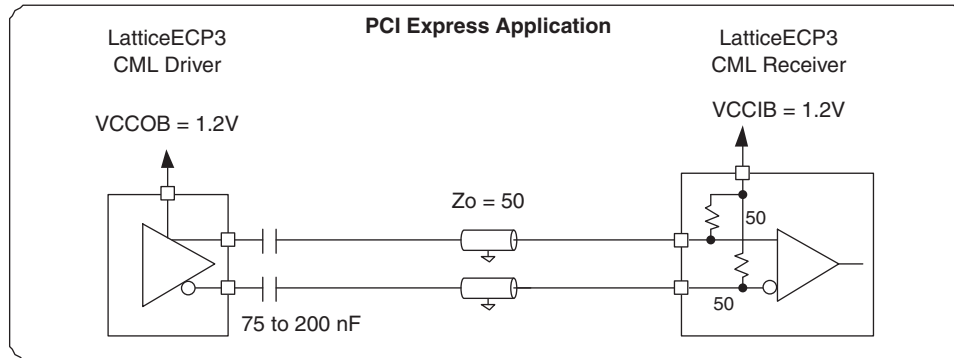


Table 8-20. Differential PCI Express Specifications

Symbol	Parameter	Min.	Nom.	Max.	Units	Comments	Location
ZTX-DIFF-DC	DC Differential TX Impedance	80	100	120	Ohm	TX DC Differential mode low impedance. ZTX-DIFF-DC is the small signal resistance of the transmitter measured at a DC operating point that is equivalent to that established by connecting a 100 Ohm resistor from D+ and D- while the TX is driving a static logic one or logic zero.	Internal
ZRX-DIFF-DC	DC Differential Input Impedance	80	100	120	Ohm	RX DC Differential mode impedance during all LTSSM states. When transmitting from a Fundamental Reset to Detect, (the initial state of the LTSSM), there is a 5 ms transition time before receiver termination values must be met on all un-configured lanes of a port.	Internal
CTX	AC Coupling Capacitor	75		200	nF	All transmitters shall be AC coupled. The AC coupling is required either within the media or within the transmitting component itself.	External

### PCI Express Electrical Idle Transmission

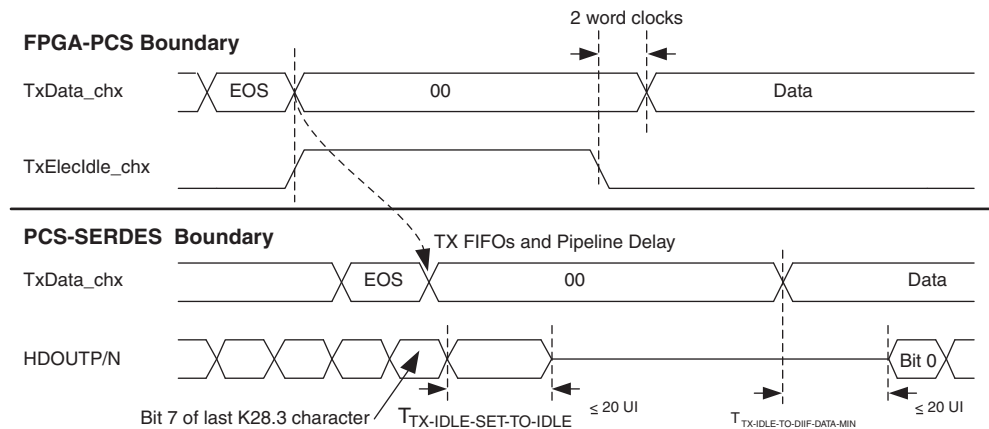
Electrical Idle is a steady state condition where the transmitter P and N voltages are held constant at the same value (i.e., the Electrical Idle Differential Peak Output Voltage, VTX-IDLE-DIFFp, is between 0 and 20mV). Electrical Idle is primarily used in power saving and inactive states.

Per the PCI Express base specification, before a transmitter enters Electrical Idle, it must always send the Electrical Idle Ordered Set (EIOS), a K28.5 (COM) followed by three K28.5 (IDL). After sending the last symbol of the Electrical Idle Ordered Set, the transmitter must be in a valid Electrical Idle state as specified by TTX-IDLE-SET-TO-IDLE is less than 20UI.

To achieve this, the Electrical Idle Enable (tx\_idle\_chx\_c) from the FPGA core logic to the PCS is sent in along with each transmit data. This signal is pipelined similarly all the way to the PCS-SERDES boundary. For all valid data this signal is LOW. To initiate Electrical Idle, the FPGA logic pulls this signal HIGH on the clock after it transmits the last K28.5 (IDL) symbol. As the signal is pipelined to the PCS-SERDES boundary, the relationship between the transmit data and this signal is exactly the same as on the FPGA-PCS boundary.

14UI after the rising edge of the Electrical Idle enable signal at the PCS-SERDES boundary the last bit (bit7) of the last K28.3 (IDL) symbol is transmitted. 16UI (<20UI) later the transmit differential buffer achieves Electrical Idle state.

Figure 8-29. Transmit Electrical Idle



As long as the FPGA core logic deems that the transmitter needs to stay in Electrical Idle state it needs to clock in data (preferably all zeros) along with the Electrical Idle Enable (`tx_idle_chx_c`) signal active (HIGH). The transmitter is required to stay in the Electrical Idle state for a minimum of 50UI (20ns) (`TTX-IDLE-MIN`).

### PCI Express Electrical Idle Detection

Each channel in the quad has a loss-of-signal detector. The Electrical Idle is detected once two out of the three K28.3 (IDL) symbols in the Electrical Idle Ordered Set (EOS) have been received. After the Electrical Idle Ordered Set is received, the receiver should wait for a minimum of 50ns (`TTX-IDLE-MIN`) before enabling its Electrical Idle Exit detector.

These signals (one per channel, four per quad) should be routed through the PCS, and should be made available to the FPGA core. The required state machine(s) to support electrical idle can then be constructed in the FPGA core.

### PCI Express Receiver Detection

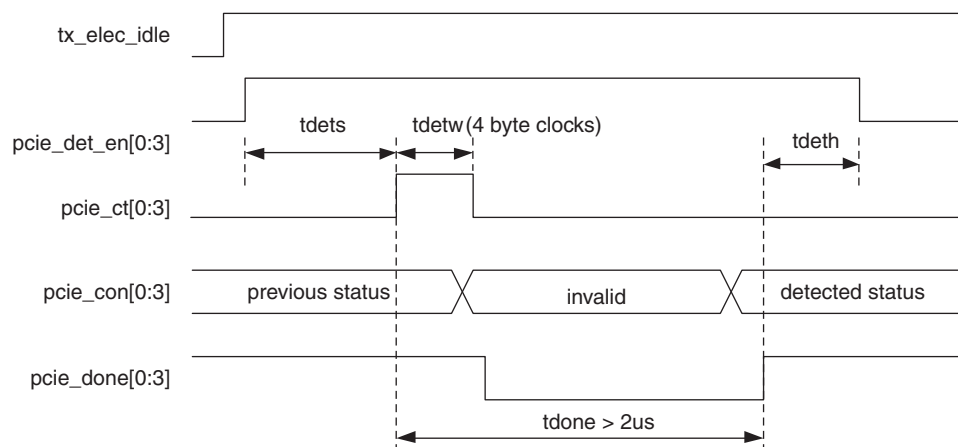
Figure 8-30 shows a Receiver Detection sequence. A Receiver Detection test can be performed on each channel of a quad independently. Before starting a Receiver Detection test, the transmitter must be put into electrical idle by setting the `tx_idle_ch#_c` input high. The Receiver Detection test can begin 120 ns after `tx_elec_idle` is set high by driving the appropriate `pci_det_en_ch#_c` high. This puts the corresponding SERDES Transmit buffer into receiver detect mode by setting the driver termination to high impedance and pulling both differential outputs to VCCOB through the high impedance driver termination.

Setting the SERDES Transmit buffer into receiver detect state takes up to 120 ns. After 120 ns, the receiver detect test can be initiated by driving the channel's `pcie_ct_ch#_c` input high for four byte (word) clock cycles. The corresponding channel's `pcie_done_ch#_s` is then cleared asynchronously. After enough time for the receiver detect test to finish has elapsed (determined by the time constant on the transmit side), the `pcie_done_ch#_s` receiver detect status port will go high and the Receiver Detect status can be monitored at the `pcie_con_ch#_s` port. If at that time the `pcie_con_ch#_s` port is high, then a receiver has been detected on that channel. If, however, the `pcie_con_ch#_s` port is low, then no receiver has been detected for that channel. Once the Receiver Detect test is complete, `tx_idle_ch#_c` can be deasserted.

Receiver detection proceeds as follows:

1. The user drives `pcie_det_en` high, putting the corresponding TX driver into receiver detect mode. This sets the driver termination to high-impedance (5K ohm) and pulls both outputs of the differential driver toward common mode through the high-impedance driver termination. The TX driver takes some time to enter this state so the `pcie_det_en` must be driven high for at least 120ns before `pcie_ct` is asserted.
2. The user drives `pcie_ct` high for four byte clocks.
3. SERDES drives the corresponding `pcie_done` low.
4. SERDES drives the internal signal (corresponding to `pcie_ct`) for as long as it is required (based on the time constant) to detect the receiver.
5. SERDES drives the corresponding `pcie_con` connection status.
6. SERDES drives the corresponding `pcie_done` high.
7. The user can use this asserted state of `pcie_done` to sample the `pcie_con` status to determine if the receiver detection was successful.

**Figure 8-30. PCI Express Mode Receiver Detection Sequence**



### PCI Express Power Down Mode

`rx_serdes_rst_ch[3:0]` reset signal should be used instead of `rx_pwrup_ch[3:0]` signal. This allows the RX termination to remain enabled at 50 Ohms so that the far-end transmitter can detect that a receiver is connected.

### PCI Express Beacon Support

This section highlights how the LatticeECP3 PCS can support Beacon detection and transmission. The PCI Express requirements for Beacon detection are presented with the PCS support for Beacon transmission and detection.

#### Beacon Detection Requirements

- Beacon is required for exit from L2 (P2) state.
- Beacon is a DC-balanced signal of periodic arbitrary data, which is required to contain some pulse widths  $\geq 2$ ns (500 MHz) and  $< 16$ us (30 KHz).
- Maximum time between pulses should be  $< 16$  us.
- DC balance must be restored within  $< 32$  us.
- For pulse widths  $> 500$  ns, the output beacon voltage level must be 6 db down from VTX-DIFFp-p (800 mV to 1200 mV).
- For pulse widths  $< 500$  ns, the output beacon voltage level must be  $\leq$  VTX-DIFFp-p and  $\geq 3.5$  db down from VTX-DIFFp-p.

---

### PCS Beacon Detection Support

- The signal loss threshold detection circuit senses if the specified voltage level exists at the receiver buffer.
- This is indicated by the rlos\_lo\_ch(0-3) signal.
- This setting can be used both for PCI Express Electrical Idle Detection and PCI Express beacon detection (when in power state P2).
- The remote transmitting device can have a beacon output voltage of 6 db down from VTX-DIFFpp (i.e., 201 mV). If this signal can be detected, then the beacon is detected.

### PCS Beacon Transmission Support

Sending the K28.5 character (IDLE) (5 ones followed by 5 zeroes) provides a periodic pulse with of 2 ns occurring every 2 ns (1.0 UI = 400 ps, multiplied by 5 = 2 ns). This meets the lower requirement. The output beacon voltage level can then be VTX-DIFFp-p. This is a valid beacon transmission.

### SDI (SMPTE) Mode

The SDI mode of the LatticeECP3 SERDES/PCS block supports all three SDI modes, SD-SDI, HD-SDI and 3G-SDI.

#### Transmit Path

- Serializer

#### Receive Path

- Deserializer
- Optional word alignment to user-defined alignment pattern.

The following data rates are the most popular in the broadcast video industry.

- SD-SDI (SMPTE259M): 270Mbps
- HD-SDI (SMPTE292M): 1.485Gbps,  $1.485\text{Gbps}/1.001 = 1.4835\text{Gbps}$
- 3G-SDI (SMPTE424M): 2.97Gbps,  $2.97\text{Gbps}/1.001 = 2.967\text{Gbps}$

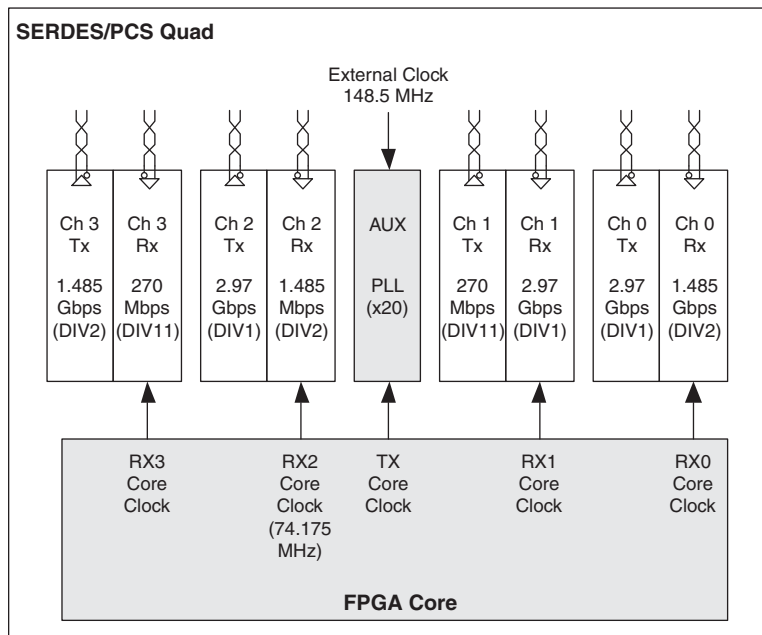
Most designers have indicated that they would like to see support for all these rates. The reason is that in a broadcast studio, or a satellite head-end or cable head-end, they do not necessarily have prior knowledge of what the RX data rate will be.

The switchover time between different rates should be as low as possible. The time to re-lock the CDR is unavoidable. In the LatticeECP3 SERDES, the PLL does not have to be re-locked. This is possible because the LatticeECP3 has per RX and TX dividers. Video links generally have a uni-directional nature (i.e., different channels can run at different rates, and more importantly, the RX and TX in the same channel can run at different rates).

Also, depending on the geography where the equipment is deployed, either the full HD/3G-SDI rates (Europe/Asia) are used while transmitting video or the fractional rates (North America - NTSC). This allows us to develop two potential solution example cases for multi-rate SMPTE support with high quad utilization.

Please note that simultaneous support of 3G/HD Full TX Rate(s) and Fractional TX Rate(s) is not possible in the same SERDES quad. In general, based on the above, geographically partitioned usage is an acceptable limitation.

Figure 8-31. Example A: 3G/HD/SD Full RX/TX Rate Support and 3G/HD Fractional TX Rate Support



To support the major application requirements, a selectable DIV per RX and TX is supported. In LatticeECP3, DIV11 has been added. One potential multi-rate configuration is to provide a 148.5MHz REFCLK from the primary pins to the TX PLL. The TX PLL would be in the x20 mode. The resulting output clock would be 2.97GHz. Then, by using the DIV2 for 1.485Gbps and DIV11 for 270Mbps, a very quick switchover can be achieved without having to re-train and lock the PLL.

## Serial RapidIO (SRIO) Mode

This section describes the operation of the Serial RapidIO mode of the SERDES/PCS block. The LatticeECP3 supports 1x and 4x Serial RapidIO applications with one PCS quad. SRIO1.0 highlights multiple frequency support, 3.125Gbps, 2.5Gbps and 1.25Gbps. The ratio of these rates is 2.5:2:1. Supporting all of these rates with integer dividers in the same quad is not possible, but the ratio of 2.5 Gbps to 1.25 Gbps is 2:1 (full rate : half rate).

### Transmit Path

- Serializer
- 8b10b encoding

### Receive Path

- Deserializer
- Word alignment based on the Sync Code Group as defined in the RapidIO Physical Layer 1x/4x LP-Serial specification.
- 8b10b decoding
- Clock Tolerance Compensation logic capable of accommodating clock domain differences

## Serial Digital Video and Out-Of-Band Low Speed SERDES Operation

The LatticeECP3 SERDES/PCS supports any data rates that are slower than what the SERDES TX PLL and RX CDR natively support (<250Mbps: Out-Of-Band signal, OOB), by bypassing the receiver CDR and associated SERDES/PCS logic (e.g., 100Mbps Fast Ethernet, SD-SDI at 143Mbps or 177Mbps). Though these out-of-band paths primarily use low data rates, higher rates can be used for other functional reasons. See the Multi-Rate SMPTE Support section of this document for more information.

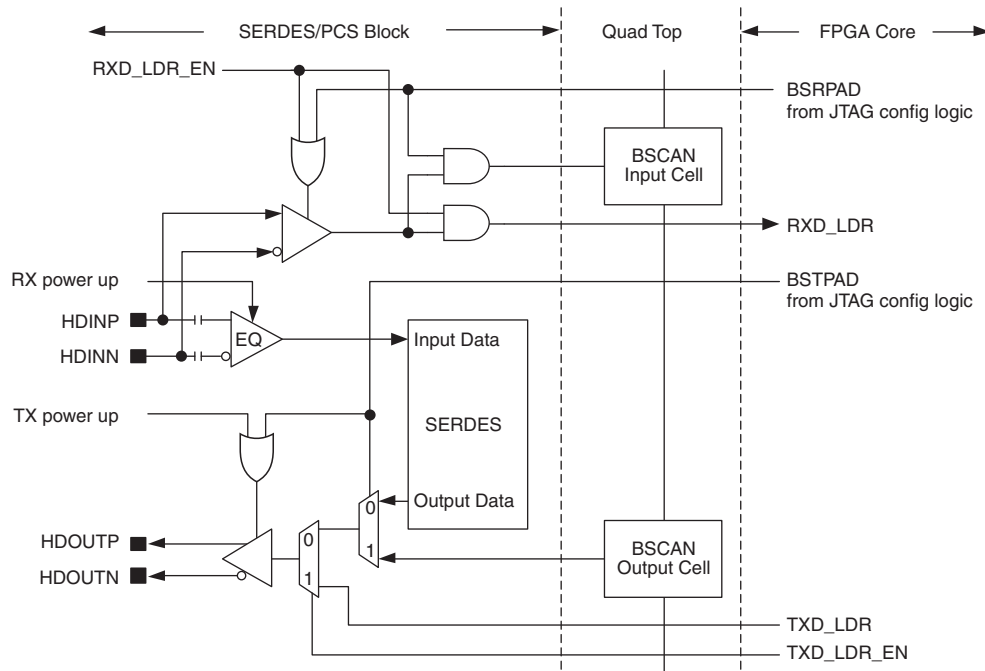
In addition, for SD-SDI, these rates sometimes must co-exist on the same differential RX pair with HD-SDI rates (i.e., SD-SDI rates may be active and then the data rate may switch over to HD-SDI rates). Since there is no way to predict which of these two rates will be in effect, it is possible to send the input data stream to two SERDES in parallel, a high-speed SERDES (already in the quad) and a lower-speed SERDES (implemented outside the quad). One possible implementation is shown in Figure 8-32.

There is an input per channel RXD\_LDR, low data rate single-ended input from the RX buffer to the FPGA core. In the core a low-speed Clock Data Recovery (CDR) block or a Data Recovery Unit (DRU) can be built using soft logic. A channel register bit, RXD\_LDR\_EN, can enable this data path. If enabled by another register bit, a signal from the FPGA can also enable this in LatticeECP3.

In the transmit direction, it is also possible to use a serializer built in soft logic in the FPGA core and use the TXD\_LDR pin to send data into the SERDES. It will be muxed in at a point just before the pre-emphasis logic near where the regular high-speed SERDES path is muxed with the boundary scan path. This is shown conceptually in Figure 8-32. The low data rate path can be selected by setting a channel register bit, TX\_LDR\_EN.

Alternatively, on the output side, the high-speed SERDES is used to transmit either high-speed data, or lower speed data using decimation (the SERDES continues to run at high-speed, but the output data can only change every nth clock where n is the decimation factor).

**Figure 8-32. Possible Implementation of Serial Digital Video Support**



---

## Open Base Station Architecture Initiative (OBSAI)

OBSAI is an open forum that is aimed at an open market for cellular base stations.

The LatticeECP3 SERDES/PCS supports most of the OBSAI features except the 3.84Gbps rate.

### Transmit Path

- Serializer
- Transmit State Machine is set to Gigabit Ethernet Mode
- 8b10b Encoding

### Receive Path

- Deserializer
- Word alignment based on IEEE 802.3-2002 1000 BASE-X defined alignment characters
- 8b10b Decoding

A Basestation Transceiver System (BTS) has four main components/modules and there are three major interfaces, or Reference Points (RPs), between them.

- **RP3** – RF Module receives signals from portable devices (terminals) and down converts it to digital data.
- **RP2** – The baseband module takes the encoded signal and processes it and sends it to the transport module, which will send it over the terrestrial network.
- **RP1** – A control module maintains coordination between these three functions.

Currently, most of the focus in the industry revolves around providing lower RF modules and power amplifiers and hence OBSAI's primary effort has been to define Reference Point 3 (RP3). In fact, the specification of interest is RP3-01, which is focused on Remote Radio Heads (RRHs).

The OBSAI RP3 electrical specification is based on the XAUI electrical specification and customized to the needs of a Base Transceiver System. The XAUI electrical interface is specified in Clause 47 of IEEE 802.3ae-2002. RP3 version 3.1 specifies the following electrical rates 3.84Gbps, 3.072Gbps, 2.304Gbps, 1.536Gbps and 0.736Gbps out of which the last four are supported.

The RP3 electrical specification defines a receiver compliance mask and provides a sample transmitter output mask. The BER should be better than  $1 \times 10^{-15}$ , which is more stringent than XAUI requirement of  $1 \times 10^{-12}$ . The RP3 electrical specification also differs from the XAUI specification in the definition of the UI. XAUI allows for a difference of +/- 100ppm. This difference does not apply to OBSAI systems since the BTSes are fully synchronous systems.

Since a BTS is a synchronous system, it is imperative to measure and calibrate the delay across any bus. OBSAI has carefully considered this and, as a result, come up with a method for synchronizing the master frame across the RP3 link. Delay calibration takes into account all factors, including processing, and buffer delay, in transmit and receive modules, as well as the latency across the link.

Another major item in the data-link layer is the synchronization between the transmitter and receiver. Synchronization ensures the actual data can be decoded successfully over the link. The frequency of errors as well as the synchronization status is constantly monitored.

RP3-01 has gone further and specified line rates that are integer multiples of 768Mbps, up to 3.84Gbps, and are considered OBSAI compatible line rates. Due to the number of line rates available, auto-negotiation between the remote RF units and the local units is defined. This extension of the specification includes Ethernet transmission between two RP3-01 nodes, mapping of RP1 information into the RP3 link because the RRH does not have a

physical RP1 link, delay measurement, synchronization between RP3-01 units, and data multiplexing across the RP3-01 link.

The delay of each functional block in the LatticeECP3 SERDES/PCS is described in the CPRI section.

## **Common Public Radio Interface (CPRI)**

The goal of CPRI is to allow base station manufacturers and component vendors to share a common protocol and more easily adapt platforms from one user to another.

### **Transmit Path**

- Serializer
- Transmit State Machine is set to Gigabit Ethernet Mode
- 8b10b Encoding

### **Receive Path**

- Deserializer
- Word alignment based on IEEE 802.3-2002 1000 BASE-X defined alignment characters
- 8b10b Decoding

Unlike OBSAI, CPRI does not specify mechanical or electrical interface requirements. In terms of scope, CPRI has a much narrower focus than OBSAI. CPRI looks solely at the link between the RRH and the baseband module(s). In CPRI nomenclature, those modules are known as Radio Equipment (RE) and Radio Equipment Control (REC), respectively. In other words, CPRI is specifying the same interface as the OBSAI RP3 specification. CPRI primarily covers the physical and data link layer of the interface. It also specifies how to transfer the user plane data, control and management (C&M) plane data and the synchronization plane data.

CPRI has had better “traction” for two reasons - the muscle of the companies backing it and the focus on just one interface link (between the RF modules and the Baseband modules) and even at that focusing primarily on the physical and data link layers.

CPRI allows four line bit rate options; 614.4Mbps, 1.2288Gbps, 2.4576Gbps and 3.072Gbps; at least one of these rates needs to be supported. The higher line rate is always compared to the one that is immediately lower.

CPRI does not have a mandatory physical layer protocol, but the protocol used must meet the BER requirement of  $1 \times 10^{-12}$ , which is less stringent than OBSAI. It also specifies the clock stability and the phase noise requirements.

CPRI also recommends two electrical variants: high voltage (HV) and low voltage (LV). HV is guided by 1000Base-CX specifications in IEEE 802.3-2002 clause 39 with 100-ohm impedance. LV is guided by XAUI. LV is recommended for all rates and will be the focus for this device.

It is important to understand two link layer requirements when dealing with the CPRI and OBSAI specifications:

- Link delay accuracy and cable delay calibration
- Startup synchronization

### **Link Delay Accuracy and Cable Delay Calibration**

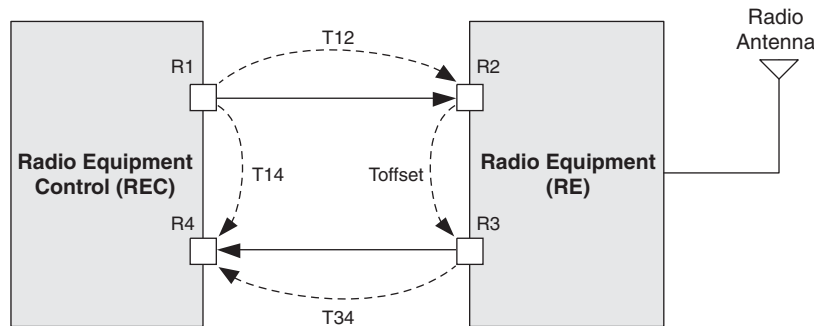
Though the following discussion largely leverages from the CPRI requirements, the same requirements also apply to OBSAI implementations.

The REs or RRHs are frequency locked to the REC or BTS. Thus, in this synchronous system it is necessary to calibrate all delays between RRHs and the BTS to meet air-interface timing requirements. The interface requires the support of the basic mechanisms to enable calibrating the cable delay on links and the round trip delay on sin-



gle- and multi-hop connections. Specifically, the reference points for delay calibration and the timing relationship between input and output signals at RE (Radio Equipment) are defined. All definitions and requirements are described for a link between REC Master Port and RE Slave Port in the single-hop scenario as shown in Figure 8-33.

**Figure 8-33. Link Between REC Master Port and RE Slave Port (Single Hop Scenario)**



Reference points R1-4 correspond to the output point (R1) and the input point (R4) of REC, and the input point (R2) and the output point (R3) of an RE terminating a particular logical connection. The antenna is shown as Ra for reference.

- T12 is the delay of the downlink signal from the output point of REC (R1) to the input point of RE (R2), essentially the downlink cable delay.
- T34 is the delay of the uplink signal from the output point of RE (R3) to the input point of the REC (R4), essentially the uplink cable delay.
- Toffset is the frame offset between the input signal at R2 and the output signal at R3.
- T14 is the frame timing difference between the output signal at R1 and the input signal at R4 (i.e., the round trip delay - RTT).

Delay measurement is accomplished using frame timing. CPRI has a 10ms frame based on the UMTS radio frame number or Node B Frame Number, also known as BFN. Each UMTS Radio Frame has 150 hyperframes (i.e., each HyperFrame is 66.67us) with the corresponding hyperframe number (HFN =  $0 \leq Z \leq 149$ ). Each hyperframe has 256 ( $0 \leq W \leq 255$ ) basic frames (i.e. each basic frame is 260.42ns = Tchip or Tc).

An RE determines the frame timing of its output signal (uplink) to be the fixed offset (Toffset) relative to the frame timing of its input signal (downlink). Toffset is an arbitrary value, which is greater than or equal to 0 and less than 256 Tc (it cannot slip beyond a hyperframe). Different REs may use different values for Toffset. REC knows the value of Toffset of each RE in advance (pre-defined value or RE informs REC by higher layer message).

To determine T14, the downlink BFN and HFN from REC to RE is given back in uplink from the RE to the REC. In the case of an uplink-signaled error condition, the REC treats the uplinks BFN and HFN as invalid. So,  $T14 = T12 + Toffset + T34$ .

As stated earlier the system is synchronous. Further, assuming that hyperframes are of fixed length and the RRH-BTS interconnect (cable length) is equal in both directions (i.e.,  $T12 = T34$ , both optical fibers are in one bundle), the interconnect delay devolves down to  $(T14 - Toffset)/2$ . The method for determining T14 has been discussed earlier. So the major component that affects delay calibration is Toffset. Thus, the interconnect delay is the difference in hyperframe arrival and departure times measured at each side of the link.

Delay calibration requirements are driven by 3GPP and UTRAN requirements specifically requirements R-21 in the CPRI specification (CPRI v3.0 page 20), which states that the accuracy of the round trip delay measurement of the cable delay of one link is  $\pm Tc/16$ . Additionally, requirement R-20 states that the round trip time absolute accuracy of the interface, excluding the round trip group delay on the transmission medium (excluding the cable length), shall

meet a similar requirement ( $\pm T_c/16$  for T14). Taking into account the previous discussion, the absolute link delay accuracy in the downlink between REC master port and RE slave port excluding the cable length is half of the above requirement ( $\pm T_c/32$  or approximately 8ns (8.138ns)). Thus, both T14 and Toffset need absolute accuracy less than  $\pm 8$ ns.

Next it is important to determine how many bits of uncertainty can be acceptable for the different rates. Essentially, the various CPRI and OBSAI bit rates can be multiplied by 8.138ns to determine the number of bits worth of indeterminism/variance is acceptable. The impact of this will become clear subsequently when the SERDES serial/parallel data path is discussed.

Most SERDES have a certain level of uncertainty that is introduced in the serializing and de-serializing process. Thus, a SERDES with 16-bit bus architecture may have twice the delay uncertainty as a SERDES with a 8-bit architecture because the number of bits per word is doubled.

TX and RX latency respectively in Table 8-23 is listed. The table also lists the variability between the latency. This variability directly contributes to the absolute delay accuracy required from earlier discussion. The variability comes from three sources: TX FPGA Bridge FIFO, RX FPGA Bridge FIFO and RX Clock Tolerance Compensation FIFO. Since the CPRI system is a synchronous system, the RX CTC FIFO is bypassed and the RX recovered clock is used.

The remaining contributors to the latency variability are the FPGA Bridge FIFO. This FIFO can be bypassed if the interface to the FPGA is 8-bit bus mode. In 16-bit interface mode, the FPGA Bridge FIFO cannot be bypassed because the 2:1 gearing is done via the FIFO.

## SONET/SDH

Synchronous Optical Networking (SONET) and Synchronous Digital Hierarchy (SDH) are standardized multiplexing protocols that transfer data over optical fiber or via an electrical interface. SONET generic criteria are detailed in the Telcordia Technologies Generic Requirements document GR-253-CORE. Generic criteria applicable to SONET and other transmission systems (e.g., asynchronous fiber optic systems or digital radio systems) are found in Telcordia GR-499-CORE. SONET and SDH were originally designed to transport circuit mode communications (e.g., T1, T3) from a variety of different sources. The primary difficulty in doing this prior to SONET was that the synchronization sources of these different circuits were different. This meant each circuit was operating at a slightly different rate and with different phase. SONET allowed for the simultaneous transport of many different circuits of differing origin within a single framing protocol.

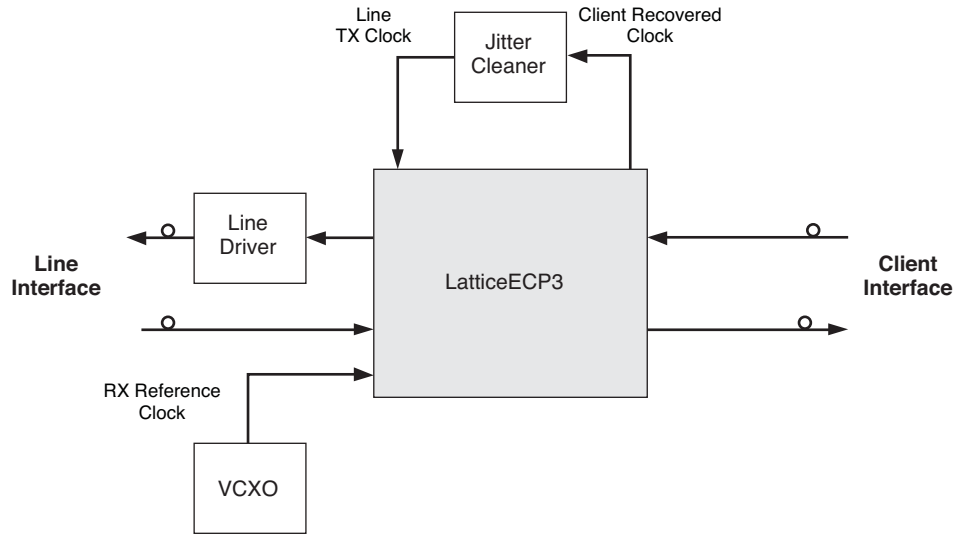
The LatticeECP3 SERDES/PCS offers transceivers capable of supporting three SONET/SDH data rates, STS-3/STM-1 (155.52 Mbps), STS-12/STM-4 (622.08 Mbps) and STS-48/STM-16 (2.488 Gbps). 8-bit SERDES mode is used for SONET/SDH applications.

In order to be SONET/SDH line-compliant, external components are required with the LatticeECP3. An external line driver is required on the output of the SERDES. To filter out high frequency jitter from the incoming data stream, a jitter cleaner can be applied to the recovered clock before using it as the transmit reference clock.

For chip-to-chip or backplane applications, the external line driver and the clock jitter cleaner are not required.

Figure 8-34 shows the line-side solution using external components.

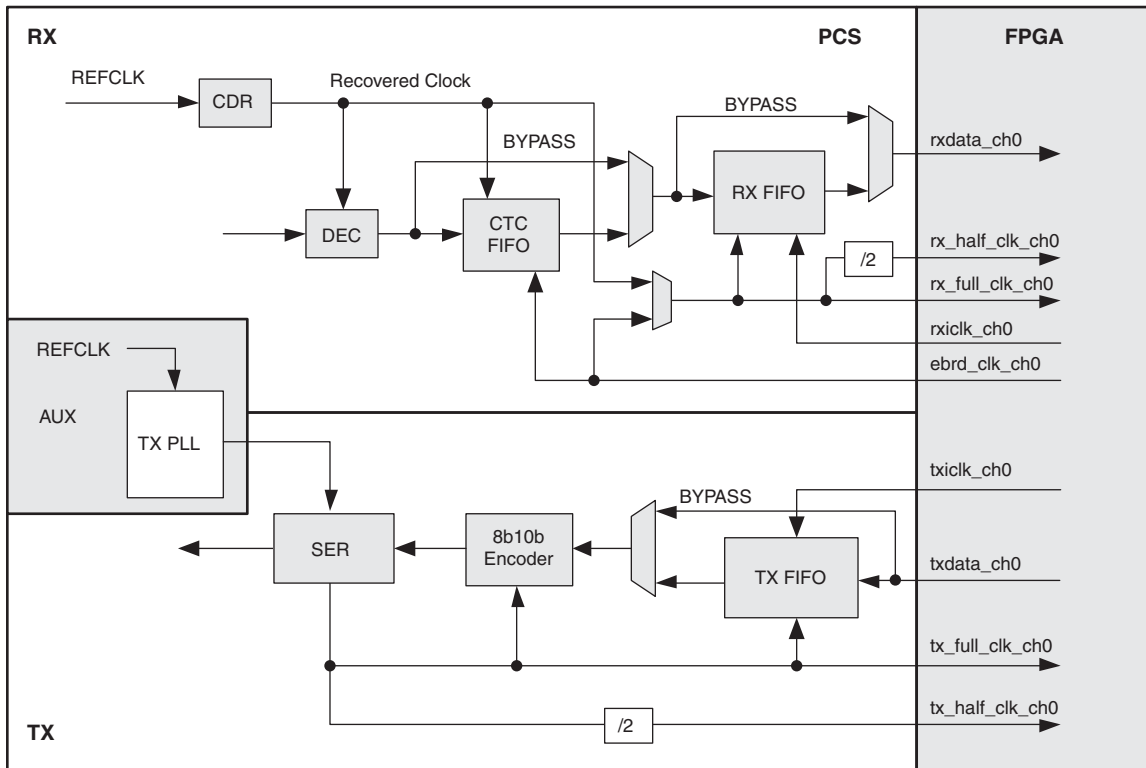
**Figure 8-34. SONET/SDH Line Interface**



**FPGA Interface Clocks**

Figure 8-35 shows a conceptual diagram of the later stage of the PCS core and the FPGA Bridge and the major clocks that cross the boundary between PCS and the FPGA.

**Figure 8-35. Conceptual PCS/FPGA Clock Interface Diagram**



In the above diagram and in the subsequent clock diagrams in this section, please note that suffix “i” indicates the index [3:0] i.e., one for each channel.

It is a requirement that if any of the selectors to the clock muxes are changed by writes to register bits, the software agent will reset the logic clocked by that muxed clock.

The PCS outputs 16 clocks. There are two transmit clocks (per channel) and two receive clocks (per channel). The two transmit clocks provide full rate and half rate clocks and are all derived from the TX PLL. There are also two clocks (full and half) per receive channel. All 16 clocks can be used as local (secondary) or global (primary) clocks for the FPGA logic as required. tx\_half\_clks are used when the gearing is in 2:1 mode. As described in Table 8-5, only tx\_full\_clk\_ch0 and tx\_half\_clk\_ch0 can drive the primary clock routing directly. Other channel clocks can also drive the primary clock net but general routing is used. All of the tx\_full\_clk\_ch[3:0] and tx\_half\_clk\_ch[3:0] signals can drive the secondary clock net by applying a USE\_SECONDARY clocking preference. General routing is also used to drive Secondary clock net.

The transmit clock is used on the write port of the TX FIFO (or Phase Shift FIFO, depending on the case). One of the two receive clocks is connected to the read clock of the RX FIFO. The other clock is used on the read port of the CTC FIFO and potentially on the write port of the RX FIFO (depending on the case). Based on whether the CTC and the TX FIFO are bypassed and whether the PCS is in 8bit/10bit interface mode or 16bit/20bit interface mode, four use cases are possible. The active paths are highlighted with weighted lines. It is also indicated how many and what kind of clock trees are required. There are some modes that would more commonly be preferred by the user.

This section describes the operation of the six supported cases. The cases are outlined in Table 8-21.

**Table 8-21. Six Interface Cases Between the SERDES/PCS Quad and the FPGA Core**

Interface	Data Width	RX CTC FIFO	RX Phase-Shift/ Down-Sample FIFO	TX Phase-Shift/ Up-Sample FIFO
Case I-a <sup>2</sup>	8/10 bit	Yes	Yes	Yes
Case I-b <sup>2</sup>	8/10 bit	Bypass	Yes	Yes
Case I-c <sup>2</sup>	8/10 bit	Yes	Bypass	Bypass
Case I-d <sup>2</sup>	8/10 bit	Bypass	Bypass	Bypass
Case II-a <sup>1, 2</sup>	16/20 bit	Yes	Yes	Yes
Case II-b <sup>1, 2</sup>	16/20 bit	Bypass	Yes	Yes

1. When using a 16/20-bit datapath width, the TX phase-shift (upsample) FIFO and the RX phase-shift FIFO (downsample) are always used. They cannot be bypassed. It is not required that both RX and TX have the same FPGA interface datapath width simultaneously. There is independent control available. For the sake of brevity, they have been represented together in the same use case.
2. The TX phase-shift (upsample) FIFO and the RX phase-shift FIFO (downsample) don't need to be bypassed together. They are independently controllable. Again, for the sake of brevity, they have been represented here in the same case.

### 2:1 Gearing

For guaranteed performance of the FPGA global clock tree, it is recommended to use a 16/20-bit wide interface for SERDES line rates greater than 2.5Gbps. In this interface, the FPGA interface clocks are running at half the byte clock frequency.

Even though the 16/20-bit wide interface running at half the byte clock frequency can be used with all SERDES line rates, the 8/10-bit wide interface is preferred when the SERDES line rate is low enough to allow it (2.5Gbps and below) because this results in the most efficient implementation of IP in the FPGA core.

The decision matrix for the six interface cases is explained in Table 8-22.

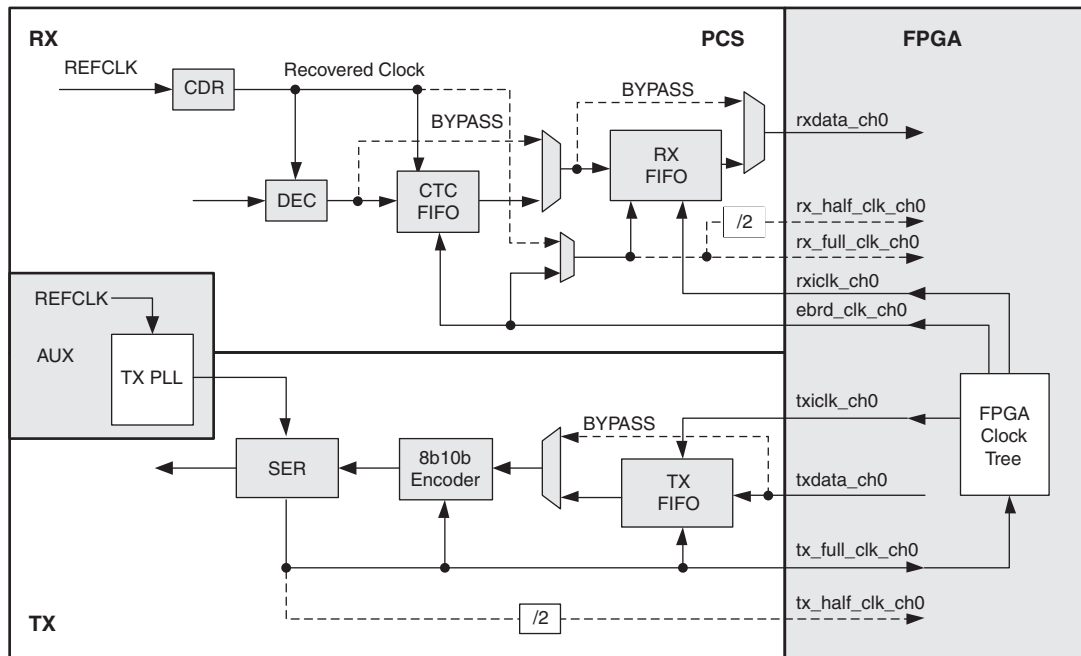
**Table 8-22. Decision Matrix for Six Interface Cases**

SERDES Line Rate	Datapath Width	Multi-Channel Alignment Required?	CTC Required?	RX FIFO Required?	Interface Case
2.5 Gbps and below	8/10 bit (1:1 gearing)	No, single-channel link	Yes	Yes	Case I_a <sup>1</sup>
				No	Case I_c <sup>1</sup>
			No	Yes	Case I_b <sup>2</sup>
				No	Case I_d <sup>2</sup>
		Yes, multi-channel link	Yes	Case I_b <sup>3</sup>	
			No	Case I_d <sup>3</sup>	
3.2 Gbps and below	16/20 bit (2:1 gearing)	No, single-channel link	Yes	Yes	Case II_a <sup>4</sup>
			No	Yes	Case II_b <sup>5</sup>
		Yes, multi-channel link	Must bypass, not available	Yes	Case II_b <sup>6</sup>

1. This case is intended for single-channel links at line rates of 2.5 Gbps and lower (8/10-bit wide interface) that require clock tolerance compensation in the quad. CTC is required when both ends of the link have separate reference clock sources that are within +/- 300ppm of each other. Case I\_a is used if the IP in the core requires the RX phase-shift FIFO. Case I\_b is used if the IP does not require this FIFO.
2. This case is intended for single-channel links at line rates of 2.5Gbps and lower (8/10-bit wide interface) that do NOT require clock tolerance compensation in the quad. CTC is not required when both ends of the link are connected to the same reference clock source. This is often the case for chip-to-chip links on the same circuit board. There is exactly 0ppm difference between the reference clocks and so CTC is not required and can be bypassed. CTC is also not required in the quad when this function is performed by the IP in the core.
3. This case is intended for multi-channel links at line rates of 2.5Gbps and lower (8/10-bit wide interface). Multi-channel alignment MUST be done in the FPGA design. Since multi-channel alignment must be done prior to CTC, the CTC FIFO in the quad MUST be bypassed when multi-channel alignment is required and so both multi-channel alignment and CTC (if required) are done by the FPGA design.
4. This case is intended for single-channel links at line rates of 3.2Gbps and lower that require a 2:1 gearbox between the quad and the FPGA core (16/20 bit wide interface). Clock tolerance compensation is included in the quad. CTC is required when both ends of the link have separate reference clock sources that are within +/- 300ppm of each other.
5. This case is intended for single-channel links at line rates of 3.2Gbps and lower that require a 2:1 gearbox between the quad and the FPGA core (16/20-bit wide interface). Clock tolerance compensation is NOT included in the quad. CTC is not required when both ends of the link are connected to the same reference clock source. This is often the case for chip-to-chip links on the same circuit board. There is exactly 0ppm difference between the reference clocks and so CTC is not required and can be bypassed. CTC is also not required in the quad when this function is performed by the FPGA design.
6. This case is intended for multi-channel links at line rates of 3.2Gbps and lower that require a 2:1 gearbox between the quad and the FPGA core (16/20-bit wide interface). Multi-channel alignment MUST be done by the FPGA design. Since multi-channel alignment must be done prior to CTC, the CTC FIFO in the quad MUST be bypassed when multi-channel alignment is required and so both multi-channel alignment and CTC (if required) are done by the FPGA design.

### Case I\_a: 8/10-Bit, CTC FIFO and RX/TX FIFOs Not Bypassed

Figure 8-36. 8/10-Bit, CTC FIFO and RX/TX FIFOs NOT Bypassed



1. The TX FIFO acts as a Phase Shift FIFO only in this case.
2. The RX FIFO acts as a Phase Shift FIFO only in this case.
3. The quad-level full rate clock from the TX PLL (tx\_full\_clk) has direct access to the FPGA center clock mux. This is a relatively higher performance path. A Global Clock Tree out of the Center Clock Mux is used to clock the user's interface logic in the FPGA. Some leaf nodes of the clock tree are connected to the FPGA Transmit Input clock (txicl\_k), the CTC FIFO Read Clock per Channel (ebrd\_clk) through the FPGA Receive Input clock (rxicl\_k). This case is possibly the most common single-channel use case.

#### Example of Clock and Data Signals Interface in FPGA Logic (Case I\_a)

Below is a portion of the SERDES/PCS module instantiation in the top module which describes how clock and data ports are mapped in Verilog.

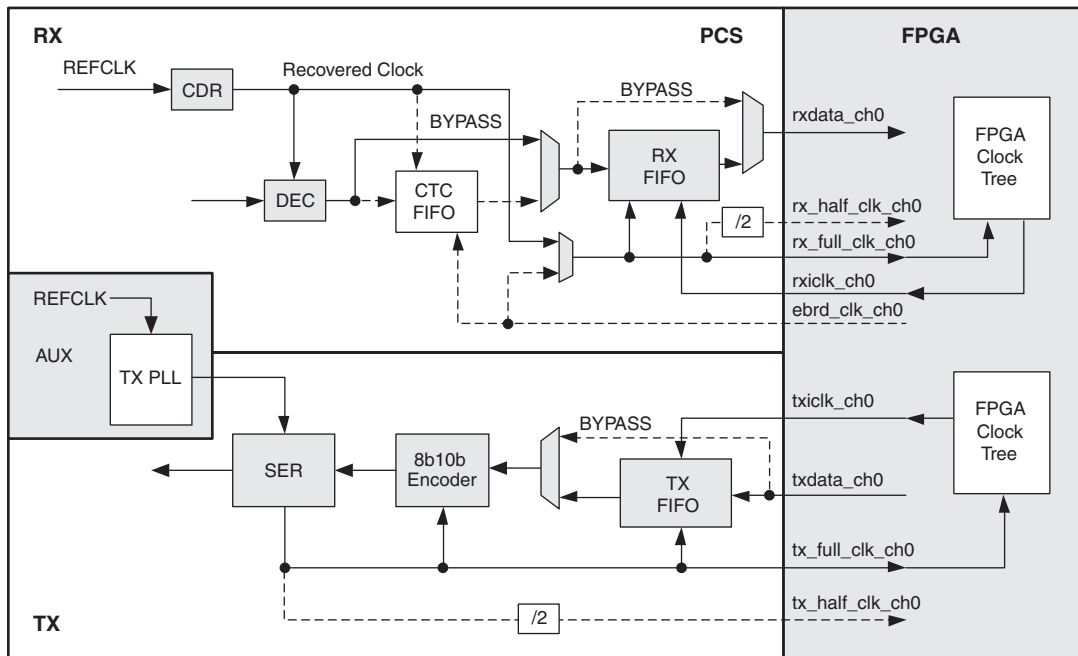
```
.txicl_k_ch0(txclk),
.rxicl_k_ch0(txclk),
.rx_full_clk_ch0(),
.tx_full_clk_ch0(txclk),
.tx_half_clk_ch0(),
.txdata_ch0(txdata_2_pcs),
.rxdata_ch0(rxdata_from_pcs),
.tx_k_ch0(txkcntl_2_pcs),
.rx_k_ch0(rxkcntl_from_pcs),
```

ebrd\_clk\_ch0 is routed automatically by the software, depending on the case.

Note that tx\_full\_clk\_ch0 uses wire name 'txclk' and feeds both txi\_clk\_ch0 and rxi\_clk\_ch0, as shown in Figure 8-36.

**Case I\_b: 8/10-Bit, CTC FIFO Bypassed**

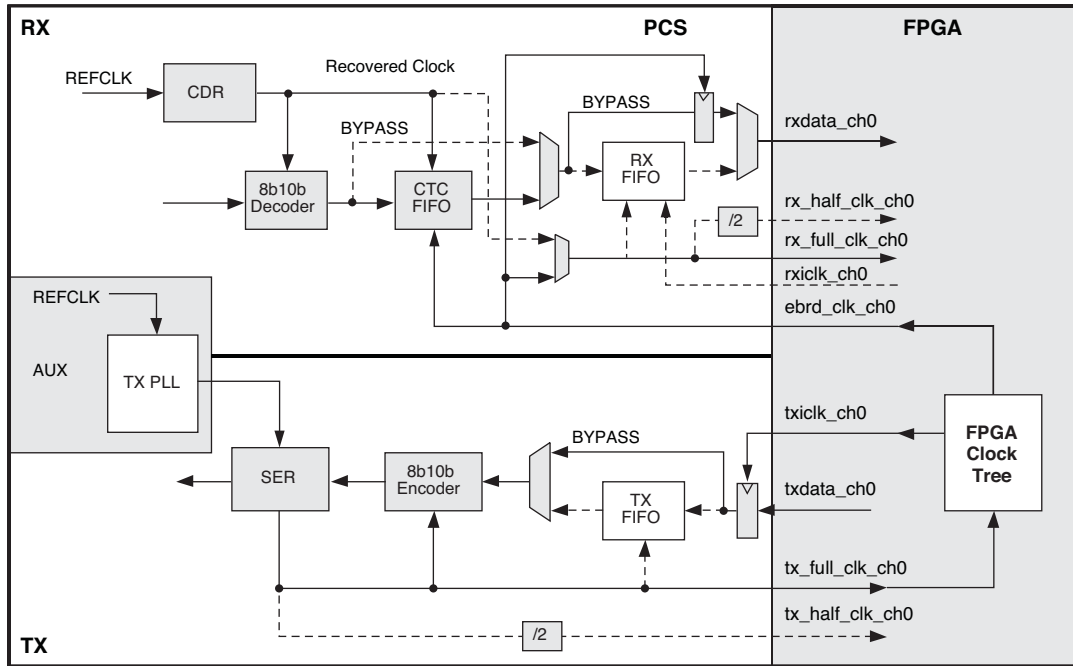
*Figure 8-37. 8b/10-Bit, CTC FIFO Bypassed*



1. The TX FIFO acts as a Phase Shift FIFO only in this case.
2. The RX FIFO acts as a Phase Shift FIFO only in this case.
3. The TX FPGA Channel input clock is clocked similarly as in the previous case using a clock tree driven by a direct connection of the full rate transmit FPGA output clock to the FPGA center clock mux. Once the CTC FIFO is bypassed, the recovered clock needs to control the write port of the RX FIFO. The recovered clock of each channel may need to drive a separate local or global clock tree (i.e., up to four local or global clock trees per quad). The clock tree will then drive the FPGA receive clock input to control the read port of the RX FIFO. The reason for bypassing the CTC FIFO in this case is most likely for doing multi-channel alignment in the FPGA core. It implies that CTC using an elastic buffer will be done in the FPGA core. The CTC FIFOs can be written by either the recovered clocks or by a master recovered clock. The read of the CTC FIFO will be done using the TX clock via the TX clock tree.

**Case I\_c: 8/10-Bit, RX/TX FIFO Bypassed**

*Figure 8-38. 8/10-Bit, RX/TX FIFO Bypassed*

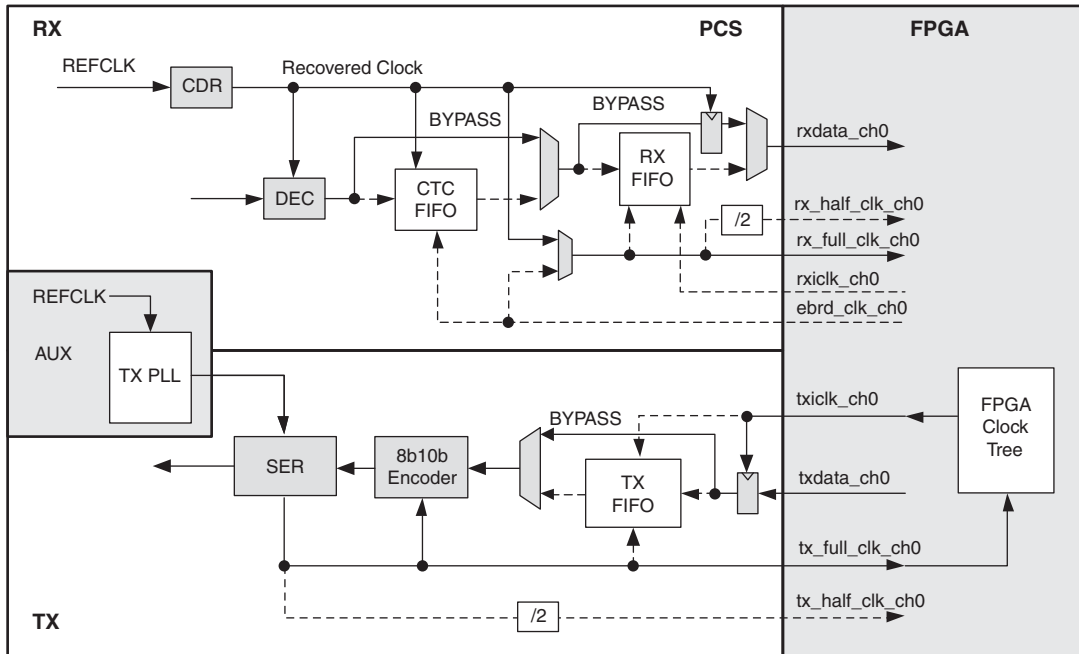


1. The TX channel clocking is similar to the previous two cases. On the RX channel, the FPGA input clock is now ebrd\_clk\_i. The FPGA TX clock tree drives this clock. In this case, ebrd\_clk\_i is automatically routed by the software.



**Case I\_d: 8/10-Bit, CTC FIFO and RX/TX FIFOs Bypassed**

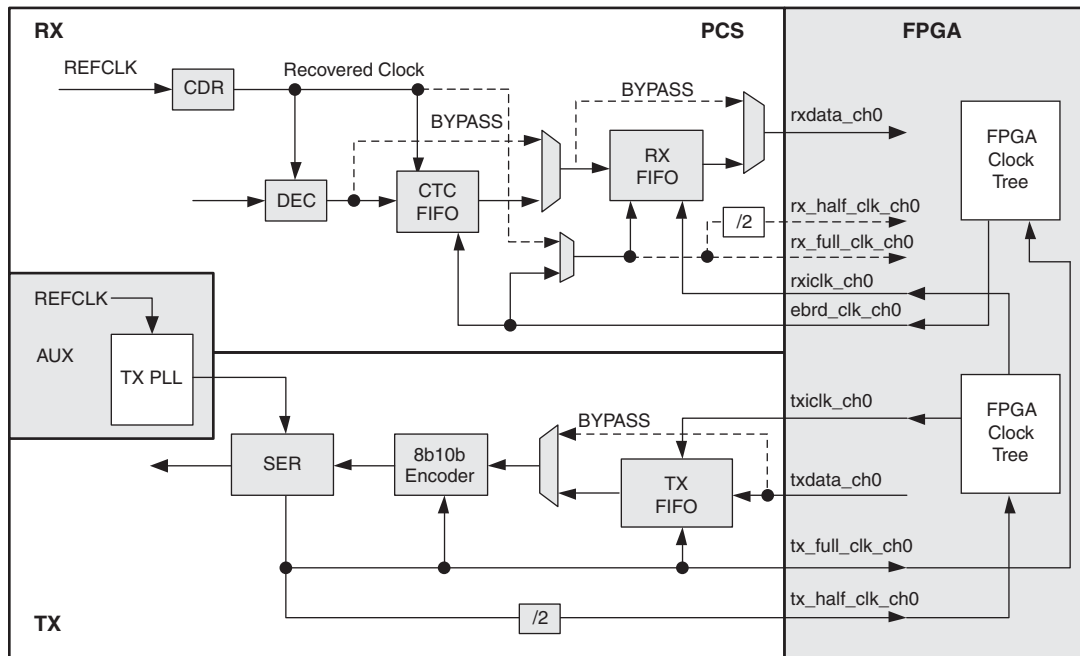
*Figure 8-39. 8b/10-Bit, CTC FIFO and RX/TX FIFOs Bypassed*



1. FPGA clock trees can be interchangeably thought of as clock domains in this case. The TX channel clocking is similar to the previous three cases. On the RX channel, the recovered channel RX clock is sent out to the FPGA. This case is useful for supporting video applications.

**Case II\_a: 16/20-bit, CTC FIFO and RX/TX FIFOs NOT Bypassed**

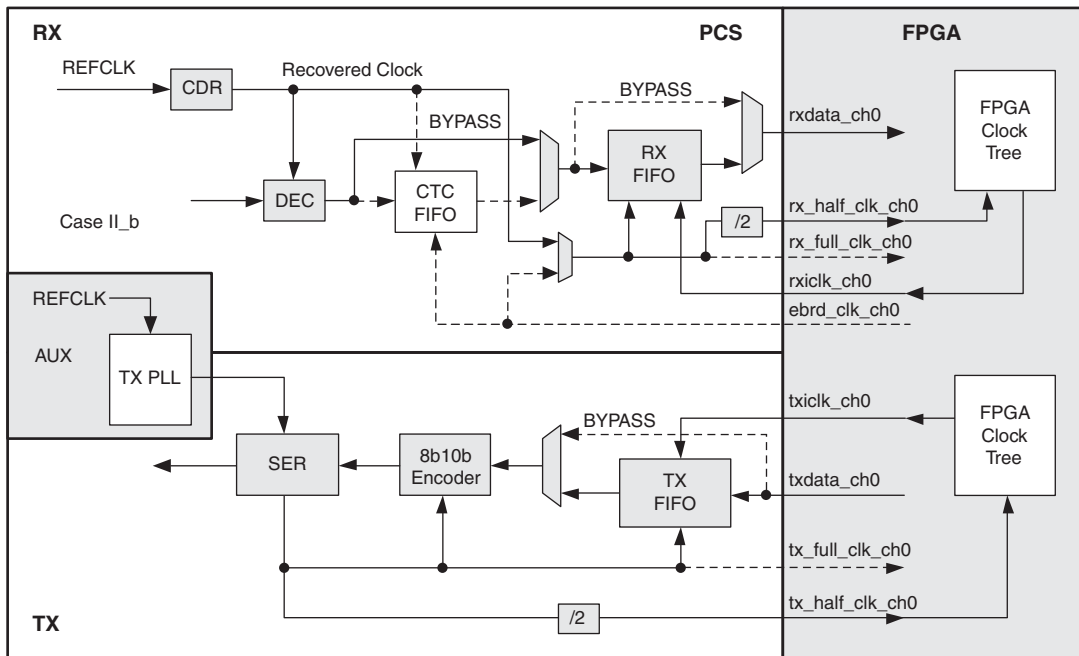
*Figure 8-40. 16/20-bit, CTC FIFO and RX/TX FIFOs NOT Bypassed*



1. The TX FIFO acts both as a Phase Shift FIFO and Upsample FIFO in this case.
2. The RX FIFO acts both as a Phase Shift FIFO and Downsample FIFO in this case.
3. This is a very common single channel use case when the FPGA is unable to keep up with full byte frequency. Two clock trees are required. These clock trees are driven by direct access of transmit full-rate clock and transmit half-rate clock to the FPGA clock center mux. The full-rate clock tree drives the CTC FIFO read port and the RX FIFO write port. The half-rate clock tree drives the RX FIFO and the FPGA logic.

**Case II\_b: 16/20-bit, CTC FIFO Bypassed**

*Figure 8-41. 16/20-bit, CTC FIFO Bypassed*



1. The TX FIFO acts both as a Phase Shift FIFO and Upsample FIFO in this case.
2. The RX FIFO is acting both as a Phase Shift FIFO and Downsample FIFO in this case.
3. This is a very common multi-channel alignment use case when the FPGA is unable to keep up with full byte frequency. The receive clock trees (up to four) can be local or global. They are running a half-rate clock. The transmit clock tree is driven by direct access of the transmit half-rate clock to the FPGA clock center mux.

## SERDES/PCS Block Latency

Table 8-23 describes the latency of each functional block in the transmitter and receiver. Latency is given in parallel clock cycles. Figure 8-42 shows the location of each block.

**Table 8-23. SERDES/PCS Latency Breakdown**

Item	Description	Min.	Avg.	Max.	Fixed	Bypass	Units
<b>Transmit Data Latency<sup>1</sup></b>							
T1	FPGA Bridge – 1:1 gearing with different clocks	1	3	5	—	1	word clk
	FPGA Bridge – 1:1 gearing with the same clocks	—	—	—	3	1	word clk
	FPGA Bridge – 2:1 gearing	1	3	5	—	—	word clk
T2	8b10b Encoder	—	—	—	2	1	word clk
T3	SERDES Bridge transmit	—	—	—	2	1	word clk
T4	Serializer: 8-bit mode	—	—	—	15 + $\Delta 1$	—	UI + ps
	Serializer: 10-bit mode	—	—	—	18 + $\Delta 1$	—	UI + ps
T5	Pre-emphasis ON	—	—	—	1 + $\Delta 2$	—	UI + ps
	Pre-emphasis OFF	—	—	—	0 + $\Delta 3$	—	UI + ps
<b>Receive Data Latency<sup>2</sup></b>							
R1	Equalization ON	—	—	—	$\Delta 1$	—	UI + ps
	Equalization OFF	—	—	—	$\Delta 2$	—	UI + ps
R2	Deserializer: 8-bit mode	—	—	—	10 + $\Delta 3$	—	UI + ps
	Deserializer: 10-bit mode	—	—	—	12 + $\Delta 3$	—	UI + ps
R3	SERDES Bridge receive	—	—	—	2	1	word clk
R4	Word alignment <sup>3</sup>	3.1	—	4	—	—	word clk
R5	8b10b decoder	—	—	—	1	1	word clk
R6	Clock Tolerance Compensation	7	15	23	1	1	word clk
R7	FPGA Bridge – 1:1 gearing with different clocks	1	3	5	—	1	word clk
	FPGA Bridge – 1:1 gearing with same clocks	—	—	—	3	1	word clk
	FPGA Bridge – 2:1 gearing	1	3	5	—	—	word clk

1.  $\Delta 1 = -245\text{ps}$ ,  $\Delta 2 = +88\text{ps}$ ,  $\Delta 3 = +112\text{ps}$ .

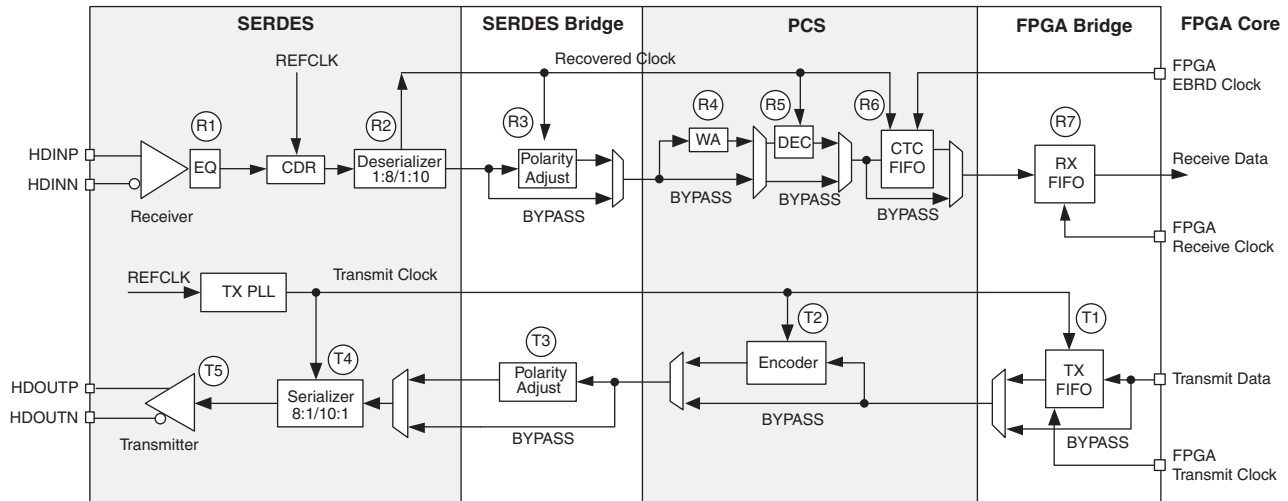
2.  $\Delta 1 = +118\text{ps}$ ,  $\Delta 2 = +132\text{ps}$ ,  $\Delta 3 = +700\text{ps}$ .

3. Table 8-24 shows word aligner latency depending on word alignment offset. The exact offset can be found in channel status register, CH\_22, bit [3:0].

**Table 8-24. Word Aligner Latency vs. Offset**

wa_offset[3:0] (CH_22[3:0])	Latency (Word Clock)
0	4.0
1	3.9
2	3.8
3	3.7
4	3.6
5	3.5
6	3.4
7	3.3
8	3.2
9	3.1

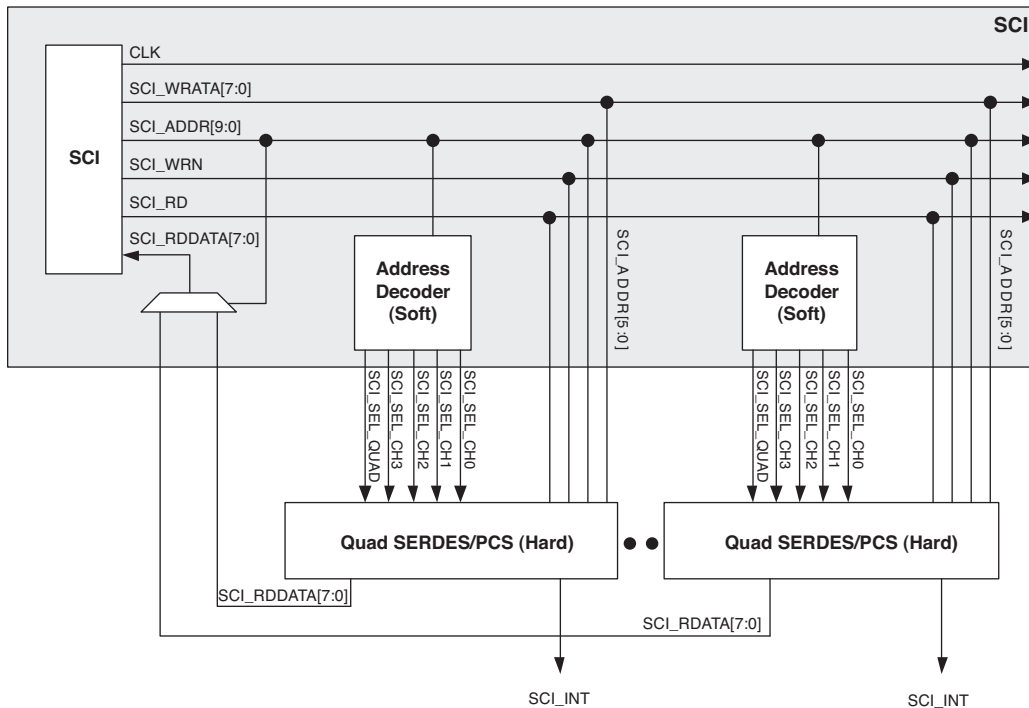
**Figure 8-42. Transmitter and Receiver Latency Block Diagram**



### SERDES Client Interface

The SCI allows the SERDES/PCS quad to be controlled by registers as opposed to the configuration memory cells. It is a simple register configuration interface. The block diagram of the SCI that resides in the FPGA core is shown in Figure 8-43.

**Figure 8-43. SCI Interface Block Diagram**



The interface logic that resides in the FPGA core should be developed by users per their interface scheme. Contact Lattice Technical Support for example code.

The SCI\_ADDR bus is six bits wide within the block. The bus width at the block boundary is 11 bits. The upper five bits are used for quad block selection and channel selection. Table 8-25 shows the SCI address map for the SERDES quad.

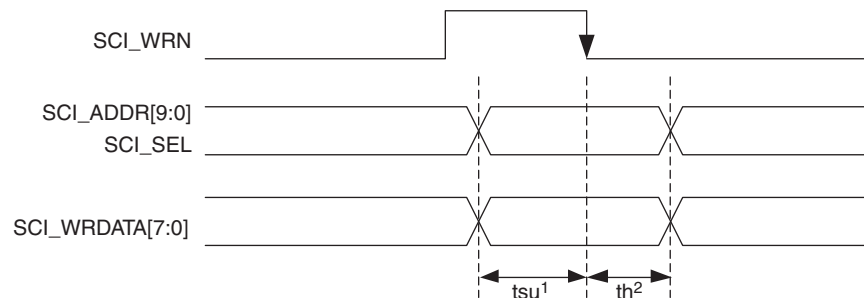
Refer to Appendix A and Appendix B for SERDES/PCS register address and bit descriptions.

**Table 8-25. SCI Address Map for Up to Four SERDES/PCS Quads**

Address Bits	Description
SCI_ADDR[5:0]	Register address bits 000000 = select register 0 000001 = select register 1 ... 111110 = select register 62 111111 = select register 63
SCI_ADDR[8:6]	Channel address bits 000 = select channel 0 001 = select channel 1 010 = select channel 2 011 = select channel 3 100 = select Quad 101 = Unused 110 = Unused 111 = Unused
SCI_ADDR[10:9]	Quad address bits 00 = select Quad A 01 = select Quad B 10 = select Quad C 11 = select Quad D

Read and write operations through this interface are asynchronous. In the WRITE cycle the write data and write address must be set up and held in relation to the falling edge of the SCI\_WR. In the READ cycle the timing has to be in relation with the SCI\_RD pulse. Figures 8-44 and 8-45 show the WRITE and READ cycles, respectively.

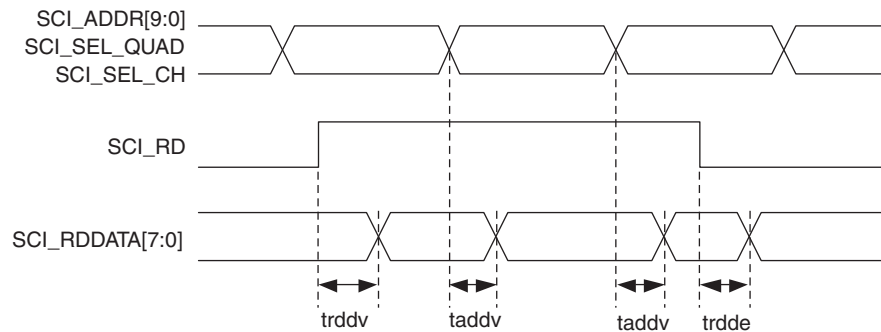
**Figure 8-44. SCI WRITE Cycle, Critical Timing**



1. tsu is the setup time for address and write data prior to the falling edge of the write strobe.
2. th is the hold time for address and write data after the falling edge of the write strobe.

Note: To avoid accidental writing to control registers, registers should be used at the SCI input ports to drive them low at power-up reset.

**Figure 8-45. SCI READ Cycle, Critical Timing**



**Table 8-26. Timing Parameters**

Parameter	Typical Value	Units
tsu, trddv, tadv	1.127	ns
th, trdde	0.805	ns

The SCI interface is as simple as memory read/write. Here is an example of the pseudo code:

Write:

- Cycle 1: Set sci\_addr[5:0], sciw\_data[7:0], sci\_sel = 1'b1
- Cycle 2: Set sci\_wrn from 0 ≥ 1
- Cycle 3: Set sci\_wrn from 1 ≥ 0, sci\_sel = 1'b0

Read:

- Cycle 1: Set sci\_addr[5:0], sci\_sel = 1'b1
- Cycle 2: Set sci\_rd from 0 ≥ 1
- Cycle 3: Obtain reading data from sci\_rddata[7:0]
- Cycle 4: Set sci\_rd from 1 ≥ 0

## Interrupts and Status

The status bit may be read via the SCI, which is a byte wide and thus reads the status of eight interrupt status signals at a time. The SCI\_INT signal goes high to indicate that an interrupt event has occurred. The user is then required to read the QIF status register that indicates whether the interrupt came from the quad or one of the channels. This register is not cleared on read. It is cleared when all interrupt sources from the quad or channel are cleared. Once the aggregated source of the interrupt is determined, the user can read the registers in the associated quad or channel to determine the source of the interrupt. Tables 8-27 and 8-28 list all the sources of interrupt.

**Table 8-27. Quad Interrupt Sources**

Quad SCI_INT Source	Description	Register Name
int_qd_out	Quad Interrupt. If there is an interrupt event anywhere in the quad this register bit will be active. This register bit is cleared when all interrupt events have been cleared.	PCS Quad Status Register QD_20
int_ch_out[0:3]	Channel Interrupt. If there is an interrupt event anywhere in the respective channel this register bit will be active. These register bits are cleared when all interrupt sources in the respective channel have been cleared.	PCS Quad Status Register QD_20
ls_sync_statusn_[0:3]_int ls_sync_status_[0:3]_int	Link Status Low (out of sync) channel interrupt Link Status High (in sync) channel interrupt	PCS Quad Interrupt Status Register QD_22
~PLOL, PLOL	Interrupt generated on ~PLOL and PLOL - PLL Loss of Lock	SERDES Quad Status Register QD_25

**Table 8-28. Channel Interrupt Sources**

Channel SCI_INT Source	Description	Register Name
fb_tx_fifo_error_int fb_rx_fifo_error_int cc_overnun_int cc_underrun_int	FPGA Bridge TX FIFO Error Interrupt FPGA Bridge RX FIFO Error Interrupt CTC FIFO Overrun and Underrun Interrupts	PCS Channel General Interrupt Status Register CH_23
pci_det_done_int rlos_lo_int ~rlos_lo_int rlol_int ~rlol_int	Interrupt generated for pci_det_done Interrupt generated for rlos_lo Interrupt generated for ~rlos_lo Interrupt generated for rlol Interrupt generated for ~rlol	SERDES Channel Interrupt Status Register CH_2A

### SERDES Client Interface Application Example

Lattice ORCAstra FPGA configuration software is a PC-based GUI that allows users to configure the operational mode of a Lattice FPGA by programming control bits in the FPGA registers.

SERDES/PCS status information is displayed on-screen in real time, and any configuration can be saved to control registers for additional testing. Use of the GUI does not interfere with the programming of the FPGA core portion. More information and downloadable files for ORCAstra can be found on the Lattice Semiconductor website at [www.latticesemi.com/products/designsoftware/orcastra.cfm](http://www.latticesemi.com/products/designsoftware/orcastra.cfm).

Users can get a complete LatticeECP3 ORCAstra interface design from IPexpress. In addition to the HDL file that contains the ORCAstra interface, a file called “chip.v” (for Verilog) that wraps the ORCAstra interface is also provided.

### Dynamic Configuration of the SERDES/PCS Quad

The SERDES/PCS quad can be controlled by registers that are accessed through the optional SERDES Client Interface.

When controlled by the configuration memory cells, it is a requirement that the SERDES/PCS quads must reach a functional state after configuration is complete, without further intervention from the user. This means that any special reset sequences that are required to initialize the SERDES/PCS quad must be handled automatically by the hardware. In other words, use of the SCI is optional. The SERDES/PCS quad does NOT assume that the soft IP is present in the FPGA core.

## SERDES Debug Capabilities

### PCS Loopback Modes

The LatticeECP3 family provides three loopback modes controlled by control signals at the PCS/FPGA interface for convenient testing of the external SERDES/board interface and the internal PCS/FPGA logic interface. Two loopback modes are provided to loop received data back onto the transmit data path. The loopback modes are useful



for checking the high-speed serial SERDES package pin connections as well as the embedded SERDES and/or PCS logic.

**RX-to-TX Serial Loopback Mode**

Loops serial receive data back onto the transmit buffer without passing through the CDR or de-serializer. Selecting the RX-to-TX Serial Loopback option in the IPexpress GUI will set both the LB\_CTL[1:0] to '10' and TDRV\_DAT\_SEL[1:0] register bits to '11' (see Tables 8-81 and 8-84).

**TX-to-RX Serial Loopback Mode**

This mode loops back serial transmit data back onto the receiver CDR block. Selecting the TX-to-RX Serial Loopback option in the IPexpress GUI will set LB\_CTL[1:0] to '01' (see Table 8-81).

**SERDES Parallel Loopback Mode**

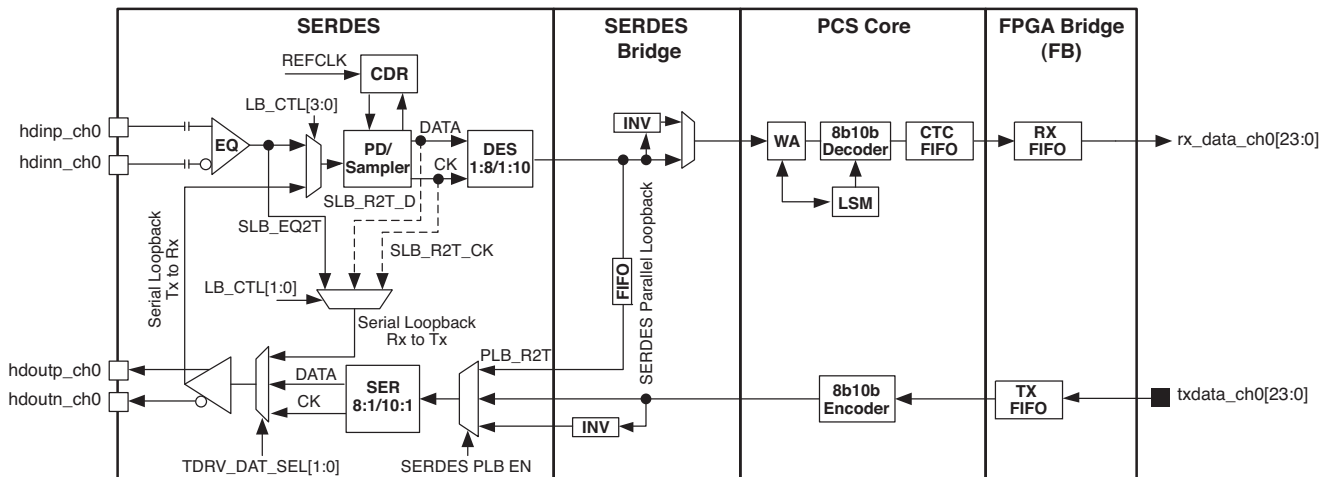
Loops parallel receive data back onto the transmit data path without passing through the PCS logic. When disabled in the IPexpress GUI, the parallel loopback mode can be dynamically controlled from the FPGA core control signals sb\_felb\_ch[3:0]\_c and sb\_pfifo\_lp[3:0]\_c.

If the dynamic feature of this loopback mode is not used, the two control signals should be tied to ground.

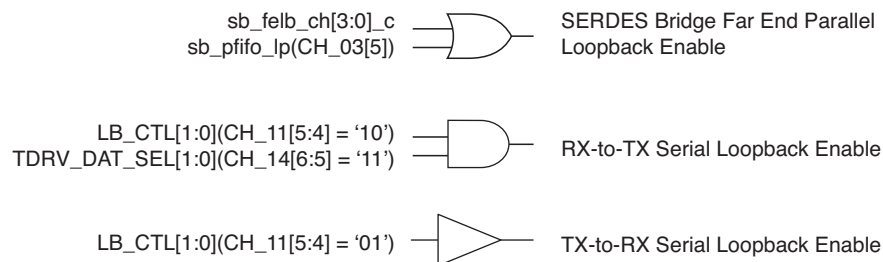
When enabled in the loopback mode in the IPexpress GUI, the control register bit sb\_pfifo\_lp(CH\_03[5]) is set and the loopback mode is set. The control signals from the FPGA core, sb\_felb\_ch[3:0]\_c and sb\_felb\_rst\_ch[3:0]\_c, are not available in the PCS module.

Refer to the Figure 8-46 for the discussion above.

**Figure 8-46. Three Loopback Modes**



**Figure 8-47. Loopback Enable Signals**



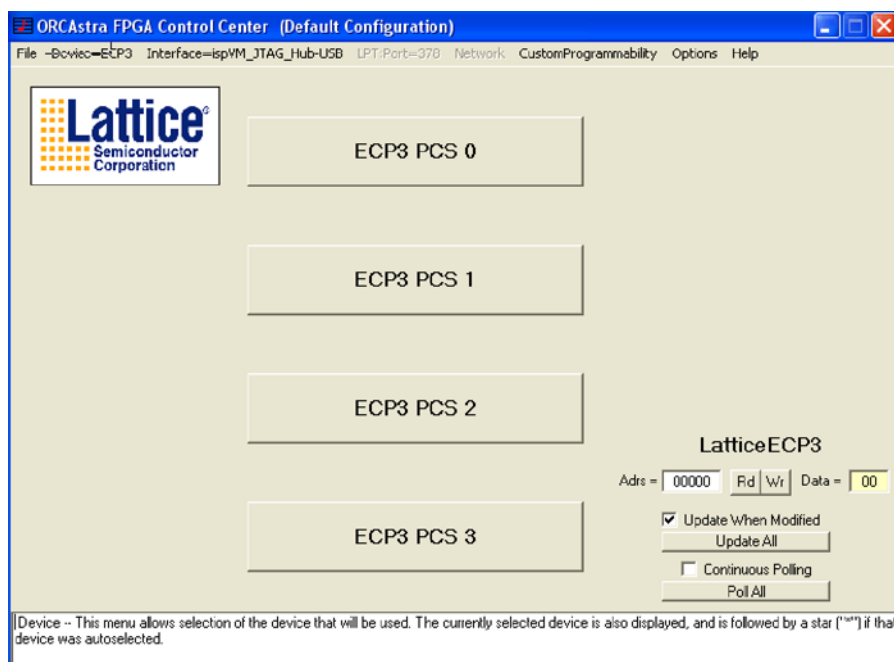
## ORCAstra

Lattice ORCAstra software helps you quickly explore configuration options without going through a lengthy re-compile process or making changes to your board. Configurations created in the GUI can be saved to memory and re-loaded for later use. To use ORCAstra, the ORCASTR module from IPexpress must be created and used in the FPGA design.

A macro capability is also available to support script-based configuration and testing. The GUI can also be used to display system status information in real time. Use of the ORCAstra software does not interfere with the programming of the FPGA.

Figure 8-48 shows the ORCAstra GUI top-level window. Users can read and write in this window without going through the subwindows for each PCS channel by read and write data at the address cell. When invoked, ORCAstra will automatically recognize the device type. Or, device types can be selected under the device pull-down menu.

**Figure 8-48. ORCAstra Top-Level Screen Shot**



By default, the data box shown in Figure 8-48 follows Big Endian byte order (i.e., the most significant bit is placed on the left). Users can change to Little Endian order by selecting **Display Data Reversed in Data Box** under the **Options** tab.

Click on the tab **Interface=None** and select **1 ispVM JTAG Hub USB Interface** from the drop-down list.

Then select the **C2 0A 80 80** from the **Select Target JTAG Device** window.

Figure 8-49. JTAG Device Selection



Then click **OK** in the ORCAstra Hub I/O window.

Figure 8-50. Hub ID Selection

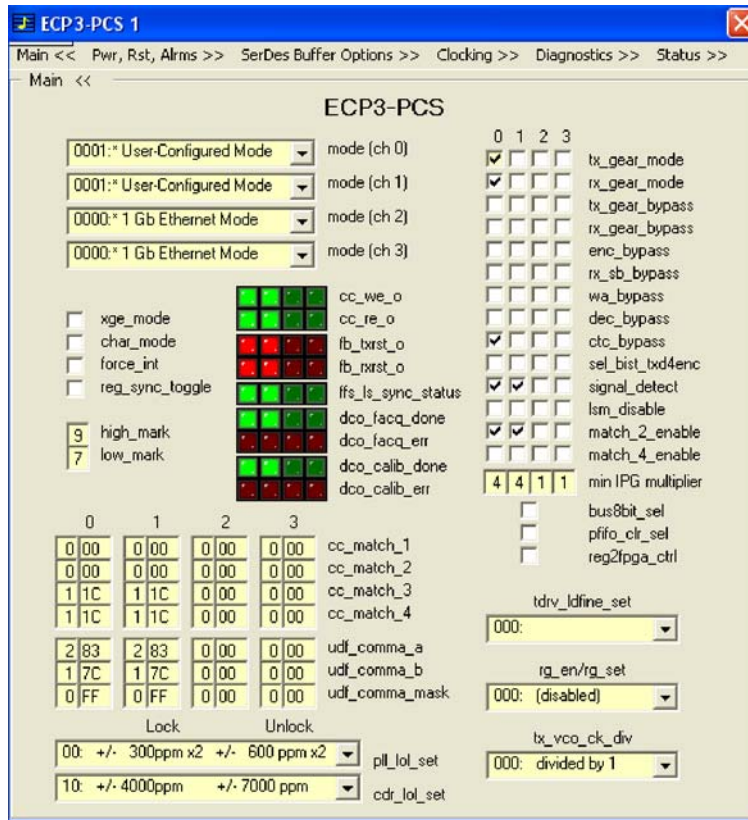


The figure shows there are activities on channel 0 and channel 1. In this example, we assume that the PCS SCI address is mapped to Quad 0 in the design.

Double-clicking on the **PCS0** (Quad 0) button will open the main window as shown in Figure 8-51.

These standard Windows menus control the selection of the device and interface. They also support various configuration options, including setting up and saving configurations with stored files.

**Figure 8-51. ORCAstra Main Window**

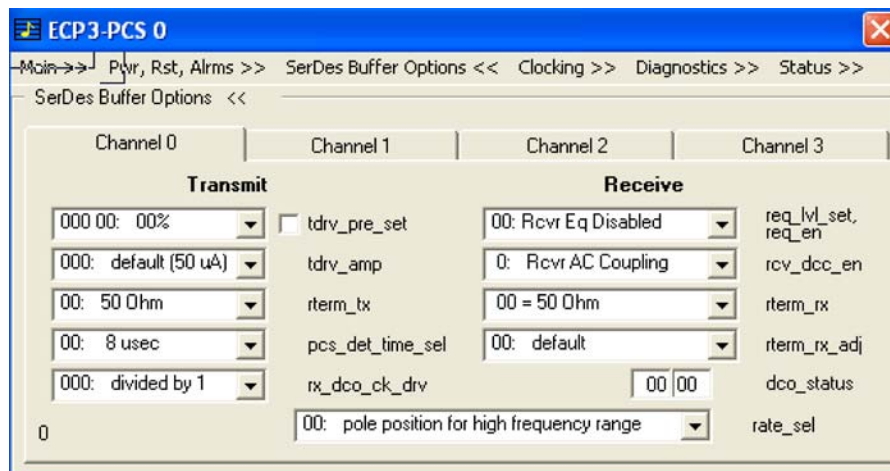


**Control Boxes and Buttons, Status Boxes and the Text Window**

Moving the cursor over a control box and clicking the left mouse button sets the control bits. Both the bit location and function for the selected box are displayed in the text window and will match those of the register map tables in the [LatticeECP3 Family Data Sheet](#). Only the function is displayed when the cursor is over the bit name. Status boxes are similar to control boxes but have an LED appearance and a colored background.

Figure 8-52 shows the SERDES Buffer Options window. Configuration options can be selected from the pull-down menu.

**Figure 8-52. SERDES Buffer Options Window**



More information and downloadable files for ORCAstra can be found on the Lattice Semiconductor website at the following address: [www.latticesemi.com/products/designsoftware/orcastra.cfm](http://www.latticesemi.com/products/designsoftware/orcastra.cfm).

## Other Design Considerations

### Simulation of the SERDES/PCS

**Table 8-29. Simulation Model Locations**

Simulator	Model Location
Active-HDL	ispTOOLS\cae_library\simulation\blackbox\pcsc-aldec.zip
ModelSim	ispTOOLS\cae_library\simulation\blackbox\pcsd-mti_6.0-V1-1.zip
NC-Verilog	ispTOOLS\cae_library\simulation\blackbox\pcsd-ncv.zip
VCS	ispTOOLS\cae_library\simulation\blackbox\PCSD_sim.vp.zip

### 16/20-Bit Word Alignment

The PCS receiver cannot recognize the 16-bit word boundary. When Word Aligner is enabled, the PCS can only do BYTE alignment. The 16-bit word alignment should be done in the FPGA fabric and is fairly straight forward. The simulation model works in the same way. It can be enhanced if users implement an alignment scheme as described below.

For example, if transmit data at the FPGA interface are:

YZABCDEFGHIJKLM... (each letter is a byte, 8-bit or 10-bit)

Then the incoming data in PCS after 8b10b decoder and before rx\_gearbox are:

YZABCDEFGHIJKLM...

After rx\_gearbox, they can become:

1. {ZY} {BA} {DC} {FE} {HG} {JI} {LK} ....

or

2. {AZ} {CB} {ED} {GF} {IH} {KJ} {ML} ...

Clearly, sequence 2 is not aligned. It has one byte offset, but 16/20-bit alignment is needed. Let's say the special character 'A' should be always placed in the lower byte.

Flopping one 20-bit data combines with the current 16/20-bit data to form 32/40-bit data as shown below:

1. {DCBA} {HGFE} {LKJI} ...

^  
| \*\*Found the A in lower 10-bit, set the offset to '0`, send out aligned data 'BA`

Next clock cycle:

{FEDC} {JIHG} {NMLK} ...

^  
| \*\*send out aligned data 'DC'

etc.

After the 16/20-bit alignment, the output data are:

{ZY} {BA} {DC} {FE} {HG} {JI} {LK} ...

2. {CBAZ} {GFED} {KJIH} ....

^  
| \*\*Found the A in upper 10-bit, set the offset to '10', send out aligned data 'BA'

Next clock cycle:

{EDCB} {IHGF} {MLKJ} ...  
^  
| \*\*send out aligned data 'DC'

etc.

After the 20-bit alignment, the output data are:

{ZY} {BA} {DC} {FE} {HG} {JI} {LK} ...

*Note: The LSB of a 8/10-bit byte or a 16/20-bit word is always transmitted first and received first.*

For sample 16/20-bit word alignment code, send your request to [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com).

### Unused Quad/Channel and Power Supply

On unused quads and channels, VCCA should be powered up. VCCIB, VCCOB, HDINP/N, HDOUTP/N and REF-CLKP/N should be left floating. Unused channel outputs are tristated, with approximately 10 KOhm internal resistor connecting between the differential output pair. During configuration, HDOUTP/N are pulled high to VCCOB.

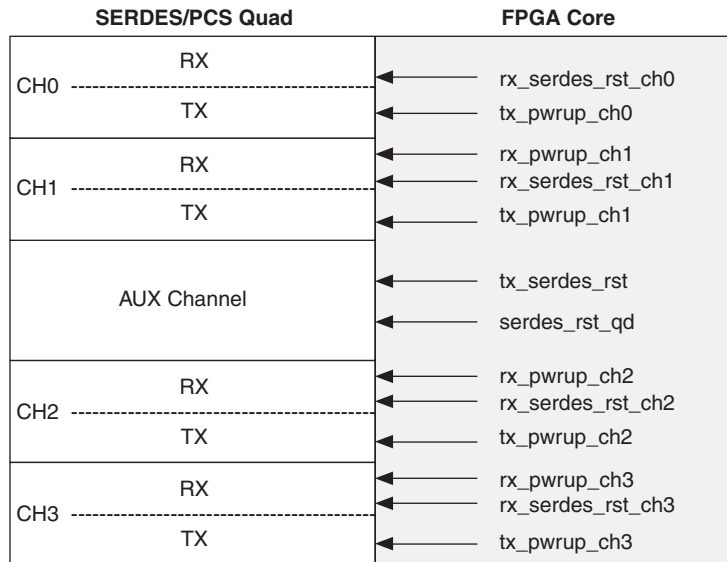
Even when a channel is used as Rx Only mode or Tx Only mode, both the VCCOB and VCCIB of the channel must be powered up. Unused SERDES is configured in power down mode by default.

### Reset and Power-Down Control

The SERDES quad has reset and power-down controls for the entire macro and also for each transmitter and receiver as shown in Figure 8-53. The reset signals are active high and the power-down is achieved by driving the pwrup signals low. The operation of the various reset and power-down controls are described in the following sections.

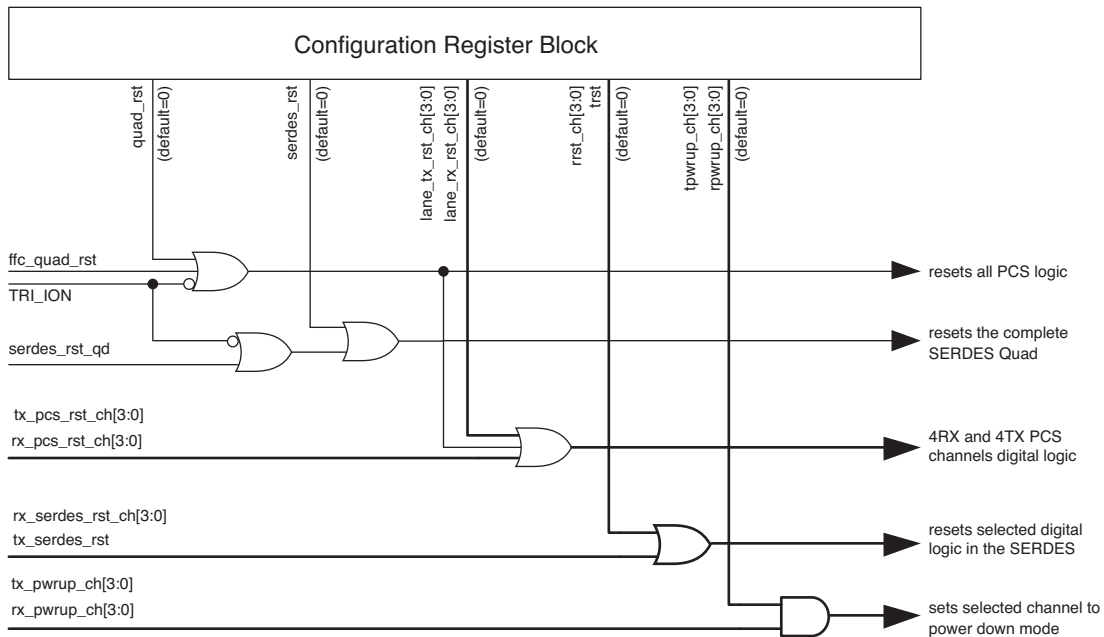
*Note: When the device is powering up and the chip level power-on-reset is active, the SERDES control bits (in the PCS) will be cleared (or will take on their default value). This will put the SERDES quad into the power-down state.*

**Figure 8-53. SERDES/PCS Quad Reset and Power-Down Controls**



Typically, all resets are via power-on reset and various FPGA fabric resets. The reset logic is shown in Figure 8-54 and Table 8-30.

**Figure 8-54. SERDES/PCS Reset Diagram**



**Table 8-30. SERDES/PCS Reset Table**

Reset Signals		PCS <sup>1</sup> TX	PCS <sup>1</sup> RX	SERDES TX	SERDES RX	PCS CTRL Registers	TX PLL	CDR PLL
FPGA	Control Register							
tx_pcs_rst_ch[3:0]_c	lane_tx_rst[3:0]	X						
rx_pcs_rst_ch[3:0]_c	lane_rx_rst[3:0]		X					
rst_qd_c	quad_rst	X	X	X	X		X	X
serdes_rst_qd_c	serdes_rst			X	X		X	X
rx_serdes_rst_ch[3:0]_c	rrst[3:0] <sup>2</sup>				X			X
tx_serdes_rst_c	trst						X <sup>3</sup>	
TRI_ION (configuration)		X	X	X	X	X		

1. Includes SB (SERDES Bridge), PCS core and FB (FPGA Bridge) sub-blocks.
2. For internal use only. This reset should always be tied to '0' unless there is a need to reset the CDR PLL.
3. tx\_serdes\_rst\_c reset does not reset the TX PLL. It only forces tx\_pll\_lol\_qd\_s to go high.

**Table 8-31. Reset Controls Description<sup>1, 2, 3</sup>**

Rest Signal		Description
FPGA	Control Register	
rst_qd_c	quad_rst	Active-high, asynchronous input. Resets all SERDES channels including the auxiliary channel and PCS. This reset includes serdes_rst, txpll, cdr, lane_tx_rst, and lane_rx_rst.
serdes_rst_qd_c	serdes_rst	Active-high, asynchronous input to the SERDES quad. Gated with software register bit. This reset is for the SERDES block only and TXPLL and CDRPLL are included.
tx/rx_pcs_rst_ch[3:0]_c	lane_tx/rx_rst[0:3]	Active-high, asynchronous input. Resets individual TX/RX channel in SB, PCS core and FB blocks.
rx_serdes_rst_ch[3:0]_c	rrst[0:3]	Resets loss-of-lock (rlo), loss-of-signal and calibration circuits.
tx_serdes_rst_c	trst	Resets the loss-of-lock of AUX PLL (plo).

1. For all channels in the quad running in full-data-rate mode, parallel side clocks are guaranteed to be in-phase.
2. For all channels in the quad running in half-data-rate mode, each channel has a separate divide-by-two circuit. Since there is no mechanism in the quad to guarantee that these divide-by-two circuits are in phase after de-assertion of "serdes\_rst", the PCS design should assume that the dividers (and therefore the parallel side clocks) are NOT in phase.
3. In half-data-rate mode, since there is no guarantee that the parallel side clocks are in phase, this may add channel-to-channel skew to both transmit and receive sides of a multi-channel link.

**Table 8-32. Reset Pulse Specification**

Parameter	Description	Min.	Typ.	Max.	Units
t <sub>SERDES_RST_QD</sub>	Quad SERDES Reset high time	1			us
t <sub>RX_PCS_RST</sub>	Channel RX PCS Reset high time	3			ns
t <sub>TX_PCS_RST</sub>	Channel TX PCS Reset high time	3			ns
t <sub>RX_SERDES_RST</sub>	Channel RX SERDES reset high time	3			ns
t <sub>TX_SERDES_RST</sub>	Quad TX SERDES reset high time	3			ns

### Power-Down Control Description

Each RX and TX channel can be individually powered-down by a software register bit or a control signal from the FPGA. The individual channel power-down control bits will only power-down selected blocks within the SERDES macro and the high-speed I/O buffers.



**Table 8-33. Power-Down Control Description**

Signal		Description
FPGA	Register	
serdes_pd		Active-low asynchronous input to the SERDES quad, acts on all channels including the auxiliary channel. When driven low, it powers down the whole macro including the transmit PLL. All clocks are stopped and the macro power dissipation is minimized. After release, both the TX and RX reset sequences should be followed.
tx_pwrup_ch[0:3]_c	tpwrup[0:3]	Active-high transmit channel power-up – Powers up the serializer and output driver. After release, the TX reset sequence should be followed.
rx_pwrup_ch[0:3]_c	rpwrup[0:3]	Active-high receive channel power-up – Powers up CDR, input buffer (equalizer and amplifier) and loss-of-signal detector. After release, the RX reset sequence should be followed.

**Table 8-34. Power-Down/Power-Up Timing Specification**

Parameter	Description	Min.	Typ.	Max.	Units
t <sub>PWRDN</sub>	Power-down time after serdes_pd	20			ns
t <sub>PWRUP</sub>	Power-up time after serdes_pd	20			ns

## SERDES/PCS RESET

### Reset Sequence and Reset State Diagram

After power-up and configuration, all SERDES resets and FPGA resets are applied.

### Reset Sequence Generation

Reset Sequence is included in the IPExpress GUI (available in Diamond 1.1 and later versions).

We recommend to select the reset sequence generation option in IPExpress as described in the Control Setup tab section. The HDL file generated for the SERDES/PCS will include the Tx Reset State Machine and Rx Reset State Machine.

### Lock Status Signals Definitions

tx_pll_lo_lol_qd_s:	: 1 = TX PLL loss of lock : 0 = TX PLL lock It takes 1,400,000 UI to declare the lock of TX PLL
rx_cdr_lo_lol_ch[3:0]_s	: 1 = CDR loss of lock : 0 = Lock maintained It takes 400,000 reference clock cycles (worst case) to declare the lock of CDR PLL
rx_los_low_ch[3:0]_s	: 1 = Loss of signal detection for each channel : 0 = Signal detected

The rx\_cdr\_lo\_lol\_ch[3:0]\_s status signal is an indicator of the CDR lock status as defined above. However, during the CDR locking process the CDR PLL will lock to a reference clock when there is no input data present. This avoids ignoring the input data when it is restored.

In order to ensure the presence of input data during CDR lock status checking, it is recommended to use the rx\_los\_low\_ch[3:0]\_s signal in conjunction with the rx\_cdr\_lo\_lol\_ch[3:0]\_s signal.

### TX Reset Sequence

1. QUAD\_RESET: At power up, assert rst\_qd\_c and tx\_pcs\_rst\_ch[3:0]\_c.
2. WAIT\_FOR\_TIMER1: Start TIMER1. Wait for a minimum 20 ns.
3. CHECK\_PLOL: Release rst\_qd\_c.
4. WAIT\_FOR\_TIMER2: Start TIMER2. If TIMER2 expires and the TX PLL is not locked, go to step 1.
5. NORMAL: Release tx\_pcs\_rst\_ch#\_c. If tx\_pll\_lo\_lol\_qd\_s goes high during normal operation, go to step 1.

### RX Reset Sequence

1. WAIT\_FOR\_PLOL: Wait until the TX PLL locks and receive data is present. rx\_serdes\_rst\_ch[3:0]\_c is set to 0 because rx\_los\_low[3:0]\_s goes high when it is asserted.
2. RX\_SERDES\_RESET: Assert rx\_serdes\_rst\_ch[3:0]\_c and rx\_pcs\_rst\_ch[3:0]\_c.
3. WAIT\_FOR\_TIMER1: Wait for a minimum of 3 ns.
4. CHECK\_LOL\_LOS: Release rx\_serdes\_rst\_ch\_c. Reset TIMER2.

5. WAIT\_FOR\_TIMER2: Wait for both `cdr_lo_lol_ch[3:0]_s` and `rx_los_low_ch[3:0]` to go low. If there is a transition in `rx_lo_lol_los` (`rx_cdr_lo_lol_ch_s` || `rx_los_low_ch_s`), go to step 4. If TIMER2 expires with `rx_lo_lol_los = 1`, go to step 1.

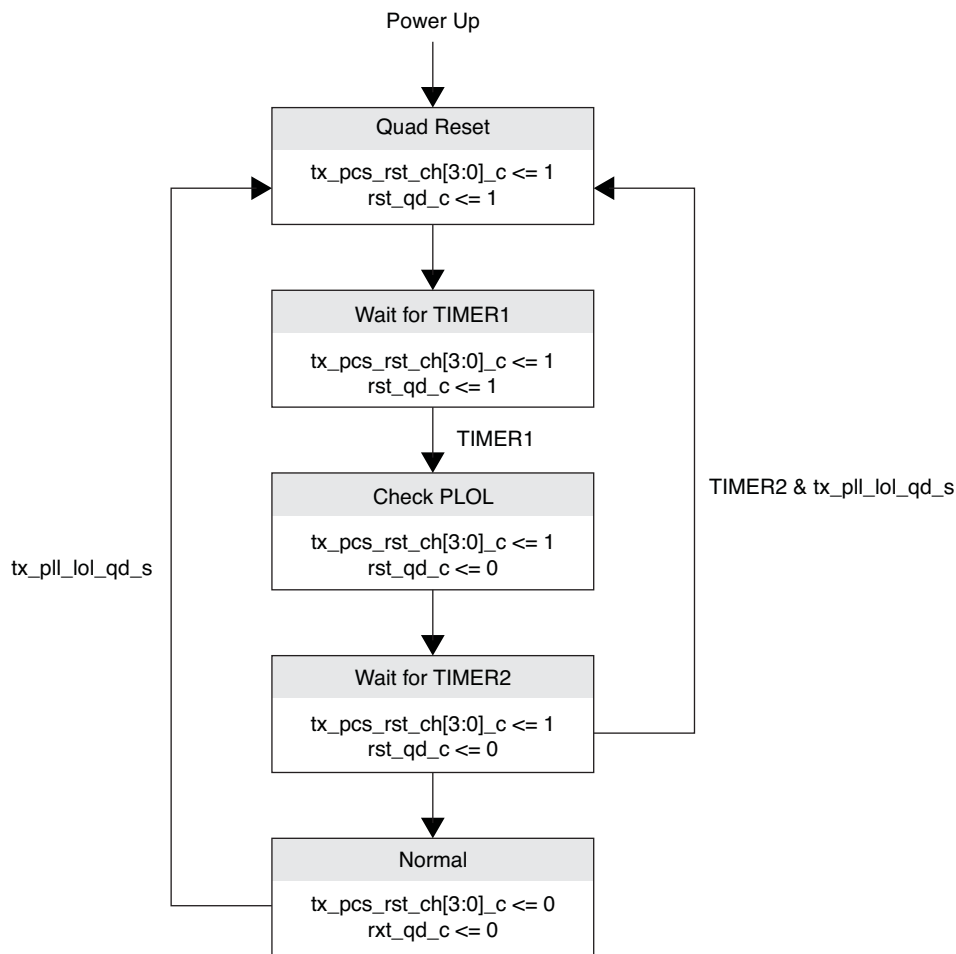
6. NORMAL: Release `rx_pcs_rst_ch_c`. If `rx_lo_lol_los` goes high, go to step 1.

*Note: The RX reset sequence provides the CDR re-locking feature when the input data source is interrupted during normal operation. The RX reset can be applied by channel base.*

The reset sequence state diagrams are described in Figures 8-55 and 8-56.

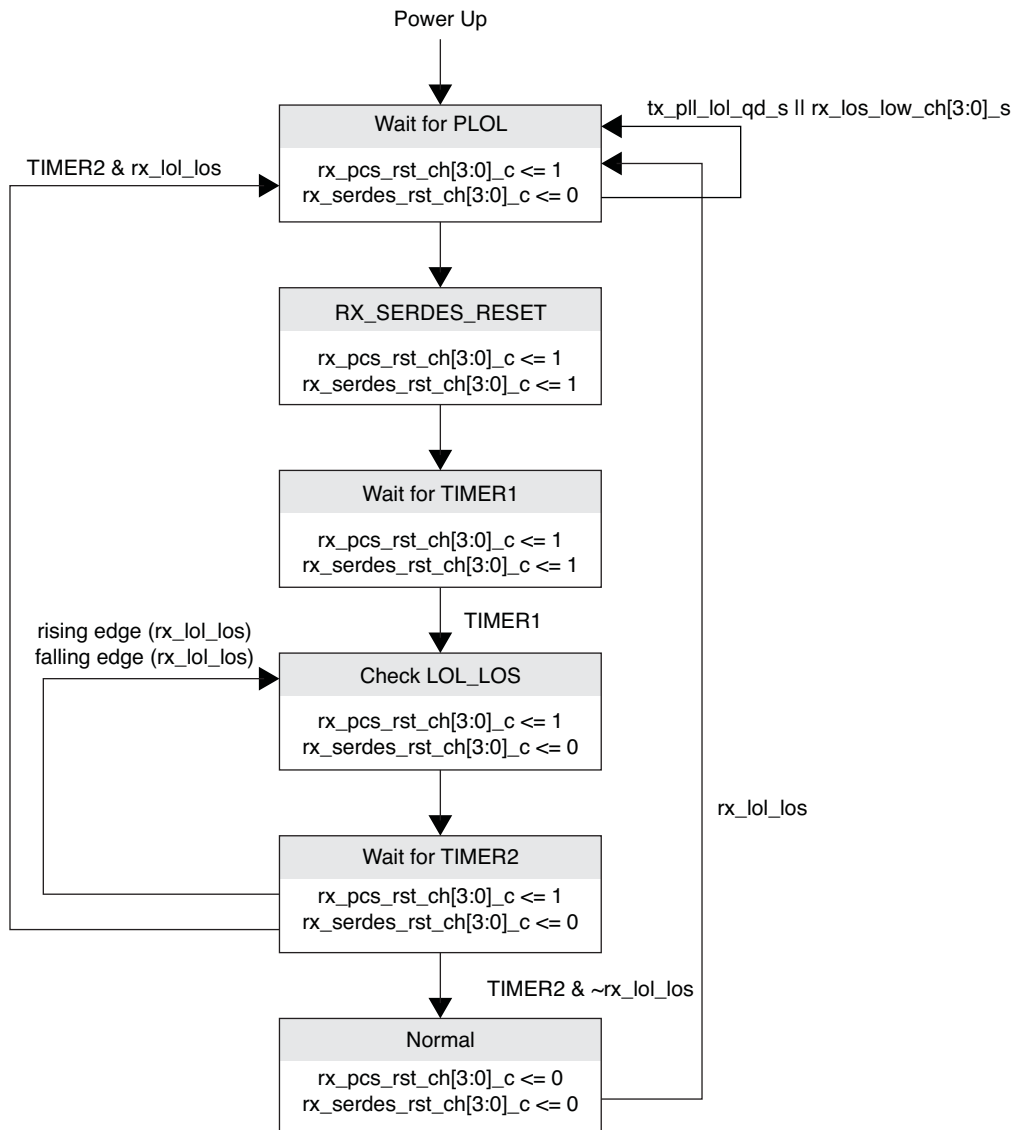
A set of [sample reset sequence code](#) is available on the Lattice web site.

**Figure 8-55. TX Reset State Diagram**



Notes:  
 TIMER 1: `rst_qd_c` asserted for a minimum of 20 ns.  
 TIMER 2: Time to declare TX PLL lock: 1,400,000 UI.

**Figure 8-56. RX Reset State Diagram**



**Notes:**

TIMER 1: rx\_serdes\_rst\_ch[3:0]\_c asserted for minimum 3 ns.

TIMER 2: Time for rx\_lol\_los signal to stay low (400,000 reference clock cycles). Any FPGA clock can be used to satisfy the timer requirement.

In the diagram above, rx\_lol\_los is defined as rx\_cdr\_lol\_ch[3:0]\_s || rx\_los\_low\_ch[3:0]\_s.

The tx\_pll\_lol\_qd\_s input to the state diagram RTL code should be tied low in Rx Only mode or when the recovered clock is used as the Tx PLL reference clock, as in SDI applications.

When multiple receiver channels rx\_serdes\_rst\_ch[3:0]\_c are to be asserted, it is recommended to activate the reset signals one channel at a time. Simultaneous resetting of multiple receiver SERDES channels may cause a current surge in the SERDES/PCS quad.

The rx\_los\_low output from SERDES may be triggered for some input streams with continuous zeros, like the SDI pathological pattern. For such applications, the rx\_los\_low input to the reset state machine must be connected to the carrier detect output (must be inverted) of the cable equalizer if available. In general, CD=1 means carrier is present. So this signal must be inverted to replace rx\_los\_low. If a cable equalizer is not available, users may tie it to zero but in this case, the CDR can lock to a local reference clock when there is no input data present.

### Power Supply Sequencing Requirements

When using the SERDES with 1.5V VCCIB or VCCOB, the SERDES should not be left in a steady state condition with the 1.5V power applied and the 1.2V power not applied. Both the 1.2V and the 1.5V power should be applied to the SERDES at nominally the same time. The normal variation in ramp\_up times of power supplies and voltage regulators is not a concern.

### References

- TN1033, [High-Speed PCB Design Considerations](#)
- TN1114, [Electrical Recommendations for Lattice SERDES](#)
- HB1009, [LatticeECP3 Family Handbook](#)
- DS1021, [LatticeECP3 Family Data Sheet](#)

### Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

### Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
March 2009	01.1	Updated Data Bus Usage for Each Mode table.
		Updated Reference Clock Usage Block Diagram.
June 2009	01.2	tdrv_amp attribute setting changed to tdrv attribute.
		Added External Link State Machine Option section.
		Added Idle Insert for Gigabit Ethernet Mode section.
		GUI screen shots updated.
		Word alignment latency updated.
November 2009	01.3	Updated Reset Sequence State Diagrams are provided, tx and rx side separately.
		los settings are updated.
		Added SONET support.
		Removed Far End Parallel Loopback feature.
		Data rate ranges are re-defined.
		Reduced minimum data rate to 150 Mbps.
February 2010	01.4	Reset sequence state diagram updated.
March 2010	01.5	Added Generation Options tab in IPexpress GUI.
June 2010	01.6	Updated for Lattice Diamond design software support.
		refclk_to/from_nq signals removed from clock diagram.
December 2010	01.7	Added Reset Sequence Generation option in the IPexpress GUI.
		Added detailed information about comma mask.
July 2011	01.8	Standards Supported by SERDES table – updated information for Gigabit Ethernet, SGMII, 10-Bit SERDES, 8-Bit SERDES, Generic 8b10b.
		Transmit Data Bus section updated.
		Receive Data Bus section updated.
		Updated footnotes for Loss of Signal Detector figure.

**Revision History (Continued)**

Date	Version	Change Summary
July 2011(cont.)	01.8 (cont.)	SERDES_PCS GUI Attributes - Quad Tab Setup table – Added SGMII to Protocol Range.
		SERDES/PCS Latency Breakdown table – updated gearing information.
		Simulation Model Locations table – Added VCS row.
		Updated SERDES/PCS Reset Diagram.
		Updated footnotes of the Rx Reset State Diagram.
		Updated information for bit 5 in the SERDES Control Register QD_0A table.
		Updated information for bits 3 and 2:0 in the SERDES Control Register CH_13 table.
		Added footnote to PCS Status Register CH_21 table.
Attribute Cross Reference Table – Updated attribute values for TX_DATARATE_RANGE and CHn_RX_DATARATE_RANGE. Updated first footnote.		
September 2011	01.9	SERDES Control Register CH_14, corrected default value of tpwrup register bit.
		Attribute Cross-Reference Table, removed all int_all reserved bits and los hi bits.
		PCS Control Register QD_02, updated default values.
November 2011	02.0	SERDES_PCS I/O Descriptions table – Updated descriptions for tx_full_clk_ch[3:0] and tx_half_clk_ch[3:0].
		Updated Tx Lane-to-Lane Skew text section.
		Updated word alignment control bullet in the Word Alignment (Byte Boundary Detect) section.
		Updated External Link State Machine Option text section.
		Power-Down Control Description table – Updated description for tx_pwrup_ch[0:3]_c signal.
		Appendix A, SERDES Control Register QD_0B table – Updated description for Bit 7.
		Appendix A, PCS Control Register CH_01 table – Updated description for Bit 7.
Appendix C, Attribute Cross-Reference table – Updated attribute value for {CHn_RXWA, CHn_ILSM}.		
February 2012	02.1	Updated document with new corporate logo.
April 2012	02.2	Updated Number of SERDES/PCS Quads per LatticeECP3 Device table for 328-ball csBGA.
		SERDES_PCS I/O Descriptions table – Updated description for refclk2fpga.
		SERDES_PCS GUI Attributes – SERDES Advanced Setup Tab table – Updated footnote 3.
		Added new Reference Clock to FPGA Core and Reset Sequence text section.
		Updated Generic 8b10b Mode text section.
		Updated FPGA Interface Clocks text section.
		Appendix A, Channel Interface Registers Map table – Updated information for CH_16, CH_17 and CH_23.
Appendix A, SERDES Control Register CH_11 table – Changed Bit 3:2 to “Reserved”.		

**Revision History (Continued)**

Date	Version	Change Summary
May 2012	02.3	Clarified data pattern requirements for LSM synchronization in generic 8b10b mode.
August 2012	02.4	Added LatticeECP3-17EA 328 csBGA limited channels availability.
		SERDES_PCS I/O Descriptions table – Added footnote 3.
		RX Reset State Diagram footnote updated.
June 2013	02.5	Added note on possible lsm_status_ch(0:3)_s glitches.
		Updated note on the SERDES/PCS GUI – PCS Advanced2 Setup Tab table.
		Updated the Protocol-Specific SERDES Setup Options table.
		Updated the SERDES Control Register CH_10 table.
		Added information on the SERDES Equalizer and Pre-Emphasis settings.
		Updated Technical Support Assistance information.

## Appendix A. Configuration Registers

### Quad Registers Overview

Table 8-35. Quad Interface Registers Map

BA	Register Name	D7	D6	D5	D4	D3	D2	D1	D0
<b>Per Quad PCS Control Registers</b>									
00	QD_00	reg_sync_toggle	force_int	char_mode	xge_mode				
01	QD_01	internal use only							
02	QD_02	high_mark[3]	high_mark[2]	high_mark[1]	high_mark[0]	low_mark[3]	low_mark[2]	low_mark[1]	low_mark[0]
03	QD_03						pfifo_clr_sel	internal use only	internal use only
04	QD_04	internal use only							
05	QD_05	internal use only							
06	QD_06	internal use only							
07	QD_07	internal use only							
08	QD_08	internal use only							
09	QD_09	ls_sync_status_3_int_ctl	ls_sync_status_2_int_ctl	ls_sync_status_1_int_ctl	ls_sync_status_0_int_ctl	ls_sync_statusn_3_int_ctl	ls_sync_statusn_2_int_ctl	ls_sync_statusn_1_int_ctl	ls_sync_statusn_0_int_ctl
<b>Per Quad SERDES Control Registers</b>									
0A	QD_0A	internal use only	reserved	tx_refck_sel	refck_dcc_en	refck_rterm		refck_out_sel[1]	refck_out_sel[0]
0B	QD_0B	refck25x	bus8bit_sel	reserved	reserved	reserved	reserved	refck_mode[1]	refck_mode[0]
0C	QD_0C	reserved	reserved	reserved	reserved	reserved	reserved	cdr_loi_sel[1]	cdr_loi_sel[0]
0D	QD_0D	internal use only	internal use only	internal use only	pll_loi_sel[1]	pll_loi_sel[0]	tx_vco_ck_div[2]	tx_vco_ck_div[1]	tx_vco_ck_div[0]
0E	QD_0E	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only
0F	QD_0F	plol_int_ctl	-plol_int_ctl	reserved	reserved	reserved	reserved	reserved	reserved
<b>Per Quad Clock Reset Registers</b>									
10	QD_10	reserved	reserved	reserved	reserved	serdes_pd	serdes_rst	quad_rst	trst
11	QD_11	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
<b>Per Quad PCS Status Registers</b>									
20	QD_20				int_qd_out	int_ch[3]	int_ch[2]	int_ch[1]	int_ch[0]
21	QD_21	ls_sync_status_3	ls_sync_status_2	ls_sync_status_1	ls_sync_status_0	ls_sync_statusn_3	ls_sync_statusn_2	ls_sync_statusn_1	ls_sync_statusn_0
22	QD_22	ls_sync_status_3_int	ls_sync_status_2_int	ls_sync_status_1_int	ls_sync_status_0_int	ls_sync_statusn_3_int	ls_sync_statusn_2_int	ls_sync_statusn_1_int	ls_sync_statusn_0_int
23	QD_23	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only
24	QD_24	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only
<b>Per Quad SERDES Status Registers</b>									
25	QD_25	plol	-plol	reserved	reserved	reserved	reserved	reserved	reserved
26	QD_26	plol_int	-plol_int	reserved	reserved	reserved	reserved	reserved	reserved
27	QD_27	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
28	QD_28	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved

### Per Quad PCS Control Registers Details

Table 8-36. PCS Control Register QD\_00

Bit	Name	Description	Type	Default
7	reg_sync_toggle	Transition = Reset the four TX serializers to minimize TX lane-to-lane skew Level = Normal operation of TX serializers	RW	0
6	force_int	1 = Force to generate interrupt signal 0 = Normal operation	RW	0
5	char_mode	1 = Enable SERDES characterization mode 0 = Disable SERDES characterization mode	RW	0
4	xge_mode	1 = Selects 10Gb Ethernet 0 = Depends on Channel Mode Selection	RW	0
3:0	Reserved			

Table 8-37. PCS Control Register QD\_01

Bit	Name	Description	Type	Default
7:0	Internal use only			



**Table 8-38. PCS Control Register QD\_02**

Bit	Name	Description	Type	Default
7:4	high_mark[3:0]	Clock compensation FIFO high water mark. Mean is 4'b1000.	RW	4'b1001
3:0	low_mark[3:0]	Clock compensation FIFO low water mark. Mean is 4'b1000.	RW	4'b0111

**Table 8-39. PCS Control Register QD\_03**

Bit	Name	Description	Type	Default
7:3	Reserved			
2	pfifo_clr_sel	1 = pfifo_clr signal or channel register bit clears the FIFO 0 = pfifo_error internal signal self clears the FIFO	RW	0
1	Internal use only			
0	Internal use only			

**Table 8-40. PCS Control Register QD\_04**

Bit	Name	Description	Type	Default
7:0	Internal use only			

**Table 8-41. PCS Control Register QD\_05**

Bit	Name	Description	Type	Default
7:0	Internal use only			

**Table 8-42. PCS Control Register QD\_06**

Bit	Name	Description	Type	Default
7:0	Internal use only			

**Table 8-43. PCS Control Register QD\_07**

Bit	Name	Description	Type	Default
7:0	Internal use only			

**Table 8-44. PCS Control Register QD\_08**

Bit	Name	Description	Type	Default
7:0	Internal use only			

**Table 8-45. PCS Control Register QD\_09**

Bit	Name	Description	Type	Default
7	ls_sync_status_3_int_ctl	1 = Enable interrupt for ls_sync_status_3 (in sync) 0 = Disable interrupt for ls_sync_status_3 (in sync)	RW	0
6	ls_sync_status_2_int_ctl	1 = Enable interrupt for ls_sync_status_2 (in sync) 0 = Disable interrupt for ls_sync_status_2 (in sync)	RW	0
5	ls_sync_status_1_int_ctl	1 = Enable interrupt for ls_sync_status_1 (in sync) 0 = Disable interrupt for ls_sync_status_1 (in sync)	RW	0
4	ls_sync_status_0_int_ctl	1 = Enable interrupt for ls_sync_status_0 (in sync) 0 = Disable interrupt for ls_sync_status_0 (in sync)	RW	0
3	ls_sync_statusn_3_int_ctl	1 = Enable interrupt for ls_sync_status_3 when it goes low (out of sync) 0 = Disable interrupt for ls_sync_status_3 when it goes low (out of sync)	RW	0
2	ls_sync_statusn_2_int_ctl	1 = Enable interrupt for ls_sync_status_2 when it goes low (out of sync) 0 = Disable interrupt for ls_sync_status_2 when it goes low (out of sync)	RW	0
1	ls_sync_statusn_1_int_ctl	1 = Enable interrupt for ls_sync_status_1 when it goes low (out of sync) 0 = Disable interrupt for ls_sync_status_1 when it goes low (out of sync)	RW	0
0	ls_sync_statusn_0_int_ctl	1 = Enable interrupt for ls_sync_status_0 when it goes low (out of sync) 0 = Disable interrupt for ls_sync_status_0 when it goes low (out of sync)	RW	0

### Per Quad PCS Control Registers Details

**Table 8-46. SERDES Control Register QD\_0A**

Bit	Name	Description	Type	Default
7	Internal use only			
6	Reserved		RW	0
5	TX_REFCK_SEL	TxPLL reference clock select 0 = REFCLKP/N 1 = FPGA core	RW	0
4	Reserved			
3	REFCK_RTERM	Termination at reference clock input buffer 0 = High impedance 1 = 50 OHm	RW	1
2	Reserved		RW	0
1	REFCLK_OUT_SEL[1]	0 = refclk2fpga output disable 1 = refclk2fpga output enable	RW	0
0	REFCLK_OUT_SEL[0]	0 = tx_refck_local output enable 1 = tx_refck_local_output disable	RW	0

Note: Refer to Figure 8-9 for Reference Clock Select Control signals.

**Table 8-47. SERDES Control Register QD\_0B**

Bit	Name	Description	Type	Default
7	REFCK25X	1 = Internal high-speed bit clock is 25x 0 = See REFCK_MODE	RW	0
6	BUS8BIT_SEL	1 = Select 8-bit bus width 0 = Select 10-bit bus width	RW	0
5	Reserved		RW	0
4	Reserved		RW	0
3	Reserved		RW	0
2	Reserved		RW	0
1:0	REFCK_MODE[1:0]	If REFCK25X = 0, then: 00 = Internal high-speed bit clock is 20x 01 = Internal high-speed bit clock is 10x 10 = Internal high-speed bit clock is 16x 11 = Internal high-speed bit clock is 8x  If REFCK25X = 1, then: xx = Internal high-speed bit clock is 25x	RW	00

**Table 8-48. PCS Control Register QD\_0C**

Bit	Name	Description	Type	Default										
7:2	Reserved													
1:0	CDR_LOL_SET[1:0]	CDR loss-of-lock setting <table border="0" style="width: 100%;"> <tr> <td style="text-align: center;">Lock</td> <td style="text-align: center;">Unlock</td> </tr> <tr> <td>00 = +/-1000ppm x2</td> <td>+/-1500ppm x2</td> </tr> <tr> <td>01 = +/-2000ppm x2</td> <td>+/-2500ppm x2</td> </tr> <tr> <td>10 = +/-4000ppm</td> <td>+/-7000ppm</td> </tr> <tr> <td>11 = +/-300ppm</td> <td>+/-450ppm</td> </tr> </table>	Lock	Unlock	00 = +/-1000ppm x2	+/-1500ppm x2	01 = +/-2000ppm x2	+/-2500ppm x2	10 = +/-4000ppm	+/-7000ppm	11 = +/-300ppm	+/-450ppm	RW	00
Lock	Unlock													
00 = +/-1000ppm x2	+/-1500ppm x2													
01 = +/-2000ppm x2	+/-2500ppm x2													
10 = +/-4000ppm	+/-7000ppm													
11 = +/-300ppm	+/-450ppm													

**Table 8-49. PCS Control Register QD\_0D**

Bit	Name	Description	Type	Default
7:6	Internal use only			
5	Internal use only			
4:3	PLL_LOL_SET[1:0]	00 = +/- 1350ppm x2 01 = +/- 2400ppm x2 10 = +/- 6800ppm 11 = +/- 400ppm	RW	0
2:0	TX_VCO_CK_DIV[2:0]	VCO output frequency select 00x = Divided by 1      01x = Divided by 2 100 = Divided by 4      101 = Divided by 8 110 = Divided by 16     111 = Divided by 32	RW	0

**Table 8-50. PCS Control Register QD\_0E**

Bit	Name	Description	Type	Default
7:0	Internal use only			

**Table 8-51. PCS Control Register QD\_0F**

Bit	Name	Description	Type	Default
7	PLOL_INT_CTL	1 = Interrupt enabled for loss of lock on PLOL 0 = Interrupt disabled for loss of lock on PLOL	RO CR	0
6	-PLOL_INT_CTL	1 = Interrupt enabled for obtaining lock on PLOL 0 = Interrupt disabled for obtaining lock on PLOL	RO CR	0
5:0	Reserved			

### Per Quad Reset and Clock Control Registers Details

**Table 8-52. PCS Control Register QD\_10**

Bit	Name	Description	Type	Default
7:4	Reserved			
3	serdes_pd	0 = Assert power down	RW	1
2	serdes_rst	1 = Assert serdes reset	RW	0
1	quad_rst	1 = Assert quad reset	RW	0
0	trst	1 = TX reset	RW	0

**Table 8-53. PCS Control Register QD\_11**

Bit	Name	Description	Type	Default
7:0	Reserved			

### Per Quad PCS Status Registers Details

**Table 8-54. PCS Status Register QD\_20**

Bit	Name	Description	Type	Int?
7:6	Reserved			
5	ion_delay	0 = Delayed global resetn from tri_ion	RO	No
4	int_qd_out	1 = Per quad interrupt status	RO	No
3:0	int_ch_out[3:0]	1 = Per channel interrupt status	RO	No

**Table 8-55. PCS Status Register QD\_21**

Bit	Name	Description	Type	Int?
7	ls_sync_status_3	1 = Alarm generated on sync_status_3 0 = Alarm not generated on sync_status_3	RO	Yes
6	ls_sync_status_2	1 = Alarm generated on sync_status_2 0 = Alarm not generated on sync_status_2	RO	Yes
5	ls_sync_status_1	1 = Alarm generated on sync_status_1 0 = Alarm not generated on sync_status_1	RO	Yes
4	ls_sync_status_0	1 = Alarm generated on sync_status_0 0 = Alarm not generated on sync_status_0	RO	Yes
3	ls_sync_statusn_3	1 = Alarm generated on sync_status_3 when it goes low (out of sync) 0 = Alarm not generated on sync_status_3 when it goes low (out of sync)	RO	Yes
2	ls_sync_statusn_2	1 = Alarm generated on sync_status_2 when it goes low (out of sync) 0 = Alarm not generated on sync_status_2 when it goes low (out of sync)	RO	Yes
1	ls_sync_statusn_1	1 = Alarm generated on sync_status_1 when it goes low (out of sync) 0 = Alarm not generated on sync_status_1 when it goes low (out of sync)	RO	Yes
0	ls_sync_statusn_0	1 = Alarm generated on sync_status_0 when it goes low (out of sync) 0 = Alarm not generated on sync_status_0 when it goes low (out of sync)	RO	Yes

**Table 8-56. PCS Interrupt Status Register QD\_22**

Bit	Name	Description	Type	Int?
7	ls_sync_status_3_int	1 = Interrupt generated on sync_status_3 0 = Interrupt not generated on sync_status_3	RO CR	Yes
6	ls_sync_status_2_int	1 = Interrupt generated on sync_status_2 0 = Interrupt not generated on sync_status_2	RO CR	Yes
5	ls_sync_status_1_int	1 = Interrupt generated on sync_status_1 0 = Interrupt not generated on sync_status_1	RO CR	Yes
4	ls_sync_status_0_int	1 = Interrupt generated on sync_status_0 0 = Interrupt not generated on sync_status_0	RO CR	Yes
3	ls_sync_statusn_3_int	1 = Interrupt generated on sync_status_3 when it goes low (out of sync) 0 = Interrupt not generated on sync_status_3 when it goes low (out of sync)	RO CR	Yes
2	ls_sync_statusn_2_int	1 = Interrupt generated on sync_status_2 when it goes low (out of sync) 0 = Interrupt not generated on sync_status_2 when it goes low (out of sync)	RO CR	Yes
1	ls_sync_statusn_1_int	1 = Interrupt generated on sync_status_1 when it goes low (out of sync) 0 = Interrupt not generated on sync_status_1 when it goes low (out of sync)	RO CR	Yes
0	ls_sync_statusn_0_int	1 = Interrupt generated on sync_status_0 when it goes low (out of sync) 0 = Interrupt not generated on sync_status_0 when it goes low (out of sync)	RO CR	Yes

**Table 8-57. PCS Status Register QD\_23**

Bit	Name	Description	Type	Default
7:0	Internal use only			

**Table 8-58. PCS Status Register QD\_24**

Bit	Name	Description	Type	Default
7:0	Internal use only			

### Per Quad SERDES Status Registers Details

**Table 8-59. SERDES Status Register QD\_25**

Bit	Name	Description	Type	Int?
7	PLOL	1 = PLL loss-of-lock	RO	Yes
6	-PLOL	1 = PLL lock obtained	RO	Yes
5:0	Reserved			

**Table 8-60. SERDES Interrupt Status Register QD\_26**

Bit	Name	Description	Type	Int?
7	PLOL_INT	1 = Interrupt generated on PLOL 0 = Interrupt not generated on PLOL	RO CR	Y
6	-PLOL_INT	1 = Interrupt generated on -PLOL 0 = Interrupt not generated on -PLOL	RO CR	Y
5:0	Reserved			

**Table 8-61. SERDES Status Register QD\_27**

Bit	Name	Description	Type	Default
7:0	Reserved			

**Table 8-62. SERDES Status Register QD\_28**

Bit	Name	Description	Type	Default
7:0	Reserved			

## Channel Registers Overview

Table 8-63. Channel Interface Registers Map

BA	Register Name	D7	D6	D5	D4	D3	D2	D1	D0
<b>Per Channel General Control Registers</b>									
00	CH_00					rio_mode	pcie_mode	fc_mode	uc_mode
01	CH_01	word_align_enable	internal use only	internal use only	ge_an_enable		internal use only	invert_tx	invert_rx
02	CH_02	pfifo_clr	pcie_ei_en	pcs_det_time_sel[1]	pcs_det_time_sel[0]	rx_gear_mode	tx_gear_mode	rx_ch	tx_ch
03	CH_03		sb_bypass	sb_pfifo_lp	internal use only	enc_bypass	internal use only	tx_gear_bypass	fb_loopback
04	CH_04	lsm_sel	ilsm_en	rx_gear_bypass	ctc_bypass	dec_bypass	wa_bypass	rx_sb_bypass	sb_loopback
05	CH_05	min_ipg_cnt[1]	min_ipg_cnt[0]	match_4_enable	match_2_enable				
06	CH_06	cc_match_1[7]	cc_match_1[6]	cc_match_1[5]	cc_match_1[4]	cc_match_1[3]	cc_match_1[2]	cc_match_1[1]	cc_match_1[0]
07	CH_07	cc_match_2[7]	cc_match_2[6]	cc_match_2[5]	cc_match_2[4]	cc_match_2[3]	cc_match_2[2]	cc_match_2[1]	cc_match_2[0]
08	CH_08	cc_match_3[7]	cc_match_3[6]	cc_match_3[5]	cc_match_3[4]	cc_match_3[3]	cc_match_3[2]	cc_match_3[1]	cc_match_3[0]
09	CH_09	cc_match_4[7]	cc_match_4[6]	cc_match_4[5]	cc_match_4[4]	cc_match_4[3]	cc_match_4[2]	cc_match_4[1]	cc_match_4[0]
0A	CH_0A	cc_match_4[9]	cc_match_4[8]	cc_match_3[9]	cc_match_3[8]	cc_match_2[9]	cc_match_2[8]	cc_match_1[9]	cc_match_1[8]
0B	CH_0B	udf_comma_mask[7]	udf_comma_mask[6]	udf_comma_mask[5]	udf_comma_mask[4]	udf_comma_mask[3]	udf_comma_mask[2]	udf_comma_mask[1]	udf_comma_mask[0]
0C	CH_0C	udf_comma_a[7]	udf_comma_a[6]	udf_comma_a[5]	udf_comma_a[4]	udf_comma_a[3]	udf_comma_a[2]	udf_comma_a[1]	udf_comma_a[0]
0D	CH_0D	udf_comma_b[7]	udf_comma_b[6]	udf_comma_b[5]	udf_comma_b[4]	udf_comma_b[3]	udf_comma_b[2]	udf_comma_b[1]	udf_comma_b[0]
0E	CH_0E	udf_comma_a[9]	udf_comma_a[8]	udf_comma_b[9]	udf_comma_b[8]	udf_comma_mask[9]	udf_comma_mask[8]		
0F	CH_0F					cc_underrun_int_ctl	cc_ouerrun_int_ctl	fb_rx_fifo_error_int_ctl	fb_tx_fifo_error_int_ctl
<b>Per Channel SERDES Control Registers</b>									
10	CH_10	req_en	req_lv_set	rcv_dcc_en	rate_sel[1]	rate_sel[0]	rx_dco_ck_div[2]	rx_dco_ck_div[1]	rx_dco_ck_div[0]
11	CH_11	internal use only	internal use only	lb_ctl[1]	lb_ctl[0]	internal use only	internal use only	rterm_rx[1]	rterm_rx[0]
12	CH_12	tdrv_amp[2]	tdrv_amp[1]	tdrv_amp[0]	tdrv_pre_set[4]	tdrv_pre_set[3]	tdrv_pre_set[2]	tdrv_pre_set[1]	tdrv_pre_set[0]
13	CH_13	ldr_core2tx_sel				internal use only	internal use only	internal use only	internal use only
14	CH_14	tx_div11_sel	tdrv_dat_sel[1]	tdrv_dat_sel[0]	tdrv_ppre_en	rterm_tx[1]	rterm_tx[0]	rate_mode_tx	tpwrap
15	CH_15	internal use only	internal use only	internal use only	internal use only	ldr_rx2core_en	rx_refck_sel	rate_mode_rx	rpwrap
16	CH_16	rx_div11_sel	rlos_sel				rlos_lset[2]	rlos_lset[1]	rlos_lset[0]
17	CH_17		pci_det_done_int_ctl	rlos_lo_int_ctl	-rlos_lo_int_ctl			riol_int_ctl	-riol_int_ctl
<b>Per Channel Clock Reset Registers</b>									
18	CH_18	internal use only	internal use only				rrst	lane_rx_rst	lane_tx_rst
19	CH_19				tx_f_clk_dis	tx_h_clk_en	rx_f_clk_dis	rx_h_clk_en	sel_sd_rx_clk
<b>Per Channel General Status Registers</b>									
20	CH_20					cc_underrun	cc_ouerrun	fb_rx_fifo_error	fb_tx_fifo_error
21	CH_21	prbs_error_cnt[7]	prbs_error_cnt[6]	prbs_error_cnt[5]	prbs_error_cnt[4]	prbs_error_cnt[3]	prbs_error_cnt[2]	prbs_error_cnt[1]	prbs_error_cnt[0]
22	CH_22					wa_offset[3]	wa_offset[2]	wa_offset[1]	wa_offset[0]
23	CH_23					cc_underrun_int	cc_ouerrun_int	fb_rx_fifo_error_int	fb_tx_fifo_error_int
24	CH_24		ffs_ls_sync_status	fb_rxrst_o	fb_txrst_o			cc_re_o	cc_we_o
25	CH_25								
<b>Per Channel SERDES Status Registers</b>									
26	CH_26		pcie_det_done	rlos_lo	-rlos_lo	rlos_hi	-rlos_hi	riol	-riol
27	CH_27	internal use only	internal use only	internal use only	internal use only			cdr_train_done	pci_connect
28	CH_28	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only
29	CH_29	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only	internal use only
2A	CH_2A		pci_det_done_int	rlos_lo_int	-rlos_lo_int	rlos_hi_int	-rlos_hi_int	riol_int	-riol_int
2B	CH_2B								
2C	CH_2C								

**Per Channel PCS Control Registers Details**
**Table 8-64. PCS Control Register CH\_00**

Bit	Name	Description	Type	Default
7:4	Reserved			
3	rio_mode	1 = Selects RapidIO mode 0 = Selects other mode (10GbE, 1GbE)	RW	0
2	pcie_mode	1 = Selects PCI Express mode 0 = Selects other mode (RapidIO, 10GbE, 1GbE)	RW	0
1	fc_mode	1 = Selects Fibre Channel mode 0 = Selects other mode (PCI Express, RapidIO, 10GbE, 1GbE)	RW	0
0	uc_mode	1 = Selects User Configured (G8B10B, 8BSER only, 10BSER only) mode 0 = Selects other mode (Fibre Channel, PCI Express, RapidIO, 10GbE, 1GbE)	RW	0

**Table 8-65. PCS Control Register CH\_01**

Bit	Name	Description	Type	Default
7	word_align_enable	1 = Enable continuous comma alignment 0 = Disable continuous comma alignment	RW	0
6	Internal use only			
5	Internal use only			
4	ge_an_enable	1 = Enable GbE Auto-negotiation 0 = Disable GbE Auto-negotiation	RW	0
3	Reserved			
2	Internal use only			
1	invert_tx	1 = Invert transmit data 0 = Don't invert transmit data	RW	0
0	invert_rx	1 = Invert received data 0 = Don't invert received data	RW	0



**Table 8-66. PCS Control Register CH\_02**

Bit	Name	Description	Type	Default
7	pfifo_clr	1 = Clears PFIFO if quad register bit pfifo_clr_sel is set to 1. This signal is or'ed with interface signal pfifo_clr. 0 = Normal operation	RW	0
6	pcie_ei_en	1 = PCI Express Electrical Idle enabled 0 = Normal operation	RW	0
5:4	pcs_det_time_sel[1:0]	PCS connection detection time 11 = 16us 10 = 4us 01 = 2us 00 = 8us	RW	0
3	rx_gear_mode	1 = Enable 2:1 gearing for receive path on selected channels 0 = Disable 2:1 gearing for receive path on selected channels		
2	tx_gear_mode	1 = Enable 2:1 gearing for transmit path on selected channels 0 = Disable 2:1 gearing for transmit path on selected channels	RW	0
1	rx_ch	1 = Received output can be monitored on the test characterization pins. The test characterization mode (bit 6 in PCS controller register QD_03) should be set to '1'.	RW	0
0	tx_ch	1 = Transmit PCS inputs are sourced from test characterization ports. The test characterization mode should be enabled.	RW	0

**Table 8-67. PCS Control Register CH\_03**

Bit	Name	Description	Type	Default
7	Reserved			
6	sb_bypass	1 = Bypass TX SERDES Bridge 0 = Normal operation	RW	0
5	sb_pfifo_lp	1 = Enable parallel loopback from RX to TX via parallel FIFO 0 = Normal operation	RW	0
4	internal use only			
3	enc_bypass	1 = Bypass 8b10b encoder 0 = Normal operation		
2	internal use only			
1	tx_gear_bypass	1 = Bypass PCS TX gear 0 = Normal operation	RW	0
0	internal use only			

**Table 8-68. PCS Control Register CH\_04**

Bit	Name	Description	Type	Default
7	lsm_sel	1 = Select external RX link state machine 0 = Select internal link state machine	RW	0
6	ilsm_en	1 = Force enabling the RX link state machine 0 = Force disabling the RX link state machine.	RW	0
5	rx_gear_bypass	1 = Bypass PCS RX gear 0 = Normal operation	RW	0
4	ctc_bypass	1 = Bypass clock tolerance compensation 0 = Select Normal data	RW	0
3	dec_bypass	1 = Bypass 8b10b decoder 0 = Normal operation		
2	wa_bypass	1 = Bypass word alignment 0 = Normal operation	RW	0
1	rx_sb_bypass	1 = Bypass RX SERDES bridge 0 = Normal operation	RW	0
0	sb_loopback	1 = Enable loopback in the PCS from TX to RX in SERDES bridge 0 = Normal operation	RW	0

**Table 8-69. PCS Control Register CH\_05**

Bit	Name	Description	Type	Default
7:6	min_ipg_cnt[1:0]	Minimum IPG to enforce	RW	11
5	match_4_enable	1 = Enable four character skip matching (using match 4, 3, 2, 1)	RW	0
4	match_2_enable	1 = Enable two character skip matching (using match 4, 3)	RW	1
3:0	Reserved			

**Table 8-70. PCS Control Register CH\_06**

Bit	Name	Description	Type	Default
7:0	cc_match_1[7:0]	Lower bits of user-defined clock compensator skip pattern 1	RW	8'h00

**Table 8-71. PCS Control Register CH\_07**

Bit	Name	Description	Type	Default
7:0	cc_match_2[7:0]	Lower bits of user-defined clock compensator skip pattern 2	RW	8'h00

**Table 8-72. PCS Control Register CH\_08**

Bit	Name	Description	Type	Default
7:0	cc_match_3[7:0]	Lower bits of user-defined clock compensator skip pattern 3	RW	8'hBC

**Table 8-73. PCS Control Register CH\_09**

Bit	Name	Description	Type	Default
7:0	cc_match_4[7:0]	Lower bits of user-defined clock compensator skip pattern 4	RW	8'h50

**Table 8-74. PCS Control Register CH\_0A**

Bit	Name	Description	Type	Default
7:6	cc_match_4[9:8]	Upper bits of user-defined clock compensator skip pattern 4 [9] = Disparity error [8] = K control	RW	2'b01
5:4	cc_match_3[9:8]	Upper bits of user-defined clock compensator skip pattern 3 [9] = Disparity error [8] = K control	RW	2'b01
3:2	cc_match_2[9:8]	Upper bits of user-defined clock compensator skip pattern 2 [9] = Disparity error [8] = K control	RW	2'b00
1:0	cc_match_1[9:8]	Upper bits of user-defined clock compensator skip pattern 1 [9] = Disparity error [8] = K control	RW	2'b00

**Table 8-75. PCS Control Register CH\_0B**

Bit	Name	Description	Type	Default
7:0	udf_comma_mask[7:0]	Lower bits of user-defined comma mask	RW	8'hFF

**Table 8-76. PCS Control Register CH\_0C**

Bit	Name	Description	Type	Default
7:0	udf_comma_a[7:0]	Lower bits of user-defined comma character 'a'	RW	8'h83

**Table 8-77. PCS Control Register CH\_0D**

Bit	Name	Description	Type	Default
7:0	udf_comma_b[7:0]	Lower bits of user-defined comma character 'b'	RW	8'h7C

**Table 8-78. PCS Control Register CH\_0E**

Bit	Name	Description	Type	Default
7:6	udf_comma_a[9:8]	Upper bits of user-defined comma character 'a'	RW	2'b10
5:4	udf_comma_b[9:8]	Upper bits of user-defined comma character 'b'	RW	2'b01
3:2	udf_comma_mask[9:8]	Upper bits of user-defined comma mask	RW	2'b11 <sup>1</sup>
1:0	Reserved			

1. In most applications, K28.5 is used as the comma character. The default value of the mask is 11111111. In G8B10B mode, any comma can be used so the mask will be 1111111100 to detect any of the three comma characters, K28.1, 28.5, 28.7.

**Table 8-79. PCS Interrupt Control Register CH\_0F**

Bit	Name	Description	Type	Default
7:4	Reserved			
3	cc_underrun_int_ctl	1 = Enable interrupt for cc_underrun 0 = Disable interrupt for cc_underrun	RW	0
2	cc_overnun_int_ctl	1 = Enable interrupt for cc_overnun 0 = Disable interrupt for cc_overnun	RW	0
1	fb_rx_fifo_error_int_ctl	1 = Enable interrupt on empty/full condition in the receive FPGA bridge FIFO	RW	0
0	fb_tx_fifo_error_int_ctl	1 = Enable interrupt on empty/full condition in the transmit FPGA bridge FIFO	RW	0

### Per Channel SERDES Control Registers Details

Unless indicated, all channels must be reset after writing any SERDES Control Register.

**Table 8-80. SERDES Control Register CH\_10**

Bit	Name	Description	Type	Default
7	REQ_EN	1= Enable receiver equalization 0 = Disable receiver equalization	RW	0
6	REQ_LVL_SET	Level setting for equalization 1 = Long-reach equalization 0 = Mid-length route equalization	RW	0
5	RCV_DCC_EN	1 = Enable receiver DC coupling 0 = Enable receiver AC coupling	RW	0
4:3	RATE_SEL[1:0]	Equalizer pole position select: 00 = HIGH Equalization 01 = MED Equalization 10 = LOW Equalization 11 = Reserved	RW	00
2:0	RX_DCO_CK_DIV[2:0]	VCO output frequency select: 00x = Divide by 1 01x = Divide by 2 100 = divide by 4 101 = Divide by 8 110 = divide by 16 111 = Divide by 32	RW	000

**Table 8-81. SERDES Control Register CH\_11**

Bit	Name	Description	Type	Default
7:4	LB_CTL[3:0]	Loop back control: [3] = internal use only [2] = internal use only [1] = slb_eq2t_en, serial LB from equalizer to driver enable [0] = slb_t2r_en, serial tx to rx LB enable	R/W	4'h0
3:2	Reserved			
1:0	RTERM_RX[1:0]	00 = HiZ, 01 = 50 Ohm, 10 = 60 Ohm, 11 = 75 Ohm	R/W	2'b01

**Table 8-82. SERDES Control Register CH\_12**

Bit	Name	Description	Type	Default
7:5	TDRV_AMP[2:0]	CML driver amplitude setting: 000 = 0% 001 = +8% 010 = +11% 011 = +20% 100 = -17% 101 = -12% 110 = -10% 111 = -6%	RW	000
4:0	TDRV_PRE_SET[4:0]	TX drive pre-emphasis level setting [2:0] 000 = 0% 001 = 5% 010 = 12% 011 = 18% 100 = 25% 101 = 33% 110 = 40% 111 = 48% [4:3] Fine tune (added to any value chosen by bits[2:0] settings above) 00 = 0% 01 = 2% 10 = 3% 11 = 5%	RW	5'b00000

**Table 8-83. SERDES Control Register CH\_13**

Bit	Name	Description	Type	Default
7	ldr_core2tx_sel	1 = Select low speed serial data from FPGA core	RW	0
6	pden_sel	Reserved	RW	0
5:4	Reserved			
3	TDRV_AMP_BOOST	TX Drive amplitude boost 0 = 0% 1 = -25%	RW	0
2:0	TDRV_DRVCUR_SET[2:0]	000 = 48% 001 = 30% 010 = 60% 011 = 50% 100 = 0% 101 = -7% 110 = 19% 111 = 8%	RW	100

**Table 8-84. SERDES Control Register CH\_14**

Bit	Name	Description	Type	Default
7	TX_DIV11_SEL	0 = Full-rate selection for transmit (high definition SMPTE) 1 = Divide-by-11 selection for transmit (standard definition SMPTE)	RW	0
6:5	TDRV_DAT_SEL [1:0]	Driver output select: 00 = Data from Serializer muxed to driver (normal operation) 11 = Serial LB from equalizer to driver if slb_eq2t_en='1'	RW	00
4	TDRV_PRE_EN	1 = TX driver pre-emphasis enable 0 = TX driver pre-emphasis disable	RW	0
3:2	RTERM_TX[1:0]	TX resistor termination select. Disable when PCI Express mode is selected. 0x = 5K OHm 10 = 50 Ohm 11 = 75 OHm	RW	10
1	RATE_MODE_TX	0 = Full-rate selection for transmit 1 = Half-rate selection for transmit	RW	0
0	tpwrup	0 = Power-down transmit channel 1 = Power-up transmit channel	RW	0

**Table 8-85. SERDES Control Register CH\_15**

Bit	Name	Description	Type	Default
7	Internal use only			
6	Internal use only			
5	Internal use only			
4	Internal use only			
3	ldr_rx2core_en	1 = Enables boundary scan input path for routing the high-speed receive inputs to a lower-speed SERDES in the FPGA (for out-of-band applications)	RW	0
2	rx_refck_sel	RX CDR reference clock select 0 = REFCLKP/N 1 = FPGA core	RW	0
1	RATE_MODE_RX	0 = Full-rate selection for receive 1 = Half-rate selection for receive	RW	0
0	rpwrup	0 = Power-down receive channel 1 = Power-up receive channel	RW	0

**Table 8-86. SERDES Control Register CH\_16**

Bit	Name	Description	Type	Default
7	RX_DIV11_SEL	0 = Full-rate selection for receive (high-definition SMPTE) 1 = Divide by 11 selection for receive (standard-definition SMPTE)	RW	0
6	rlos_sel	1 = Select rlos_hi 0 = Select rlos_lo	RW	0
5:3	Reserved		RW	000
2:0	RLOS_LSET[2:0]	LOS detector reference current adjustment for smaller swing 000 = default 010 = +15% 011 = +25%	RW	010

**Table 8-87. SERDES Interrupt Control Register CH\_17**

Bit	Name	Description	Type	Default
7	Reserved			
6	pci_det_done_int_ctl	1 = Enable interrupt for detection of far-end receiver for PCI Express	RW	0
5	rlos_lo_int_ctl	1 = Enable interrupt for RX loss of signal when input levels fall below the programmed LOW threshold (using rlos_set)	RW	0
4	-rlos_lo_int_ctl	1 = Enable interrupt for RX loss of signal when input level meets or is greater than programmed LOW threshold	RW	0
3	Reserved			
2	Reserved			
1	rlo_l_int_ctl	1 = Enable interrupt for receiver loss of lock	RW	0
0	-rlo_l_int_ctl	1 = Enable interrupt when receiver recovers from loss of lock	RW	0

### Per Channel Reset and Clock Control Registers Details

**Table 8-88. Reset and Clock Control Register CH\_18**

Bit	Name	Description	Type	Default
7	Internal use only			
6	Internal use only			
5:3	Reserved			
2	rrst	1 = RX reset	RW	0
1	lane_rx_rst	1 = Assert reset signal to receive logic	RW	0
0	lane_tx_rst	1 = Assert reset signal to transmit logic	RW	0

**Table 8-89. Reset and Clock Control Register CH\_19**

Bit	Name	Description	Type	Default
7:5	Reserved			
4	tx_f_clk_dis	1 = Disable tx_f_clk	RW	0
3	tx_h_clk_en	1 = Enable tx_h_clk	RW	0
2	rx_f_clk_dis	1 = Disable rx_f_clk	RW	0
1	rx_h_clk_en	1 = Enable rx_h_clk	RW	0
0	sel_sd_rx_clk	1 = Select sd_rx_clk	RW	0

**Per Channel PCS Status Registers Details**
**Table 8-90. PCS Status Register CH\_20**

Bit	Name	Description	Type	Int?
7:5	Reserved			
4	pfifo_error	1 = Parallel FIFO error 0 = No parallel FIFO error	RO	Yes
3	cc_underrun	1 = CTC FIFO underrun 0 = CTC FIFO not underrun	RO	Yes
2	cc_overrun	1 = CTC FIFO overrun 0 = CTC FIFO not overrun	RO	Yes
1	fb_rx_fifo_error	1 = FPGA bridge (FB) RX FIFO overrun 0 = FB RX FIFO not overrun	RO	Yes
0	fb_tx_fifo_error	1 = FPGA bridge (FB) TX FIFO overrun 0 = FB TX FIFO not overrun	RO	Yes

**Table 8-91. PCS Status Register CH\_21**

Bit	Name	Description	Type	Int?
7:0	prbs_errors <sup>1</sup>	Count of the number of PRBS errors. Clears to zero on read. Sticks at flip-flop.	RO CR	No

1. Built-in PRBS generator and checker are for internal use only.

**Table 8-92. PCS Status Register CH\_22**

Bit	Name	Description	Type	Int?
7:4	Reserved			
3:0	wa_offset[3:0]	Word aligner offset	RO	No

**Table 8-93. PCS Interrupt Status Register CH\_23**

Bit	Name	Description	Type	Int?
7:4	Reserved			
3	cc_underrun_int	1 = Interrupt generated on cc_underrun 0 = Interrupt not generated on cc_underrun	RO CR	Yes
2	cc_overrun_int	1 = Interrupt generated on cc_overrun 0 = Interrupt not generated on cc_overrun	RO CR	Yes
1	fb_rx_fifo_error_int	1 = Interrupt generated on fb_rx_fifo_error 0 = Interrupt not generated on fb_rx_fifo_error	RO CR	Yes
0	fb_tx_fifo_error_int	1 = Interrupt generated on fb_tx_fifo_error 0 = Interrupt not generated on fb_tx_fifo_error	RO CR	Yes



**Table 8-94. PCS Status Register CH\_24**

Bit	Name	Description	Type	Int?
7	Reserved			
6	ffs_ls_sync_status	1 = Sync in the link state machine 0 = Not sync in the LSM	RO	No
5	fb_rxrst_o	1 = FPGA bridge RX normal operation 0 = FPGA bridge RX reset	RO	No
4	fb_txrst_o	1 = FPGA bridge TX normal operation 0 = FPGA bridge TX reset	RO	No
3	Reserved			
2	Reserved			
1	cc_re_o	1 = CTC FIFO read enable 0 = CTC FIFO read disable	RO	No
0	cc_we_o	1 = CTC FIFO write enable 0 = CTC FIFO write disable	RO	No

**Table 8-95. PCS Status Register CH\_25**

Bit	Name	Description	Type	Int?
7	Reserved			

## Per Channel SERDES Status Registers Details

**Table 8-96. SERDES Status Register CH\_26**

Bit	Name	Description	Type	Int?
7	Reserved			
6	pci_det_done	1 = Receiver detection process not completed by SERDES transmitter 0 = Receiver detection process completed by SERDES transmitter	RO CR	Yes
5	rlos_lo	1 = Indicates that the input signal detected by receiver is below the programmed LOW threshold	RO CR	Yes
4	-rlos_lo	1 = Indicates that the input signal detected by receiver is greater or equal to the programmed LOW threshold	RO CR	Yes
3	Reserved		RO CR	Yes
2	Reserved		RO CR	Yes
1	rlol	1 = Indicates CDR loss of lock to data. CDR is locked to reference clock	RO	Yes
0	-rlol	1 = Indicates that CDR has locked to data	RO	Yes

**Table 8-97. SERDES Status Register CH\_27**

Bit	Name	Description	Type	Int?
7	Internal use only			
6	Internal use only			
5	Internal use only			
4	Internal use only			
3:2	Reserved			
1	cdr_trained	1 = Indicates CDR training is done	RO	No
0	pci_connect	1 = Receiver detected by SERDES transmitter (at the transmitter device) 0 = Receiver not detected by SERDES transmitter (at the transmitter device)	RO	No

**Table 8-98. SERDES Status Register CH\_28**

Bit	Name	Description	Type	Int?
7:0	Internal use only			

**Table 8-99. SERDES Status Register CH\_29**

Bit	Name	Description	Type	Int?
7:0	Internal use only			

**Table 8-100. SERDES Interrupt Status Register CH\_2A**

Bit	Name	Description	Type	Int?
7	Reserved			
6	pci_det_done_int	1 = Interrupt generated for pci_det_done	RO CR	Yes
5	rlos_lo_int	1 = Interrupt generated for rlos_lo	RO CR	Yes
4	-rlos_lo_int	1 = Interrupt generated for -rlos_lo	RO CR	Yes
3	Reserved		RO CR	Yes
2	Reserved		RO CR	Yes
1	rlol_int	1 = Interrupt generated for rlol	RO CR	Yes
0	-rlol_int	1 = Interrupt generated for -rlol	RO CR	Yes

**Table 8-101. PCS Status Register CH\_2B**

Bit	Name	Description	Type	Int?
7	Reserved			

**Table 8-102. PCS Status Register CH\_2C**

Bit	Name	Description	Type	Int?
7	Reserved			

## Appendix B. Register Settings for Various Standards

### Per Channel Register Settings for Various Standards

*Table 8-103. Per Channel Register Settings for Various Standards*

Character	1GbE	10GbE	1GFC	PCI-Ex	RapidIO
K23.7 (F7)	Carrier extend			PAD	
K27.7 (FB)	SOP	ST		Start TLP	A (align)
K28.0 (1C)		SKIP R		SKIP	SC
K28.1 (3C)				FTS	
K28.2 (5C)		SoS		Start DLP	
K28.3 (7C)		ALIGN A		IDLE	PD
K28.4 (9C)		SEQ			
K28.5 (BC)	+D5.6 or D16.2 = IDLE	SYNC K	+D21.4+D21.5 +D21.5 = IDLE	COMMA (used for alignment)	K
K28.6 (DC)					
K28.7 (FC)					R (skip)
K29.7 (FD)	EOP	T		END	
K30.7 (FE)	ERR	ERR		END BAD	

### Per Quad Register Settings for Various Standards

*Table 8-104. Per Quad Register Settings for Various Standards*

Register	1GbE	10GbE	1G, 2G FC	PCI-Ex 1x	PCI-Ex 4x	RapidIO 1x	RapidIO 4x
comma_a_lo	hex 03	hex 03	hex 03	hex 03	hex 03	hex 03	hex 03
comma_b_lo	hex FC	hex FC	hex FC	hex FC	hex FC	hex FC	hex FC
comma_mask_lo	hex 7F	hex 7F	hex 7F	hex 7F	hex 7F	hex 7F	hex 7F

## Appendix C. Attribute Cross Reference Table

Table 8-105. Attribute Cross-Reference Table

Independent Attribute Name	Dependent Attribute Name	Attribute Value	Register Map
QUAD_MODE	{Qn_REFCK_NQ_EN} <sup>8</sup>	SINGLE : {0} MASTER : {1} SLAVE : {1} SLAVE_END : {0}	{QD_0b[2]}
CHn_PROTOCOL	{10G_MODE, CHn_PROT_MODE, CHn_RX_DET, CHn_GE_AN_EN}	GIGE : {0, 0000,00,1} FC : {0,0010,00,0} XAUI : {1,0000,00,0} SRIO : {0,1000,00,0} PCIE : {0,0100,00,0} SDI : {0,0001,00,0} G8B10B : {0,0001,00,0} 10BSER : {0,0001,00,0} 8BSER : {0,0001,00,0} CPRI : {0,0001,00,0} OBSAI : {0,0001,00,0}	{QD_00[4], CH_00[3:0], CH_02[5:4], CH_01[4]}
CHn_MODE	{CHn_TXPWDNB, CHn_RXPWDNB}	RXTX : {11} RXONLY : {01} TXONLY : {10} DISABLED : {00}	{CH_14[0], CH_15[0]}
TX_DATARATE_RANGE	{PLL_DIV}	LOWLOW : {110} LOW : {101} MEDLOW : {100} MED : {010} MEDHIGH : {010} HIGH : {000}	{QD_0D[2:0]}
CHn_RX_DATARATE_RANGE	{CHn_CDR_DIV}	LOWLOW : {110} LOW : {101} MEDLOW : {100} MED : {010} MEDHIGH : {000} HIGH : {000}	{CH_10[2:0]}
REFCK_MULT	{REFCK25X, REFCK_MODE}	8X : {0,11} 10X : {0,01} 16X : {0,10} 20X : {0,00} 25X : {1,00}	{QD_0b[7], QD_0b[1:0]}
CHn_RX_DATA_RATE	{CHn_RX_RATE_MODE, CHn_RX_DIV11}	FULL : {00} DIV2 : {10} DIV11 : {01}	{CH_15[1], CH_16[7]}
CHn_TX_DATA_RATE	{CHn_TX_RATE_MODE, CHn_TX_DIV11}	FULL : {00} DIV2 : {10} DIV11 : {01}	{CH_14[1], CH_14[7]}
CHn_TX_DATA_WIDTH	{CHn_TXCLKF, CHn_TXCLKH, CHn_TX_GEAR}	8 : {0,1,0} 10 : {0,1,0} 16 : {0,1,1} 20 : {0,1,1}	{CH_19[4], CH_19[3], CH_02[2]}
CHn_RX_DATA_WIDTH	{CHn_RXCLKF, CHn_RXCLKH, CHn_RX_GEAR}	8 : {0,0,0} 10 : {0,0,0} 16 : {1,1,1} 20 : {1,1,1}	{CH_19[2], CH_19[1], CH_02[3]}
CHn_TX_FIFO		DISABLED : {1} ENABLED : {0}	{CH_03[1]}
CHn_RX_FIFO		DISABLED : {1} ENABLED : {0}	{CH_04[5]}

**Table 8-105. Attribute Cross-Reference Table (Continued)**

Independent Attribute Name	Dependent Attribute Name	Attribute Value	Register Map
PLL_SRC	{TXREFCK_NQ_SEL, TXREFCK_SEL}	REFCLK_EXT : {0,0} REFCLK_CORE: {0,1} REFCLK_NQ : {1,0} <sup>8</sup>	{QD_0B[3], QD_0A[5]}
CHn_CDR_SRC	{RXREFCK_NQ_SEL, CHn_RXREFCK_SEL, CHn_TRAIN_EN, CHn_TRAIN_DIV }	REFCLK_EXT : {0,0,0,0} REFCLK_CORE: {0,1,0,0} REFCLK_NQ : {1,0,0,0} <sup>8</sup> TRAIN_DIV4 : {0,0,1,0} <sup>8</sup> TRAIN_DIV8 : {0,0,1,1} <sup>8</sup>	{QD_0B[4], CH_15[2], CH_15[7], CH_15[6]}
CHn_TDRV <sup>7</sup>	{CHn_TDRV_AMP, CHn_TDRV_DRVCUR_SET, CHn_TDRV_AMP_BOOST}	-4: {110,100,1} -3: {100,101,0} -2: {100,100,0} -1: {101,100,0} 0: {000,100,0} 1: {001,100,0} 2: {011,100,0} 3: {100,000,0} 4: {000,000,0}	{CH_12[7:5], CH_13[2:0], CH_13[3]}
CHn_TDRV (for PCI Express Protocol only)	{CHn_TDRV_AMP, CHn_TDRV_DRVCUR_SET, CHn_TDRV_AMP_BOOST, CHn_TDRV_PRE_EN, CHn_TDRV_PRE_SET}	2: {100,000,0,1,00101}	{CH_12[7:5], CH_13[2:0], CH_13[3], CH_14[4], CH_12[4:0]}
CHn_TX_PRE	{CHn_TDRV_PRE_EN, CHn_TDRV_PRE_SET}	DISABLED: {0,00000} 0: {1,00000} 1: {1,00001} 2: {1,00010} 3: {1,00011} 4: {1,00100} 5: {1,00101} 6: {1,00110} 7: {1,00111}	{CH_14[4], CH_12[4:0]}
CHn_RTERM_TX		50: {10} 75: {11} 5K: {0X}	{CH_14[3:2]}
CHn_RX_EQ	{CHn_REQ_EN, CHn_REQ_LVL_SET, CHn_RATE_SEL}	DISABLED : {0,0,00} MID_LOW : {1,0,10} MID_MED : {1,0,01} MID_HIGH : {1,0,00} LONG_LOW : {1,1,10} LONG_MED : {1,1,01} LONG_HIGH: {1,1,00}	{CH_10[7], CH_10[6], CH_10[4:3]}
CHn_RTERM_RX	{CHn_RX_RTERM}	50 : {01} 60 : {10} 75 : {11} HIGH: {00}	{CH_11[1:0]}
CHn_RX_DCC		AC: {0} DC: {1}	{CH_10[5]}
CHn_LOS_THRESHOLD_LO <sup>1</sup>	{CHn_RLOS_E}	0: {0000} 1: {0001} 2: {0010} 3: {0011} 4: {0100} 5: {0101} 6: {0110} 7: {0111}	{mc1_ser_ctl_chN[75], CH_16[2:0]}
PLL_TERM		50: {1} 2K: {0}	{QD_0A[3]}

**Table 8-105. Attribute Cross-Reference Table (Continued)**

Independent Attribute Name	Dependent Attribute Name	Attribute Value	Register Map
PLL_DCC		AC: {0} DC: {1}	{QD_0A[4]}
PLL_LOL_SET		0: {00} 1: {01} 2: {10} 3: {11}	{QD_0D[4:3]}
CHn_TX_SB	{CHn_TXPOL, CHn_TXSBBYP}	DISABLED: {0,0} ENABLED : {1,0}	{CH_01[1], CH_03[6]}
CHn_RX_SB	{CHn_RXPOL, CHn_RXSBBYP}	DISABLED: {0,0} ENABLED : {1,0}	{CH_01[0], CH_04[1]}
CHn_TX_8B10B		ENABLED : {0} DISABLED: {1}	{CH_03[3]}
CHn_RX_8B10B		ENABLED : {0} DISABLED: {1}	{CH_04[3]}
CHn_COMMA_A		Note 2	{QD_0C[0:7], QD_0E[6:7]}
CHn_COMMA_B		Note 2	{QD_0D[0:7], QD_0E[4:5]}
CHn_COMMA_M		Note 2	{QD_0B[0:7], QD_0E[2:3]}
CHn_RXWA		DISABLED: {1} ENABLED : {0}	{CH_04[2]}
CHn_ILSM		DISABLED: {1} ENABLED : {0}	{CH_04[7]}
CHn_CTC	{CHn_RXRECCLK}	ENABLED : {0,0} DISABLED{1,1}	{CH_19[0] , CH_04[4]}
CHn_CC_MATCH1		Note 2	{QD_0A[1:0], CH_06[7:0]}
CHn_CC_MATCH2		Note 2	{QD_0A[3:2], CH_07[7:0]}
CHn_CC_MATCH3		Note 2	{QD_0A[5:4], CH_08[7:0]}
CHn_CC_MATCH4		Note 2	{QD_0A[7:6], QD_09[7:0]}
CHn_CC_MATCH_MODE	{CHn_MATCH_2_EN, CHn_MATCH_4_EN}	1: {0,0} 2: {1,0} 4: {0,1}	{CH_05[4], CH_05[5]}
CHn_CC_MIN_IPG		0: {00} 1: {01} 2: {10} 3: {11}	{CH_05[7:6]}

**Table 8-105. Attribute Cross-Reference Table (Continued)**

Independent Attribute Name	Dependent Attribute Name	Attribute Value	Register Map
CCHMARK		0 : {0000} 1 : {0001} 2 : {0010} 3 : {0011} 4 : {0100} 5 : {0101} 6 : {0110} 7 : {0111} 8 : {1000} 9 : {1001} 10 : {1010} 11 : {1011} 12 : {1100} 13 : {1101} 14 : {1110} 15 : {1111}	{QD_02[7:4]}
CCLMARK		0 : {0000} 1 : {0001} 2 : {0010} 3 : {0011} 4 : {0100} 5 : {0101} 6 : {0110} 7 : {0111} 8 : {1000} 9 : {1001} 10 : {1010} 11 : {1011} 12 : {1100} 13 : {1101} 14 : {1110} 15 : {1111}	{QD_02[3:0]}
CHn_SSLB		DISABLED : {0000, 00} ENABLED_EQ2T : {0010, 11} ENABLED_T2R : {0001, 00}	{CH_11[7:4], CH_14[6:5]}
CHn_SPLBPORTS <sup>4</sup>	{PFIFO_CLR_SEL, CHn_SB_PFIFO_LP}	DISABLED : {0,0} ENABLED : {1,1}	{QD_03[2], CH_03[5]}
QD_REFCK2CORE		DISABLED : {0} ENABLED : {1}	{QD_0a[1]}
INT_ALL	{PLOLINT, PLOLNINT, CHn_PCIDETINT, CHn_RLOSLINT, CHn_RLOSLNINT, CHn_RLOLINT, CHn_RLOLNINT, CHn_LSSYNCINT, CHn_LSSYNCNINT, CHn_TXFIFOINT, CHn_RXFIFOINT, CHn_CCORUNINT, CHn_CCURUNINT}	DISABLED : { 0,0,0,0,0,0,0,0,0000,0000, 0,0,0,0} ENABLED : { 1,1,1,1,1,1,1,1,1111,1111, 1,1,1,1}	{QD_OF[7], QD_OF[6], CH_17[6], CH_17[5], CH_17[4], CH_17[1], CH_17[0], QD_09[7:4], QD_09[3:0], CH_OF[0], CH_OF[1], CH_OF[2], CH_OF[3]}
CHn_LDR	{CHn_LDR_RX_EN, CHn_LDR_TX_SEL}	DISABLED : {0,0} RXTX : {1,1} RXONLY : {1,0} TXONLY : {0,1}	{CH_15[3], CH_13[7]}
{PLL_SRC, CHn_CDR_SRC}	{REFCKLOCAL}	Note 5	{QD_0A[0]}

**Table 8-105. Attribute Cross-Reference Table (Continued)**

Independent Attribute Name	Dependent Attribute Name	Attribute Value	Register Map
{CHn_TX_8B10B, CHn_RX_8B10B}	{BUS8BIT_SEL}	Note 6	{QD_0B [6]}
{CHn_RXWA, CHn_ILSM}	{CHn_SIG_DET}	{DISABLED, X} <sup>3</sup> {0} {ENABLED, DISABLED} {0} {ENABLED, ENABLED} {1}	{CH_04 [6]}
{CHn_RXWA, CHn_ILSM}	{CHn_C_ALIGN}	{DISABLED, X} <sup>23</sup> {0} {ENABLED, DISABLED} {0} {ENABLED, ENABLED} {0}	{CH_01 [7]}

- rx\_los\_low will only show that a signal has been detected for data rates above 1 Gbps with a maximum CID (Consecutive Identical Digits) of 7 bits (i.e., a minimum input signal transition density as is sent by 8b10b). rx\_los\_low is only supported with a default setting of rlos\_lset[2:0] = 2 for all protocols except PCI Express and SDI. For PCI Express, 2 and 3 are supported. In SDI mode, it is recommended to use the carrier detect output signal (/CD) from the external SDI cable equalizer. rlos\_hset is not supported.
- 10-bit symbol code default / populated by the user in the “PCS Advanced Setup” configuration GUI. Since the 10-bit symbol code representation in the GUI is LSB to MSB, the bit representation is swapped appropriately in the table for software use.
- X = don't care.
- If any of the channels is enabled, the quad bit will be set to 1.
- If PLL\_SRC and all of the enabled CHn\_CDR\_SRC is set to REFCLK\_CORE then this bit should be 1, else this bit should be 0.
- If any of the enabled channels CHn\_TX\_8B10B or CHn\_RX\_8B10B are DISABLED then this bit should be 1 or else this bit should be set to 0.
- The TDRV\_AMP attribute has been replaced by TDRV. When the .lpc file is opened, the software will automatically replace the TDRV\_AMP attribute with TDRV. If the user attempts to re-compile without re-generating the PCS module, the software will generate an error in the auto-make.log file. Users should either edit the .txt file or re-generate the PCS module using IPexpress.
- refclk\_to/from\_nq signals are for internal use only.

**Table 8-106. Protocol-Specific SERDES Setup Options**

Protocol	DATARATE	DATARATE Range	REFCK Multiplier	Data Width	RX Equalization <sup>1</sup>
GbE	1.25G	MED	10x, 20x, 25x	8, 16	DISABLE, MID_MED, LONG_MED
SGMII	1.25G	MED	10x, 20x, 25x	8	
PCI Express	2.5G	HIGH	25x, 20x	8, 16	MID_LOW, DISABLE, MID_HIGH, LONG_HIGH
XAUI	3.125	HIGH	20x, 10X	16	
CPRI	614.4M, 1.2288G, 2.4576G, 3.072G	MEDLOW, MED, MEDHIGH, HIGH	10x, 20x, 25x	8, 16	DISABLE, MID_LOW, MID_MED, MID_HIGH, LONG_LOW, LONG_MED, LONG_HIGH
SDI	270M, 1.485G, 2.97G	LOW, MED, HIGH	20x	10, 20	DISABLE, MID_LOW, MID_MED, MID_HIGH, LONG_LOW, LONG_MED, LONG_HIGH
G8B10B	ANY_VALUE	LOWLOW, LOW, MEDLOW, MED, MEDHIGH, HIGH	10x, 20x, 25x	8, 16	DISABLE, MID_LOW, MID_MED, MID_HIGH, LONG_LOW, LONG_MED, LONG_HIGH
8-Bit SERDES			8x, 16x	8, 16	
10-Bit SERDES			10x, 20x, 25x	8, 16	
USER_DEF			8x, 10x, 16x, 20x, 25x	8, 16	

1. Rx Equalization is based on user signal traces. MID ~20", LONG ~40". LOW/MED/HIGH is amount of equalization user needs to test out.



---

## Appendix D. Lattice Diamond Overview

This appendix discusses the use of Lattice Diamond design software for projects that include the LatticeECP2M SERDES/PCS module.

For general information about the use of Lattice Diamond, refer to the [Lattice Diamond User Guide](#).

If you have been using ispLEVER software for your FPGA design projects, Lattice Diamond may look like a big change. But if you look closer, you will find many similarities because Lattice Diamond is based on the same toolset and work flow as ispLEVER. The changes are intended to provide a simpler, more integrated, and more enhanced user interface.

### Converting an ispLEVER Project to Lattice Diamond

Design projects created in ispLEVER can easily be imported into Lattice Diamond. The process is automatic except for the ispLEVER process properties, which are similar to the Diamond strategy settings, and PCS modules. After importing a project, you need to set up a strategy for it and regenerate any PCS modules.

### Importing an ispLEVER Design Project

Make a backup copy of the ispLEVER project or make a new copy that will become the Diamond project.


1. In Diamond, choose **File > Open > Import ispLEVER Project**.
2. In the ispLEVER Project dialog box, browse to the project's .syn file and open it.
3. If desired, change the base file name or location for the Diamond project. If you change the location, the new Diamond files will go into the new location, but the original source files will not move or be copied. The Diamond project will reference the source files in the original location.

The project files are converted to Diamond format with the default strategy settings.

### Adjusting PCS Modules

PCS modules created with IPexpress have an unusual file structure and need additional adjustment when importing a project from ispLEVER. There are two ways to do this adjustment. The preferred method is to regenerate the module in Diamond. However this may upgrade the module to a more recent version. An upgrade is usually desirable but if, for some reason, you do not want to upgrade the PCS module, you can manually adjust the module by copying its .txt file into the implementation folder. If you use this method, you must remember to copy the .txt file into any future implementation folders.

### Regenerate PCS Modules

1. Find the PCS module in the Input Files folder of File List view. The module may be represented by an .lpc, .v, or .vhd file.
2. If the File List view shows the Verilog or VHDL file for the module, and you want to regenerate the module, import the module's .lpc file:
  - a. In the File List view, right-click the implementation folder (  ) and choose **Add > Existing File**.
  - b. Browse for the module's .lpc file, **<module\_name>.lpc**, and select it.
  - c. Click **Add**. The .lpc file is added to the File List view.
  - d. Right-click the module's Verilog or VHDL file and choose **Remove**.
3. In File List, double-click the module's .lpc file. The module's IPexpress dialog box opens.
4. In the bottom of the dialog box, click **Generate**. The Generate Log tab is displayed. Check for errors and close.

In File List, the .lpc file is replaced with an .ipx file. The IPexpress manifest (.ipx) file is new with Diamond. The .ipx file keeps track of the files needed for complex modules.

### Using IPexpress with Lattice Diamond

Using IPexpress with Lattice Diamond is essentially the same as with ispLEVER.

The configuration GUI tabs are all the same except for the Generation Options tab. Figure 8-57 shows the Generation Options tab window.

Figure 8-57. Generation Options Tab

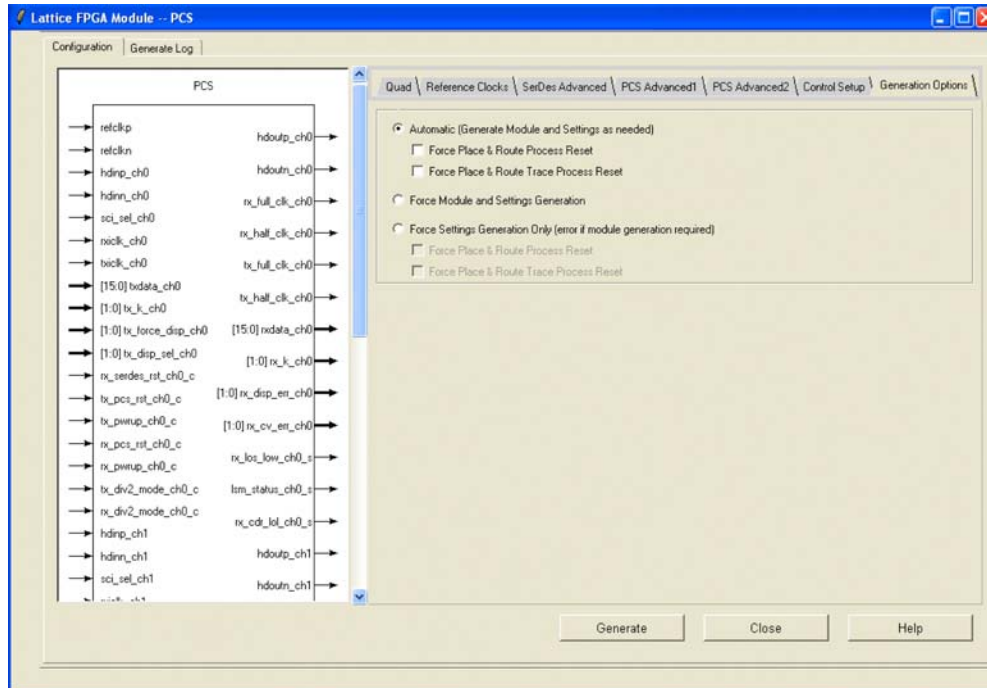


Table 8-107. SERDES\_PCS GUI Attributes – Generation Options Tab

GUI Text	Description
Automatic	Automatically generates the HDL and configuration(.txt) files as needed. Some changes do not require regenerating both files.
Force Module and Settings Generation	Generates both the HDL and configuration files.
Force Settings Generation Only	Generates only the attributes file. You get an error message if the HDL file also needs to be generated.
Force Place & Route Process Reset	Resets the Place & Route Design process, forcing it to be run again with the newly-generated PCS module.
Force Place & Route Trace Process Reset	Resets the Place & Route Trace process, forcing it to be run again with the newly-generated PCS module.

Note:  
Automatic is set as the default option. If either Automatic or Force Settings Generation Only and no sub-options (Process Reset Options) are checked and the HDL module is not generated, the reset pointer is set to Bitstream generation automatically.

After the Generation is finished, the reset marks in the process window will be reset accordingly.

---

## Creating a New Simulation Project Using Simulation Wizard

This section describes how to use the Simulation Wizard to create a simulation project (.spf) file so you can import it into a standalone simulator.

1. In Project Navigator, click **Tools > Simulation Wizard**. The Simulation Wizard opens.
2. In the Preparing the Simulator Interface page click **Next**.
3. In the Simulator Project Name page, enter the name of your project in the Project Name text box and browse to the file path location where you want to put your simulation project using the Project Location text box and Browse button.

When you designate a project name in this wizard page, a corresponding folder will be created in the file path you choose. Click **Yes** in the pop-up dialog that asks you if you wish to create a new folder.

4. Click either the Active-HDL® or ModelSim® simulator check box and click **Next**.
5. In the Process Stage page choose which type of Process Stage of simulation project you wish to create. Valid types are RTL, Post-Synthesis Gate-Level, Post-Map Gate-Level, and Post-Route Gate-level+Timing. Only those process stages that are available are activated.

Note that you can make a new selection for the current strategy if you have more than one defined in your project.

The software supports multiple strategies per project implementation which allow you to experiment with alternative optimization options across a common set of source files. Since each strategy may have been processed to different stages, this dialog allows you to specify which stage you wish to load.

6. In the Add Source page, select from the source files listed in the Source Files list box or use the browse button on the right to choose another desired source file. Note that if you wish to keep the source files in the local simulation project directory you just created, check the **Copy Source to Simulation Directory** option.
7. Click **Next** and a Summary page appears and provides information on the project selections including the simulation libraries. By default, the Run Simulator check box is enabled and will launch the simulation tool you chose earlier in the wizard in the Simulator Project Name page.
8. Click **Finish**.

The Simulation Wizard Project (.spf) file and a simulation script DO file are generated after running the wizard. You can import the DO file into your current project if desired. If you are using Active-HDL, the wizard will generate an .ado file and if you are using ModelSim, it creates and .mdo file.

*Note: PCS configuration file, (.txt) must be added in step 6.*

## Introduction

The LatticeECP3™ sysIO™ buffers give the designer the ability to easily interface with other devices using advanced system I/O standards. This technical note describes the sysIO standards available and how to implement them using Lattice's ispLEVER® design software.

## sysIO Buffer Overview

The LatticeECP3 sysIO interface contains multiple Programmable I/O Cell (PIC) blocks. Each PIC contains two Programmable I/Os (PIO), PIOA and PIOB, that are connected to their respective sysIO buffers. Two adjacent PIOs can be joined to provide a differential I/O pair (labeled as "T" and "C").

Each PIO includes a sysIO buffer and I/O logic (IOLOGIC). The LatticeECP3 sysIO buffers support a variety of single-ended and differential signaling standards. The sysIO buffer also supports the DQS strobe signal that is required for interfacing with the DDR memory. One of every 12 PIOs in the LatticeECP3 contains a delay element to facilitate the generation of DQS signals. The DQS signal from the bus is used to strobe the DDR data from the memory into input register blocks. For more information on the architecture of the sysIO buffer, refer to the [LatticeECP3 Family Data Sheet](#).

The IOLOGIC includes input, output and tristate registers that implement both single data rate (SDR) and double data rate (DDR) applications along with the necessary clock and data selection logic. Programmable delay lines and dedicated logic within the IOLOGIC are used to provide the required shift to incoming clock and data signals and the delay required by DQS inputs in DDR memory. The DDR implementation in the IOLOGIC and the DDR memory interface support are discussed in more detail in TN1180, [LatticeECP3 High-Speed I/O Interface](#).

## Supported sysIO Standards

The LatticeECP3 sysIO buffer supports both single-ended and differential standards. Single-ended standards can be further subdivided into internally ratioed standard such as LVCMOS, LVTTTL and PCI; and externally referenced standards such as HSTL and SSTL. The buffers support the LVTTTL, LVCMOS 1.2, 1.5, 1.8, 2.5 and 3.3V standards. In the LVCMOS and LVTTTL modes, the buffer has individually configurable options for drive strength, bus maintenance (weak pull-up, weak pull-down, or a bus-keeper latch). Other single-ended standards supported include SSTL and HSTL. Differential standards supported include LVDS, RSDS, BLVDS, LVPECL, differential SSTL and differential HSTL. LatticeECP3 also support mini-LVDS, PPLVDS (Point-to-Point LVDS) and TRLVDS (Transition Reduced LVDS). Table 1 lists the sysIO standards supported in the LatticeECP3 devices.

**Table 9-1. Supported Input Standards**

Input Standard	V <sub>REF</sub> (Nominal)	V <sub>CCIO</sub> (Nominal)
<b>Single Ended Interfaces</b>		
LVTTTL	—	3.3
LVC MOS33	—	3.3 <sup>1</sup>
LVC MOS25	—	2.5 <sup>1</sup>
LVC MOS18	—	1.8
LVC MOS15	—	1.5
LVC MOS12	—	—
PCI33	—	—
HSTL18 Class I, II	0.9	—
HSTL15 Class I	0.75	—
SSTL33 Class I, II	1.5	3.3, 2.5
SSTL25 Class I, II	1.25	3.3, 2.5, 1.8
SSTL18 Class I, II	0.9	—
SSTL15	0.75	—
<b>Differential Interfaces</b>		
Differential SSTL33 Class I, II	—	3.3, 2.5
Differential SSTL18 Class I, II	See Note 2	—
Differential SSTL25 Class I, II	—	3.3, 2.5, 1.8
Differential HSTL15 Class I	—	—
Differential HSTL18 Class I, II	—	—
Differential SSTL 15	See Note 2	—
LVDS	—	—
Transition Reduced LVDS	—	3.3
LVPECL	—	3.3
Bus LVDS	—	—
MLVDS	—	—

1. For LVTTTL33, LVC MOS33 and LVC MOS25, if PCICLAMP is OFF, they can be used independently of V<sub>CCIO</sub> in the top banks.
2. VREF is required when using Differential SSTL to interface to DDR memory.

**Table 9-2. Supported Output Standards**

Output Standard	Drive	V <sub>CCIO</sub> (Nominal)
<b>Single Ended Interfaces</b>		
LVTTTL	4mA, 8mA, 12mA, 16mA, 20mA	3.3
LVC MOS33	4mA, 8mA, 12mA, 16mA, 20mA	3.3
LVC MOS25	4mA, 8mA, 12mA, 16mA, 20mA	2.5
LVC MOS18	4mA, 8mA, 12mA, 16mA, 20mA <sup>3</sup>	1.8
LVC MOS15	4mA, 8mA, 12mA <sup>3</sup> , 16mA <sup>3</sup> , 20mA <sup>3</sup>	1.5
LVC MOS12	2mA, 4mA <sup>3</sup> , 6mA, 8mA <sup>3</sup> , 12mA <sup>3</sup> , 16mA <sup>3</sup> , 20mA <sup>3</sup>	1.2
PCI33	N/A	3.3
HSTL18 Class I	8mA, 12mA	1.8
HSTL18 Class II	N/A	1.8
HSTL15 Class I	4mA, 8mA	1.5
SSTL33 Class I, II	N/A	3.3
SSTL25 Class I	8mA, 12mA	2.5
SSTL25 Class II	16mA, 20mA	2.5
SSTL18 Class I	N/A	1.8
SSTL18 Class II	8mA, 12mA	1.8
SSTL15	10mA	1.5
<b>Differential Interfaces</b>		
Differential HSTL18 Class I	8mA, 12mA	1.8
Differential HSTL18 Class II	N/A	1.8
Differential HSTL15 Class I	4mA, 8mA	1.5
Differential SSTL33 Class I, II	N/A	3.3
Differential SSTL25 Class I	8mA, 12mA	2.5
Differential SSTL25 Class II	16mA, 20mA	2.5
Differential SSTL18 Class I	N/A	1.8
Differential SSTL18 Class II	8mA, 12mA	1.8
Differential SSTL15	10mA	1.5
LVDS	N/A	2.5
Point-to-Point LVDS (PPLVDS)	N/A	2.5, 3.3
RSDS, RSDSE <sup>2</sup>	N/A	2.5
Mini-LVDS <sup>1</sup>	N/A	2.5
MLVDS <sup>2</sup>	N/A	2.5
BLVDS <sup>2</sup>	N/A	2.5
LVPECL <sup>2</sup>	N/A	3.3

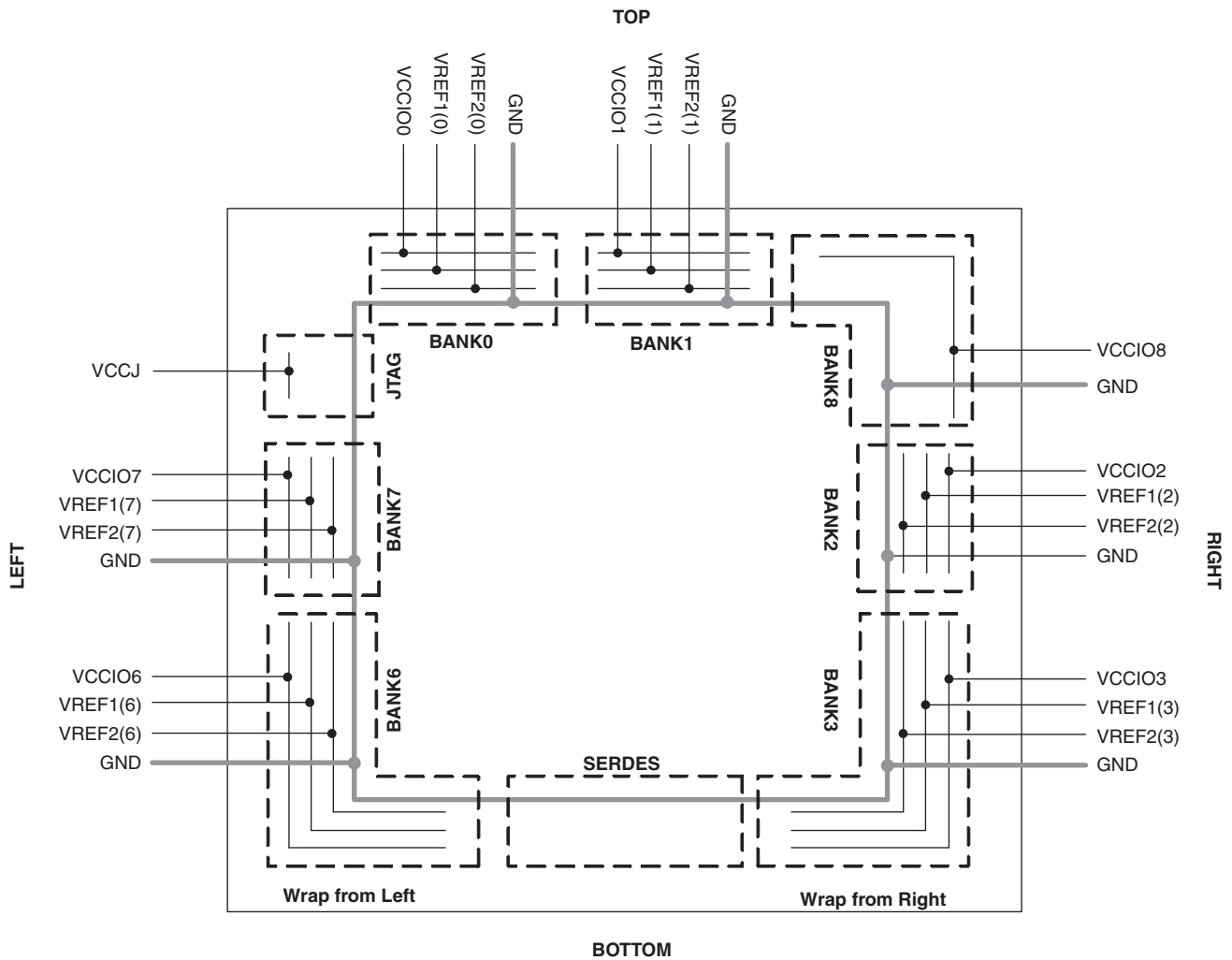
1. Multiple Drive supported using DiffDrive and MultDrive.
2. Emulated with LVC MOS drivers and external resistors.
3. This drive strength is only available when the output is configured as open-drain.

## sysIO Banking Scheme

LatticeECP3 devices have six general-purpose programmable sysIO banks and a seventh configuration bank. Each of the six general-purpose sysIO banks has a V<sub>CCIO</sub> supply voltage and two reference voltages, VREF1 and VREF2. Figure 9-1 shows the six general-purpose banks and the configuration bank with associated supplies. Bank 8 is a bank dedicated to configuration logic and has seven dedicated configuration I/Os and 14 multiplexed configuration I/Os. Bank 8 has the power supply pads (V<sub>CCIO</sub> and V<sub>CCAUX</sub>) but does not have VREF pads.

On the top and bottom banks, the sysIO buffer pair consists of two single-ended output drivers and two sets of single-ended input buffers (both ratioed and referenced). The left and right sysIO buffer pair consists of two single-ended output drivers and two sets of single-ended input buffers (both ratioed and referenced). The referenced input buffer can also be configured as a differential input. In 50% of the pairs, there is also one differential output driver. The two pads in the pair are described as “true” and “comp”, where the true pad is associated with the positive side of the differential input buffer and the comp (complementary) pad is associated with the negative side of the differential input buffer.

Figure 9-1. sysIO Banking



### V<sub>CCIO</sub> (1.2V/1.5V/1.8V/2.5V/3.3V)

There are a total of six V<sub>CCIO</sub> supplies, V<sub>CCIO0</sub> - V<sub>CCIO8</sub>. Each bank has a separate V<sub>CCIO</sub> supply that powers the single-ended output drivers and the ratioed input buffers such as LVTTTL, LVCMOS, and PCI. LVTTTL, LVCMOS3.3, LVCMOS2.5 and LVCMOS1.2 inputs also have fixed threshold options allowing them to be placed in any bank. Tables 9-4 and 9-6 list the allowed mixed-voltage support in a given bank. The V<sub>CCIO</sub> voltage applied to the bank determines the ratioed input standards that can be supported in that bank. It is also used to power the differential output drivers. In addition, V<sub>CCIO8</sub> is used to supply power to the sysCONFIG™ signals.

### V<sub>CCAUX</sub> (3.3V)

In addition to the bank V<sub>CCIO</sub> supplies, LatticeECP3 devices have a V<sub>CC</sub> core logic power supply, and a V<sub>CCAUX</sub> auxiliary supply that powers the differential and referenced input buffers. V<sub>CCAUX</sub> is used to supply I/O reference voltage requiring 3.3V to satisfy the common-mode range of the drivers and input buffers.

### V<sub>CCJ</sub> (1.2V/1.5V/1.8V/2.5V/3.3V)

The JTAG pins have a separate V<sub>CCJ</sub> power supply that is independent of the bank V<sub>CCIO</sub> supplies. V<sub>CCJ</sub> determines the electrical characteristics of the LVCMOS JTAG pins, both the output high level and the input threshold.

Table 9-3 contains a summary of the required power supplies.

**Table 9-3. Power Supplies**

Power Supply	Description	Value <sup>1</sup>
V <sub>CC</sub>	Core power supply	1.2V
V <sub>CCIO</sub>	Power supply for the I/O and configuration banks	1.2V/1.5V/1.8V/2.5V/3.3V
V <sub>CCAUX</sub>	Auxiliary power supply	3.3V
V <sub>CCJ</sub>	Power supply for JTAG pins	1.2V/1.5V/1.8V/2.5V/3.3V

1. Refer to the [LatticeECP3 Family Data Sheet](#) for recommended min. and max. values.

### Input Reference Voltage (V<sub>REF1</sub>, V<sub>REF2</sub>)

Each bank can support up to two separate V<sub>REF</sub> input voltages, V<sub>REF1</sub> and V<sub>REF2</sub>, that are used to set the threshold for the referenced input buffers. The location of these V<sub>REF</sub> pins is pre-determined within the bank. These pins can be used as regular I/Os if the bank does not require a V<sub>REF</sub> voltage.

#### VREF1 for DDR Memory Interface

When interfacing to DDR memory, the V<sub>REF1</sub> input must be used as the reference voltage for the DQS and DQ input from the memory. A voltage divider between V<sub>REF1</sub> and GND is used to generate an on-chip reference voltage that is used by the DQS transition detector circuit. This voltage divider is only present on V<sub>REF1</sub> it is not available on V<sub>REF2</sub>. For further information on the DQS transition detect logic and its implementation, refer to TN1180, [LatticeECP3 High-Speed I/O Interface](#). When not used as VREF, these predefined voltage reference pins are available as user I/O pins.

For DDR1 memory interfaces, the VREF1 should be connected to 1.25V since only SSTL25 signaling is allowed. For DDR2 memory interfaces, VREF1 should be connected to 0.9V since only SSTL18 signaling is allowed. For DDR3 memory interfaces, VREF1 should be connected to 0.75V since only SSTL15 signaling is allowed.

### VTT Termination Voltage

The VTT termination voltage on LatticeECP3 device is used for the referenced standard termination and common mode differential termination. These termination voltage pins are available on the left and right of the device only. Use of VTT is optional, these pins should be left floating if termination to VTT is not required. The allowable range for VTT is from 0.5V to 1.25V, independent of the value of V<sub>CCIO</sub>. The user decides the best termination voltage to apply to VTT. Many applications will choose VTT to be nominally equal to the switching threshold of the interface standard being used, with a tolerance of +/- 5% (and this is usually equal to half of the driver supply voltage). VTT Termination can be dynamic for bidirectional pins (enabled when output buffer is put in tristate) or static (always on).

### Hot Socketing Support

The I/Os located on the top and bottom sides of the device are fully hot socketable. The top side of the device simultaneously supports hot-socketing, mixed voltage support within a bank and programmable clamp diodes for supporting PCI. The I/Os located on the left and right sides of the device do not support hot socketing. See the [LatticeECP3 Family Data Sheet](#) for hot socketing (IDK) requirements.



## Mixed Voltage Support in a Bank

The LatticeECP3 sysIO buffer is connected to three parallel ratioed input buffers. These three parallel buffers are connected to  $V_{CCIO}$ ,  $V_{CCAUX}$  and to  $V_{CC}$ , giving support for thresholds that track with  $V_{CCIO}$  as well as fixed thresholds for 3.3V ( $V_{CCAUX}$ ) and 1.2V ( $V_{CC}$ ) inputs. This allows the input threshold for ratioed buffers to be assigned on a pin-by-pin basis, rather than have it track with  $V_{CCIO}$ . This option is available for all 1.2V, 2.5V and 3.3V ratioed inputs and is independent of the bank  $V_{CCIO}$  voltage on the top banks when PCICLAMP is OFF. In the left and right banks, the PCICLAMP is always enabled to clamp any currents when  $V_{INPUT}$  is higher than  $V_{CCIO}$ . Hence, only 1.2 inputs and 2.5 inputs are allowed independent of  $V_{CCIO}$  as long as the  $V_{INPUT}$  is less than  $V_{CCIO}$ . For example, if the bank  $V_{CCIO}$  is 1.8V and PCICLAMP is OFF, it is possible to have 1.2V and 3.3V ratioed input buffers with fixed thresholds, as well as 2.5V ratioed inputs with tracking thresholds on the top bank. On the left and right banks, when  $V_{CCIO}$  is 1.8V, it is possible to have only 1.2V with fixed thresholds. But if the  $V_{CCIO}$  on the left and right sides is 3.3V, it is possible to have a 1.2V input with fixed thresholds as well as 2.5V with tracking thresholds.

Prior to device configuration, the ratioed input thresholds track the bank  $V_{CCIO}$ . This option only takes effect after configuration. Output standards within a bank are always set by  $V_{CCIO}$  but can drive a lower output standard into a device that is tolerant up to that  $V_{CCIO}$ . Tables 9-4 and 9-5 shows the sysIO standards that can be mixed in the same bank.

**Table 9-4. Mixed Voltage Support in Top Banks**

$V_{CCIO}$	Input sysIO Standards <sup>1, 2, 3, 4, 5</sup>					Output sysIO Standards <sup>6</sup>				
	1.2V	1.5V	1.8V	2.5V	3.3V	1.2V	1.5V	1.8V	2.5V	3.3V
1.2V	Yes			Yes	Yes	Yes				
1.5V	Yes	Yes		Yes	Yes		Yes			
1.8V	Yes		Yes	Yes	Yes			Yes		
2.5V	Yes			Yes	Yes				Yes	
3.3V	Yes			Yes	Yes					Yes

- Mixed voltage input support is available on the top banks only when PCICLAMP is OFF
- All differential input buffers except LVPECL33 and TRLVDS can be supported in banks independent of  $V_{CCIO}$ . LVPECL33 can be placed on top side independent of  $V_{CCIO}$  when PCICLAMP is OFF.
- 1.5V and 1.8V HSTL and SSTL reference inputs can be supported on banks with any  $V_{CCIO}$ .
- 2.5V SSTL reference inputs can be supported on banks with  $V_{CCIO}$  set to 1.8V, 2.5V or 3.3V.
- 3.3V SSTL reference inputs can be supported on banks with  $V_{CCIO}$  set to 2.5V or 3.3V.
- When output is configured as open drain it can be placed in bank independent of  $V_{CCIO}$ .

**Table 9-5. Mixed Voltage Support in Left and Right Banks**

$V_{CCIO}$	Input sysIO Standards <sup>1, 2, 3, 4</sup>					Output sysIO Standards <sup>5</sup>				
	1.2V	1.5V	1.8V	2.5V	3.3V	1.2V	1.5V	1.8V	2.5V	3.3V
1.2V	Yes					Yes				
1.5V	Yes	Yes					Yes			
1.8V	Yes		Yes					Yes		
2.5V	Yes			Yes					Yes	
3.3V	Yes			Yes	Yes					Yes

- All differential input buffers except LVPECL33 and TRLVDS can be supported on banks with  $V_{CCIO}$  set to 2.5V or 3.3V. LVPECL and TRLVDS require  $V_{CCIO}$  of 3.3V. If the  $V_{CCIO}$  is set to 1.8V or 1.5V, this reduces the  $V_{CM}$  (max) and  $V_{IN}$  (max) to approximately 1.7V as the PCICLAMP is always enabled on this side when the  $V_{INPUT} > V_{CCIO}$  of the bank.
- 1.5V and 1.8V HSTL and SSTL reference inputs can be supported on banks with any  $V_{CCIO}$ .
- 2.5V SSTL reference inputs can be supported on banks with  $V_{CCIO}$  set to 1.8V, 2.5V or 3.3V.
- 3.3V SSTL reference inputs can be supported on banks with  $V_{CCIO}$  set to 2.5V or 3.3V.
- When output is configured as open drain it can be placed in bank independent of  $V_{CCIO}$ .

## sysIO Standards Supported Per Bank

Table 9-6. I/O Standards Supported per Bank

Description	Top Side	Right Side	Bottom Side	Left Side
Types of I/O Buffers	Single-ended	Single-ended and Differential	Single-ended	Single-ended and Differential
Single-Ended Standards Outputs	LVTTTL LVC MOS33 LVC MOS25 LVC MOS18 LVC MOS15 LVC MOS12 SSTL15 SSTL18 Class I, II SSTL25 Class I, II SSTL33 Class I, II HSTL15 Class I HSTL18_I, II	LVTTTL LVC MOS33 LVC MOS25 LVC MOS18 LVC MOS15 LVC MOS12 SSTL15 SSTL18 Class I, II SSTL25 Class I, II SSTL33 Class I, II HSTL15 Class I HSTL18 Class I, II	LVTTTL LVC MOS33 LVC MOS25 LVC MOS18 LVC MOS15 LVC MOS12 SSTL15 SSTL18 Class I, II SSTL2 Class I, II SSTL3 Class I, II HSTL15 Class I HSTL18 Class I, II	LVTTTL LVC MOS33 LVC MOS25 LVC MOS18 LVC MOS15 LVC MOS12 SSTL15 SSTL18 Class I, II SSTL2 Class I, II SSTL3 Class I, II HSTL15 Class I HSTL18 Class I, II
Differential Standards Outputs	LVC MOS33D  SSTL15D SSTL18D Class I, II SSTL25D Class I, II SSTL33D Class I, II HSTL15D Class I HSTL18D Class I, II  LVDS25E <sup>1</sup> LVPECL <sup>1</sup> BLVDS <sup>1</sup> RSDSE <sup>1</sup>	LVC MOS33D  SSTL15D SSTL18D Class I, II SSTL25D Class I, II SSTL33D Class I, II HSTL15D Class I HSTL18D Class I, II  LVDS <sup>2,3</sup> RSDS <sup>2</sup> Mini-LVDS <sup>2</sup> PPLVDS <sup>2</sup> (point-to-point) LVDS25E <sup>1</sup> LVPECL <sup>1</sup> BLVDS <sup>1</sup> RSDSE <sup>1</sup>	LVC MOS33D  SSTL15D SSTL18D Class I, II SSTL25D Class I, II, SSTL33D Class I, II HSTL15D Class I HSTL18D Class I, II  LVDS25E <sup>1</sup> LVPECL <sup>1</sup> BLVDS <sup>1</sup> RSDSE <sup>1</sup>	LVC MOS33D  SSTL15D SSTL18D Class I, II SSTL25D Class I, II, SSTL33D_I, II HSTL15D Class I HSTL18D Class I, II  LVDS <sup>2</sup> RSDS <sup>2</sup> Mini-LVDS <sup>2</sup> PPLVDS <sup>2</sup> (point-to-point) LVDS25E <sup>1</sup> LVPECL <sup>1</sup> BLVDS <sup>1</sup> RSDSE <sup>1</sup>
Inputs	All Single-ended and Differential TRLVDS (Transition Reduced LVDS)	All Single-ended, Differential	All Single-ended, Differential	All Single-ended, Differential
Clock Inputs	All Single-ended, Differential	All Single-ended, Differential	All Single-ended, Differential	All Single-ended, Differential
Hot Socketing	Yes	No	Yes	No
Equalization on Inputs	No <sup>4</sup>	Yes <sup>4</sup>	No	Yes
ISI Correction	For DDR3 memory	For DDR3 memory	For DDR3 memory	For DDR3 memory
On Chip Termination	No	On-Chip Parallel Termination  On-Chip Differential Termination	No	On-Chip Parallel Termination  On-Chip Differential Termination
PCI Support	PCI33 with or without clamp <sup>5</sup>	PCI33 with clamp	PCI33 with clamp	PCI33 with clamp

1. These differential standards are implemented by using a complementary LVC MOS driver with the external resistor pack.

2. Available on 50% of the I/Os in the bank.

3. I/Os in Bank 8 are shared with sysCONFIG pins and do not support true LVDS and DDR registers.

4. I/Os in Bank 8 do not support equalization.

5. I/Os in Bank 8 do not have a programmable clamp setting. PCI clamp is always on in Bank 8.

## sysIO Buffer Configurations

This section describes the various sysIO features available on the LatticeECP3 FPGAs.

### Bus Maintenance Circuit

Each of the LVCMOS, LVTTTL and PCI types of inputs has a weak pull-up, weak pull-down and weak bus keeper capability. The pull-up and pull-down settings offer a fixed characteristic which is useful in creating wired logic such as wired ORs. However, current can be slightly higher than other options depending on the signal state. The bus keeper option latches the signal in the last driven state, holding it at a valid level with minimal power dissipation. You can also choose to turn off the bus maintenance circuitry, minimizing power dissipation and input leakage. Note that in this case it is important to ensure that inputs are driven to a known state to avoid unnecessary power dissipation in the input buffer.

On the outputs, the weak pull-ups are on at all times. Users have the option to turn off the pull-up setting in the software.

### Programmable Drive

Each LVCMOS or LVTTTL, as well as some of the referenced (SSTL and HSTL) output buffers, has a programmable drive strength option. This option can be set for each I/O independently. The drive strength settings available are 2mA, 4mA, 6mA, 8mA, 12mA, 16mA and 20mA. Actual options available vary by I/O voltage. The user must consider the maximum allowable current per bank and the package thermal limit current when selecting the drive strength. Table 9-7 shows the available drive settings for each of the output standards.

**Table 9-7. Programmable Drive Values for Single-Ended Buffers**

Single-Ended I/O Standard	Programmable Drive (mA)
HSTL15_I/ HSTL15D_I	4, 8
HSTL18_I/ HSTL18D_I	8, 12
SSTL25_I/ SSTL25D_I	8, 12
SSTL25_II/ SSTL25D_II	16, 20
SSTL18_II/SSTL18D_II	8, 12
LVCMOS12 (with PCI Clamp OFF)	4, 8, 12, 16, 20
LVCMOS12 (with PCI Clamp ON)	2, 6
LVCMOS15 (with PCI Clamp OFF)	4, 8, 12, 16, 20
LVCMOS15 (with PCI Clamp ON)	4, 8
LVCMOS18 (with PCI Clamp OFF)	4, 8, 12, 16, 20
LVCMOS18 (with PCI Clamp ON)	4, 8, 12, 16
LVCMOS25	4, 8, 12, 16, 20
LVCMOS33	4, 8, 12, 16, 20
LVTTTL	4, 8, 12, 16, 20

### Programmable Slew Rate

Each LVCMOS or LVTTTL output buffer pin also has a programmable output slew rate control that can be configured for either low noise or high-speed performance. Each I/O pin has an individual slew rate control. This allows designers to specify slew rate control on a pin-by-pin basis. This slew rate control affects both the rising edge and the falling edges.

### Open Drain Control

All LVCMOS and LVTTTL output buffers can be configured to function as open drain outputs. The user can implement an open drain output by turning on the OPENDRAIN attribute in the software.

---

## Differential SSTL and HSTL Support

The single-ended driver associated with the complementary “C” pad can optionally be driven by the complement of the data that drives the single-ended driver associated with the true pad. This allows a pair of single-ended drivers to be used to drive complementary outputs with the lowest possible skew between the signals. This is used for driving complementary SSTL and HSTL signals (as required by the differential SSTL and HSTL clock inputs on synchronous DRAM and synchronous SRAM devices, respectively). This capability is also used in conjunction with off-chip resistors to emulate LVPECL, and BLVDS output drivers.

## PCI Support with Programmable PCICLAMP

Each sysIO buffer can be configured to support PCI33. The buffers on the top of the device have an optional PCI clamp diode that may optionally be specified in the ispLEVER design tools. The programmable PCICLAMP can be turned ON or OFF. This option is available on each I/O independently only on the top side banks.

For the other three sides of the device, the PCICLAMP is always ON.

## Differential I/O Support

50% of the sysIO buffer pairs on the left and right edges contain a differential output driver that can optionally drive the pads in the pair. The standards support on these differential output pairs is as follows:

- LVDS
- Point to Point LVDS (PPLVDS)
- Mini-LVDS
- RSDS

All the other pins on all the sides of the device can support Emulated Differential standards using complementary LVCMOS drivers with external resistors. The standards supported using differential output pairs is as follows:

- BLVDS
- LVDS25E
- RSDSE
- LVPECL

The [LatticeECP3 Family Data Sheet](#) lists the LVCMOS drivers and external resistor requirements to implement these emulated standards. The data sheet also lists the electrical specifications supported for all differential standards.

## Differential SSTL and HSTL

All single-ended sysIO buffers pairs support differential SSTL and HSTL. Refer to the [LatticeECP3 Family Data Sheet](#) for a detailed description of the Differential HSTL and SSTL implementations.

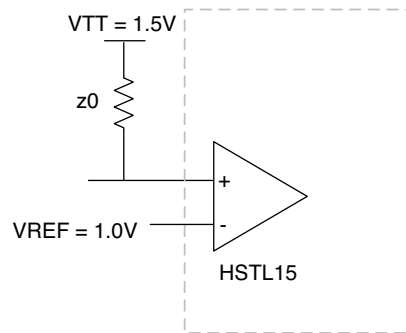
## Differential LVCMOS33

All single-ended sysIO buffer pairs also support the LVCMOS33D (Differential) standard with configurable drive strength and slew settings. This generic 3.3V differential buffer allows the user to implement any type of 3.3V differential buffer by configuring the drive strength and calculating the external resistor values as per the application requirements.

## GTL+ Input Support

GTL+ inputs can be supported using either the SSTL15 or HSTL15\_I input standard with VREF set to 1.0V and external VTT termination to 1.5V. GTL+ inputs implemented using this method can support the maximum speed listed for the SSTL and HSTL standards in the [LatticeECP3 Family Data Sheet](#). GTL+ outputs are not supported in the LatticeECP3 device.

Figure 9-2. GTL+ Input Buffer Emulation Using HSTL15 Input



### On-Chip Termination

LatticeECP3 devices support on-chip Parallel Termination to VTT as well Common mode differential termination using the VTT pin. On-chip termination is available on the left and right sides of the device. The VTT pin should be left to float when using common mode differential termination. Termination can be set to be dynamic for bidirectional buffers where the termination is active only when the output buffer is disabled through the tristate control. External termination to VTT should be used when implementing the DDR2 and DDR3 memory interfaces.

### Equalization Setting

Equalization filtering is available for single-ended inputs on both true and complementary I/Os, and for differential inputs on the true I/Os. Equalization is required to compensate for the difficulty of sampling alternating logic transitions with a relatively slow slew rate. It is useful for the input DDR modes used in DDR3 memory and fast SPI4.2 mode signaling. It is available on the left and right sides.

Equalization acts as a tunable filter, with settings determining the level of correction. There are four settings available: Zero (none), One, Two and Three. The equalization logic resides in the I/O buffers. Each I/O can have a unique equalization setting within a DQS-12 group for DDR3 memory.

### Software sysIO Attributes

sysIO attributes can be specified in the HDL, using the Preference Editor GUI or in the ASCII Preference file (.prf) directly. The appendices of this document provide examples of how these can be assigned using each of the methods described above. This section describes each of these attributes in detail.

### IO\_TYPE

This attribute is used to set the sysIO standard for an I/O. The  $V_{CCIO}$  required to set these I/O standards are embedded in the attribute name itself. There is no separate attribute to set the VCCIO requirements. Table 9-8 lists the available I/O types.

**Table 9-8. IO\_TYPE Attribute Values**

sysIO Signaling Standard	IO_TYPE
DEFAULT	LVCMOS25
LVDS 2.5V	LVDS25 <sup>2</sup>
Point to Point LVDS	PPLVDS <sup>2</sup>
Mini-LVDS	MINILVDS <sup>2</sup>
RSDS	RSDS <sup>2</sup>
Transition Reduced LVDS	TRLVDS <sup>3</sup>
Emulated LVDS 2.5V	LVDS25E <sup>1</sup>
Bus LVDS 2.5V	BLVDS25 <sup>1</sup>
LVPECL 3.3V	LVPECL33 <sup>1</sup>
Emulated RSDS	RSDSE <sup>1</sup>
MLVDS	MLVDS
HSTL18 Class I and II	HSTL18_I, HTSL18_II
Differential HSTL 18 Class I and II	HSTL18D_I HSTL18D_II
HSTL 15 Class I	HSTL15_I
Differential HSTL 15 Class I	HSTL15D_I
SSTL 33 Class I and II	SSTL33_I, SSTL33_II
Differential SSTL 33 Class I and II	SSTL33D_I SSTL33D_II
SSTL 25 Class I and II	SSTL25_I SSTL25_II
Differential SSTL 25 Class I and II	SSTL25D_I SSTL25D_II
SSTL 18 Class I and II	SSTL18_I SSTL18_II
Differential SSTL 18 Class I and II	SSTL18D_I SSTL18D_II
SSTL 15	SSTL15
LVTTTL	LVTTTL33
3.3V LVC MOS	LVC MOS33
3.3V LVC MOS Differential	LVC MOS33D
2.5V LVC MOS	LVC MOS25
1.8V LVC MOS	LVC MOS18
1.5V LVC MOS	LVC MOS15
1.2V LVC MOS	LVC MOS12
3.3V PCI	PCI33

1. These differential standards are implemented by using the complementary LVC MOS driver and external resistor pack.
2. Supported on 50% of the pairs on the left and right sides of the device.
3. Only inputs supported. Available only on the top side of the device.

## OPENDRAIN

LVC MOS and LVTTTL IO standards can be set to Open Drain configuration by using the OPENDRAIN attribute. When configuring I/Os on the left and right banks to be Open Drain, it is required that the external pull-up be less than the bank  $V_{CCIO}$ .

**Table 9-9. Open Drain Attribute Values**

Attribute	Values	Default
OPENDRAIN	ON, OFF	OFF

## DRIVE

The DRIVE attribute will set the programmable drive strength for the output standards that have programmable drive capability

**Table 9-10. DRIVE Settings**

Output Standard	DRIVE (mA)	Default (mA)
HSTL15_I/ HSTL15D_I	4, 8	8
HSTL18_I/ HSTL18D_I	8, 12	12
SSTL25_I/ SSTL25D_I	8, 12	8
SSTL25_II/ SSTL25D_II	16, 20	16
SSTL18_II/SSTL18D_II	8, 12	12
LVCOS12 (with OPENDRAIN)	2, 6	6
LVCOS12 (without OPENDRAIN)	4, 8, 12, 16, 20	12
LVCOS15 (with OPENDRAIN)	4, 8	8
LVCOS15 (without OPENDRAIN)	4, 8, 12, 16, 20	12
LVCOS18 (with OPENDRAIN)	4, 8, 12, 16	12
LVCOS18 (without OPENDRAIN)	4, 8, 12, 16, 20	12
LVCOS25	4, 8, 12, 16, 20	12
LVCOS33	4, 8, 12, 16, 20	12
LVTTTL	4, 8, 12, 16, 20	12

## DIFFDRIVE

The DIFFDRIVE setting is used to set the differential drive setting for the Mini-LVDS driver when the driver setting needs to be adjusted to support variation in external termination. An I/O bank can have differential outputs with the same DIFFDRIVE setting. Differential outputs with different DIFFDRIVE settings cannot be placed in the same I/O bank.

**Table 9-11. DIFFDRIVE Values**

I/O Standard	MULTDRIVE Values	Default
MINILVDS	1.6, 1.65, 1.7, 1.75, 1.81, 1.87, 1.93, 2.0	1.6
LVDS	1.75	1.75
RSDS	2.0	2.0
PPLVDS	2.0	2.0

## MULTDRIVE

DIFFDRIVE only partially supports variation of Mini-LVDS driver current. Therefore, in addition to DIFFDRIVE, MULTDRIVE settings must be used to adjust the output drive strength of Mini-LVDS.

**Table 9-12. MULTDRIVE Values**

I/O Standard	MULTDRIVE Values	Default
MINILVDS	1x, 2x, 3x, 4x	1x
LVDS	2x	2x
RSDS	1x	1x
PPLVDS	1x	1x

## TERMINATEVTT

This attribute is used to set the on-chip parallel termination to VTT for reference buffer inputs. VTT pins in corresponding banks should be connected externally to the correct level. VTT of the bank should be left floating if this termination is not used.

**Table 9-13. TERMINATEVTT Values**

Attribute	Values	Default
TERMINATEVTT	OFF, 40, 50, 60	OFF

## DIFFRESISTOR

This attribute is used to set the on-chip differential termination using common mode termination to VTT. This on-chip termination is optimized to work primarily for the LVDS I/O type. When the DIFFRESISTOR attribute is set, the VTT pin should be left floating.

**Table 9-14. DIFFRESISTOR Values**

Attribute	Values	Default
DIFFRESISTOR	OFF, 80, 100, 120	OFF

## EQ\_CAL

This attribute is used to set the Equalization setting available on the input pins on the left and right sides of the device. EQ\_CAL is not available in Bank 8.

**Table 9-15. EQ\_CAL Values**

Attribute	Values	Default
EQ_CAL	0, 1, 2, 3, 4	0

## PULLMODE

The PULLMODE attribute is available for all the LVTTLL and LVCMOS inputs and outputs. This attribute can be enabled for each I/O independently.

**Table 9-16. PULLMODE Values**

PULL Options	PULLMODE Value
Pull up (Default)	UP
Pull Down	DOWN
Bus Keeper	KEEPER
Pull Off	NONE

**Table 9-17. PULLMODE Settings**

Buffer	Values	Default
Input	UP, DOWN, KEEPER, NONE	UP
Output	UP, DOWN, KEEPER, NONE	UP

## PCICLAMP

PCICLAMP is available on all the pads of the device. On the top of the device the PCICLAMP setting can be optionally turned OFF. The rest of the banks only support the PCICLAMP value ON.



**Table 9-18. PCICLAMP Values**

Attribute	Values	Default
PCI33	ON, OFF	ON

## SLEWRATE

The SLEWRATE attribute is available for all LVTTTL and LVCMOS output drivers. Each I/O pin has an individual slew rate control. This allows designers to specify slew rate control on a pin-by-pin basis.

**Table 9-19. Slew Rate Values**

Attribute	Values	Default
SLEWRATE	FAST, SLOW	SLOW

## INBUF

By default, all the unused input buffers are disabled. The INBUF attribute is used to enable the unused input buffers when performing a boundary scan test. This is a global attribute and can be globally set to ON and OFF.

- Values: ON, OFF
- Default: OFF

## FIXEDEDELAY

This attribute can be used in HDL to enable the input Fixed Delay on the input of an SDR Input register. When set to TRUE, you can achieve a zero hold time on the input register.

- Value: TRUE, FALSE
- Default: FALSE

## DIN/DOUT

This attribute can be used when an I/O register needs to be assigned. Using DIN asserts an input register and using DOUT asserts an output register in the design. By default, the software will attempt to assign the I/O registers if applicable. Users can turn this OFF by using a synthesis attribute or the ispLEVER Preference Editor. These attributes can only be applied on registers.

## LOC

This attribute can be used to make pin assignments to the I/O ports in the design. This attribute is only used when the pin assignments are made in HDL source. Pins can also be assigned directly using the GUI in the Preference Editor of the software. See the appendices for further information.

## Design Considerations and Usage

This section discusses some of design rules and considerations that need to be taken into account when designing with the LatticeECP3 sysIO buffer

### Banking Rules

- If  $V_{CCIO}$  or  $V_{CCJ}$  for any bank is set to 3.3V, it is recommended that it be connected to the same power supply as  $V_{CCAUX}$ , thus minimizing leakage.
- If  $V_{CCIO}$  or  $V_{CCJ}$  for any bank is set to 1.2V, it is recommended that it be connected to the same power supply as  $V_{CC}$ , thus minimizing leakage.
- When implementing DDR memory interfaces, the  $V_{REF1}$  of the bank is used to provide reference to the interface pins and cannot be used to power any other referenced inputs.

- Only the top bank supports programmable PCI clamps.
- On the top banks, all legal input buffers should be independent of bank  $V_{CCIO}$  except for 1.8V and 1.5V buffers, which require a bank  $V_{CCIO}$  of 1.8V and 1.5V. On the left and right banks, 1.2V input buffers can be assigned to a bank independent of  $V_{CCIO}$ . 2.5V input buffers can be assigned to banks with  $V_{CCIO}$  2.5V and 3.3V. All other input buffers depend on the  $V_{CCIO}$  to the bank.
- When DIFFRESISTOR is used, the VTT pin for that bank should be left floating.
- When TERMINATEVTT is used, VTT should be connected to the correct voltage depending on the IO\_TYPE set. For example, for SSTL18 standards, VTT should be connected to 0.9V.
- Both TERMINATEVTT and DIFFRESISTOR use the VTT pin, hence these are mutually exclusive in a bank.
- Equalization is only available on the left and right banks.
- PCICLAMP is programmable on the top-side banks 0 and 1. For all other banks, PCICLAMP is always ON.

### Differential I/O Rules

- All banks can support LVDS input buffers. Only the banks on the right and left sides (Banks 2, 3, 6 and 7) will support True Differential output buffers. The banks on the top and bottom will support the LVDS input buffers but will not support True LVDS outputs. The user can use emulated LVDS output buffers on these banks.
- All banks support emulated differential buffers using the external resistor pack and complementary LVCMOS drivers.
- Only 50% of the I/Os on the left and right sides can provide LVDS, mini-LVDS, PPLVDS and RSDS output buffer capability. See the [LatticeECP3 Family Data Sheet](#) for the pin listing for all the true differential pairs.
- The IO\_TYPE attribute for a differential buffer can only be assigned to the TRUE pad. The ispLEVER design tool will automatically assign the other I/O of the differential pair to the complementary pad.
- TRLVDS inputs are only supported on the top banks.
- LVDS, MINILVDS, RSDS, PPDS cannot co-exist in one bank.
- An I/O bank can only have differential outputs with the same DIFFDRIVE setting. Differential outputs with different DIFFDRIVE settings cannot be placed in the same I/O bank.
- DIFFRESISTOR termination is only available on the left and right sides. If enabled, the VTT of the bank should be left floating. Referenced inputs cannot be used in this bank when DIFFRESISTOR is enabled.

### Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
May 2009	01.1	Updated Mixed Voltage Support in Top Banks table.
		Updated Mixed Voltage Support in Left and Right Banks table.
		Added FIXEDDELAY attribute support for SDR registers.
August 2009	01.2	Updated On-Chip Termination text section.
		Updated DIFFRESISTOR text section.
		Updated Banking Rules bullets.
April 2010	01.3	Removed support for programmable PCICLAMP, equalization and VREF pins in Bank 8.
June 2010	01.4	Added Appendix C - sysIO Attributes Using the Diamond Spreadsheet View User Interface.
November 2010	01.5	Updated Supported Input Standards table.
		Updated Hot Socketing Support text section.
		Updated Hot Socketing row of the I/O Standards Supported per Bank table.
		Updated first footnote in the Mixed Voltage Support in Left and Right Banks table.
March 2011	01.6	Added support for GTL+ input standard using HSTL input buffer.
April 2011	01.7	Updated to clarify PCICLAMP programmability and DRIVE settings available with and without OPENDRAIN.
February 2012	01.8	Updated document with new corporate logo.
March 2012	01.9	Verilog Synplicity Attribute Syntax table – Corrected information in the Syntax column.
June 2012	02.0	Supported Input Standards table – Updated VCCIO column and removed GTL+ row.
		Supported Output Standards table – Updated VCCIO information for Point-to-Point LVDS.
		Mixed Voltage Support in Top Banks table – Updated footnote 2.
		Mixed Voltage Support in Left and Right Banks table – Updated footnote 1.
March 2013	02.1	Mixed Voltage Support in Top Banks table – Updated footnote 2.
July 2013	02.2	Updated Technical Support Assistance information.

## Appendix A. HDL Attributes

Using HDL attributes, you can assign the sysIO attributes directly in your source code. You will need to use the attribute definition and syntax for the synthesis vendor you are planning to use. Below is a list of the sysIO attributes, syntax and examples for the Synplify Pro® synthesis tool. This section only lists the sysIO buffer attributes for these devices. You can refer to the Synplify Pro user manual for a complete list of synthesis attributes. You can access these manuals through the ispLEVER software Help.

### VHDL Synplify Pro

This section lists syntax and examples for all the sysIO attributes in VHDL when using Synplify Pro synthesis tools.

#### Syntax

**Table 9-20. VHDL Attribute Syntax for Synplify Pro**

Attribute	Syntax
IO_TYPE	attribute IO_TYPE: string; attribute IO_TYPE of Pinname: signal is "IO_TYPE Value";
OPENDRAIN	attribute OPENDRAIN: string; attribute OPENDRAIN of Pinname: signal is "OpenDrain Value";
DRIVE	attribute DRIVE: string; attribute DRIVE of Pinname: signal is "Drive Value";
DIFFDRIVE	attribute DIFFDRIVE: string; attribute DIFFDRIVE of Pinname: signal is "Diffdrive Value";
MULTIDRIVE	attribute MULTIDRIVE: string; attribute MULTIDRIVE of Pinname: signal is "MULTIDRIVE Value";
EQ_CAL	attribute EQ_CAL: string; attribute EQ_CAL of Pinname: signal is "EQ_CAL Value";
TERMINATEVTT	attribute TERMINATEVTT: string; attribute TERMINATEVTT of Pinname: signal is "TERMINATEVTT Value";
DIFFRESISTOR	attribute DIFFRESISTOR: string; attribute DIFFRESISTOR of Pinname: signal is "DIFFRESISTOR Value";
PULLMODE	attribute PULLMODE: string; attribute PULLMODE of Pinname: signal is "Pullmode Value";
PCICLAMP	attribute PCICLAMP: string; attribute PCICLAMP of Pinname: signal is "PCIClamp Value";
SLEWRATE	attribute PULLMODE: string; attribute PULLMODE of Pinname: signal is "Slewrates Value";
DIN	attribute DIN: string; attribute DIN of Pinname: signal is " ";
DOUT	attribute DOUT: string; attribute DOUT of Pinname: signal is " ";
LOC	attribute LOC: string; attribute LOC of Pinname: signal is "pin_locations";
FIXEDELAY	attribute FIXEDELAY:string; attribute FIXEDELAY of Pinname: signal is "FIXEDELAY Value";

#### Examples

##### IO\_TYPE

```
--***Attribute Declaration***
ATTRIBUTE IO_TYPE: string;

--***IO_TYPE assignment for I/O Pin***
```

---

```
ATTRIBUTE IO_TYPE OF portA: SIGNAL IS "PCI33";
ATTRIBUTE IO_TYPE OF portB: SIGNAL IS "LVCMOS33";
ATTRIBUTE IO_TYPE OF portC: SIGNAL IS "LVDS25";
```

### **OPENDRAIN**

```
--***Attribute Declaration***
ATTRIBUTE OPENDRAIN: string;
--***Open Drain assignment for I/O Pin***
ATTRIBUTE OPENDRAIN OF portB: SIGNAL IS "ON";
```

### **DRIVE**

```
--***Attribute Declaration***
ATTRIBUTE DRIVE: string;
--***DRIVE assignment for I/O Pin***
ATTRIBUTE DRIVE OF portB: SIGNAL IS "20";
```

### **DIFFDRIVE**

```
--***Attribute Declaration***
ATTRIBUTE DIFFDRIVE: string;
--*** DIFFDRIVE assignment for I/O Pin***
ATTRIBUTE DIFFDRIVE OF portB: SIGNAL IS "2.0";
```

### **MULTDRIVE**

```
--***Attribute Declaration***
ATTRIBUTE MULTDRIVE: string;
--*** MULTDRIVE assignment for I/O Pin***
ATTRIBUTE MULTDRIVE OF portB: SIGNAL IS "2X";
```

### **EQ\_CAL**

```
--***Attribute Declaration***
ATTRIBUTE EQ_CAL: string;
--*** EQ_CAL assignment for I/O Pin***
ATTRIBUTE EQ_CAL OF portB: SIGNAL IS "1";
```

### **TERMINATEVTT**

```
--***Attribute Declaration***
ATTRIBUTE TERMINATEVTT: string;
--*** TERMINATEVTT assignment for I/O Pin***
ATTRIBUTE TERMINATEVTT OF portB: SIGNAL IS "40";
```

### **DIFFRESISTOR**

```
--***Attribute Declaration***
ATTRIBUTE DIFFRESISTOR: string;
--*** DIFFRESISTOR assignment for I/O Pin***
ATTRIBUTE DIFFRESISTOR OF portB: SIGNAL IS "80";
```

---

**PULLMODE**

```
--***Attribute Declaration***  
ATTRIBUTE PULLMODE : string;  
--***PULLMODE assignment for I/O Pin***  
ATTRIBUTE PULLMODE OF portA: SIGNAL IS "DOWN";  
ATTRIBUTE PULLMODE OF portB: SIGNAL IS "UP";
```

**PCICLAMP**

```
--***Attribute Declaration***  
ATTRIBUTE PCICLAMP: string;  
--***PULLMODE assignment for I/O Pin***  
ATTRIBUTE PCICLAMP OF portA: SIGNAL IS "OFF";
```

**SLEWRATE**

```
--***Attribute Declaration***  
ATTRIBUTE SLEWRATE : string;  
--*** SLEWRATE assignment for I/O Pin***  
ATTRIBUTE SLEWRATE OF portB: SIGNAL IS "FAST";
```

**DIN/DOUT**

```
--***Attribute Declaration***  
ATTRIBUTE din : string;  
ATTRIBUTE dout : string;  
--*** din/dout assignment for I/O Pin***  
ATTRIBUTE din OF input_vector: SIGNAL IS " ";  
ATTRIBUTE dout OF output_vector: SIGNAL IS " ";
```

**LOC**

```
--***Attribute Declaration***  
ATTRIBUTE LOC : string;  
--*** LOC assignment for I/O Pin***  
ATTRIBUTE LOC OF input_vector: SIGNAL IS "E3,B3,C3 ";
```

**FIXEDELAY**

```
--***Attribute Declaration***  
ATTRIBUTE FIXEDELAY : string;  
--*** FIXEDELAY assignment for I/O Pin***  
ATTRIBUTE FIXEDELAY OF portA: SIGNAL IS "True";
```

## Verilog Synplicity

This section lists syntax and examples for all the sysIO Attributes in Verilog using Synplicity® synthesis tool.

### Syntax

**Table 9-21. Verilog Synplicity Attribute Syntax**

Attribute	Syntax
IO_TYPE	PinType PinName /* synthesis IO_TYPE="IO_Type Value"*/;
OPENDRAIN	PinType PinName /* synthesis OPENDRAIN = "OpenDrain Value"*/;
DRIVE	PinType PinName /* synthesis DRIVE="Drive Value"*/;
DIFFDRIVE	PinType PinName /* synthesis DIFFDRIVE = "DIFFDRIVE Value"*/;
MULTDRIVE	PinType PinName /* synthesis MULTDRIVE = "MULTDRIVE Value"*/;
EQ_CAL	PinType PinName /* synthesis EQ_CAL = "EQ_CAL Value"*/;
TERMINATEVTT	PinType PinName /* synthesis TERMINATEVTT = "TERMINATEVTT Value"*/;
DIFFRESISTOR	PinType PinName /* synthesis DIFFRESISTOR = "DIFFRESISTOR Value"*/;
PULLMODE	PinType PinName /* synthesis PULLMODE="Pullmode Value"*/;
PCICLAMP	PinType PinName /* synthesis PCICLAMP = "PCIClamp Value"*/;
SLEWRATE	PinType PinName /* synthesis SLEWRATE="Slewrates Value"*/;
DIN	PinType PinName /* synthesis DIN=" "*/;
DOUT	PinType PinName /* synthesis DOUT=" "*/;
LOC	PinType PinName /* synthesis LOC="pin_locations"*/;
FIXEDELAY	PinType PinName/*synthesis FIXEDELAY = "FIXEDELAY value"*/;

### Examples

#### //IO\_TYPE, PULLMODE, SLEWRATE and DRIVE assignment

```
output portB /*synthesis IO_TYPE="LVCMOS33" PULLMODE = "UP" SLEWRATE = "FAST"
DRIVE = "20"*/;
output portC /*synthesis IO_TYPE="LVDS25" */;
```

#### //DIFFRESISTOR

```
input portB /*synthesis IO_TYPE="LVDS" DIFFRESISTOR="80" */;
```

#### //DIFFDRIVE, MULTDRIVE

```
output portB /*synthesis IO_TYPE="MINILVDS" DIFFDRIVE="2.0" MULTDRIVE="2X"*/;
```

#### //TERMINATEVTT, EQ\_CAL

```
input portB /*synthesis IO_TYPE="SSTL15" TERMINATEVTT="60" EQ_CAL="2"*/;
```

#### //OPENDRAIN

```
output portA /*synthesis OPENDRAIN = "ON"*/;
```

#### //PCICLAMP

```
output portA /*synthesis IO_TYPE="PCI33" PCICLAMP = "OFF"*/;
```

#### //FIXEDELAY

```
input portB /*synthesis FIXEDELAY = "true" */;
```

**// Place the flip-flops near the load input**

```
input load /* synthesis din="" */;
```

**// Place the flip-flops near the outload output**

```
output outload /* synthesis dout="" */;
```

**//IO pin location**

```
input [3:0] DATA0 /* synthesis loc="E3,B1,F3"*/;
```

**//Register pin location**

```
reg data_in_ch1_buf_reg3 /* synthesis loc="R40C47" */;
```

**//Vectored internal bus**

```
reg [3:0] data_in_ch1_reg /*synthesis loc ="R40C47,R40C46,R40C45,R40C44" */;
```



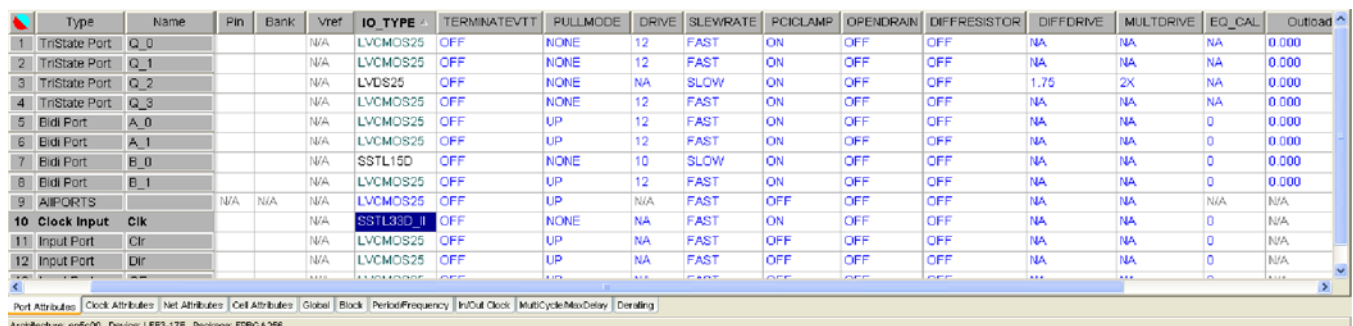
## Appendix B. sysIO Attributes Using the ispLEVER Design Planner User Interface

sysIO buffer attributes can be assigned using the Design Planner Spreadsheet View available in the ispLEVER design tool. If you are using Lattice Diamond™ design software, see Appendix C. The Pin Attribute Sheet lists all the ports in a design and all the available sysIO attributes as preferences. Click on each of these cells for a list of all the valid I/O preferences for that port. Each column takes precedence over the next. Therefore, when you choose a particular IO\_TYPE, the columns for the DRIVE, PULLMODE, SLEW-RATE and other attributes will only list the valid combinations for that IO\_TYPE. Pin locations can be locked using the pin location column of the Pin Attribute sheet. Right-click on a cell to list all the available pin locations. The Design Planner will also run a DRC check to check for any incorrect pin assignments.

You can enter the DIN/ DOUT preferences using the Cell Attributes Sheet of the Design Planner. All the preferences assigned using the Design Planner are written into the logical preference file (.lpf).

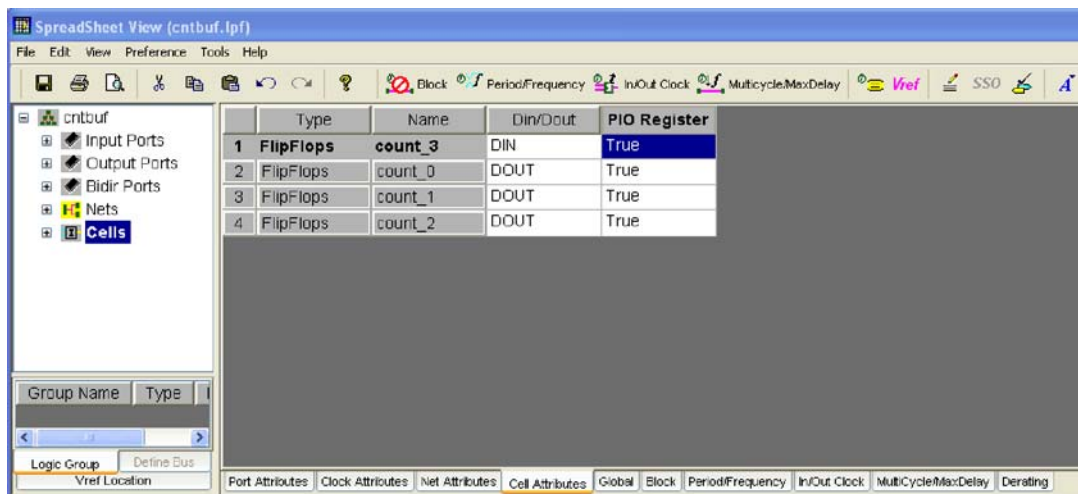
Figures 9-3 and 9-4 show the Port Attribute Sheet and the Cell Attribute Sheet views of the Design Planner. For further information on how to use the Design Planner, refer to the ispLEVER Help documentation, available in the Help menu option of the software.

**Figure 9-3. Port Attributes Tab of Design Planner Spreadsheet View**



Type	Name	Pin	Bank	Vref	IO_TYPE	TERMINATEVTT	PULLMODE	DRIVE	SLEWRATE	POICLAMP	OPENDRAIN	DIFFRESISTOR	DIFFDRIVE	MULTDRIVE	EQ_CAL	Outload
1	TriState Port	Q_0		N/A	LVCMOS25	OFF	NONE	12	FAST	ON	OFF	OFF	NA	NA	NA	0.000
2	TriState Port	Q_1		N/A	LVCMOS25	OFF	NONE	12	FAST	ON	OFF	OFF	NA	NA	NA	0.000
3	TriState Port	Q_2		N/A	LVDS25	OFF	NONE	NA	SLOW	ON	OFF	OFF	1.75	2X	NA	0.000
4	TriState Port	Q_3		N/A	LVCMOS25	OFF	NONE	12	FAST	ON	OFF	OFF	NA	NA	NA	0.000
5	Bidi Port	A_0		N/A	LVCMOS25	OFF	UP	12	FAST	ON	OFF	OFF	NA	NA	0	0.000
6	Bidi Port	A_1		N/A	LVCMOS25	OFF	UP	12	FAST	ON	OFF	OFF	NA	NA	0	0.000
7	Bidi Port	B_0		N/A	SSTL15D	OFF	NONE	10	SLOW	ON	OFF	OFF	NA	NA	0	0.000
8	Bidi Port	B_1		N/A	LVCMOS25	OFF	UP	12	FAST	ON	OFF	OFF	NA	NA	0	0.000
9	AIOPORTS		N/A	N/A	LVCMOS25	OFF	UP	NA	FAST	OFF	OFF	OFF	NA	NA	N/A	N/A
10	Clock Input	Clk		N/A	SSTL33D_I	OFF	NONE	NA	FAST	ON	OFF	OFF	NA	NA	0	N/A
11	Input Port	Clr		N/A	LVCMOS25	OFF	UP	NA	FAST	OFF	OFF	OFF	NA	NA	0	N/A
12	Input Port	Dir		N/A	LVCMOS25	OFF	UP	NA	FAST	OFF	OFF	OFF	NA	NA	0	N/A

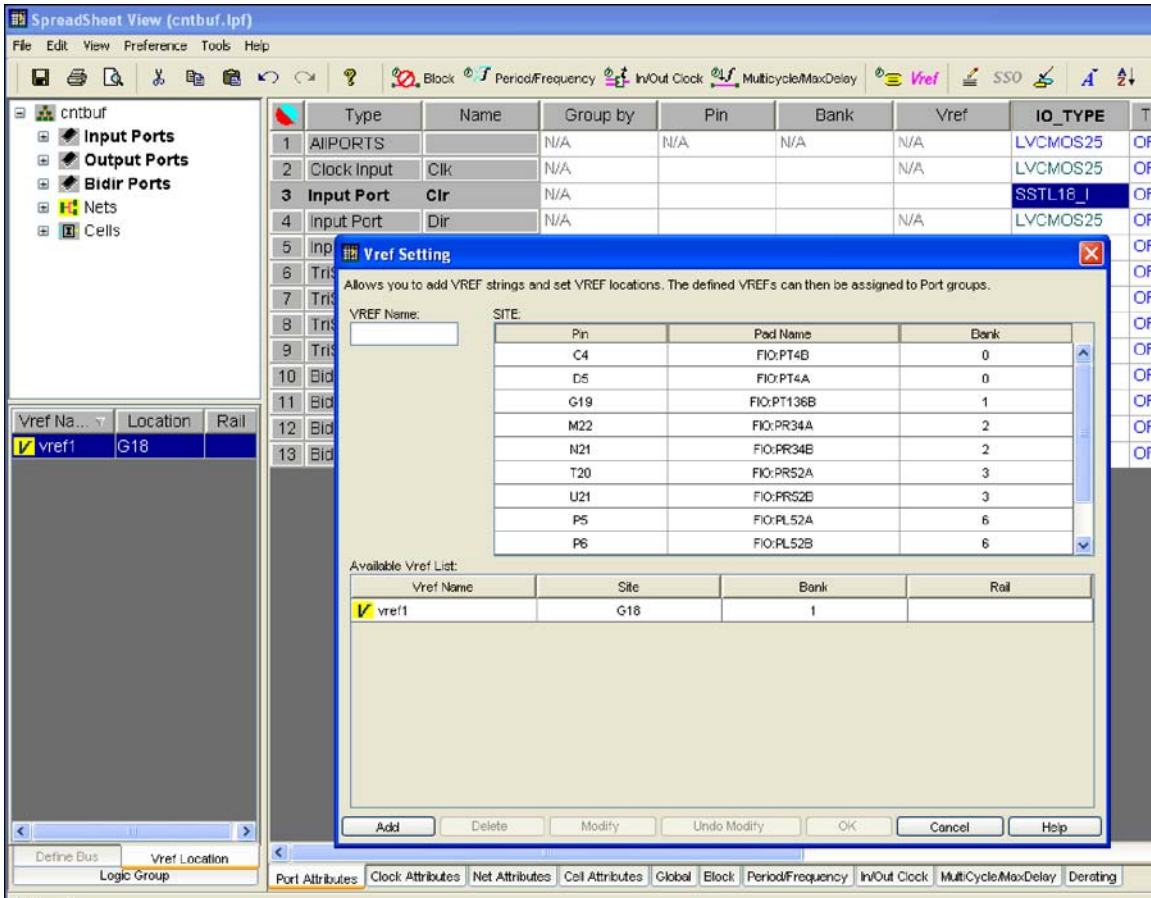
**Figure 9-4. Cell Attributes Tab**



Type	Name	Din/Dout	PIO Register	
1	FlipFlops	count_3	DIN	True
2	FlipFlops	count_0	DOUT	True
3	FlipFlops	count_1	DOUT	True
4	FlipFlops	count_2	DOUT	True

Users can assign V<sub>REF</sub> for a bank using the V<sub>REF</sub> Setting option in the Design Planner. See the software online help for a more detailed description of this setting.

Figure 9-5. VREF Assignment in Design Planner



## Appendix C. sysIO Attributes Using the Diamond Spreadsheet View User Interface

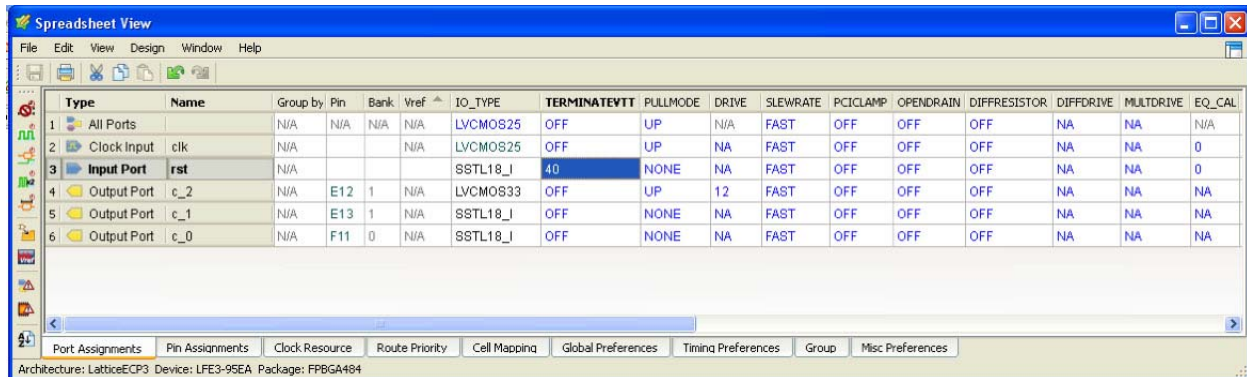
sysIO buffer attributes can be assigned using the Spreadsheet view in the Lattice Diamond design software. The Port Assignments Sheet lists all the ports in a design and all the available sysIO attributes in multiple columns. Click on each of these cells for a list of all the valid I/O preferences for that port. Each column takes precedence over the next. Therefore, when you choose a particular IO\_TYPE, the columns for the DRIVE, PULLMODE, SLEWRATE and other attributes will only list the valid entries for that IO\_TYPE.

Pin locations can be locked using the Pin column of the Port Assignments sheet or using the Pin Assignments sheet. You can right-click on a cell and go to Assign Pins to see a list of available pins.

The Spreadsheet View also has an option to run a DRC check to check for any incorrect pin assignments. You can enter the DIN/ DOUT preferences using the Cell Mapping tab. All the preferences assigned using the Spreadsheet view are written into the logical preference file (.lpf).

Figure 9-5 shows the Port Assignments Sheet of the Spreadsheet view. For further information on how to use the Spreadsheet view, refer to the Diamond Help documentation, available in the Help menu option of the software.

**Figure 9-6. Port Attributes Tab of SpreadSheet View**



Type	Name	Group by	Pin	Bank	Vref	IO_TYPE	TERMINATEVT	PULLMODE	DRIVE	SLEWRATE	PCICLAMP	OPENDRAIN	DIFFRESISTOR	DIFFDRIVE	MULTDRIVE	EQ_CAL	
1	All Ports	N/A	N/A	N/A	N/A	LVCMS0825	OFF	UP	N/A	FAST	OFF	OFF	OFF	NA	NA	N/A	
2	Clock Input	clk			N/A	LVCMS0825	OFF	UP	NA	FAST	OFF	OFF	OFF	NA	NA	0	
3	Input Port	rst				SSTL18_I	40	NONE	NA	FAST	OFF	OFF	OFF	NA	NA	0	
4	Output Port	c_2			E12	1	N/A	LVCMS0833	OFF	UP	12	FAST	OFF	OFF	OFF	NA	NA
5	Output Port	c_1			E13	1	N/A	SSTL18_I	OFF	NONE	NA	FAST	OFF	OFF	OFF	NA	NA
6	Output Port	c_0			F11	0	N/A	SSTL18_I	OFF	NONE	NA	FAST	OFF	OFF	OFF	NA	NA

Users can create a VREF pin using the Spreadsheet view as shown in Figure 9-7 and then assign VREF for a bank using the VREF Column in the Ports Assignment Tab of the Spreadsheet View as show in Figure 9-8. See the Diamond online help for a more detailed description of this setting.

**Figure 9-7. Creating a VREF in Spreadsheet View**

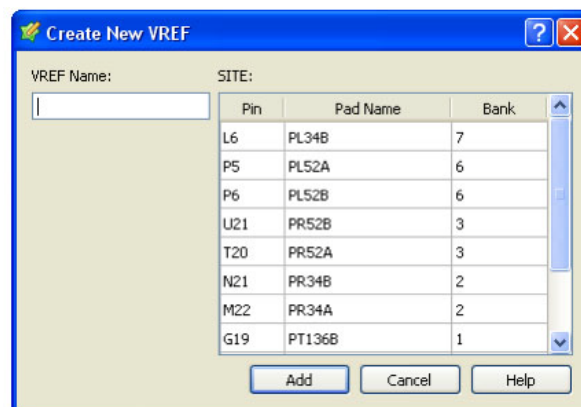


Figure 9-8. Assigning a VREF for an Input Port in Spreadsheet View

	Type	Name	Group by	Pin	Bank	Vref	IO_TYPE	TERMINATEVTT
1	All Ports		N/A	N/A	N/A	N/A	LVC MOS25	OFF
2	Input Port	Dir	N/A			N/A	LVC MOS25	OFF
3	Clock Input	Clk	N/A			N/A	LVC MOS25	OFF
4	Input Port	Clr	N/A			VREF1:L5(7)	SSTL18_I	OFF
5	Input Port	OE	N/A			N/A	LVC MOS25	OFF

Port Assignments | Pin Assignments | Clock Resource | Route Priority | Cell Mapping | Global Preferences | Timing Preferences | Gr

## Introduction

This technical note describes the clock resources available in the LatticeECP3™ device architecture. Details are provided for primary clocks, secondary clocks, edge clocks, and general routing, as well as clock elements such as PLLs, DLLs, clock dividers and more.

The number of PLLs, DLLs and DDR-DLLs for each device is listed in Table 10-1.

**Table 10-1. Number of PLLs, DLLs and DDR-DLLs**

Parameter	Description	ECP3-17	ECP3-35	ECP3-70	ECP3-95	ECP3-150
Number of GPLLs	General purpose PLL	2	4	10	10	10
Number of DLLs	General Purpose DLL	2	2	2	2	2
Number of DDR-DLLs	DLL for DDR applications	2	2	2	2	2

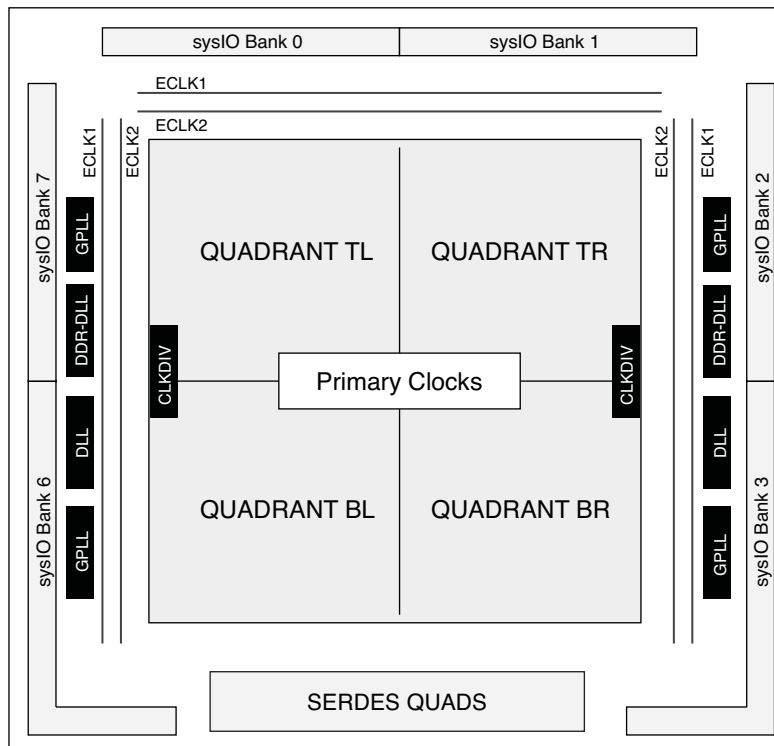
## Clock/Control Distribution Network

LatticeECP3 devices provide global clock distribution in the form of eight quadrant-based primary clocks and flexible secondary clocks. Two edge clocks are also provided on the left, right and top edges of the device. Other clock sources include clock input pins, general logic, PLLs, DLLs, DCSs, and clock dividers.

## LatticeECP3 Top-Level View

Figure 10-1 provides a view of the primary clocking structure of the LatticeECP3-35 device.

**Figure 10-1. LatticeECP3 Clocking Structure (LatticeECP3-35)**



## Primary Clocks

The LatticeECP3 architecture provides eight primary global clock nets, all eight of which are available to all quadrants; however, each quadrant can have any of these eight nets independent of the others, or they can be tied to other vertically or horizontally adjacent quadrants. Two of these clocks provide the Dynamic Clock Selection (DCS) feature. The six primary clocks without DCS can be specified in the Pre-map Preference Editor as 'Primary Pure' and the two DCS clocks as 'Primary-DCS'.

The sources of primary clocks include the following (refer also to Figure 10-35).

- GPLL outputs
- DLL outputs
- PCS TX CLKs
- CLKDIV outputs
- PCLK PIOs

## Secondary Clocks

The LatticeECP3 secondary clocks are a flexible region-based clocking resource. A region is an area over which a secondary clock operates. Each region has eight possible sources to drive secondary clock routing. Note that secondary/regional clock net boundaries do not always coincide with primary/quadrant clock net boundaries; for example, the ECP3-17 has three rows of regions, but two rows of quadrants.

There are eight secondary clock muxes per region. Each mux has inputs from eight different sources. Seven of these are from internal nodes. The eighth input comes from a primary clock pin. The input sources are not necessarily located in the same region as the mux. This structure enables global use of secondary clocks.

The sources of secondary clocks are:

- Dedicated PCLK clock pins:
  - PCLKT0\_0
  - PCLKT1\_0
  - PCLKT2\_0
  - PCLKT3\_0
  - PCLKT6\_0
  - PCLKT7\_0
- Internal nodes

Table 10-2 lists the number of secondary clock regions in LatticeECP3 devices and Figure 10-2 shows how the secondary clocks are organized into regions for a typical LatticeECP3 device.

**Table 10-2. Number of Secondary Clock Regions**

Parameter	ECP3-17	ECP3-35	ECP3-70	ECP3-95	ECP3-150
Number of regions	16	16	20	20	36
Layout (Vertical x Horizontal)	4x4	4x4	5x4	5x4	6x6

## Edge Clocks

The LatticeECP3 device has two Edge Clocks (ECLK) each on the left, right, and top sides of the device. There are no edge clocks located on the bottom side of the device because of the SERDES blocks located there. These clocks, which have low injection time and skew, are used to clock I/O registers. Edge clock resources are designed for high speed I/O interfaces with high fanout capability. In the list of edge clock sources, shown below, there is a

special provision made for signals from the left-side PLLs and DLLs to feed all edge clock banks. This is especially useful in DDR applications which allow I/Os to be distributed across three edges of the device. Refer to Appendix B for detailed information on ECLK locations and connectivity.

The sources of edge clocks are:

- **For Left and Right Edge Clocks:**

- Dedicated clock pins
- GPLL outputs:
  - LatticeECP-35: Both PLLs on each side can drive that same side's edge clock
  - Others: Second and third PLL on each side can drive that same side's edge clock
- GPLL input pins
- DLL outputs
- LEFT side top PLL and DLL can drive both sides ("bridging" capability)
- Internal nodes

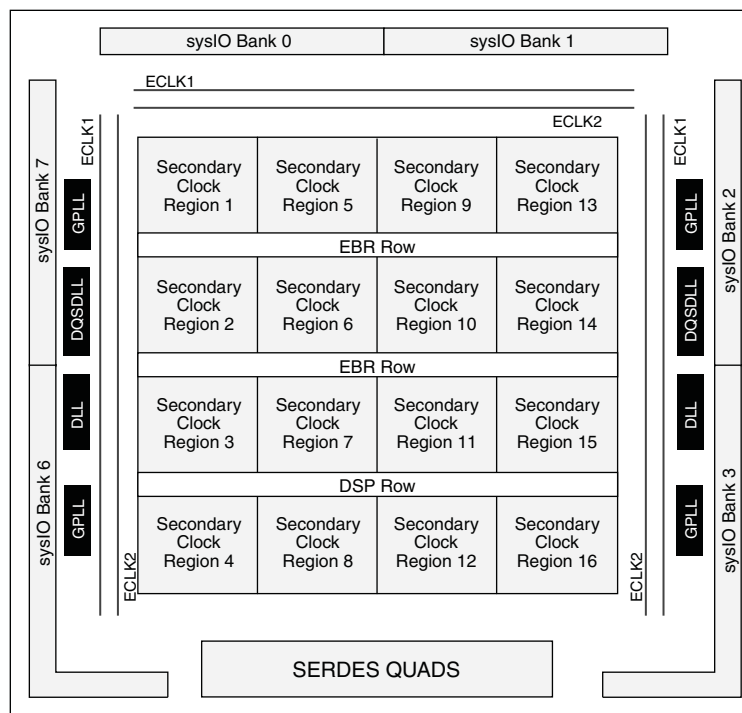
- **For Top Edge Clocks:**

- Dedicated clock pins
- Top left or top right GPLL outputs
- LEFT side top PLL and DLL ("bridging" capability)
- Internal nodes

Edge clocks can directly drive the secondary clock resources and general routing resources. Refer to Figure 10-36 for detailed information on edge clock routing. The edge clocks on the left and right edges can drive the primary clock resources through the CLKDIV blocks.

Figure 10-2 shows the structure of the secondary clocks and edge clocks for a typical device.

**Figure 10-2. LatticeECP3 Secondary Clocks and Edge Clocks (ECP3-35)**



## General Routing for Clocks

The LatticeECP3 architecture supports the ability to use the general routing, normally used for data routing, as a clock resource. This resource is intended to be used for small areas of the design to allow additional flexibility in linking dedicated clocking resources and building very small clock trees.

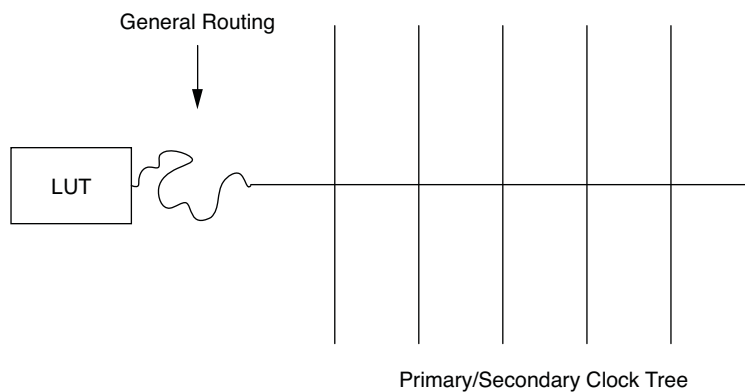
The general routing of the LatticeECP3 is optimized for data routing. Clocks can be routed on this resource, but will not have the same performance as the dedicated clocking resources. Due to the large amount of connectivity the place and route of this resource will not have as tight a skew as the dedicated clocking resources. For this reason it is best to limit the distance of a general routing based clock as well as the number of loads.

### Additional Connectivity for Dedicated Clock Resources

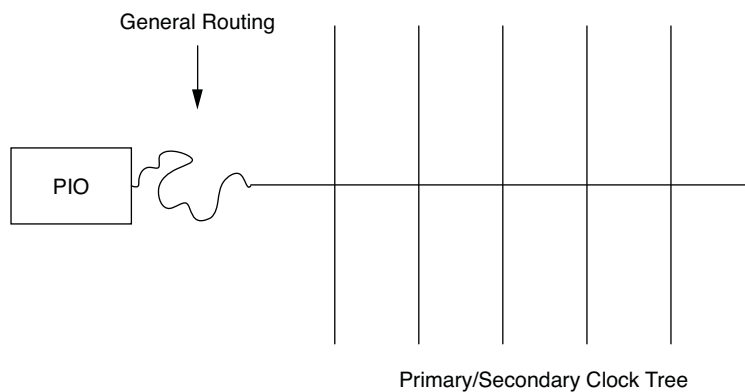
The dedicated clocking resources in the LatticeECP3 all have preferred entry points for the clock signal. Sometimes these entry points cannot be used by the user. To allow flexible connectivity to the dedicated resources the general routing can also be used an entry point. This is useful when creating a clock gate or clock MUX implemented in FPGA LUTs.

Figures 10-3 and 10-4 provide examples of using general routing to bridge a clock source to a dedicated clock resource. Figure 10-3 uses a LUT such as would be used for a clock gate or MUX. Figure 10-4 uses a PIO, or FPGA I/O input pin, to bridge to a dedicated clock resource. This is the structure that would be found if a non-preferred input pin was selected.

**Figure 10-3. General Routing Connection to Dedicated Clock Resource from LUT**



**Figure 10-4. General Routing Connection to Dedicated Clock Resource from PIO**



*Note: General Routing cannot be used in the source path for edge clocks. Edge clocks are high-speed resources which require clean and duty-cycle balanced sources. The Place and Route software in Lattice Diamond® will not allow general routing in the source path to edge clocks.*



When using the general routing to reach a dedicated resource the place and route process will issue a warning. Below is an example warning message.

```
WARNING - par: Regional clock signal "myclk" drives a PIO comp.
          Generic routing may have to be used to route to the PIO load of
          this regional clock.
```

These warnings are intended to alert the user that general routing is being utilized to route a portion of the clock before it arrives at a dedicated resource.

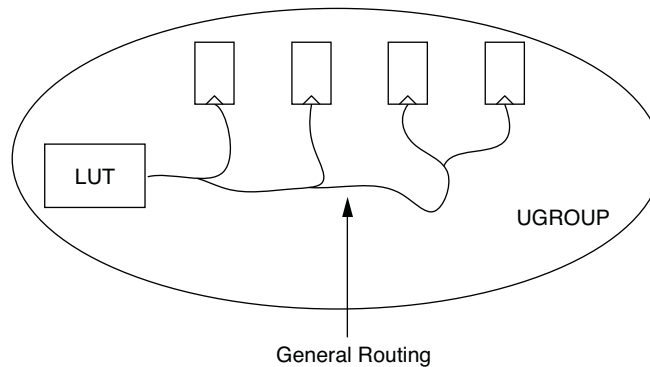
### Very Small Clock Domains

General routing can be used to make very small clock domains. As discussed previously, the general routing requires a large number of connections to allow flexible data routing. These additional connections will increase the skew that can occur when implementing clocks. Due to this skewed reality of the general routing it is best to limit the number of loads on the clock.

Small clock domains should be grouped together using a UGROUP preference (see the **Diamond Help >Constraints Reference Guide > Preferences > UGROUP**). A UGROUP preference is a placement constraint which Place & Route will utilize to place all of the components in the group close together. This constraint will attempt to keep the routing distance to a minimum and thereby reduce the amount of clock skew that can occur between the destinations.

Figure 10-5 shows an example of using general routing for a small clock domain.

**Figure 10-5. Small Clock Domain Example with General Routing**



### Static Timing Analysis of General Routing Clocks

All LatticeECP3 designs require the user to run static timing analysis using the Trace process. When using general routing for clocks it is necessary to generate both a setup and hold time Trace report. These setup and hold checks will ensure that any skew induced by the general routing connections will be accounted for by the timing tools and reported to the user.

In the Diamond Strategy for Place & Route Trace make sure the Analysis Option is set to “Standard Setup and Hold Analysis”. This option will create a Trace report which includes both the setup and hold times of the requested number of worst-case paths per preference. The user should examine the general routing based clocks carefully to ensure that these paths meet their timing preferences.

### Specifying Clocks in the Design Tools

If desired, designers can specify the clock resources, primary, secondary or edge to be used to distribute a given clock source. Figure 10-6 illustrates how this can be done in the ispLEVER® Design Planner Spreadsheet View (or in Spreadsheet View in the Lattice Diamond design software). Alternatively, the resources can be specified by using corresponding preferences in the preference file.

## Global Primary Clock and Quadrant Primary Clock

### Global Primary Clock

If a primary clock is not assigned as a quadrant clock, the software assumes it is a global clock.

There are six Global Primary/Pure clocks and two Global Primary/DCS clocks available.

### Primary-Pure and Primary-DCS

Primary Clock Net can be assigned to either Primary-Pure (CLK0 to CLK5) or Primary-DCS (CLK6 and CLK7).

### Quadrant Primary Clock

Any primary clock may be assigned to a quadrant clock. The clock may be assigned to a single quadrant or to two adjacent quadrants (not diagonally adjacent).

When a quadrant clock net is used, the user must ensure that the registers each clock drives can be assigned in that quadrant without any routing issues.

In the quadrant primary clocking scheme, the maximum number of primary clocks is 32. Note, however, that these are not global primary clocks; the maximum number of global primary clocks is eight.

Primary quadrant clocks can be preferenced in the .lpf file by using this format:

```
USE PRIMARY [PURE | DCS] [NET | PORT "<net or port name>" [quadrant_type] ];
```

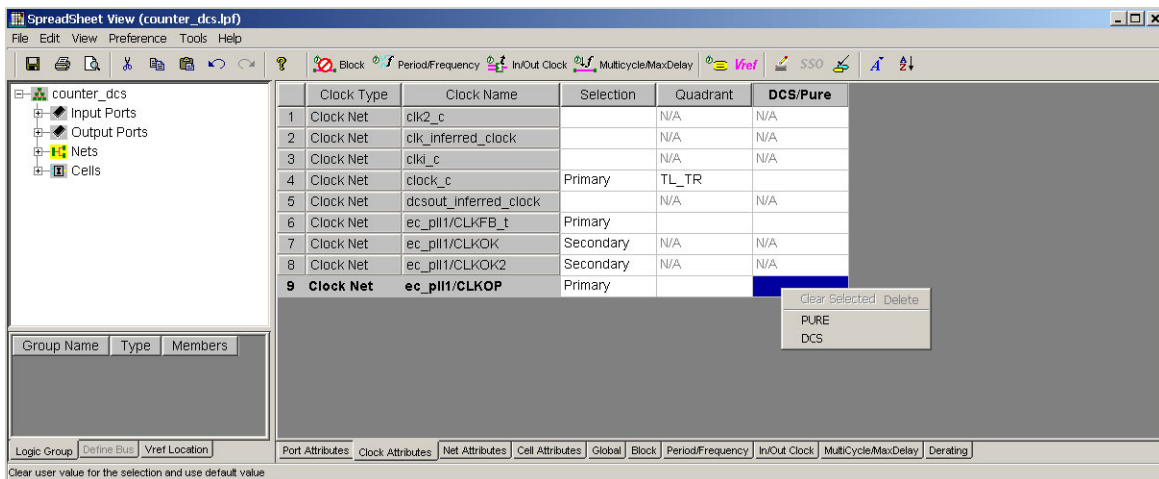
where:

```
<net or port name> = specific clock net name <string>
[quadrant_type] ::= [ QUADRANT_TL | QUADRANT_TR | QUADRANT_BL | QUADRANT_BR ]
QUADRANT_TL ::= Top left corner of the FPGA
QUADRANT_TR ::= Top right corner of the FPGA
QUADRANT_BL ::= Bottom left corner of the FPGA
QUADRANT_BR ::= Bottom right corner of the FPGA
```

One or two quadrants can be specified, for example:

```
USE PRIMARY NET "clk" QUADRANT_TL QUADRANT_TR;
```

**Figure 10-6. ispLEVER Design Planner Spreadsheet View (see Appendix C Figure 10-38 for Diamond Equivalent)**



Refer to “[Appendix A. Primary Clock Sources and Distribution](#)” on page 40 for detailed clock network diagrams.

## Global Secondary Clock and Regional Secondary Clocks

### Global Secondary Clocks

Secondary clocking is regional. However, if no secondary regions are defined, software will combine all secondary clock regions into a signal global secondary clock structure comprising up to eight secondary clock/CE/LSR sources. If nets are not assigned to secondary regions, the software will automatically assign up to eight clock/CE/LSR nets to use secondary resources in this global secondary clock structure.

### Regional Secondary Clocks

Secondary clock resources are regional, meaning there are different clock regions within the device as a whole. Each LatticeECP3 device has a different number of regions based on the size. Below is a basic diagram of the secondary clock region resources for each LatticeECP3.

#### LatticeECP3-17K / 35K:

R1C1	R1C	R1C3	R1C4
R2C1	R2C2	R2C3	R2C4
R3C1	R3C2	R3C3	R3C4
R4C1	R4C2	R4C3	R4C4

#### LatticeECP3-70K / 95K:

R1C1	R1C2	R1C3	R1C4
R2C1	R2C2	R2C3	R2C4
R3C1	R3C2	R3C3	R3C4
R4C1	R4C2	R4C3	R4C4
R5C1	R5C2	R5C3	R5C4

#### LatticeECP3-150K:

R1C1	R1C2	R1C3	R1C4	R1C5	R1C6
R2C1	R2C2	R2C3	R2C4	R2C5	R2C6
R3C1	R3C2	R3C3	R3C4	R3C5	R3C6
R4C1	R4C2	R4C3	R4C4	R4C5	R4C6
R5C1	R5C2	R5C3	R5C4	R5C5	R5C6
R6C1	R6C2	R6C3	R6C4	R6C5	R6C6

### Secondary Region Clock Preferecing

In order to use secondary clock regions, the user must create a secondary clock region, then preference their clock to it.

To create a region, the format is:

```
REGION "<region name>" CLKREG "CLKREG_R1C1" <# of Columns to Span> <# of Rows to Span>;
USE SECONDARY NET "<clock net>" <region name>;
```

where:

<region name> = A unique region name given to each region preference.

<clock net> = Specific clock/CE/LSR name.

"CLKREG\_R1C1" = The row/column starting point as defined above for each device.

<# of Columns to Span> = Number of columns deep the secondary clock region will encompass. It can be from 1 to 6, depending on which device is used.

<# of Rows to Span> = Number of rows wide the secondary clock region will encompass. It can be from 1 to 6, depending on which device is used.

For example, we will create a region in a LatticeECP3-150K device and assign a clock to it.

```
REGION "D_QUAD_R" CLKREG "CLKREG_R1C4" 5 3;

USE SECONDARY NET "clk1" D_QUAD_R;
```

This will create a secondary clock region that is five columns deep, by three rows wide, starting at R1C4 and clk1 will reside only in that area. Any logic that is clocked by clk1 will be placed within that region. If it cannot be placed in that region a DRC error will occur in place and route. The diagram of this region looks like this:

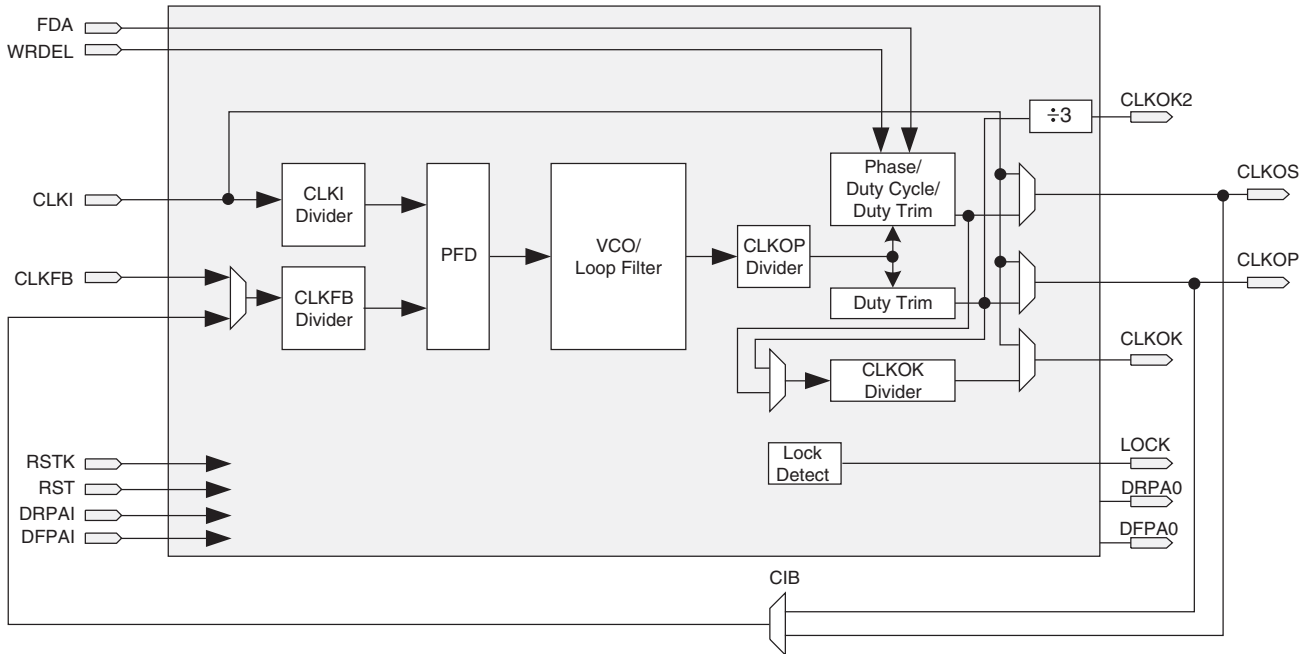
R1C1	R1C2	R1C3	R1C4	R1C5	R1C6
R2C1	R2C2	R2C3	R2C4	R2C5	R2C6
R3C1	R3C2	R3C3	R3C4	R3C5	R3C6
R4C1	R4C2	R4C3	R4C4	R4C5	R4C6
R5C1	R5C2	R5C3	R5C4	R5C5	R5C6
R6C1	R6C2	R6C3	R6C4	R6C5	R6C6

## sysCLOCK™ PLL

The LatticeECP3 PLL provides features such as clock injection delay removal, frequency synthesis, phase/duty cycle adjustment, and dynamic delay adjustment. Figure 10-7 shows the block diagram of the LatticeECP3 PLL.

Generally, the best way to add a PLL to a design is by using IPexpress™; the user simply provides information to the tool via a GUI, and the tool performs all calculations and design rule checks. It then generates a package that can be added to your design in the HDL language of your choice.

Figure 10-7. LatticeECP3 PLL Block Diagram



## Functional Description

### PLL Divider and Delay Blocks

#### Input Clock (CLKI) Divider

The CLKI divider is used to control the input clock frequency into the PLL block. This is also referred to as the M-Divider. The divider setting directly corresponds to the divisor of the output clock. The input and output of the input divider must be within the input and output frequency ranges specified in the [LatticeECP3 Family Data Sheet](#). This is checked by IPExpress.

#### Feedback Loop (CLKFBO) Divider

The CLKFBO divider is used to divide the feedback signal. This is also referred to as the N-Divider. Effectively, this multiplies the output clock, because the divided feedback must speed up to match the input frequency into the PLL block. The PLL block increases the output frequency until the divided feedback frequency equals the input frequency. The input and output of the feedback divider must be within the input and output frequency ranges specified in the [LatticeECP3 Family Data Sheet](#). This is checked by IPExpress.

#### Output Clock (CLKOP) Divider

The CLKOP divider serves the dual purposes of squaring the duty cycle of the VCO output and scaling up the VCO frequency into the 500MHz to 1000MHz range to minimize jitter. This is also referred to as the V-Divider.

#### CLKOK Divider

The CLKOK divider acts as a source for the global clock nets. This is also referred to as the K-Divider. It divides the CLKOP or CLKOS signal (user selectable) of the PLL by the value of the divider to produce lower frequency clock.

#### CLKOK2 Divider

The CLKOK2 divider always works off the CLKOP signal and has a fixed value of 3. The CLKOK2 signal can be used for generating 140 MHz from 420 MHz to support SPI4.2 or for other uses. The first rising edge of CLKOK2 is aligned to the first falling edge of CLKOP (after reset) and the falling edge of CLKOK2 is aligned to the third rising edge of CLKOP. This will show up as a skew between CLKOP and CLKOK2 equal to one-half of the CLKOP period. CLKOK2 is available to be routed as a primary clock.

### Phase Adjustment and Duty Cycle Select (Static Mode)

Users can program CLKOS with Phase and Duty Cycle options. Phase adjustment can be done in 22.5° steps. The duty cycle resolution is 1/16th of a period except 1/16th, 15/16th and 16/16th duty cycle options, which are not supported to avoid minimum pulse violation.

### Dynamic Phase Adjustment (DPHASE) and Dynamic Duty Cycle (DDUTY) Select

The Phase Adjustment and Duty Cycle Select can be controlled in dynamic mode. When this mode is selected, both the Phase Adjustment and Duty Cycle Select must be in dynamic mode. If only one of the features is to be used in dynamic mode, users can manually set the other control inputs to the fixed logic levels of their choice.

### Duty Trim Adjustment

With the LatticeECP3 device family, the duty cycle can be fine-tuned with the Duty Trim Adjustment.

### Fine Delay Adjust

This optional feature is controlled by the input port, WRDEL. See information on the WRDEL input in the next section of this document.

## PLL Inputs and Outputs

### CLKI Input

The CLKI signal is the reference clock for the PLL. It must conform to the specifications in the [LatticeECP3 Family Data Sheet](#) in order for the PLL to operate correctly. The CLKI can be sourced from a dedicated dual-purpose pin or from routing. Please note it is not recommended to use a DCS output as a clock source to this input as a loss of PLL lock can occur.

### RST Input

The PLL reset occurs under two conditions. First, at power-up, the internal power-up reset signal from the configuration block resets the PLL. Second, the user-controlled PLL reset signal RST, provided as part of the PLL module, can be driven by an internally generated reset function or an external pin. This RST signal resets all internal PLL counters, flip-flops (including the M, N, V, K and CLKOK2 Dividers) and the charge pump. When RST goes inactive, the PLL will start the lock-in process, and will take  $t_{LOCK}$  time to complete the PLL lock. Figure 10-8 shows the timing diagram of the RST input. Figure 10-9 shows the timing relationship between the RST input and the CLKI divider output. RST is active high. The RST signal is optional; if unused, tie the input LOW. RST asserts asynchronously and deasserts synchronously.

**Figure 10-8. RST Input Timing Diagram**

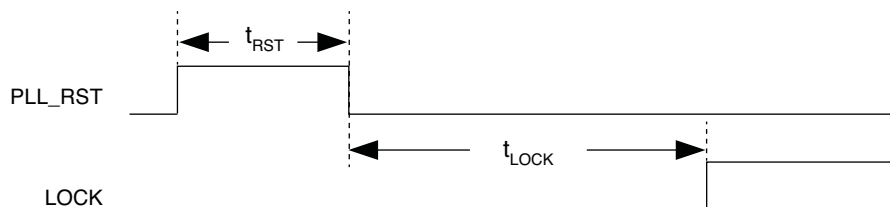
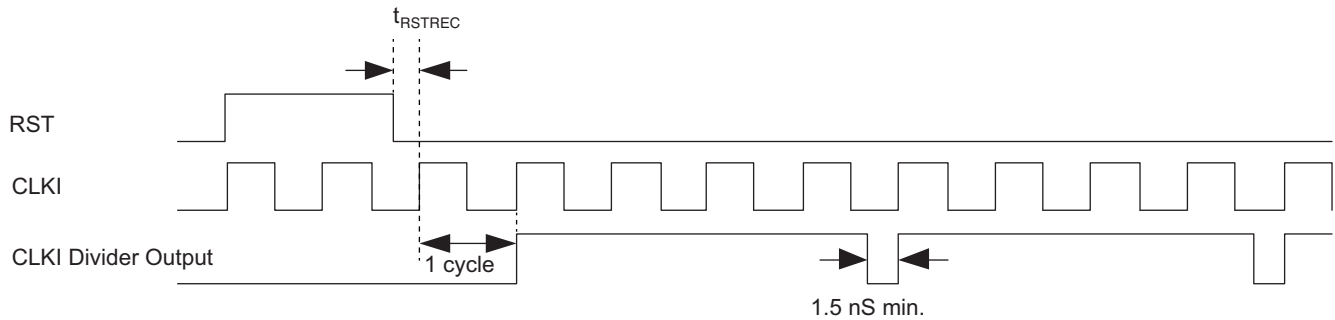


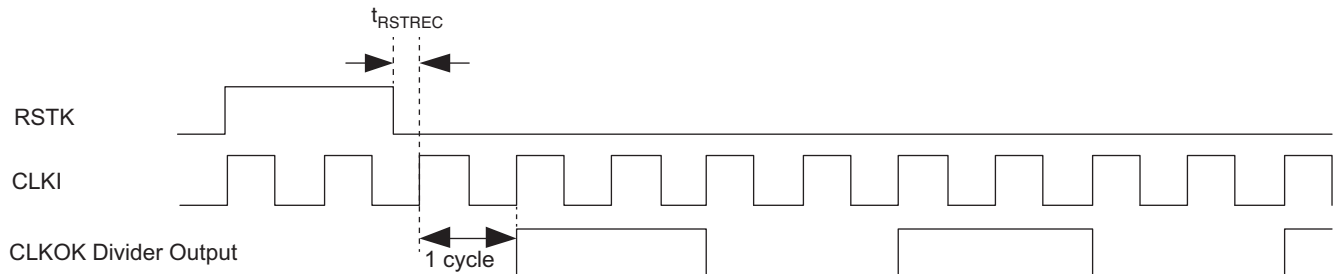
Figure 10-9. RST Input and CLKI Divider Output Timing Diagram (Example: CLKI\_DIV = 4)



## RSTK Input

RSTK is the reset input for the K-Divider (refer to Figure 10-10). This K-Divider reset is used to synchronize the K-Divider output clock to the input clock. LatticeECP3 has an optional gearbox in the I/O cell for both outputs and inputs. The K-Divider reset is useful for the gearbox implementation. RSTK is active high.

Figure 10-10. RSTK Input and CLKOK Divider Output Timing Diagram (Example: CLKOK\_DIV = 4)



## CLKFB Input

The feedback signal to the PLL, which is fed through the feedback divider, can be derived from the Primary Clock net (CLKOP), a preferred pin, directly from the CLKOP divider or from general routing. External feedback allows the designer to compensate for board-level clock alignment. Please note it is not recommended to use a DCS output as a clock source to this input as a loss of PLL lock can occur.

## CLKOP Output

The sysCLOCK PLL main clock output, CLKOP, is a signal available for selection as a primary clock.

## CLKOS Output with Phase and Duty Cycle Select

The sysCLOCK PLL auxiliary clock output, CLKOS, is a signal available for selection as a primary clock. The CLKOS is used when phase shift and/or duty cycle adjustment is desired. The programmable phase shift allows for different phase in increments of  $22.5^\circ$ . The duty select feature provides duty select in 1/16th of the clock period. This feature is also supported in Dynamic Control Mode.

## CLKOK Output with Lower Frequency

The CLKOK is used when a lower frequency is desired. It is a signal available for selection as a primary clock.

## CLKOK2 Output

The CLKOK2 divider always works off the CLKOP signal and has a fixed value of 3. The CLKOK2 signal can be used for generating 140 MHz from 420 MHz to support SPI4.2 or for other uses. The first rising edge of CLKOK2 is aligned to the first falling edge of CLKOP (after reset) and the falling edge of CLKOK2 is aligned to the third rising edge of CLKOP. This will show up as a skew between CLKOP and CLKOK2 equal to one-half of the CLKOP period. CLKOK2 is available to be routed as a primary clock.

## LOCK Output

The LOCK output provides information about the status of the PLL. After the device is powered up and the input clock is valid, the PLL will achieve lock within the specified lock time. Once lock is achieved, the PLL lock signal will be asserted. If, during operation, the input clock or feedback signals to the PLL become invalid, the PLL will lose lock. However, when the input clock completely stops, the LOCK output will remain in its last state, since it is internally registered by this clock. It is recommended to assert PLL RST to re-synchronize the PLL to the reference clock. The LOCK signal is available to the FPGA routing to implement generation of RST. ModelSim® simulation models take two to four reference clock cycles from RST release to LOCK high.

## Dynamic Phase and Dynamic Duty Cycle Adjustment

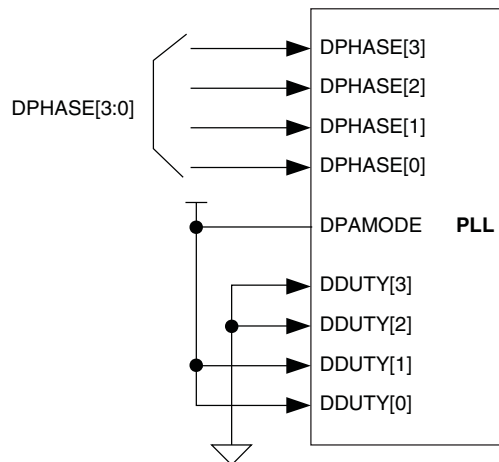
The DPHASE[3:0] port is used with the Dynamic Phase Adjustment feature to allow the user to connect a control signal to the PLL. The DDUTY[3:0] port is used with the Dynamic Duty Adjustment feature to allow the user to connect a control signal to the PLL. The DPHASE and DDUTY ports are listed in Table 10-3.

The Dynamic Phase and Dynamic Duty Cycle Adjustment features will be discussed in more detail in later sections of this document.

**Table 10-3. Dynamic Phase and Duty Cycle Adjust Ports**

Port Name	I/O	Description
DPHASE[3:0]	I	Dynamic Phase Adjust inputs
DDUTY[3:0]	I	Dynamic Duty Cycle Adjust inputs

**Figure 10-11. Example of Dynamic Phase Adjustment with a Fixed Duty Cycle of 3/16th of a Period**



## Dynamic Phase Adjustment/Duty Cycle Select

Phase Adjustment settings are described in Table 10-4.



**Table 10-4. Phase Adjustment Settings**

DPHASE[3:0]	Phase (°)
0000	0
0001	22.5
0010	45
0011	67.5
0100	90
0101	112.5
0110	135
0111	157.5
1000	180
1001	202.5
1010	225
1011	247.5
1100	270
1101	292.5
1110	315
1111	337.5

## Fine Delay Ports

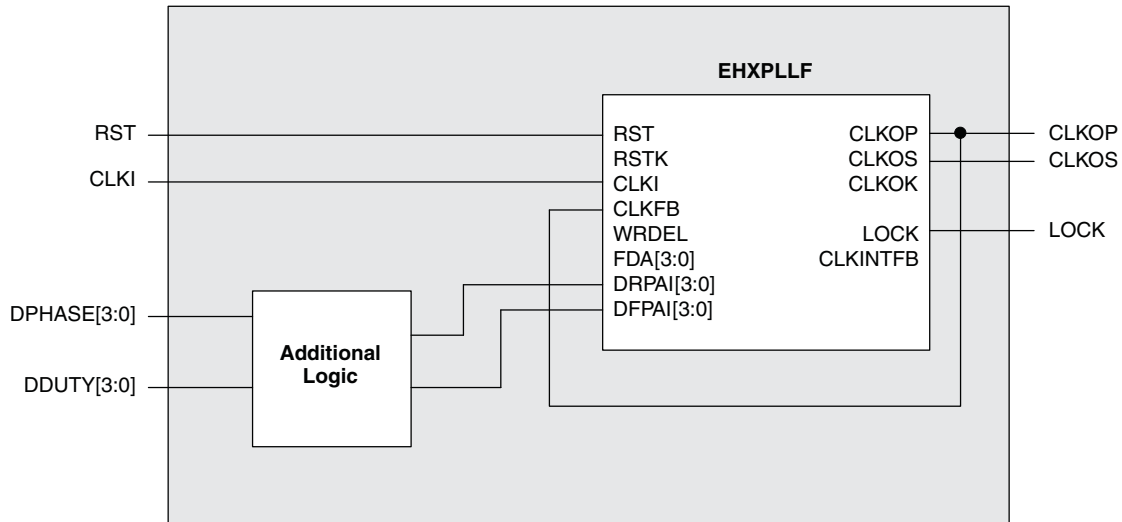
When selecting "Dynamic Mode" and enabling CLKOS, the FINEDELB0-3 ports appear. These ports allow the user to lag the CLKOS output clock with respect to the feedback clock in increments of  $t_{PA} * \text{FINEDELB}(0:3)$ . The  $t_{PA}$  values are located in the [LatticeECP3 Family Data Sheet](#).

The FINEDELA port is enabled with the FINEDELA checkbox on the GUI. This port allows you to push CLKOS ahead of the feedback clock by ~70 ps. This signal is an active high pulse.

## LatticeECP3 PLL Modules

When the user creates a PLL module using IPexpress, the module will consist of a wrapper around the PLL library element and any additional logic required for the module. Figure 10-12 is a diagram of a typical PLL module. The module port names can be different than the library element in some cases. The user will see the module port names in the IPexpress window and also in the source code file for the generated module. These are the ports that will be connected in the user's design. IPexpress also creates an instantiation template file that shows the user how to instantiate the PLL module in their design. The user can import the \*.LPC file (in ispLEVER™, or \*.IPX in Diamond) into their project or the generated source code file.

Figure 10-12. LatticeECP3 Typical PLL Module Generated by IPexpress

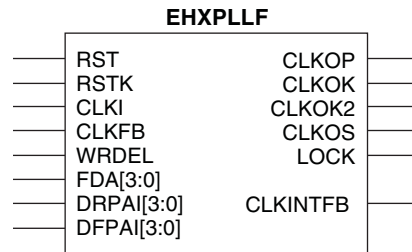


The PLL module shown in Figure 10-12 represents an example where the user has chosen to use the CLKOP and CLKOS ports, with a PLL reset signal, PLL lock signal, and dynamic phase and dynamic duty cycle. It also uses CLKOP feedback so the software will connect the CLKOP signal to the CLKFB port and use the primary clock tree to route this signal. The user would connect their signals to the CLKI, RST, DPHASE[3:0], DDUTY[3:0], CLKOP, CLKOS, and LOCK signals.

## LatticeECP3 PLL Library Definition

One PLL library element is used for LatticeECP3 PLL implementation. Figure 10-13 shows the LatticeECP3 PLL library symbols.

Figure 10-13. LatticeECP3 PLL Library Symbol



## EPLLD Design Migration from LatticeECP2 to LatticeECP3

The EPLLD generated for LatticeECP2 can be used with minor changes. If the configuration does not include Dynamic Phase and Duty Options, the migration is fully supported. If Dynamic Phase and Duty Options are included, the user must tie the DPAMODE port to ground.

### Dynamic Phase/Duty Mode

This mode sets both Dynamic Phase Adjustment and Dynamic Duty Select at the same time. There are two modes, “Dynamic Phase and Dynamic Duty” and “Dynamic Phase and 50% Duty”.

- **Dynamic Phase and 50% Duty**

This mode allows users to set up Dynamic Phase inputs only. The 50% Duty Cycle is handled internally by the ispLEVER software. The DDUTY[3:0] ports are user-transparent in this mode.

- **Dynamic Phase and Dynamic Duty**

This mode allows designers to use both DDPHASE[3:0] and DDUTY[3:0] ports to input dynamic values.

To use Dynamic Phase Adjustment with a fixed duty cycle other than a 50%, simply set the DDUTY[3:0] inputs to the desired duty cycle value. Figure 10-11 illustrates an example circuit.

**Example:** Assume a design uses dynamic phase adjustment and a fixed duty cycle select and the desired duty cycle in 3/16th of a period. The setup should be as shown in Figure 10-11.

Duty Cycle Select settings are described in Table 10-5.

**Table 10-5. Duty Cycle Select Settings**

DDUTY[3:0]	Duty Cycle (1/16th of a Period)	Comment
0000	0	Not Supported
0001	1	Not Supported
0010	2	
0011	3	
0100	4	
0101	5	
0110	6	
0111	7	
1000	8	
1001	9	
1010	10	
1011	11	
1100	12	
1101	13	
1110	14	
1111	15	Not Supported

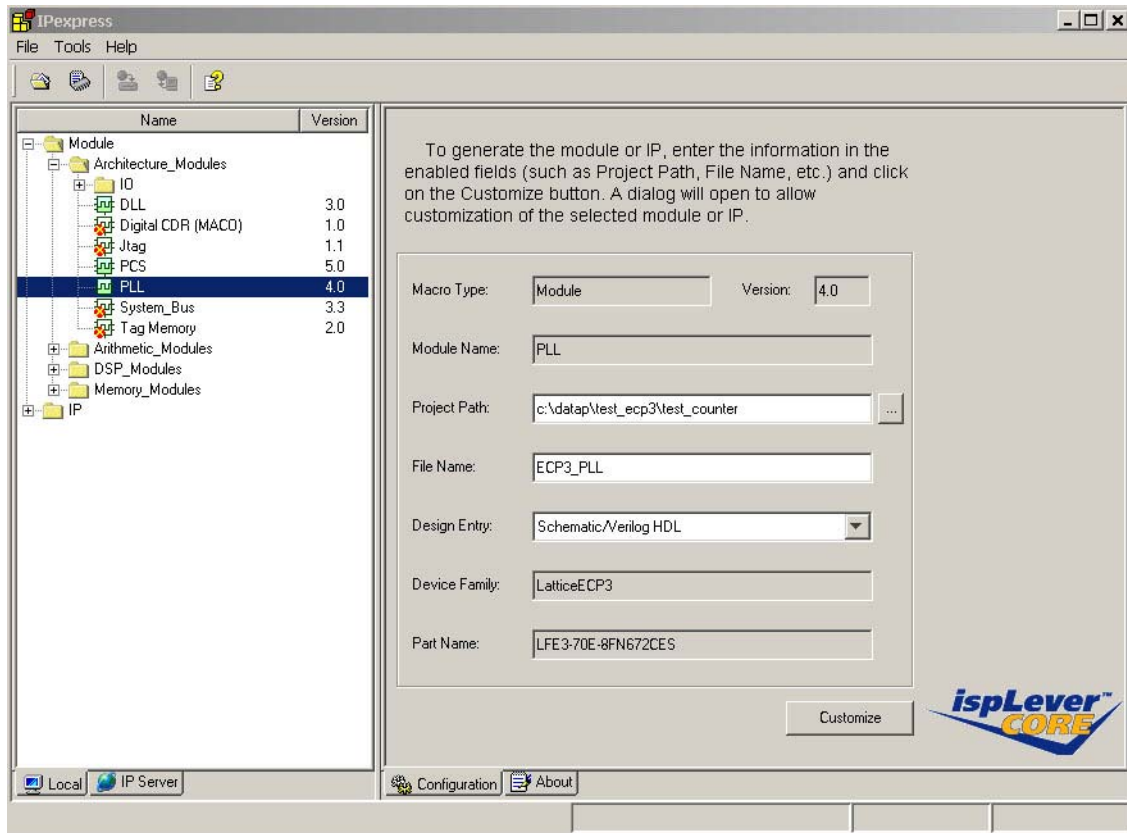
Note: PHASE/DUTY\_CTLN is selected in the GUI 'PLL Phase & Duty Options' box and if it is set to 'Dynamic Mode', then both DDPHASE[3:0] and DDUTY[3:0] inputs must be provided. If one of these inputs is a fixed value, the inputs must be tied to the desired fixed logic levels.

## PLL Usage in IPexpress

IPexpress is used to create and configure a PLL. The graphical user interface is used to select parameters for the PLL. The result is an HDL model to be used in the simulation and synthesis flow.

Figure 10-14 shows the main window when PLL is selected. The only entry required in this window is the module name. Other entries are set to the project settings. Users may change these entries, if desired. After entering the module name of choice, clicking on **Customize** will open the Configuration Tab window as shown in Figure 10-15.

Figure 10-14. ispLEVER IPexpress Main Window (see Appendix C Figure 10-39 for Diamond Equivalent)



## Configuration Tab

The Configuration Tab lists all user accessible attributes with default values set. Upon completion, clicking **Generate** will generate source and constraint files. Users may choose to use the \*.LPC file (for ispLEVER, or \*.IPX file for Diamond) to load parameters.

## Configuration Modes

There are two modes that can be used to configure the PLL in the Configuration Tab, Frequency Mode and Divider Mode.

**Frequency Mode:** In this mode, the user enters input and output clock frequencies and the software calculates the divider settings. If the output frequency entered is not achievable the nearest frequency will be displayed in the 'Actual' text box. After input and output frequencies are entered, clicking the **Calculate** button will display the divider values.

**Divider Mode:** In this mode, the user sets the divider settings with input frequency. Users must choose the CLKOP Divider value to maximize the  $f_{VCO}$  and achieve optimum PLL performance. After input frequency and divider settings are set, clicking the **Calculate** button will display the frequencies. Figure 10-15 shows the Configuration Tab.

Figure 10-15. ispLEVER LatticeECP3 PLL Configuration Tab (see Appendix C Figure 10-40 for Diamond Equivalent)

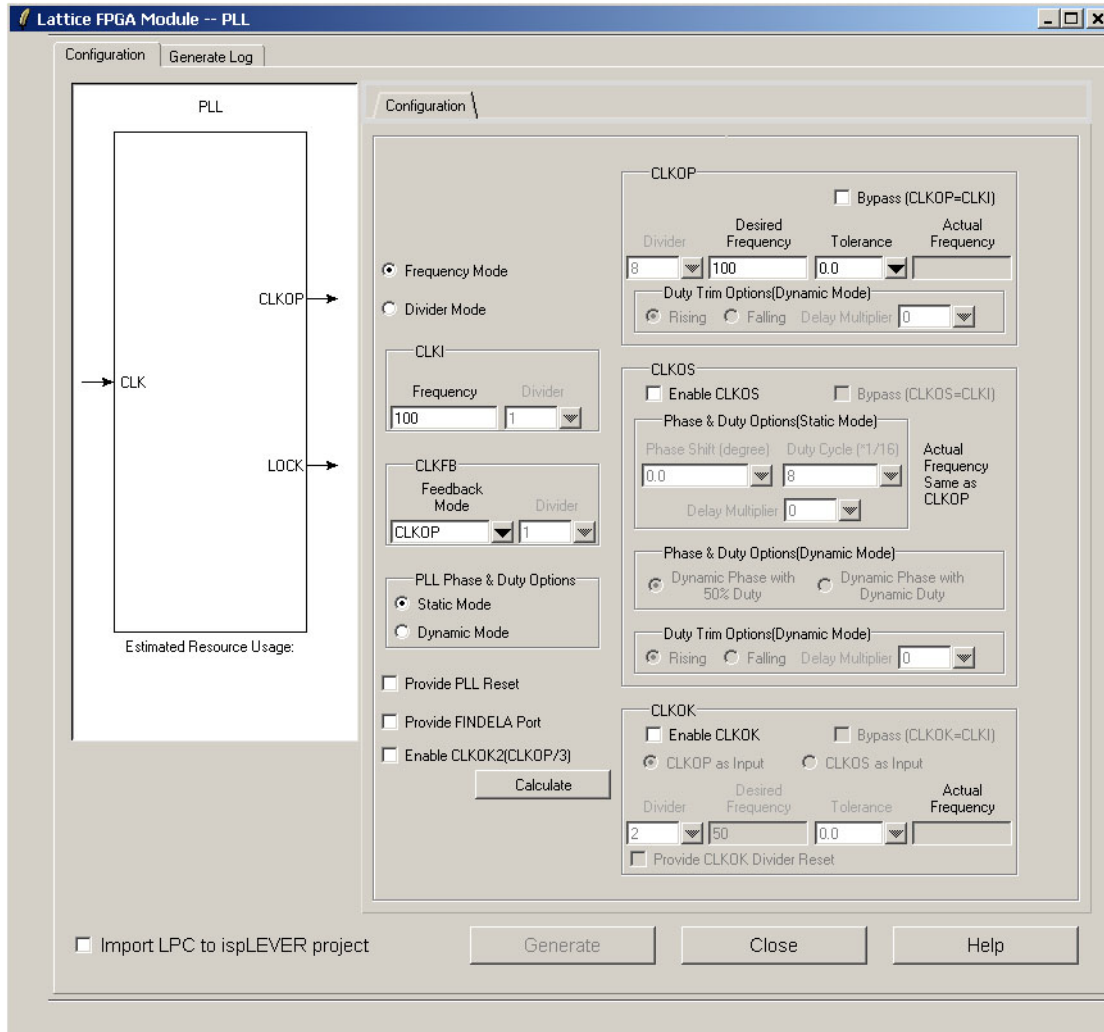


Table 10-6 describes the user parameters in the IPexpress GUI and their usage.

Table 10-6. User Parameters in the IPexpress GUI

User Parameters		Description	Range
Frequency Mode		User desired CLKI and CLKOP frequency	ON/OFF
Divider Mode		User desired CLKI frequency and dividers settings	ON/OFF
CLKI	Frequency	Input Clock frequency	2 MHz to 500 MHz
	Divider	Input Clock Divider Setting (Divider Mode)	1 to 64
CLKFB	Feedback Mode	Feedback Mode	Internal, CLKOP, CLKOS, User Clock
	Divider	Feedback Clock Divider Setting (Divider Mode)	1 to 64
PLL Phase & Duty Options	Static Mode	CLKOS Phase/Duty in Static Mode	ON/OFF
	Dynamic Mode	CLKOS Dynamic Mode Phase/Duty Setting	ON/OFF

**Table 10-6. User Parameters in the IPexpress GUI (Continued)**

User Parameters		Description	Range
CLKOP	Bypass	Bypass PLL: CLKOP = CLKI	ON/OFF
	Desired Frequency	User enters desired CLKOP frequency	4 MHz to 500 MHz
	Divider	CLKOP Divider Setting (Divider Mode)	2, 4, 8, 16, 32, 48, 64, 80, 96, 112, 128
	Tolerance	CLKOP tolerance users can tolerate	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0
	Actual Frequency	Actual frequency achievable. Read only	—
	Rising	Rising Edge Trim	ON/OFF
	Falling	Falling Edge Trim	ON/OFF
	Delay Multiplier	Number of delay steps	0 to 7
CLKOS	Enable	Enable CLKOS output clock	ON/OFF
	Bypass	Bypass PLL: CLKOS = CLKI	ON/OFF
	Phase Shift	CLKOS Static Phase Shift	0°, 22.5°, 45°..337.5°
	Duty Cycle	CLKOS Static Duty Cycle	2 to 14
	Phase and Duty Options	Dynamic Phase with 50% Duty	ON/OFF
		Dynamic Phase with Dynamic Duty	ON/OFF
	Rising	Rising Edge Trim	ON/OFF
	Falling	Falling Edge Trim	ON/OFF
Delay Multiplier	Number of Delay steps	0 to 3	
CLKOK	Enable	Enable CLKOS output clock	ON/OFF
	Bypass	Bypass PLL: CLKOK = CLKI	ON/OFF
	Desired Frequency	User enters desired CLKOK frequency	0.03125 to 250 MHz
	CLKOK input	Select input source for CLKOK	CLKOP/CLKOS
	Frequency	User enters desired CLKOK frequency	31.25 kHz to 250 MHz
	Divider	CLKOK Divider Setting	2, 4, 6, ... 126, 128
	Tolerance	CLKOK tolerance users can tolerate	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0
	Actual Frequency	Actual frequency achievable. Read only	—
CLKOK2	Enable	Enable CLKOK2 output clock	ON/OFF
Provide PLL Reset		Provide PLL Reset Port (RESET)	ON/OFF
Provide CLKOK Divide Reset		Provide CLKOK Reset Port (RSTK)	ON/OFF
Provide FINDELA Port		Provide CLKOS Fine Delay Port (WRDEL)	ON/OFF
Import LPC to ispLEVER Project		Import .lpc file to ispLEVER project	ON/OFF
Import IPX into Diamond Project		Import .lpc file to Diamond project	ON/OFF

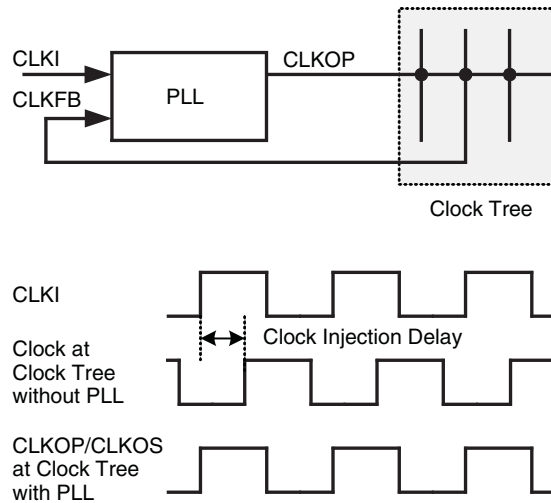
## PLL Modes of Operation

PLLs have many uses within a logic design. The two most popular are Clock Injection Removal and Clock Phase Adjustment. These two modes of operation are described below.

### PLL Clock Injection Removal

In this mode the PLL is used to reduce clock injection delay. Clock injection delay is the delay from the input pin of the device to a destination element such as a flip-flop. The phase detector of the PLL aligns the CLKI with CLKFB. If the CLKFB signal comes from the clock tree (CLKOP), then the PLL delay and the clock tree delay is removed. Figure 10-16 illustrates an example block diagram and waveform.

Figure 10-16. Clock Injection Delay Removal Application

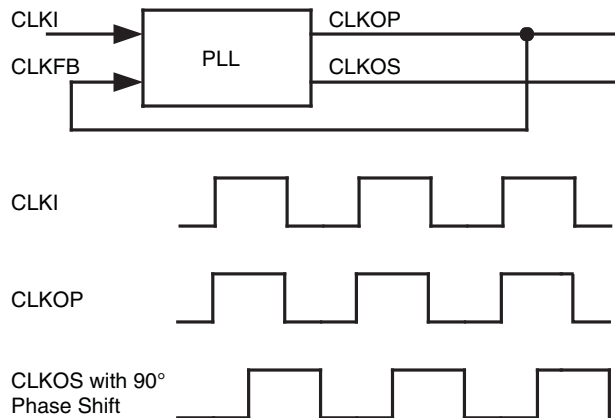


### PLL Clock Phase Adjustment

Refer to Figure 10-17. In this mode the PLL is used to create fixed phase relationships in 22.5° increments. Creating fixed phase relationships is useful for forward clock interfaces where a specific relationship between clock and data is required.

The fixed phase relationship can be used between CLKI and CLKOS or between CLKOP and CLKOS.

Figure 10-17. CLKOS Phase Adjustment from CLKOP



### IPexpress Output

There are two IPexpress outputs that are important for use in the design. The first is the <module\_name>.[v|h]d file. This is the user-named module that was generated by the tool to be used in both synthesis and simulation flows. The second is a template file, <module\_name>\_tmpl.[v|h]d. This file contains a sample instantiation of the module. This file is provided for the user to copy/paste the instance and is not intended to be used in the synthesis or simulation flows directly.

For the PLL, IPexpress sets attributes in the HDL module that are specific to the data rate selected. Although these attributes can be easily changed, they should only be modified by regenerating the package in IPexpress so that the performance of the PLL is maintained. After the map stage in the design flow, FREQUENCY preferences will be included in the preference file to automatically constrain the clocks produced from the PLL.

## Notes on PLL Usage

The GPLL CLKOP should be used as the feedback source to optimize PLL performance.

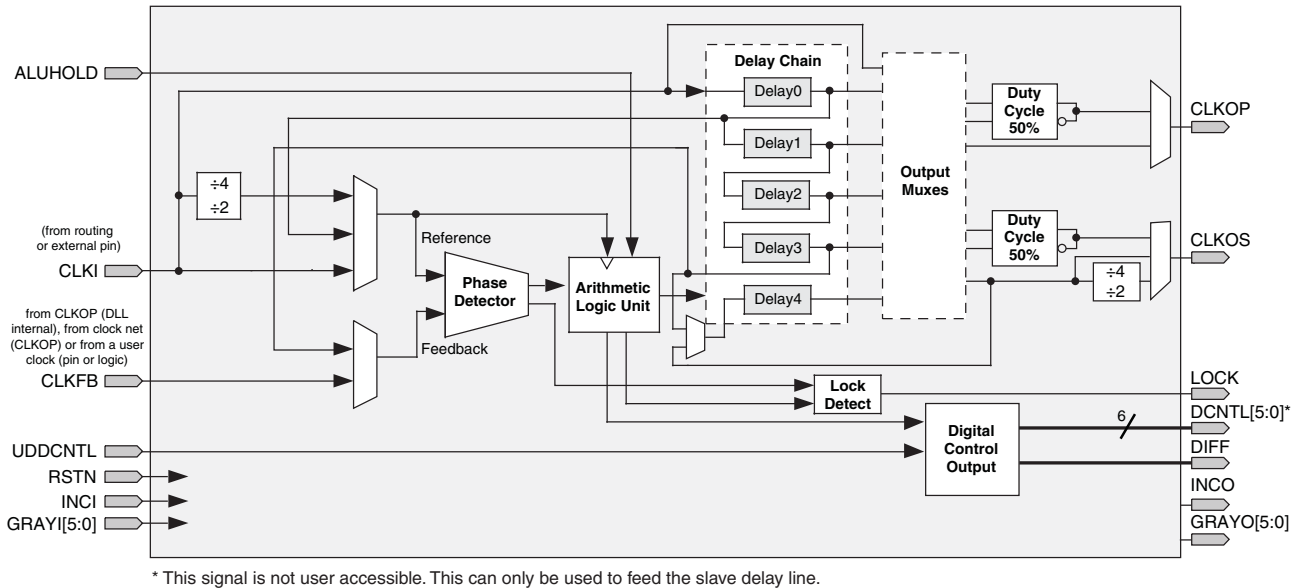
Most designers use PLLs for clock tree injection removal mode and the CLKOP should be assigned to a primary clock. This is done automatically by the software unless otherwise specified by the user.

CLKOP can route only to CLK0 to CLK5, while CLKOS/CLKOK can route to all primary clocks (CLK0 TO CLK7).

## sysCLOCK DLL

The LatticeECP3 DLL provides features such as clock injection delay removal, time reference delay (90° phase delay), and output phase adjustment. The DLL performs clock manipulation by adding delay to the CLKI input signal to create specific phase relationships. There are two types of outputs of the DLL. The first are clock signals similar to the PLL CLKOP and CLKOS. The other type of output is a delay control vector (DCNTL[5:0]). The delay control vector is connected to a Slave Delay Line (DLLDEL) element. Figure 10-18 provides a block diagram of the LatticeECP3 DLL.

Figure 10-18. LatticeECP3 DLL Block Diagram



Both clock injection delay removal and output phase adjustment use only the clock outputs of the DLL. Time reference delay modes use the delay control vector output. Specific examples of these features are discussed later in this document.

## DLL Overview

The LatticeECP3 DLL is created and configured by IPexpress. The following is a list of port names and descriptions for the DLL. There are two library elements used to implement the DLL: CIDDLLB (Clock Injection Delay), and TRDDLLB (Time Reference Delay). IPexpress will wrap one of these library elements to create a customized DLL module based on user selections.

## DLL Inputs and Outputs

### CLKI Input

The CLKI signal is the reference clock for the DLL. The CLKI input can be sourced from any type of FPGA routing and pin. The DLL CLKI input has a preferred pin per DLL which provides the lowest latency and best case performance.



**CLKFB Input**

The CLKFB input is available only if the user chooses to use a user clock signal for the feedback. If internal feedback or CLKOS/CLKOP is used for the feedback, this connection will be made inside the module. In Clock Injection Delay Removal mode, the DLL will align the input clock phase with the feedback clock phase by delaying the input clock.

**CLKOP Output**

An output of the DLL based on the CLKI rate. The CLKOP output can drive primary and edge clock routing.

**CLKOS Output**

An output of the PLL based on the CLKI rate which can be divided and/or phase shifted. The CLKOS output can drive the primary and edge clock routing.

**DCNTL[5:0] Output**

This output of the DLL is used to delay a signal by a specific amount. The DCNTL[5:0] vector can only connect to a Slave Delay Line element.

**DIFF Output**

Active high difference indicator. Active when DCNTL output is different than the internal setting and an update is needed.

**UDDCNTL Input**

This input is used to enable or disable updating of the DCNTL[5:0]. To ensure that the signal is captured by the synchronizer in the DLL block, it must be driven high for a time equal to at least two clock cycles when an update is required. If the signal is driven high and held in that state, the DCNTL[5:0] outputs are continuously updated.

**ALUHOLD Input**

This active high input stops the DLL from adding and subtracting delays to the CLKI signal. The DCNTL[5:0], CLKOP, and CLKOS outputs will still be valid, but will not change from the current delay setting.

**LOCK Output**

Active high lock indicator output. The LOCK output will be high when the CLKI and CLKFB signal are in phase. If the CLKI input stops the LOCK output will remain asserted. Since the clock is stopped, there is no clock to deassert the LOCK output. Note that this is different from the operation of the PLL, where the VCO continues to run when the input clock stops. The LOCK output transitions are glitch-free.

**RSTN**

Active low reset input to reset the DLL. The DLL can optionally be reset by the GSRN as well. It is recommended that if the DLL requires a reset, the reset should not be the same as the FPGA logic reset. Typically, logic requires that a clock is running during a reset condition. If the data path reset also resets the DLL, the source of the logic clock will stop and this may cause problems in the logic. RSTN asserts asynchronously and deasserts synchronously.

**GRAYO Output**

Gray-coded digital control bus to other DLLs. This bus, together with the GRAYI bus and INCO and INCI signals, enables DLLs to be safely cascaded. The buses are Gray-coded to prevent glitches in the transfer of the control signals. The INCO / INCI signal accompanies the GRAYO / GRAYI bus, and indicates when an incremental adjustment is being passed.

**GRAYI Input**

Gray-coded digital control bus from another DLL in time reference mode. See description of the GRAYO output bus, above.

**INCO Output**

Active high incremental indicator to other DLLs. See description of the GRAYO output bus, above.

### INCI Input

Active high incremental indicator from another DLL. See description of the GRAYO output bus, above.

### DLL Attributes

The LatticeECP3 DLL utilizes several attributes that allow the configuration of the DLL through source constraints, IPexpress and preference files. The following section details these attributes and their usage.

#### DLL Lock on Divide by 2 or Divide by 4 CLKOS Output

The LatticeECP3 DLL allows 'divide by 2' or 'divide by 4' CLKOS outputs. Two optional 'divide by 2' and 'divide by 4' blocks are placed at the CLKI input as well as the CLKOS and this enables the use of divided CLKOS in the DLL feedback path. This allows the DLL to perform clock injection removal on a 'divide by 2' or 'divide by 4' clock, which is useful for DDRX2 and DDRX4 modes of I/O buffer operation.

When this optional clock divider is used only in the CLKOS output path, it allows the DLL to output two time-aligned clocks at different frequencies. When the divider is set to divide by 2 or divide by 4, a 'dummy' delay is inserted in the CLKOP output path to match the clock to Q delay of the CLKOS divider.

#### DLL Lock Time Control

The DLL will lock when the CLKI and CLKFB phases are aligned. In a simulation environment, the lock time is fixed to 100µs (default). This value can be changed through an HDL parameter or preference (for the back annotation simulation). The DLL contains a parameter named LOCK\_DELAY which accepts an integer value for the total time in µs until the lock output goes high. Below is an example of how to set this value for front-end simulation.

#### Verilog:

```
defparam mydll.mypll_0_0.LOCK_DELAY=500;
mydll dll_inst(.CLKI(clkin), .CLKOP(clk1), .CLKOS(clk2),
```

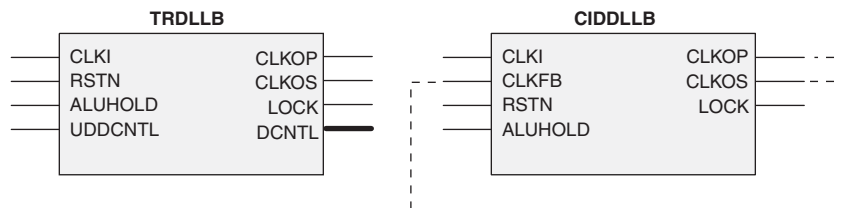
#### VHDL:

Not supported. For back annotation simulation LOCK\_DELAY needs to be set in the preference file. Below is an example for the PLL.

```
ASIC "pll/pll_0_0" TYPE "EHXPLLF" LOCK_DELAY=200;
```

### DLL Library Symbols

Figure 10-19. DLL Library Symbols



### DLL Library Definitions

The Lattice library contains library elements to allow designers to utilize the DLL. These library elements use the DLL attributes defined in the "DLL Attributes" section.

The two modes of operation are presented as library elements as listed below.

**Table 10-7. DLL Library Elements**

Library Element Name	Mode of Operation	Description
TRDLLB	Time Reference Delay DLL	This mode generates four phases of the clock, 0°, 90°, 180°, 270°, along with the control setting used to generate these phases.
CIDLLB	Clock Injection Delay DLL (Four Delay Cell Mode)	This mode removes the clock tree delay, aligning the external feedback clock to the reference clock. It has a single output coming from the fourth delay block.

## DLL Library Element I/Os

**Table 10-8. DLL Library Element I/O Descriptions**

Signal	I/O	Description
CLKI	I	Clock input pin from dedicated clock input pin, other I/O or logic block.
CLKFB	I	Clock feedback input pin from dedicated feedback input pin, internal feedback, other I/O or logic block. This signal is not user selectable.
RSTN	I	Active low synchronous reset. From dedicated pin or internal node.
ALUHOLD	I	“1” freezes the ALU. For TRDLLA and CIDLLA.
UDDCNTL	I	Active high synchronous enable signal from CIB for updating digital control to PIC delay. It must be driven high at least two clock cycles.
DCNTL[5:0]	O	Digital delay control code signals.
CLKOP	O	The primary clock output for all possible modes.
CLKOS	O	The secondary clock output with finer phase shift and/or division by 2 or by 4.
LOCK	O	Active high phase lock indicator. Lock means the reference and feedback clocks are in phase.

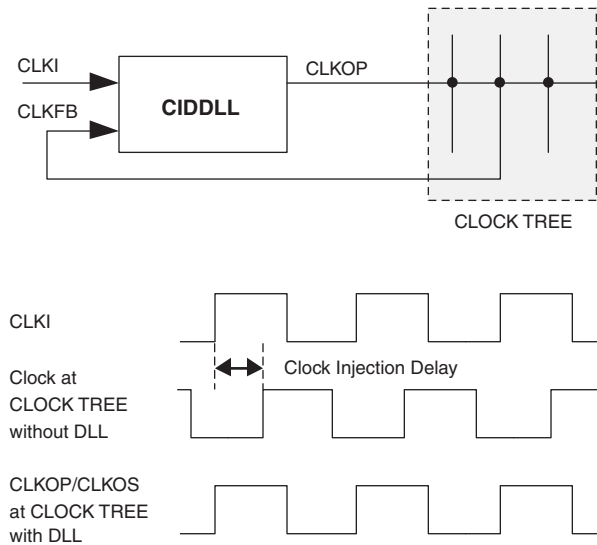
Note: Refer to the [LatticeECP3 Family Data Sheet](#) for frequency specifications.

## DLL Modes of Operation

### Clock Injection Removal Mode (CIDLLB)

The DLL can be used to reduce clock injection delay (CIDLLB). Clock injection delay is the delay from the input pin of the device to a destination element such as a flip-flop. The DLL will add delay to the CLKI input to align CLKI to CLKFB. If the CLKFB signal comes from the clock tree (CLKOP, CLKOS) then the delay of the DLL and the clock tree will be removed from the overall clock path. Figure 10-20 shows a circuit example and waveform.

Figure 10-20. Clock Injection Delay Removal via DLL



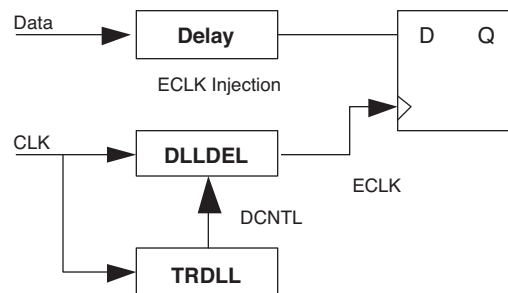
Clock injection removal mode can also provide a DCNTL port. When using the DCNTL, the DLL delay will be limited to the range of the DCNTL vector. Therefore, IPexpress will restrict the CLKI rate from 300MHz to 500MHz.

**Time Reference Delay Mode (TRDLLB: 90-Degree Phase Delay)**

The Time Reference Delay (TRDDLLB) mode of the DLL is used to calculate 90 degrees of delay to be placed on the DCNTL vector. This is a useful mode in delaying a clock 90 degrees for use in clocking a DDR type interface.

Figure 10-21 provides a circuit example of this mode.

Figure 10-21. Time Reference Delay Circuit Example

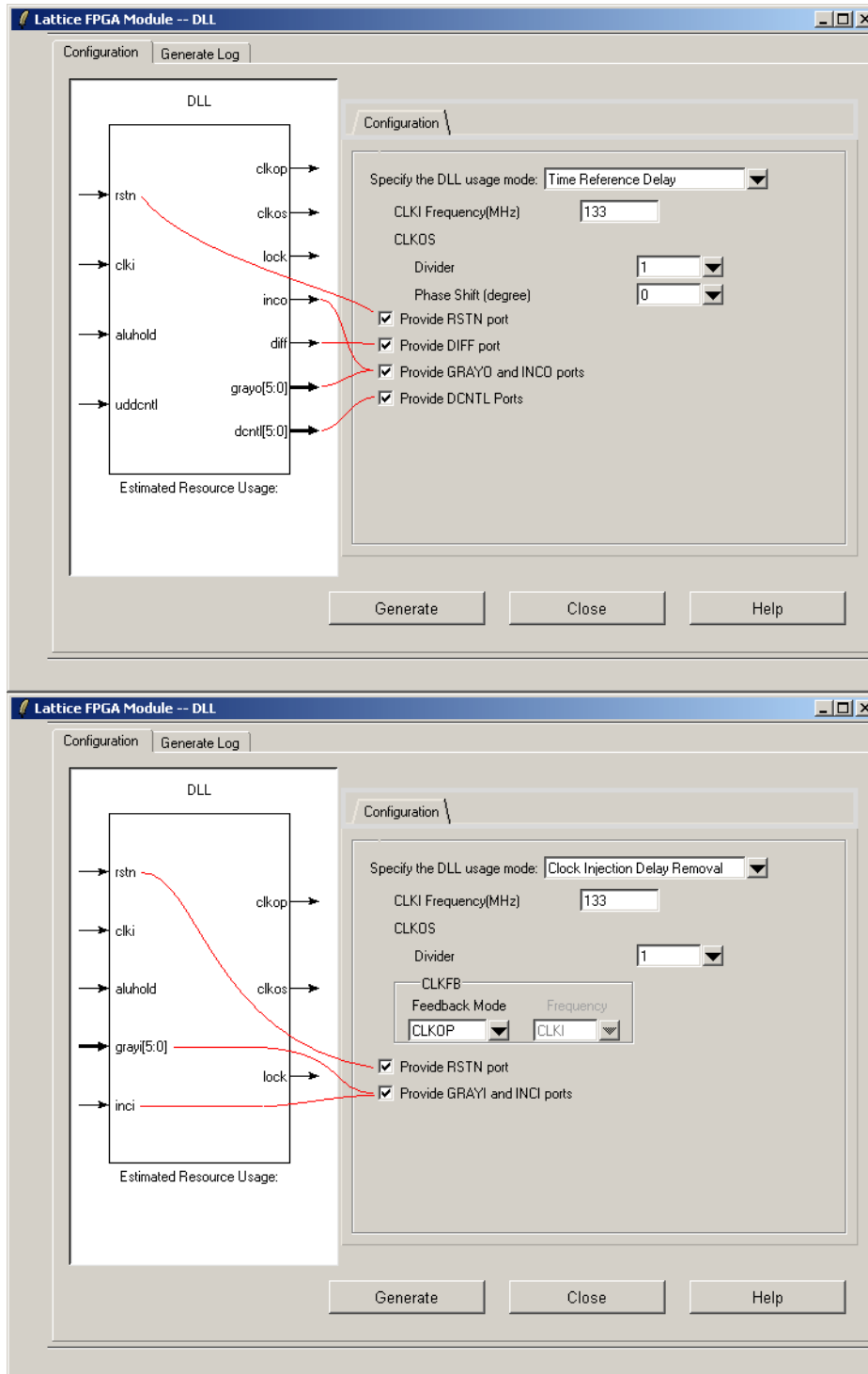


In this mode, CLKI accepts a clock input. The DLL produces a DCNTL vector that will delay an input signal by 90 degrees of a full period of the CLKI signal. This DCNTL vector can then be connected to a Slave Delay Line (DLL-DELB) to delay the signal by 90 degrees of the full period of CLKI.

**DLL Usage in IPexpress**

IPexpress is used to create and configure a DLL. The IPexpress graphical user interface, shown in Figure 10-13, allows users to select parameters for the DLL. Parameters are described in Table 10-9. The result is an HDL model to be used in the simulation and synthesis flow.

Figure 10-22. ispLEVER LatticeECP3 IPexpress DLL Configuration Tab (see Figure 10-41 for Diamond Equivalent)



**Table 10-9. User Parameters in the IPexpress DLL GUI**

User Parameter	Description	Range	Default
DLL Usage Mode	User desired operation mode	Time Reference Delay, Clock Injection Delay Removal	Clock Injection Delay Removal
CLKI Frequency (MHz)	Input CLKI frequency	133-500 MHz	133 MHz
CLKOS Divider	Output CLKOS Divider Setting	1, 2, 4	1
CLKOS Phase Shift (degrees)	Output CLKOS Phase Shift Setting	0° to 360° in 11° steps	0°
CLKFB Feedback Mode	Feedback Clock mode (source of feedback)	CLKOP, CLKOS, User Clock	CLKOP
CLKFB Frequency	Feedback Clock source frequency (CLKI divided by 1, 2 or 4)	CLKI, CLKI/2, CLKI/4	CLKI
Provide RSTN Port	Provide reset port (active-LO)	ON/OFF	ON
Provide DIFF Port	Provide DIFF port	ON/OFF	OFF
Provide GRAYO Port (Time Ref Delay mode only)	These inputs/outputs are used to safely cascade PLLs/DLLs. Refer to text for details.	ON/OFF	OFF
Provide GRAYI Port (Clk Inj Dly Rmvl mode only)		ON/OFF	OFF
Provide INCO Port (Time Ref Delay mode only)	These inputs/outputs are used to safely cascade PLLs/DLLs. Refer to text for details.	ON/OFF	OFF
Provide INCI Port (Clk Inj Dly Rmvl mode only)		ON/OFF	OFF
Provide DCNTL Port	Provide Delay Control Vector output port	ON/OFF	OFF
Import PIX to Diamond Project (Diamond only)	Import .IPX file to project	YES/NO	NO

### PLL/DLL Cascading

It is possible to connect several arrangements of PLLs and DLLs. There are three possible cascading schemes:

- PLL to PLL
- PLL to DLL
- DLL to DLL

It is not possible to connect the DLL to a PLL. The DLL produces abrupt changes on its output clocks when changing delay settings. The PLL sees this as radical phase changes that prevent the PLL from locking correctly. A DLL in Static Delay Mode can, however, be used to set fine phase delays, and it is generally best to do this with the DLL placed in front of the PLL.

### IPexpress Output

There are two outputs of IPexpress that are important for use in the design. The first is the <module\_name>.[v|vhd] file. This is the user-named module that was generated by the tool to be used in both synthesis and simulation flows. The second file is a template file <module\_name>\_tmpl.[v|vhd]. This file contains a sample instantiation of the module. This file is only provided for the user to copy/paste the instance and is not intended to be used in the synthesis or simulation flows directly.

For the PLL/DLL, IPexpress sets attributes in the HDL module created that are specific to the data rate selected. Although these attributes can easily be changed, they should only be modified by re-running the GUI so that the performance of the PLL/DLL is maintained. After the map stage in the design flow, FREQUENCY preferences will be included in the preference file to automatically constrain the clocks produced from the PLL/DLL.

## DLLDEL (Slave Delay Line)

The Slave Delay line is designed to generate the desired delay in DDR/SPI4 applications. The delay control inputs (DCNTL[5:0]) are fed from the general purpose DLL outputs. The library element definitions are described in Figure 10-23 and Table 10-10.

Figure 10-23. DLLDELB Library Symbol

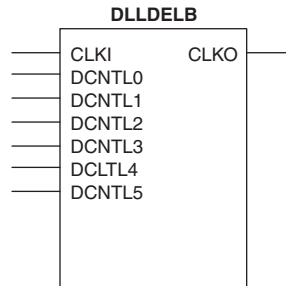


Table 10-10. DLLDELB I/O

Name	I/O	Description
CLKI	I	Clock Input
DCNTL[5:0]	I	Delay Control Bits
CLKO	O	Clock Output

### DLLDELB Declaration in VHDL Source Code

```

COMPONENT DLLDELB
  PORT
    (
      CLKI :IN std_logic;
      DCNTL0 :IN std_logic;
      DCNTL1 :IN std_logic;
      DCNTL2 :IN std_logic;
      DCNTL3 :IN std_logic;
      DCNTL4 :IN std_logic;
      DCNTL5 :IN std_logic;
      CLKO :OUT std_logic
    );
END COMPONENT;

```

```

begin
  DLLDELBinst0: DLLDELB1
    PORT MAP (
      CLKI => clkisig,
      DCNTL0 => dcntl0sig,
      DCNTL1 => dcntl1sig,
      DCNTL2 => dcntl2sig,
      DCNTL3 => dcntl3sig,
      DCNTL4 => dcntl4sig,
      DCNTL5 => dcntl5sig,
      CLKO => clkosig
    );

```

```

end

```

**DLLDELB Usage with TRDLLB - Verilog - Example**

Note: DLL0(TRDLLB) must be generated by IPexpress as a sub-module

```
module dldel_top (rst,d,clkkin,clkout,aluhold,uddcntl,q);

input rst,d,clkkin,aluhold,uddcntl;
output clkout,q;

wire [5:0]DCntl_int;
reg qint;

DLL0 dllinst0 (.clk(clkin), .aluhold(aluhold), .uddcntl(uddcntl), .clkop(), .clkos(),
               .dcntl(DCntl_int),.lock());
DLLDELB delinst0 (.CLKI(clkin),.DCNTL0(DCntl_int[0]),.DCNTL1(DCntl_int[1]),
                 .DCNTL2(DCntl_int[2]), .DCNTL3(DCntl_int[3]), .DCNTL4(DCntl_int[4]),
                 .DCNTL5(DCntl_int[5]), .CLKO(clk90)); //synthesis syn_black_box

assign clkout = clk90;
assign q = qint;

always@(posedge clk90 or negedge rst)
    if (~rst)
        qint =1'b0;
    else
        qint = d;

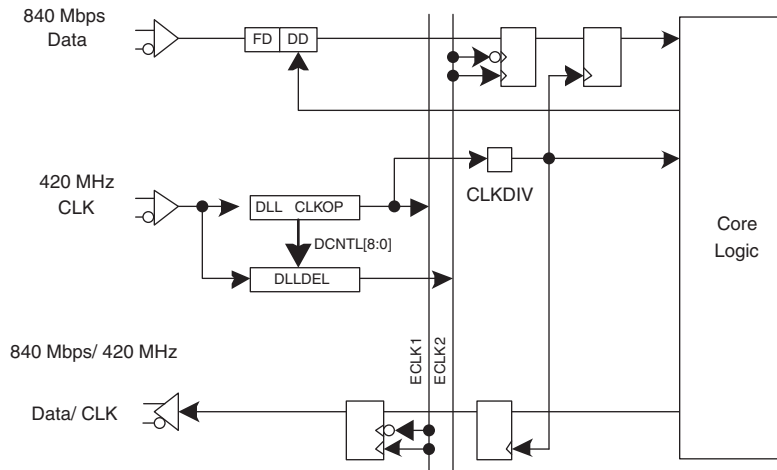
endmodule
```

**DLLDELB Application Example**

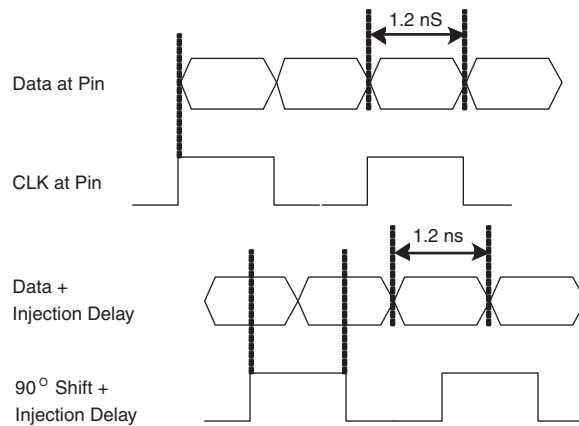
Figure 10-24 shows an example DLLDEL application. As shown in the timing diagram, DLLDEL shifts the clock by 90 degrees to center both edges in the middle of data window.



Figure 10-24. SPI4.2 and DDR Registers Interface Application



FD: Fixed Delay  
DD: Dynamic Delay  
Users can select the delay setting in IPexpress.



## DQSDLL and DQSDEL

There is another combination of DLL and Slave Delay Line, DQSDLL and DQSDEL, in the LatticeECP3 device family. This pair is similar in design and function to DLL and DLLDEL, but usage is limited to DDR implementation. For additional information, see TN1180, [LatticeECP3 High-Speed I/O Interface](#).

## Clock Dividers (CLKDIV)

The clock divider divides the high-speed clock by 1, 2, 4 or 8. All the outputs have matched input to output delay. CLKDIV can take as its input the edge clocks and the CLKOP of the PLL or DLL. The divided outputs drive the primary clock and are also available for general routing or secondary clocks. The clock dividers are used for providing the low speed FPGA clocks for shift registers (x2, x4, x8) and DDR/SPI4 I/O logic interfaces.

## CLKDIV Library Definition

Users can instantiate CLKDIV in the source code as defined in this section. Figure 10-25 and Tables 10-11 and 10-12 describe the CLKDIVB definitions.

Figure 10-25. CLKDIV Library Symbol

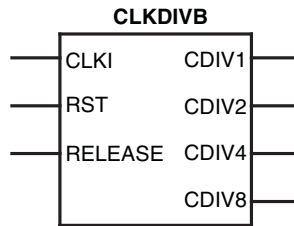


Table 10-11. CLKDIVB Port Definition

Name	Description
CLKI	Clock Input
RST	Reset Input, asynchronously forces all outputs low.
RELEASE	Releases outputs synchronously to input clock.
CDIV1	Divided BY 1 Output
CDIV2	Divided BY 2 Output
CDIV4	Divided BY 4 Output
CDIV8	Divided BY 8 Output

Table 10-12. CLKDIVB Attribute Definition

Name	Description	Value	Default
GSR	GSR Enable	ENABLED/DISABLED	DISABLED

## CLKDIV Declaration in VHDL Source Code

```

COMPONENT CLKDIVB
-- synthesis translate_off
  GENERIC (
    GSR : in String);
-- synthesis translate_on
  PORT (
    CLKI,RST, RELEASE:IN    std_logic;
    CDIV1, CDIV2, CDIV4, CDIV8:OUT  std_logic);
END COMPONENT;

attribute GSR : string;
attribute GSR of CLKDIVinst0 : label is "DISABLED";

begin

CLKDIVinst0:          CLKDIVB
-- synthesis translate_off
  GENERIC MAP(
    GSR              => "disabled"
  );
-- synthesis translate_on
  PORT MAP(
    CLKI              => CLKIsig,
    RST               => RSTsig,
    RELEASE           => RELEASEsig,
    CDIV1             => CDIV1sig,

```

```

CDIV2      => CDIV2sig,
CDIV4      => CDIV4sig,
CDIV8      => CDIV8sig
);

```

### CLKDIV Usage with Verilog - Example

```

module clkdiv_top(RST,CLKI,RELEASE,CDIV1,CDIV2,CDIV4,CDIV8);

input  CLKI,RST,RELEASE;
output CDIV1,CDIV2,CDIV4,CDIV8;

CLKDIVB CLKDIBinst0 (.RST(RST),.CLKI(CLKI),.RELEASE(RELEASE),
    .CDIV1(CDIV1),.CDIV2(CDIV2),.CDIV4(CDIV4),.CDIV8(CDIV8));

defparam CLKDIBint0.GSR = "DISABLED"

endmodule

```

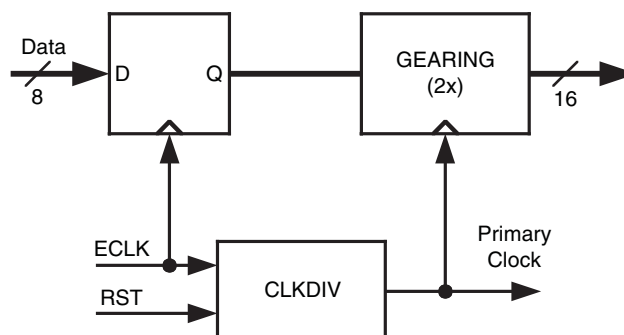
### CLKDIV Example Circuits

The clock divider (CLKDIV) can divide a clock by 2, 4 or 8 and drives a primary clock network. Clock dividers are useful for providing the low speed FPGA clocks for I/O shift registers (x2, x4) and DDR (x2, x4) I/O logic interfaces. Divide by 8 is provided for slow speed/low power operation.

To guarantee a synchronous transfer in the I/O logic, the CLKDIV input clock must be driven by an edge clock and the output must drive a primary clock. In this case, they are phase matched.

It is especially useful to synchronously reset the I/O logic when Mux/DeMux gearing is used in order to synchronize the entire data bus as shown in Figure 10-26. Using the low-skew characteristics of the edge clock routing a reset can be provided to all bits of the data bus to synchronize the Mux/DeMux gearing.

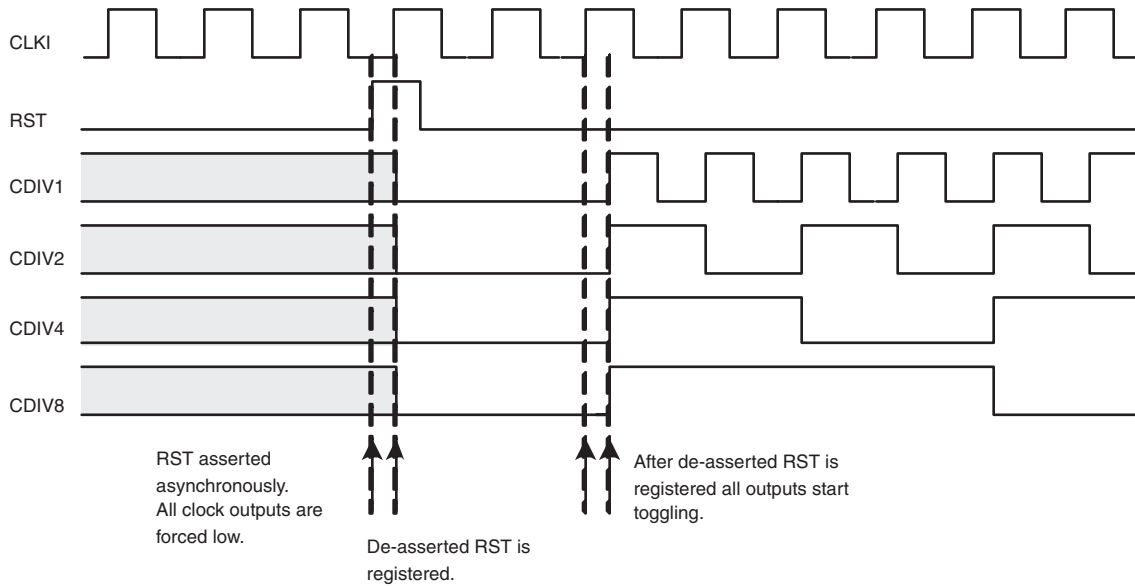
**Figure 10-26. CLKDIV Application Example**



## Reset Behavior

Figure 10-27 illustrates the asynchronous RST behavior.

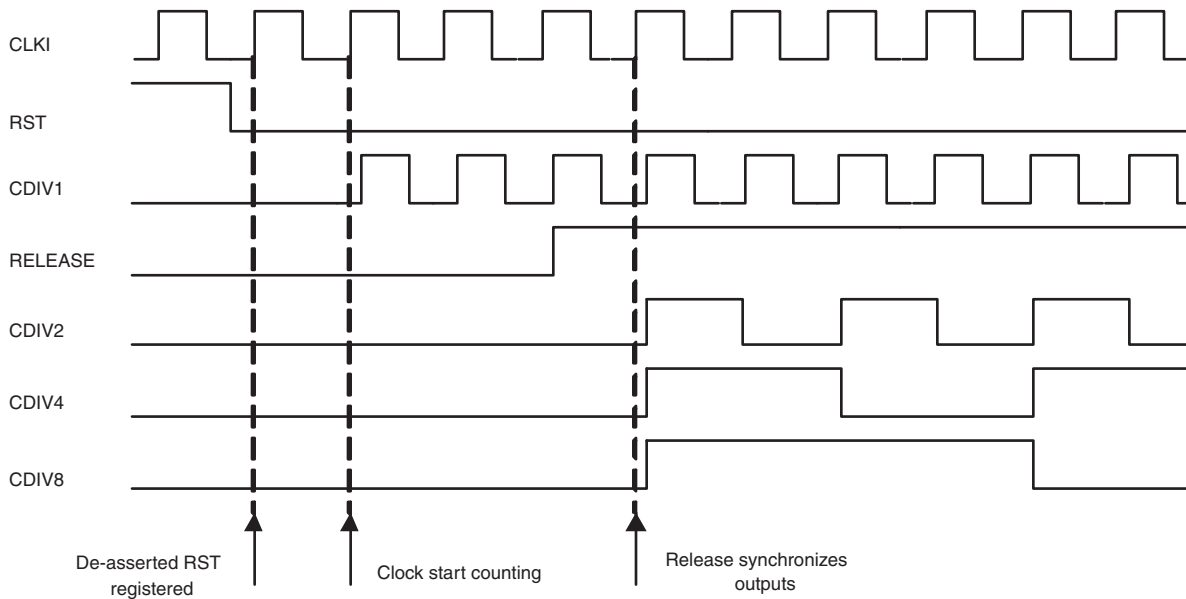
**Figure 10-27. CLKDIV Reset Behavior**



## Release Behavior

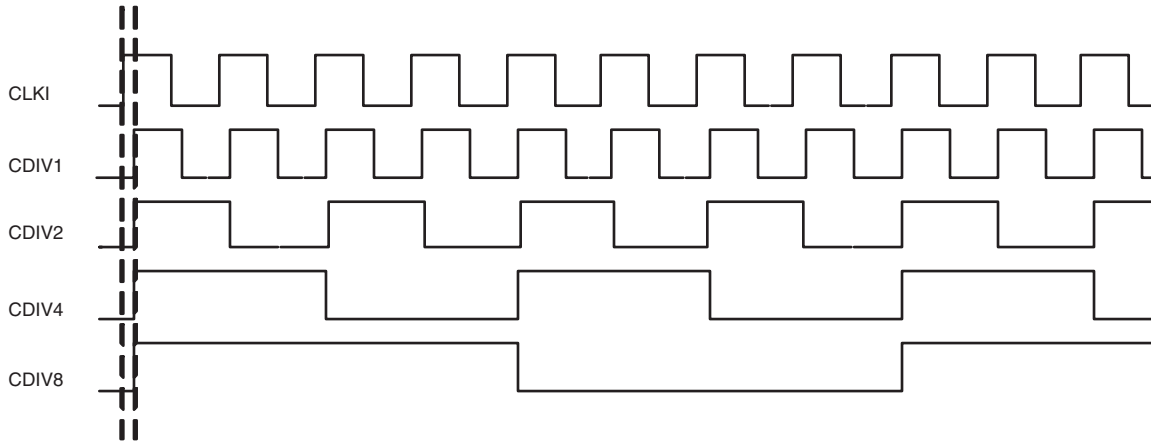
The port, “Release” is used to synchronize the all outputs after RST is de-asserted. Figure 10-28 illustrates the release behavior.

**Figure 10-28. CLKDIV Release Behavior**



## CLKDIV Inputs-to-Outputs Delay Matching

Figure 10-29. CLKDIV Inputs-to-Outputs Delay Matching



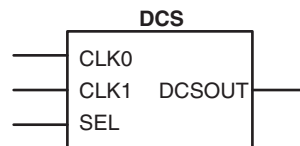
## DCS (Dynamic Clock Select)

DCS is a global clock buffer incorporating a smart multiplexer function that takes two independent input clock sources and avoids glitches or runt pulses on the output clock, regardless of where the enable signal is toggled. There are two DCSs for each quadrant.

As can be seen in Figure 10-30, the DCS inputs are driven by all the same signals that drive non-DCS clocks, except that the PLL and DLL inputs are somewhat restricted. The select input is driven by a signal from the general routing fabric. The outputs of the DCS then reach primary clock distribution via the feedlines. Figure 10-30 shows the block diagram of the DCS.

When CLK6 or CLK7 is used as a primary clock and there is only one clock input to the DCS, the DCS is assigned as a buffer mode by the software, but will still inject some delay into the net. In order to be glitchless, the DCS must have both clock inputs switching. If one of the inputs is not switching, the DCS will not be able to switch.

Figure 10-30. DCS Library Symbol



## DCS Library Definition

Table 10-13 defines the I/O ports of the DCS block. There are eight modes to select from. Table 10-14 describes how each mode is configured.

Table 10-13. DCS I/O Definition

I/O	Name	Description
Input	SEL	Input Clock Select
	CLK0	Clock input 0
	CLK1	Clock Input 1
Output	DCSOUT	Clock Output

Table 10-14. DCS Modes of Operation

Attribute Name	Description	Output		Value
		SEL=0	SEL=1	
DCS MODE	Rising edge triggered, latched state is high	CLK0	CLK1	POS
	Falling edge triggered, latched state is low	CLK0	CLK1	NEG
	Sel is active high, Disabled output is low	0	CLK1	HIGH_LOW
	Sel is active high, Disabled output is high	1	CLK1	HIGH_HIGH
	Sel is active low, Disabled output is low	CLK0	0	LOW_LOW
	Sel is active low, Disabled output is high	CLK0	1	LOW_HIGH
	Buffer for CLK0	CLK0	CLK0	CLK0
	Buffer for CLK1	CLK1	CLK1	CLK1

### DCS Timing Diagrams

Each mode performs a unique operation. The clock output timing is determined by input clocks and the edge of the SEL signal. Figure 10-31 describes the timing of each mode.

Figure 10-31. Timing Diagrams by DCS MODE

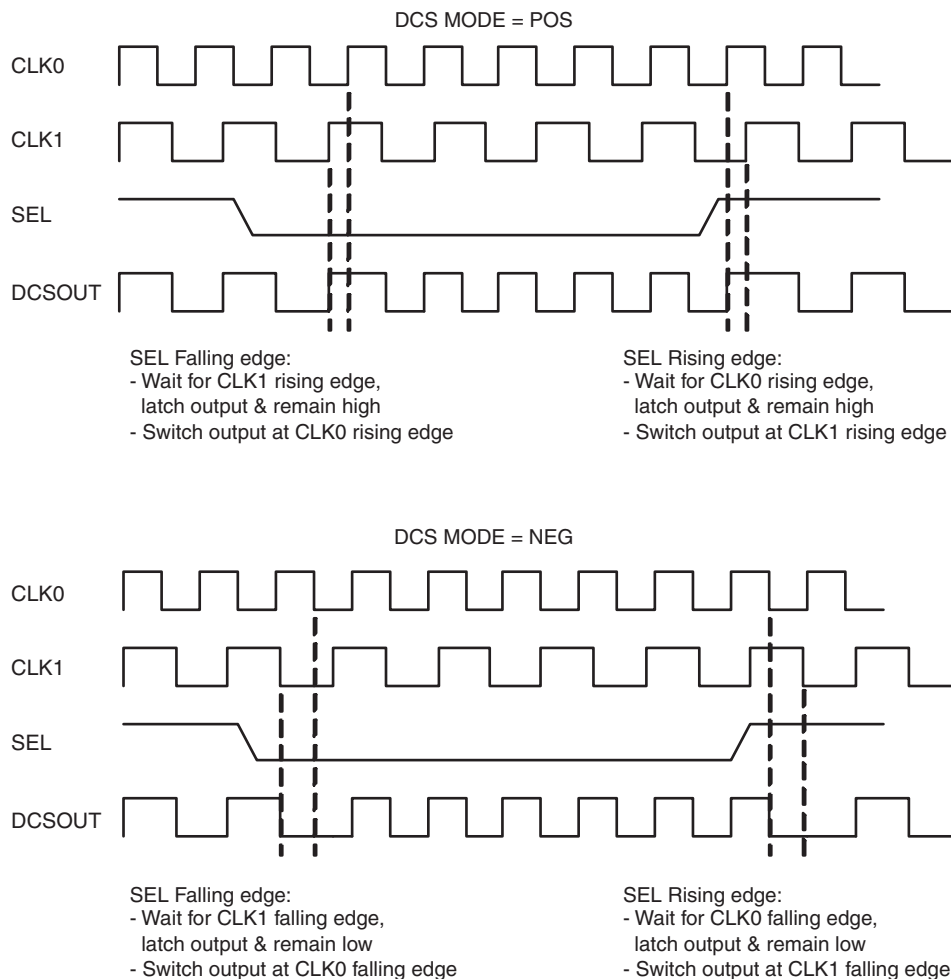
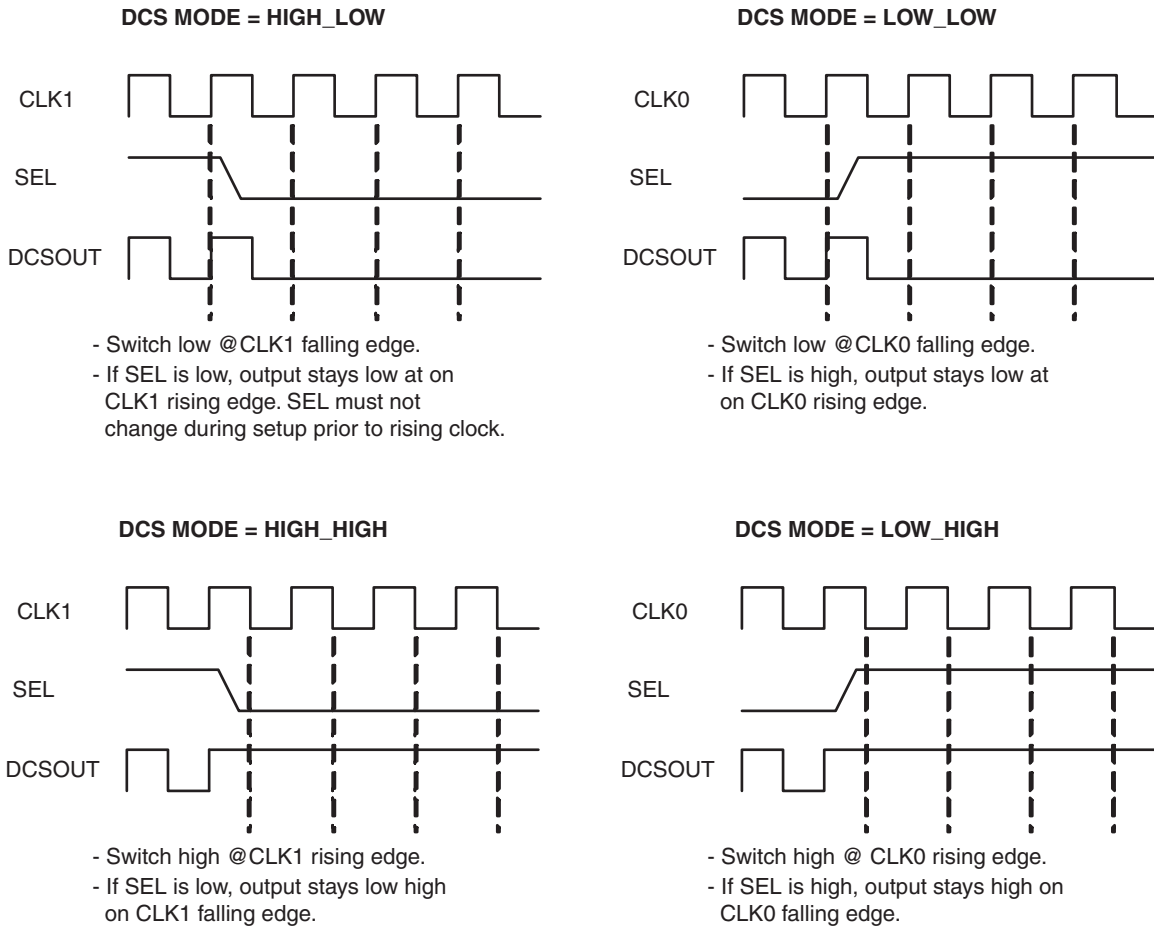


Figure 10-32. Timing Diagrams by DCS MODE (Cont.)



### DCS Usage with VHDL - Example

```

COMPONENT DCS
-- synthesis translate_off
    GENERIC (
        DCSMODE : string := "POS"
    );
-- synthesis translate_on

    PORT (
        CLK0      :IN  std_logic ;
        CLK1      :IN  std_logic ;
        SEL       :IN  std_logic ;
        DCSOUT    :OUT std_logic) ;

END COMPONENT;

attribute DCSMODE : string;
attribute DCSMODE of DCSinst0 : label is "POS";

begin

```

```
DCSInst0: DCS
-- synthesis translate_off

    GENERIC MAP (
        DCSMODE => "POS"
    );
-- synthesis translate_on

    PORT MAP (
        SEL           => clksel,
        CLK0          => dcsclk0,
        CLK1          => sysclk1,
        DCSOUT        => dcsclk
    );
```

### DCS Usage with Verilog - Example

```
module dcs(clk0,clk1,sel,dcsout);

input clk0, clk1, sel;
output dcsout;

DCS DCSInst0 (.SEL(sel),.CLK0(clk0),.CLK1(clk1),.DCSOUT(dcsout));
defparam DCSInst0.DCSMODE = "POS";

endmodule
```

### Oscillator (OSCF)

There is a dedicated oscillator in the LatticeECP3 device whose output is made available for users.

The oscillator frequency output is routed through a divider which is used as an input clock to the clock tree. The available outputs of the divider are shown in Table 10-15. The oscillator frequency output can be further divided by internal logic (user logic) for lower frequencies, if desired. The oscillator is powered down when not in use.

The output of this oscillator is not a precision clock. It is intended as an extra clock that does not require accurate clocking.

Library Element: OSCF

**Table 10-15. OSCE Port Definition**

I/O	Name	Description
Output	OSC	Oscillator Clock Output

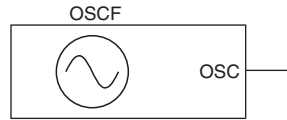
**Table 10-16. OSCE Attribute Definition**

User Attribute	Attribute Name	Value (MHz)	Default Value
Nominal Frequency	NOM_FREQ	2.5, 4.3, 5.4, 6.9, 8.1, 9.2, 10, 13, 15, 20, 26, 30, 34, 41, 45, 51, 55, 60, 130	2.5



## OSC Library Symbol (OSCF)

Figure 10-33. OSC Symbol



## OSC Usage with VHDL - Example

```
COMPONENT OSCF

    PORT (OSC:OUT    std_logic);

END COMPONENT;
begin
OSCInst0: OSCF
    PORT MAP ( OSC=>  osc_int);
```

## OSC Usage with Verilog - Example

```
module OSC_TOP(OSC_CLK);

output OSC_CLK;

OSCF OSCinst0 (.OSC(OSC_CLK));

defparam OSCinst0.NOM_FREQ = "5.4" ;

endmodule
```

## Setting Clock Preferences

Designers can use clock preferences to implement clocks to the desired performance. Preferences can be set in the ispLEVER Design Planner Spreadsheet View (or Diamond Spreadsheet View) or in preference files. Frequently used preferences are described in Appendix C.

## Power Supplies

Each PLL has its own power supply pin, VCCPLL.

## PLL/DLL Names and Preferred Pads

Table 10-17 lists the names and preferred pads for all PLLs and DLLs for each device size in the LatticeECP3 family. To obtain the pin/ball associated with each pad, refer to the pinout table for the target device in the LatticeECP3 Family Data Sheet. Here is an example of a PLL locate preference:

```
LOCATE COMP "PLL_0/PLLInst_0" SITE "PLL_R35C70";
```

Table 10-17. PLL/DLL Names and Preferred Pads

		Pad IN_A	Pad IN_B	Pad FB_A	Pad FB_B
<b>LatticeECP3-17</b>					
LUM0_GDLLT	DLL_R26C15	PL20A	PL20B	PL21A	PL21B
RUM0_GDLLT	DLL_R26C42	PR20A	PR20B	PR21A	PR21B
LUM0_GPLLT	PLL_R26C5	PL26E_C	PL26E_D	PL26E_A	PL26E_B
RUM0_GPLLT	PLL_R26C52	PR26E_C	PR26E_D	—	—

**Table 10-17. PLL/DLL Names and Preferred Pads (Continued)**

		Pad IN_A	Pad IN_B	Pad FB_A	Pad FB_B
<b>LatticeECP3-35</b>					
LUM0_GDLLT	DLL_R35C15	PL29A	PL29B	PL30A	PL30B
RUM0_GDLLT	DLL_R35C60	PR29A	PR29B	PR30A	PR30B
LUM0_GPLLT	PLL_R53C5	PL35E_C	PL35E_D	PL35E_A	PL35E_B
LLM1_GPLLT	PLL_R35C5	PL53E_C	PL53E_D	PL53E_A	PL53E_B
RLM1_GPLLT	PLL_R53C70	PR53E_C	PR53E_D	PR53E_A	PR53E_B
RUM0_GPLLT	PLL_R35C70	PR35E_C	PR35E_D	PR35E_A	PR35E_B
<b>LatticeECP3-70/95</b>					
LUM0_GDLLT	DLL_R43C15	PL37A	PL37B	PL38A	PL38B
RUM0_GDLLT	DLL_R43C132	PR37A	PR37B	PR38A	PR38B
LUM2_GPLLT	PLL_R25C5	PL25E_C	PL25E_D	PL25E_A	PL25E_B
LUM0_GPLLT	PLL_R43C5	PL43E_C	PL43E_D	PL43E_A	PL43E_B
LLM1_GPLLT	PLL_R61C5	PL61E_C	PL61E_D	PL61E_A	PL61E_B
LLM2_GPLLT	PLL_R70C5	PL70E_C	PL70E_D	PL70E_A	PL70E_B
LLM3_GPLLT	PLL_R79C5	PL79E_C	PL79E_D	PL79E_A	PL79E_B
RLM3_GPLLT	PLL_R79C142	PR79E_C	PR79E_D	PR79E_A	PR79E_B
RLM2_GPLLT	PLL_R70C142	PR70E_C	PR70E_D	PR70E_A	PR70E_B
RLM1_GPLLT	PLL_R61C142	PR61E_C	PR61E_D	PR61E_A	PR61E_B
RUM0_GPLLT	PLL_R43C142	PR43E_C	PR43E_D	PR43E_A	PR43E_B
RUM2_GPLLT	PLL_R25C142	PR25E_C	PR25E_D	PR25E_A	PR25E_B
<b>LatticeECP3-150</b>					
LUM0_GDLLT	DLL_R61C15	PL55A	PL55B	PL56A	PL56B
RUM0_GDLLT	DLL_R61C168	PR55A	PR55B	PR56A	PR56B
LUM2_GPLLT	PLL_R43C5	PL43E_C	PL43E_D	PL43E_A	PL43E_B
LUM0_GPLLT	PLL_R61C5	PL61E_C	PL61E_D	PL61E_A	PL61E_B
LLM1_GPLLT	PLL_R79C5	PL79E_C	PL79E_D	PL79E_A	PL79E_B
LLM3_GPLLT	PLL_R97C5	PL97E_C	PL97E_D	PL97E_A	PL97E_B
LLM4_GPLLT	PLL_R106C5	PL106E_C	PL106E_D	PL106E_A	PL106E_B
RLM4_GPLLT	PLL_R106C178	PR106E_C	PR106E_D	PR106E_A	PR106E_B
RLM3_GPLLT	PLL_R97C178	PR97E_C	PR97E_D	PR97E_A	PR97E_B
RLM1_GPLLT	PLL_R79C178	PR79E_C	PR79E_D	PR79E_A	PR79E_B
RUM0_GPLLT	PLL_R61C178	PR61E_C	PR61E_D	PR61E_A	PR61E_B
RUM2_GPLLT	PLL_R43C178	PR43E_C	PR43E_D	PR43E_A	PR43E_B

## Technical Support Assistance

 e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

 Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
July 2009	01.1	Updated Secondary Clocks text section.
September 2009	01.2	Updated LatticeECP3 Primary Clock Muxes figure.
November 2009	01.3	Updated Edge Clocks text section.
February 2010	01.4	Reconciled LOCK description among MachXO, LatticeXP2, LatticeECP2/M and LatticeECP3.
April 2010	01.5	Updated the Figure for Time Reference Delay Circuitry Example
		Updated example for DLLDELB Usage with TRDLLB - Verilog
		Added new section - PLL/DLL Names and Preferred Pads
June 2010	01.6	Updated for Lattice Diamond design software support.
January 2011	01.7	Added General Routing for Clocks text section.
February 2011	01.8	Added Fine Delay Ports text section.
June 2011	01.9	Output Clock (CLKOP) Divider text section – VCO frequency changed from 435MHz - 870 MHz to 500MHz - 1000MHz.
January 2012	02.0	Updated CLKI Input text section.
		Updated CLKFB Input text section.
		Updated DCS (Dynamic Clock Select) text section.
February 2012	02.1	Updated document with new corporate logo.
April 2012	02.2	Updated CLKDIV Usage with Verilog - Example.
		Updated CLKDIV Example Circuits text section.
		Updated DCS Usage with Verilog - Example.
May 2012	02.3	Updated CLKOK2 signal to clarify timing relationship to CLKOP.
September 2012	02.4	Update to fix Table 10-2, add primary clock preference examples, and clarify secondary clock regions and how to create preferences for them.
July 2013	02.2	Updated Technical Support Assistance information.

## Appendix A. Primary Clock Sources and Distribution

Figure 10-34. LatticeECP3 Primary Clock Sources and Distribution

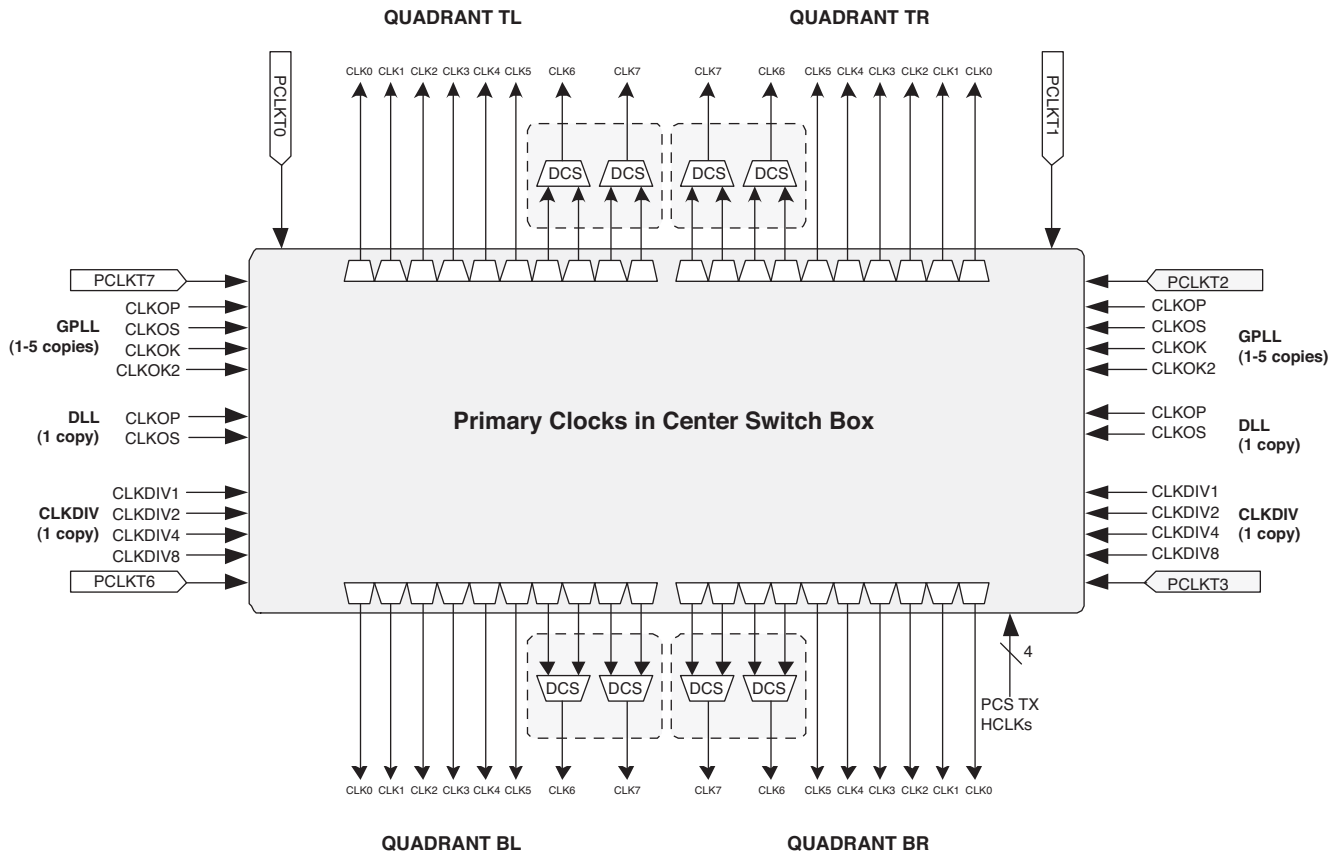
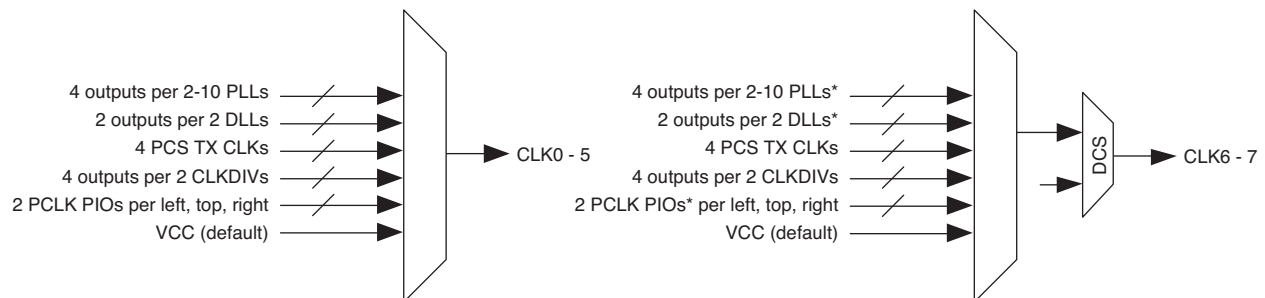


Figure 10-35. LatticeECP3 Primary Clock Muxes (Simplified)

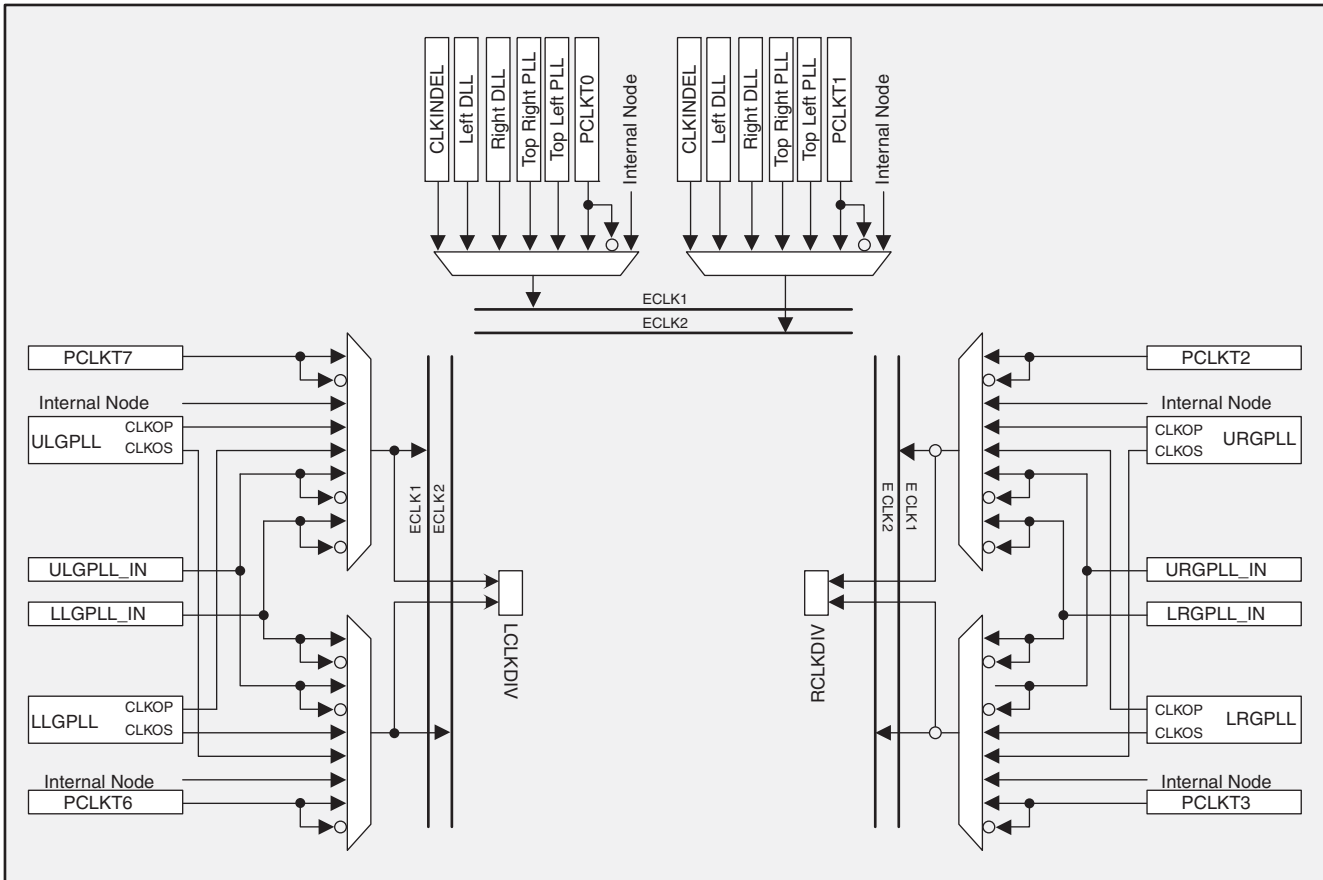


\*For CLK6 and CLK7 muxes, the PLL, DLL and PCLK PIOs, inputs are not fully populated.  
Note: Use EPIC to see exact routing

## Appendix B. PLL, CLKIDV and ECLK Locations and Connectivity

Figure 10-36 shows the locations, site names and connectivity of the PLLs, CLKDIVs and ECLKs

**Figure 10-36. PLL, CLKIDV and ECLK Locations and Connectivity**



---

## Appendix C. Lattice Diamond Usage Overview

This appendix discusses the use of Lattice Diamond design software for projects that include the LatticeECP2M SERDES/PCS module.

For general information about the use of Lattice Diamond, refer to the Lattice Diamond Tutorial.

If you have been using ispLEVER software for your FPGA design projects, Lattice Diamond may look like a big change. But if you look closer, you will find many similarities because Lattice Diamond is based on the same toolset and work flow as ispLEVER. The changes are intended to provide a simpler, more integrated, and more enhanced user interface.

### Converting an ispLEVER Project to Lattice Diamond

Design projects created in ispLEVER can easily be imported into Lattice Diamond. The process is automatic except for the ispLEVER process properties, which are similar to the Diamond strategy settings, and PCS modules. After importing a project, you need to set up a strategy for it and regenerate any PCS modules.

### Importing an ispLEVER Design Project

Make a backup copy of the ispLEVER project or make a new copy that will become the Diamond project.


1. In Diamond, choose **File > Open > Import ispLEVER Project**.
2. In the ispLEVER Project dialog box, browse to the project's .syn file and open it.
3. If desired, change the base file name or location for the Diamond project. If you change the location, the new Diamond files will go into the new location, but the original source files will not move or be copied. The Diamond project will reference the source files in the original location.

The project files are converted to Diamond format with the default strategy settings.

### Adjusting PCS Modules

PCS modules created with IPexpress have an unusual file structure and need additional adjustment when importing a project from ispLEVER. There are two ways to do this adjustment. The preferred method is to regenerate the module in Diamond. However this may upgrade the module to a more recent version. An upgrade is usually desirable but if, for some reason, you do not want to upgrade the PCS module, you can manually adjust the module by copying its .txt file into the implementation folder. If you use this method, you must remember to copy the .txt file into any future implementation folders.

### Regenerate PCS Modules

1. Find the PCS module in the Input Files folder of File List view. The module may be represented by an .lpc, .v, or .vhd file.
2. If the File List view shows the Verilog or VHDL file for the module, and you want to regenerate the module, import the module's .lpc file:
  - a. In the File List view, right-click the implementation folder (  ) and choose **Add > Existing File**.
  - b. Browse for the module's .lpc file, **<module\_name>.lpc**, and select it.
  - c. Click **Add**. The .lpc file is added to the File List view.
  - d. Right-click the module's Verilog or VHDL file and choose **Remove**.
3. In File List, double-click the module's .lpc file. The module's IPexpress dialog box opens.
4. In the bottom of the dialog box, click **Generate**. The Generate Log tab is displayed. Check for errors and close.

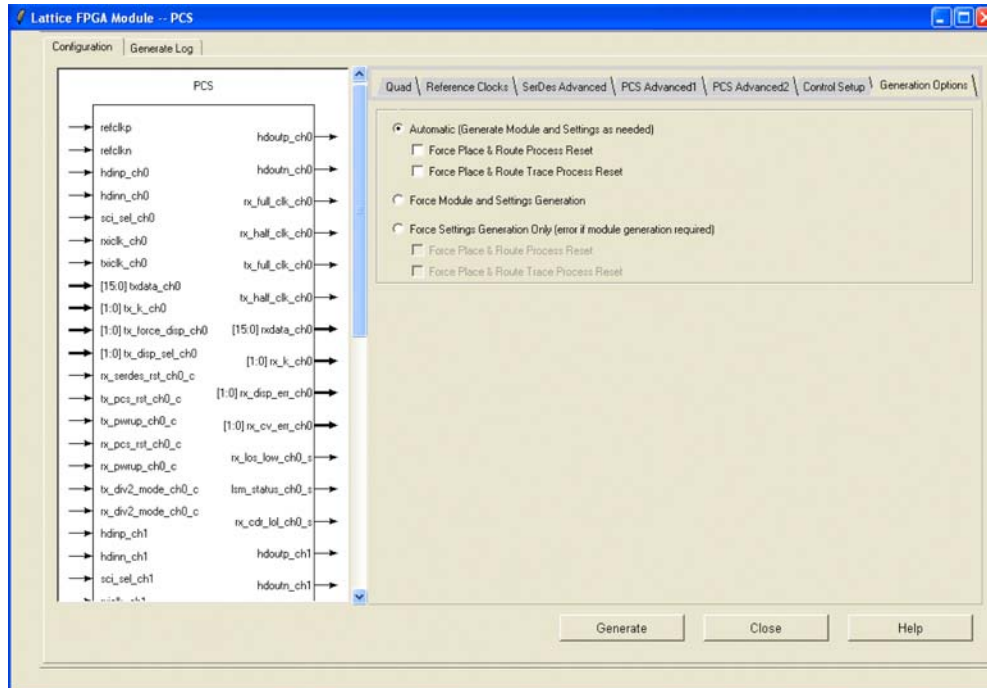
In File List, the .lpc file is replaced with an .lpx file. The IPexpress manifest (.lpx) file is new with Diamond. The .lpx file keeps track of the files needed for complex modules.

### Using IPexpress with Lattice Diamond

Using IPexpress with Lattice Diamond is essentially same as with ispLEVER.

The configuration GUI tabs are all the same except for the Generation Options tab. Figure 10-37 shows the Generation Options tab window.

**Figure 10-37. Generation Options Tab**



**Table 10-18. SERDES\_PCS GUI Attributes – Generation Options Tab**

GUI Text	Description
Automatic	Automatically generates the HDL and configuration(.txt) files as needed. Some changes do not require regenerating both files.
Force Module and Settings Generation	Generates both the HDL and configuration files.
Force Settings Generation Only	Generates only the attributes file. You get an error message if the HDL file also needs to be generated.
Force Place & Route Process Reset	Resets the Place & Route Design process, forcing it to be run again with the newly generated PCS module.
Force Place & Route Trace Process Reset	Resets the Place & Route Trace process, forcing it to be run again with the newly generated PCS module.

Note:

Automatic is set as the default option. If either Automatic or Force Settings Generation Only and no sub-options (Process Reset Options) are checked and the HDL module is not generated, the reset pointer is set to Bitstream generation automatically.

After the Generation is finished, the reset marks in the process window will be reset accordingly.

---

## Creating a New Simulation Project Using Simulation Wizard

This section describes how to use the Simulation Wizard to create a simulation project (.spf) file so you can import it into a standalone simulator.

1. In Project Navigator, click **Tools > Simulation Wizard**. The Simulation Wizard opens.
2. In the Preparing the Simulator Interface page click **Next**.
3. In the Simulator Project Name page, enter the name of your project in the Project Name text box and browse to the file path location where you want to put your simulation project using the Project Location text box and Browse button.

When you designate a project name in this wizard page, a corresponding folder will be created in the file path you choose. Click **Yes** in the popup dialog that asks you if you wish to create a new folder.

4. Click either the Active-HDL® or ModelSim® simulator check box and click **Next**.
5. In the Process Stage page choose which type of Process Stage of simulation project you wish to create. Valid types are RTL, Post-Synthesis Gate-Level, Post-Map Gate-Level, and Post-Route Gate-level+Timing. Only those process stages that are available are activated.

Note that you can make a new selection for the current strategy if you have more than one defined in your project.

The software supports multiple strategies per project implementation which allow you to experiment with alternative optimization options across a common set of source files. Since each strategy may have been processed to different stages, this dialog allows you to specify which stage you wish to load.

6. In the Add Source page, select from the source files listed in the Source Files list box or use the browse button on the right to choose another desired source file. Note that if you wish to keep the source files in the local simulation project directory you just created, check the **Copy Source to Simulation Directory** option.
7. Click **Next** and a Summary page appears and provides information on the project selections including the simulation libraries. By default, the Run Simulator check box is enabled and will launch the simulation tool you chose earlier in the wizard in the Simulator Project Name page.
8. Click **Finish**.

The Simulation Wizard Project (.spf) file and a simulation script DO file are generated after running the wizard. You can import the DO file into your current project if desired. If you are using Active-HDL, the wizard will generate an .ado file and if you are using ModelSim, it creates and .mdo file.

*Note: PCS configuration file, (.txt) must be added in step 6.*



Figure 10-38. Diamond Spreadsheet View (see Figure 10-6 for ispLEVER Equivalent)

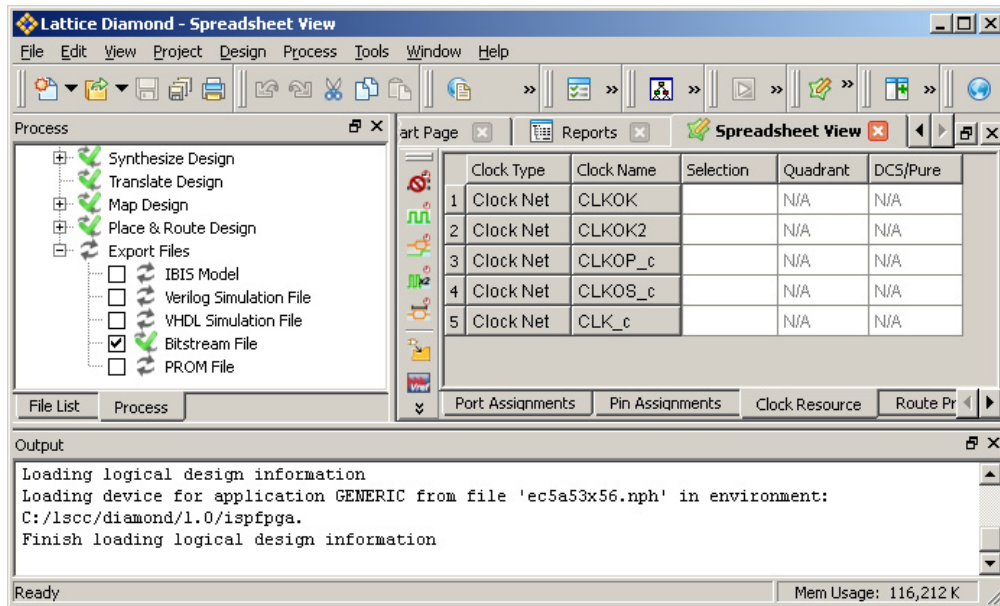


Figure 10-39. Diamond IPexpress Main Window (see Figure 10-14 for ispLEVER Equivalent)

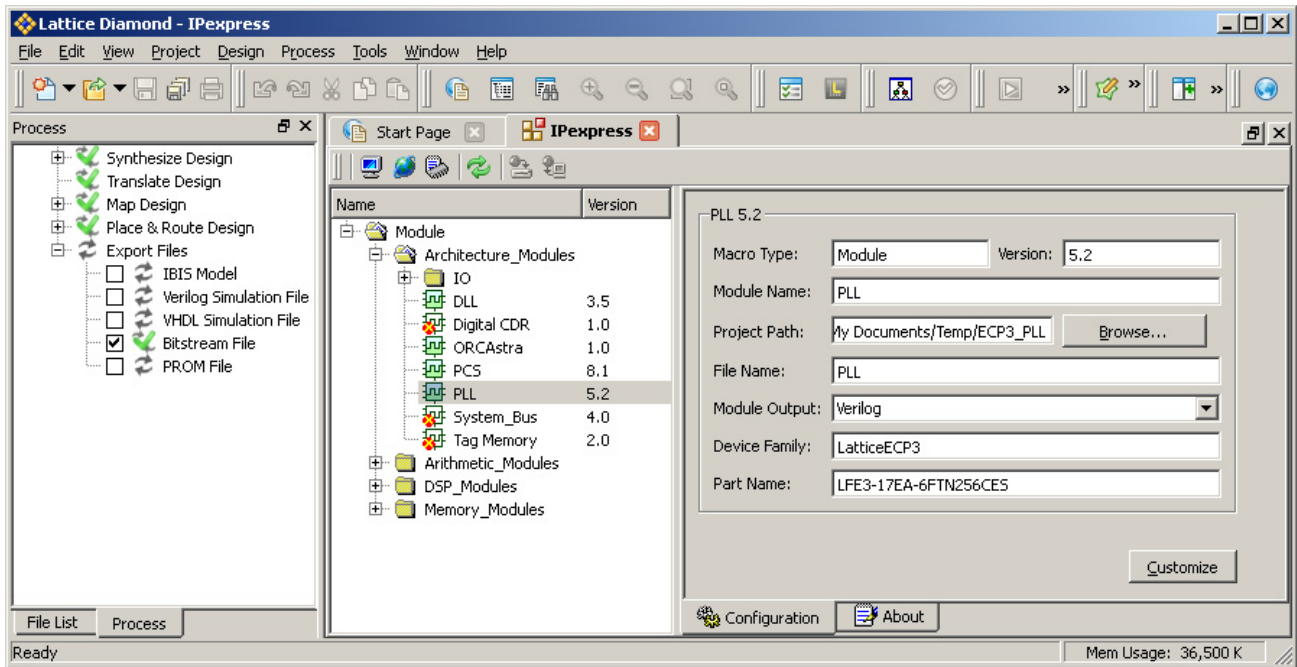


Figure 10-40. Diamond LatticeECP3 PLL Configuration Tab (see Figure 10-15 for ispLEVER Equivalent)

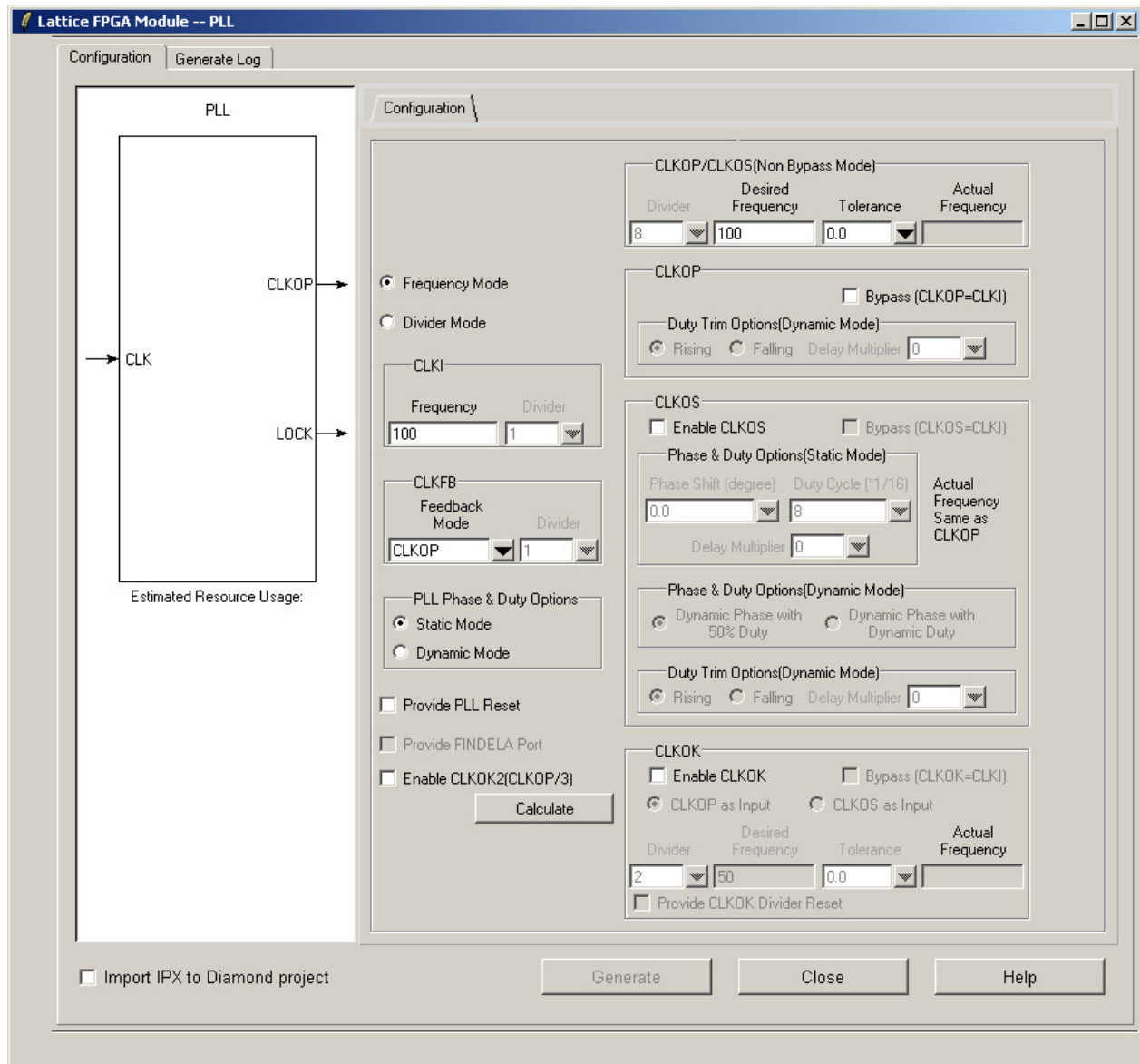
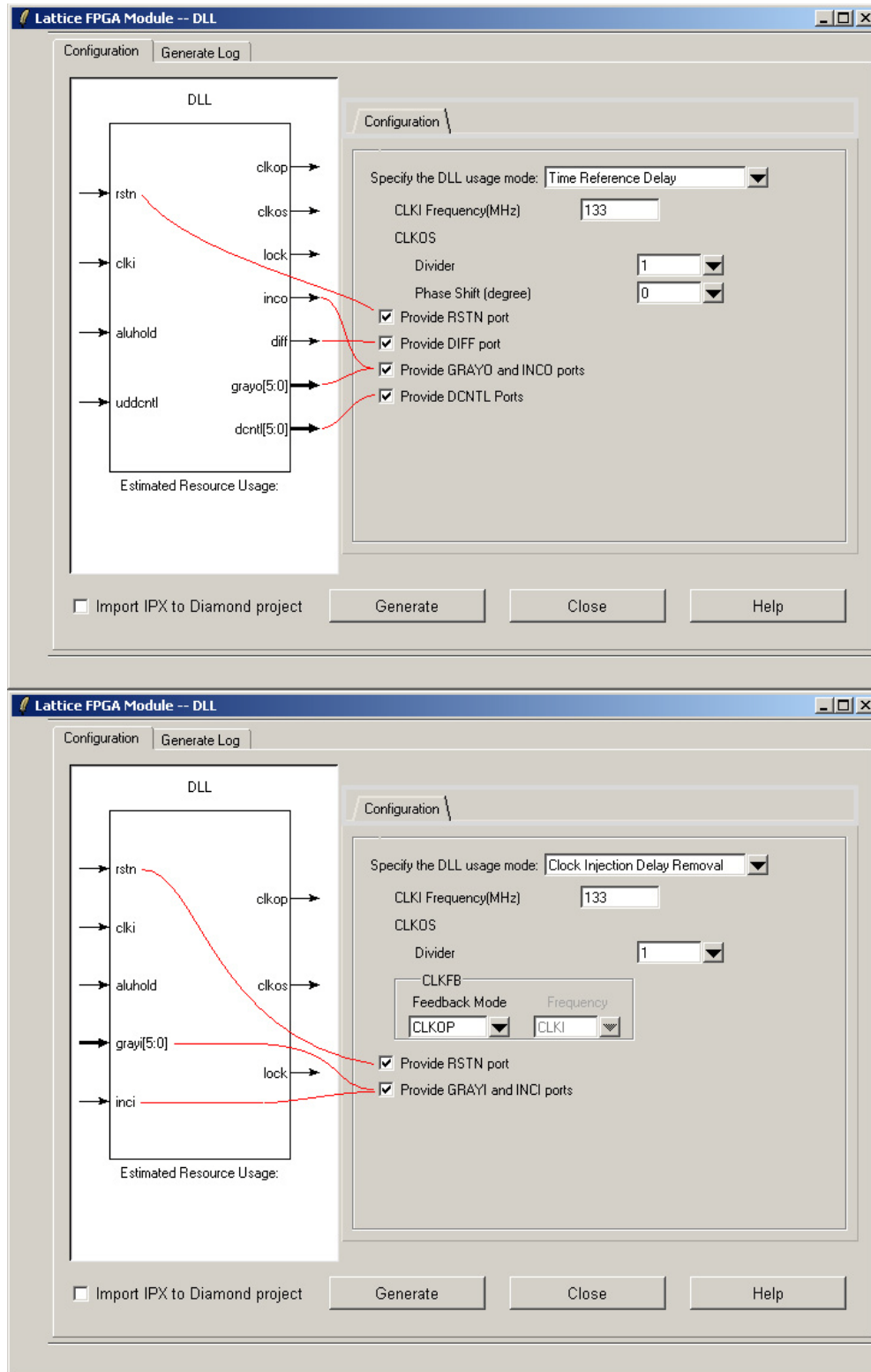


Figure 10-41. Diamond LatticeECP3 IPexpress DLL Configuration Tab (see Figure 10-22 for ispLEVER Equivalent)



## Introduction

This technical note discusses memory usage for the LatticeECP3™ family of FPGA devices. It is intended to be used by design engineers as a guide to integrating the EBR- and PFU-based memories for this device family in ispLEVER®.

The LatticeECP3 architecture provides many resources for memory-intensive applications. The sysMEM™ Embedded Block RAM (EBR) complements its distributed PFU-based memory. Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM and ROM memories can be constructed using the EBR. The LUTs and PFUs can implement Distributed Single-Port RAM, Dual-Port RAM and ROM. The internal logic of the device can be used to configure the memory elements as FIFO and other storage types.

The capabilities of the EBR Block RAM and PFU RAM are referred to in this document. Designers can utilize the memory primitives in several ways:

- Via **IPexpress™** – The IPexpress GUI allows users to specify the memory type and size required. IPexpress uses this information to construct a netlist to implement the desired memory by using one or more of the memory primitives.
- Via **PMI (Parameterizable Module Inferencing)** – PMI allows experienced users to skip the graphical interface and utilize the configurable memory primitives on-the-fly from the ispLEVER Project Navigator. The parameters and the control signals needed either in Verilog or VHDL can be set. The top-level design will have the parameters defined and signals declared so the interface can automatically generate the black box during synthesis.
- Via the **Instantiation of Memory Primitives** – Memory primitives are called directly by the top-level module and instantiated in the user's design. This is an advanced method and requires a thorough understanding of memory hook-ups and design interfaces.

The remainder of this document discusses these approaches.

## Utilizing IPexpress

Designers can utilize IPexpress to easily specify a variety of memories in their designs. These modules will be constructed using one or more memory primitives along with general purpose routing and Look-Up Tables (LUTs) as required. The primitives include:

- Single Port RAM (RAM\_DQ) – EBR-based
- Dual PORT RAM (RAM\_DP\_TRUE) – EBR-based
- Pseudo Dual Port RAM (RAM\_DP) – EBR-based
- Read Only Memory (ROM) – EBR-based
- First In First Out Memory (FIFO and FIFO\_DC) – EBR-based
- Distributed Single Port RAM (Distributed\_SPRAM) – PFU-based
- Distributed Dual Port RAM (Distributed\_DPRAM) – PFU-based
- Distributed ROM (Distributed\_ROM) – PFU/PFF-based

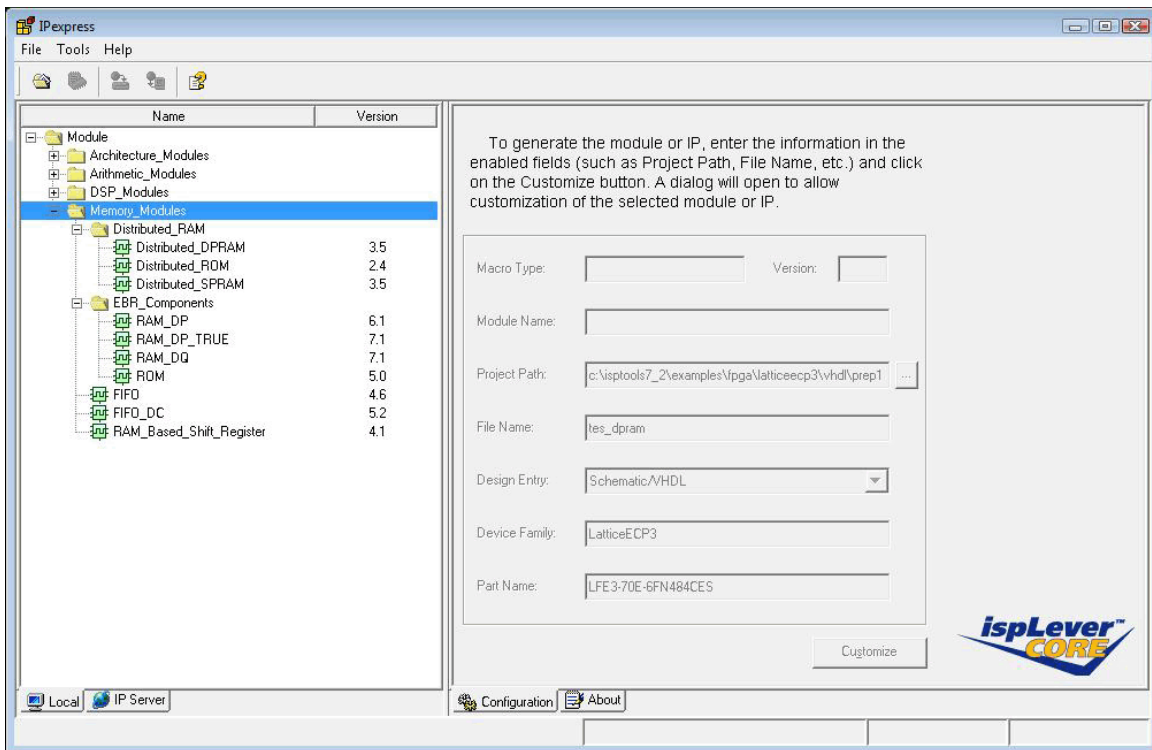
## IPexpress Flow

For generating any of these memories, create (or open) a project for the LatticeECP3 devices.

From the Project Navigator, select **Tools > Module / IP Manager**. Alternatively, users can also click the red and yellow button in the left-hand corner of the toolbar when LatticeECP3 devices are targeted in the project.

This opens the IPexpress window as shown in Figure 11-1.

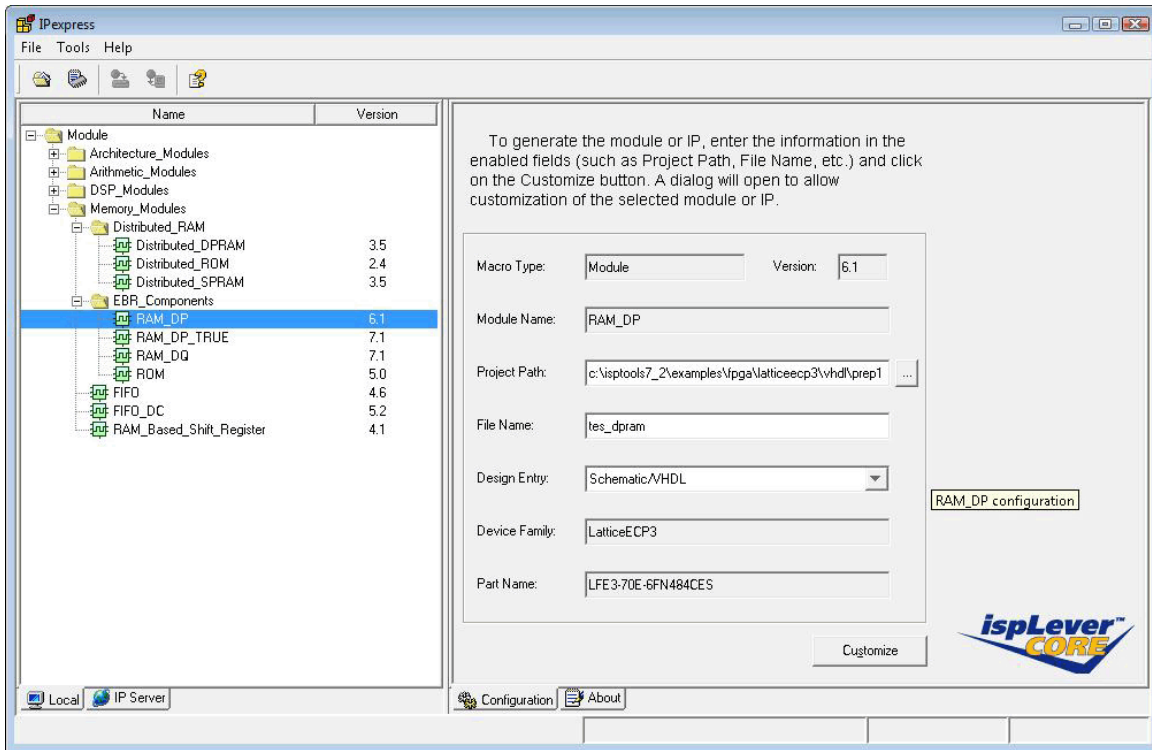
**Figure 11-1. IPexpress, Main Window**



The left pane of this window includes the Module Tree. The EBR-based Memory Modules are under the **EBR\_Components** directory and the PFU-based Distributed Memory Modules are under the **Storage\_Components** directory, as shown in Figure 11-1.

As an example, let us consider generating an EBR-based Pseudo Dual Port RAM of size 512x16. Select **RAM\_DP** under **EBR\_Components**. The right pane changes as shown in Figure 11-2.

Figure 11-2. Example: Generating Pseudo Dual Port RAM (RAM\_DP) Using IPexpress



In the right pane, options like **Device Family**, **Macro Type**, **Category** and **Module\_Name** are device- and selected-module-dependent. These cannot be changed in IPexpress.

Users can change the directory where the generated module files will be placed by clicking the **Browse** button in the **Project Path**.

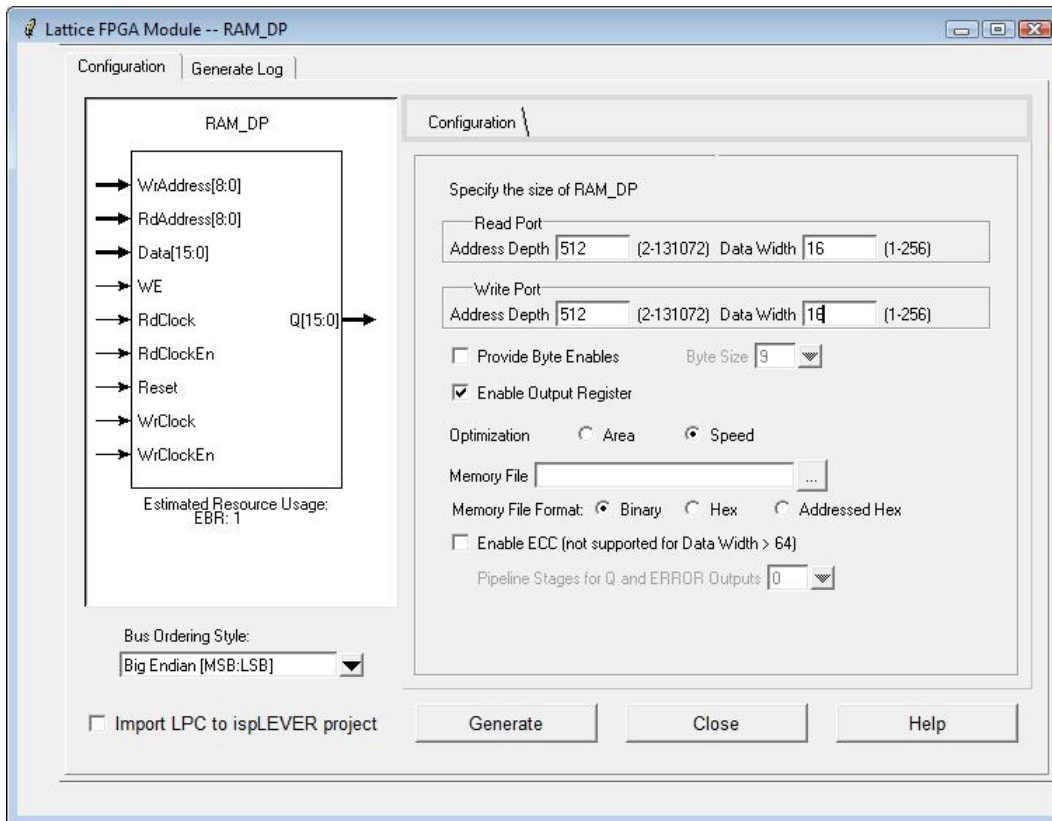
The entity name for the module to be generated can be specified in the **Module Name** text box. Users must provide this entity name.

**Design Entry**, Verilog or VHDL, by default is the same as the project type. If the project is a VHDL project, the selected Design Entry option will be “Schematic/ VHDL”, and “Schematic/ Verilog-HDL” if the project type is Verilog-HDL.

The **Device** pull-down menu allows users to select different devices within the same family, LatticeECP3 in this example.

By clicking the **Customize** button, another window opens where users can customize the RAM (Figure 11-3).

**Figure 11-3. Example: Generating Pseudo Dual Port RAM (RAM\_DP) Module Customization - General Options**



The left side of this window shows the block diagram of the module. The right side includes the **Configuration** tab where users can choose options to customize the RAM\_DP (e.g. specify the address port sizes and data widths).

Users can specify the address depth and data width for the **Read Port** and the **Write Port** in the text boxes provided. In this example, we are generating a Pseudo Dual Port RAM of size 512 x 16. Users can also create RAMs of different port widths for Pseudo Dual Port and True Dual Port RAMs.

The Input Data and the Address Control are always registered, as the hardware only supports the clocked write operation for the EBR-based RAMs. The checkbox **Enable Output Registers** inserts the output registers in the Read Data Port. Output registers are optional for EBR-based RAMs.

Users can also pre-initialize their memory with the contents specified in the **Memory File**. It is optional to provide this file in the RAM; however for ROM, the Memory File is required. These files can be of Binary, Hex or Addresses Hex format. The details of these formats are discussed in the Initialization File section of this document.

At this point, users can click the **Generate** button to generate the module they have customized. A VHDL or Verilog netlist is then generated and placed in the specified location. Users can incorporate this netlist in their designs.

Another important button is the **Load Parameters** button. IPexpress stores the parameters specified in a <module\_name>.lpc file. This file is generated along with the module. Users can click on the Load Parameters button to load the parameters of a previously-generated module to re-visit or make changes to them.

## Utilizing the PMI

Parameterizable Module Inferencing (PMI) allows experienced users to skip the graphical interface and utilize the configurable memory modules on-the-fly from the ispLEVER Project Navigator.



The parameters and control signals needed can be set in either Verilog or VHDL. The top-level design includes the defined memory parameters and declared signals. The interface can then automatically generate the black box during synthesis and ispLEVER can generate the netlist on-the-fly. Lattice memories are the same as industry standard memories, so you can get the parameters for each module from any memory-related guide, which is available through the on-line help system.

To do this, the user creates a Verilog or VHDL behavior code for the memory and the synthesis tool automatically identifies it as memory and synthesizes it as a distributed or EBR memory. Memory sizes smaller than 2K bits are automatically mapped to Distributed mode and those larger than 2K bits will be implemented using EBRs. This default option can be over-ridden using the RAM\_STYLE attribute in Synplify. Refer to Appendix A for example code.

### ECC in Memory Modules

IPexpress allows users to implement Error Check Codes in the EBR-based memory modules. There is a checkbox to enable ECC in the configuration tab for the module.

If you choose to use ECC, you will have a 2-bit error signal and the error codes are as below.

- Error[1:0]=00 – Indicates there is no error.
- Error[0]=1 – Indicates there was a 1-bit error which was fixed.
- Error[1]=1 – Indicates there was a 2-bit error which cannot be corrected.

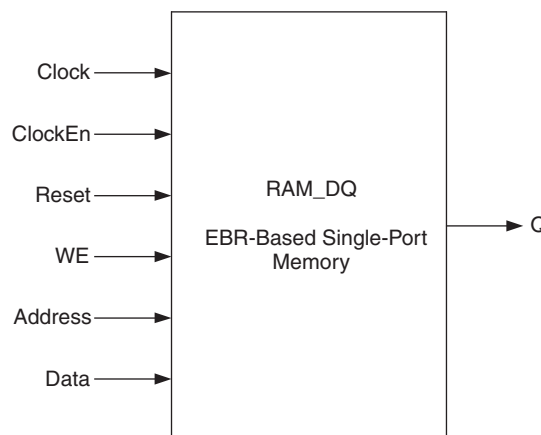
## Memory Modules

### Single-Port RAM (RAM\_DQ) – EBR Based

The EBR blocks in LatticeECP3 devices can be configured as Single-Port RAM or RAM\_DQ. IPexpress allows users to generate the Verilog-HDL or VHDL along the EDIF netlist for the memory size as per design requirements.

IPexpress generates the memory module, as shown in Figure 11-4.

**Figure 11-4. Single-Port Memory Module Generated by IPexpress**



Since the device has a number of EBR blocks, the generated module makes use of these EBR blocks, or primitives, and cascades them to create the memory sizes specified by the user in the IPexpress GUI. For sizes smaller than an EBR block, the module will be created in one EBR block and only one memory can be realized in the block. For memory sizes larger than one EBR block, multiple EBR blocks can be cascaded in depth or width (as required to create these sizes).

In Single-Port RAM mode, the inputs - data and address - are registered at the input of the memory array. The output data of the memory is optionally registered at the output.



The various ports and their definitions for Single-Port Memory are listed in Table 11-1. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM\_DQ primitive.

**Table 11-1. EBR-Based Single-Port Memory Port Definitions**

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
Clock	CLK	Clock
ClockEn	CE	Clock Enable
Address	AD[x:0]	Address Bus
Data	DI[y:0]	Data In
Q	DO[y:0]	Data Out
WE	WE	Write Enable
Reset	RST	Reset
—	CS[2:0]	Chip Select

Reset (or RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

CS, or Chip Select, is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade eight memories easily. If the memory size specified by the user requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

Each EBR block consists of 18,432 bits of RAM. The values for x (address) and y (data) of each EBR block are listed in Table 11-2.

**Table 11-2. Single-Port Memory Sizes for 18K Memories in LatticeECP3 Devices**

Single Port Memory Size	Input Data	Output Data	Address [MSB:LSB]
16,384 x 1	DI	DO	AD[13:0]
8,192 x 2	DI[1:0]	DO[1:0]	AD[12:0]
4,096 x 4	DI[3:0]	DO[3:0]	AD[11:0]
2,048 x 9	DI[8:0]	DO[8:0]	AD[10:0]
1,024 x 18	DI[17:0]	DO[17:0]	AD[9:0]
512 x 36	DI[35:0]	DO[35:0]	AD[8:0]

Table 11-3 shows the various attributes available for the Single-Port Memory (RAM\_DQ). Some of these attributes are user-selectable through the IPexpress GUI.

**Table 11-3. Single-Port RAM Attributes of LatticeECP3 Devices**

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Address depth	Address Depth Read Port	16K, 8K, 4K, 2K, 1K, 512		YES
Data Width	Data Word Width Read Port	1, 2, 4, 9, 18, 36	1	YES
Enable Output Registers	Register Mode (Pipelining) for Write Port	NOREG, OUTREG	NOREG	YES
Enable GSR	Enables Global Set Reset	ENABLE, DISABLE	ENABLE	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Memory File Format		BINARY, HEX, ADDRESSED HEX		YES
Write Mode	Read / Write Mode for Write Port	NORMAL, WRITE-THROUGH, READBEFOREWRITE <sup>1</sup>	NORMAL	YES
Chip Select Decode	Chip Select Decode for Read Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Init Value	Initialization value	0x00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 000000000000.....0xFFFF FFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFF	0x000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 00000	NO

1. READBEFOREWRITE Mode is available only for LatticeECP3-XXEA devices.

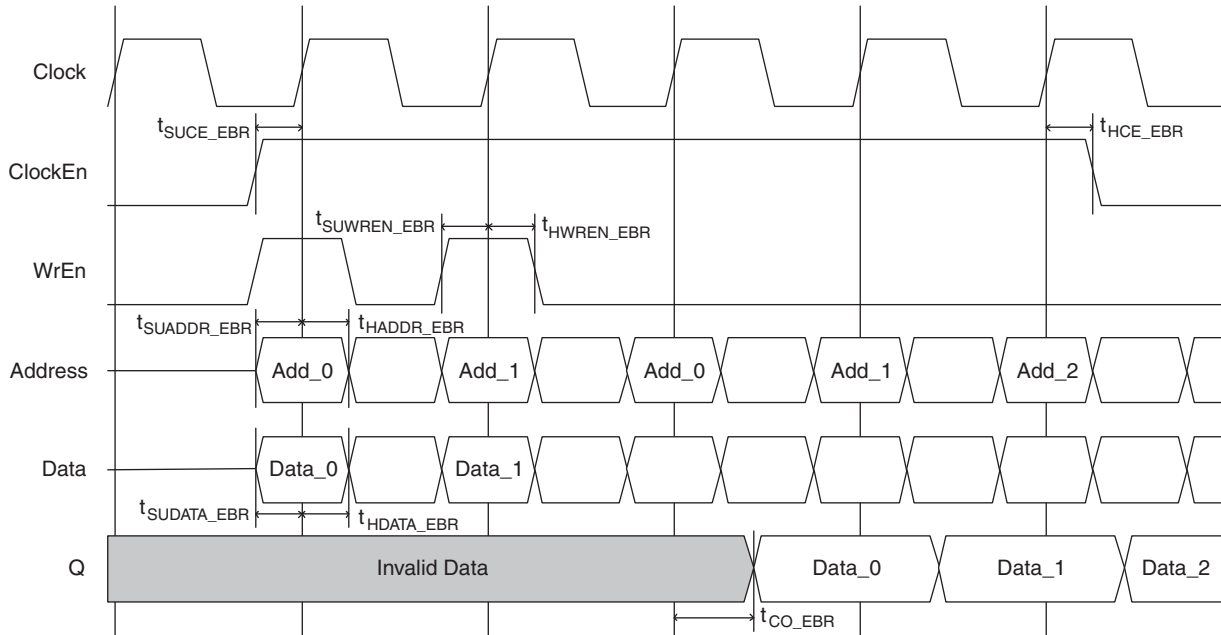
The Single-Port RAM (RAM\_DQ) can be configured as NORMAL, WRITE THROUGH or READBEFOREWRITE modes. Each of these modes affects the data coming out of port Q of the memory during the write operation followed by the read operation at the same memory location.

Additionally, users can select to enable the output registers for RAM\_DQ. Figures 11-5 to 11-8 show the internal timing waveforms for the Single Port RAM (RAM\_DQ) with these options.

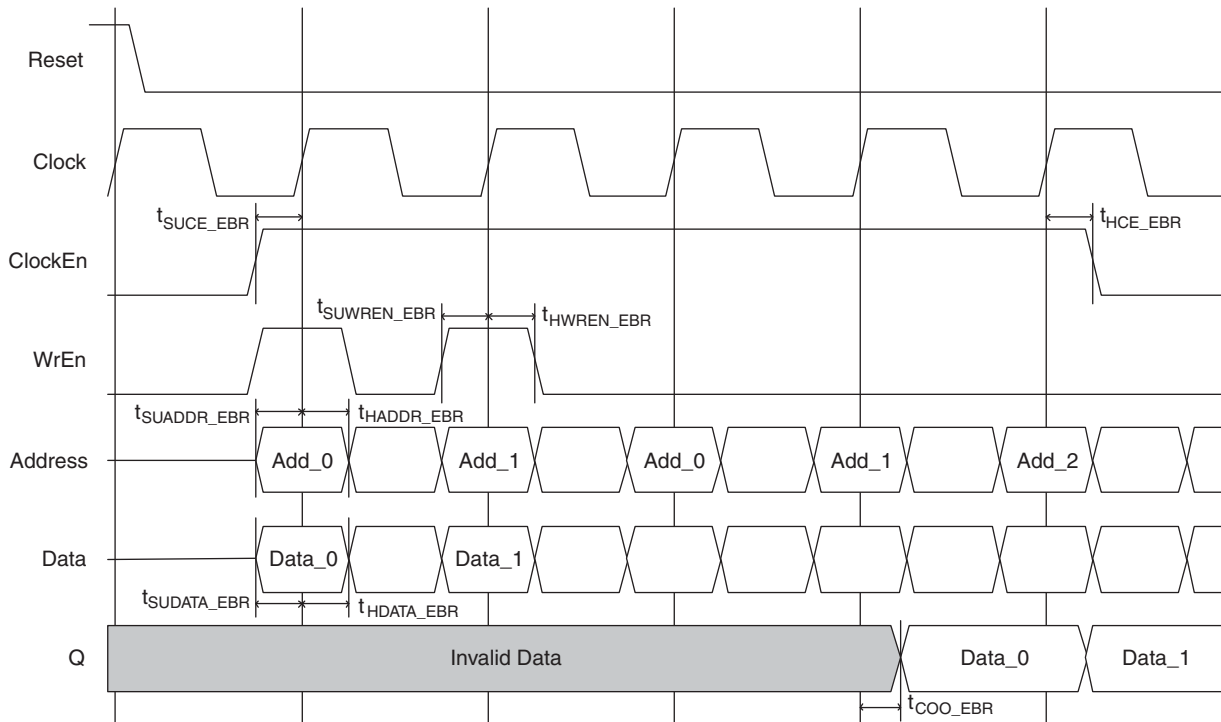
It is important that no setup and hold time violations occur on the address registers (address). Failing to meet these requirements can result in corruption of memory contents. This applies to both read and write operations.

A Post Place and Route timing report in Lattice Diamond® or ispLEVER design software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

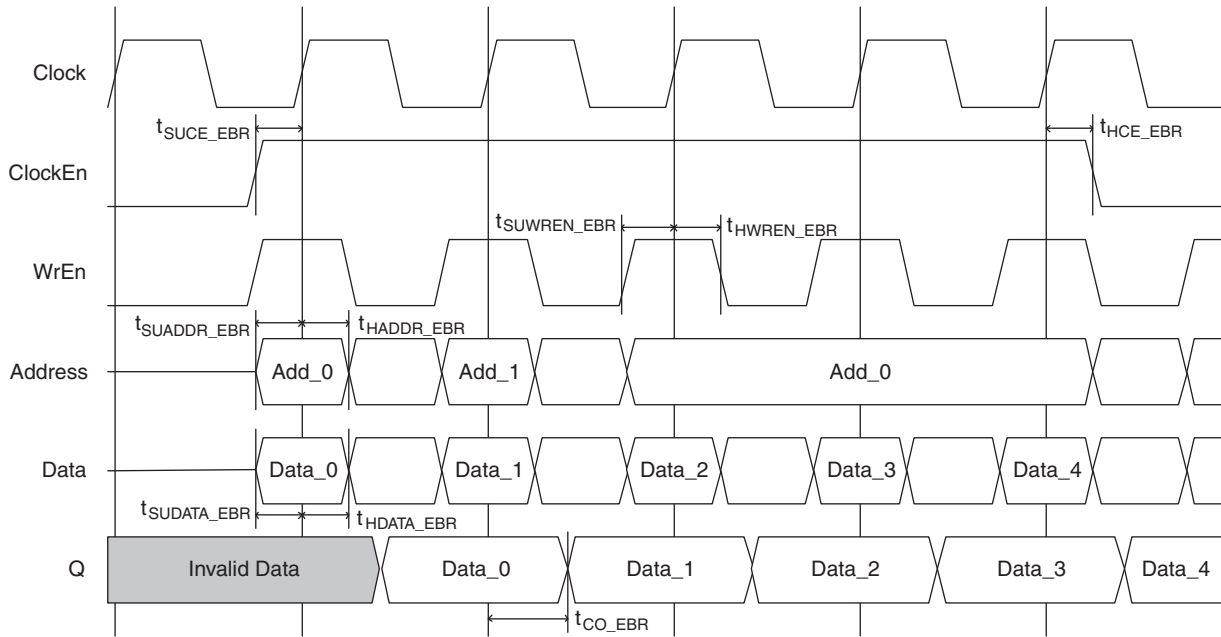
**Figure 11-5. Single Port RAM Timing Waveform in Normal (NORMAL) Mode, without Output Registers**



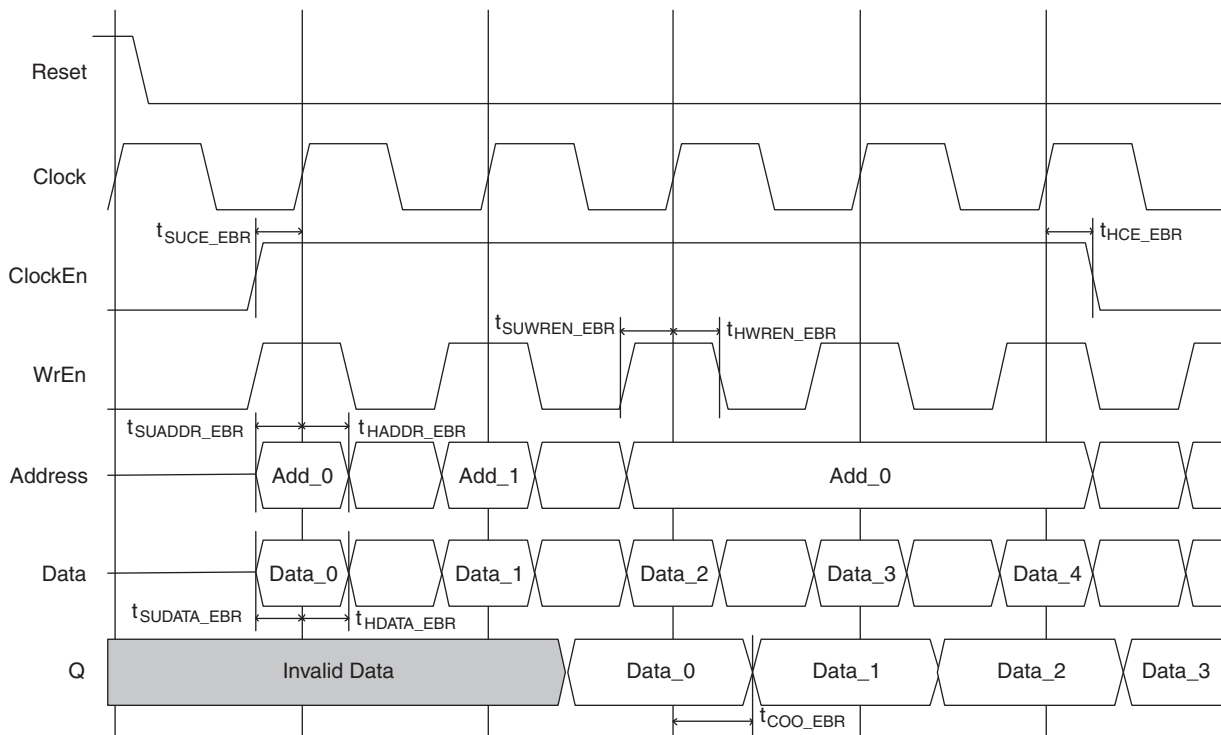
**Figure 11-6. Single Port RAM Timing Waveform in Normal (NORMAL) Mode, with Output Registers**



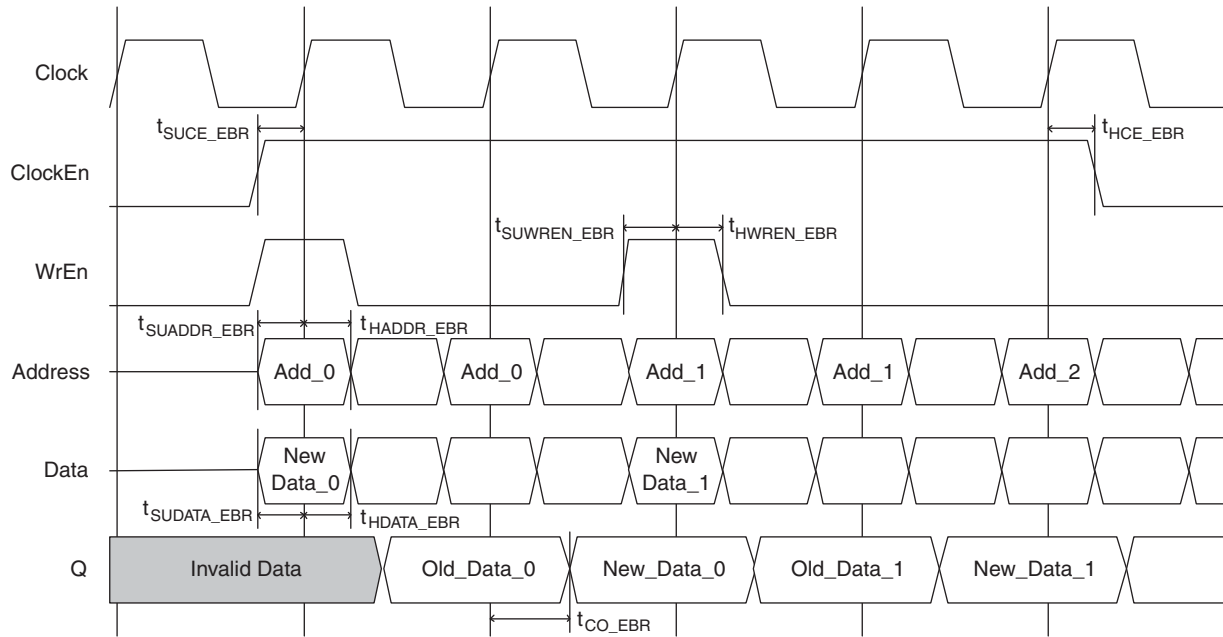
**Figure 11-7. Single Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode, without Output Registers**



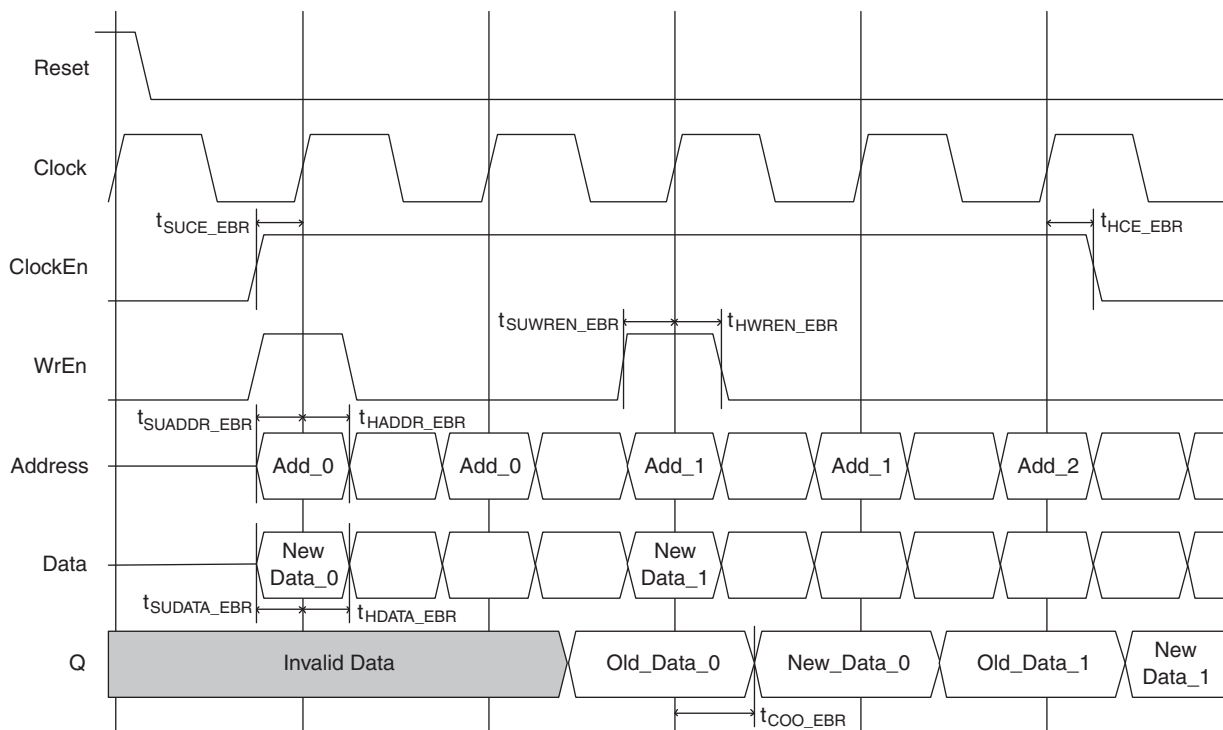
**Figure 11-8. Single Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode, with Output Registers**



**Figure 11-9. Single Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode, without Output Registers**



**Figure 11-10. Single Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode, with Output Registers**

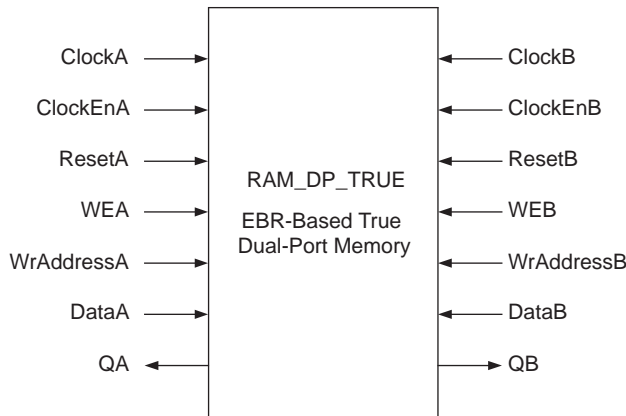


### True Dual-Port RAM (RAM\_DP\_TRUE) – EBR Based

The EBR blocks in the LatticeECP3 devices can be configured as True-Dual Port RAM or RAM\_DP\_TRUE. IPexpress allows users to generate the Verilog-HDL, VHDL or EDIF netlists for the memory size as per design requirements.

IPexpress generates the memory module, as shown in Figure 11-11.

**Figure 11-11. True Dual-Port Memory Module Generated by IPexpress**



The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than one EBR block, the module will be created in one EBR block. Where the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width as required to create these sizes.

In True Dual Port RAM mode, the inputs - data and address - are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions for True Dual-Port RAM are listed in Table 11-4. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM\_DP\_TRUE primitive.

**Table 11-4. EBR-Based True Dual-Port Memory Port Definitions**

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
ClockA, ClockB	CLKA, CLKB	Clock for PortA and PortB
ClockEnA, ClockEnB	CEA, CEB	Clock Enables for Port CLKA and CLKB
AddressA, AddressB	ADA[x:0], ADB[x:0]	Address Bus port A and port B
DataA, DataB	DIA[y:0], DIB[y:0]	Input Data port A and port B
QA, QB	DOA[y:0], DOB[y:0]	Output Data port A and port B
WEA, WEB	WEA, WEB	Write enable port A and port B
ResetA, ResetB	RSTA, RSTB	Reset for PortA and PortB
—	CSA[2:0], CSB[2:0]	Chip Selects for each port

CS, or Chip Select, is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade eight memories easily. However, if the memory size specified by the user requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU external to the EBR blocks.

Each EBR block consists of 18,432 bits of RAM. The values for x (address) and y (data) of each EBR block are listed in Table 11-5.

**Table 11-5. Dual-Port Memory Sizes for 18K Memory in LatticeECP3 Devices**

Dual Port Memory Size	Input Data Port A	Input Data Port B	Output Data Port A	Output Data Port B	Address Port A [MSB:LSB]	Address Port B [MSB:LSB]
16,384 x 1	DIA	DIB	DOA	DOB	ADA[13:0]	ADB[13:0]
8,192 x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	ADA[12:0]	ADB[12:0]
4,096 x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	ADA[11:0]	ADB[11:0]
2,048 x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	ADA[10:0]	ADB[10:0]
1,024 x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	ADA[9:0]	ADB[9:0]

Table 11-6 shows the various attributes available for True Dual-Port Memory (RAM\_DQ). Some of these attributes are user-selectable through the IPexpress GUI.

**Table 11-6. True Dual-Port RAM Attributes for LatticeECP3 Devices**

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Port A Address depth	Address Depth Port A	16K, 8K, 4K, 2K, 1K		YES
Port A Data Width	Data Word Width Port A	1, 2, 4, 9, 18	1	YES
Port B Address depth	Address Depth Port B	16K, 8K, 4K, 2K, 1K		YES
Port B Data Width	Data Word Width Port B	1, 2, 4, 9, 18	1	YES
Port A Enable Output Registers	Register Mode (Pipelining) for Port A	NOREG, OUTREG	NOREG	YES
Port B Enable Output Registers	Register Mode (Pipelining) for Port B	NOREG, OUTREG	NOREG	YES
Enable GSR	Enables Global Set Reset	ENABLE, DISABLE	ENABLE	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Memory File Format		BINARY, HEX, ADDRESSED HEX		YES
Port A Write Mode	Read / Write Mode for Port A	NORMAL, WRITE-THROUGH, READBEFOREWRITE <sup>1</sup>	NORMAL	YES
Port B Write Mode	Read / Write Mode for Port B	NORMAL, WRITE-THROUGH, READBEFOREWRITE <sup>1</sup>	NORMAL	YES
Chip Select Decode for Port A	Chip Select Decode for Port A	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Chip Select Decode for Port B	Chip Select Decode for Port B	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Init Value	Initialization value	0x00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 0000000000000000.....0xF FFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFF	0x00000000 000000000000 000000000000 000000000000 000000000000 000000000000 000000000000 000000000000 000000000000 000000000000 000000000000	NO

1. READBEFOREWRITE Mode is available only for LatticeECP3-XXEA devices.

The True Dual Port RAM (RAM\_DP\_TRUE) can be configured as NORMAL, WRITE THROUGH or READBEFOREWRITE modes. Each of these modes affects what data comes out of the port Q of the memory during the write operation followed by the read operation at the same memory location. The detailed discussions of the WRITE modes and the constraints of the True Dual Port can be found in Appendix A.

Additionally, users can select to enable the output registers for RAM\_DP\_TRUE. Figures 11-12 through 11-15 show the internal timing waveforms for the True Dual Port RAM (RAM\_DP\_TRUE) with these options.

It is important that no setup and hold time violations occur on the address registers (AddressA and AddressB). Failing to meet these requirements can result in corruption of memory contents. This applies to both read and write operations.

A Post Place and Route timing report in Lattice Diamond or ispLEVER design software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

**Figure 11-12. True Dual Port RAM Timing Waveform in Normal (NORMAL) Mode, without Output Registers**

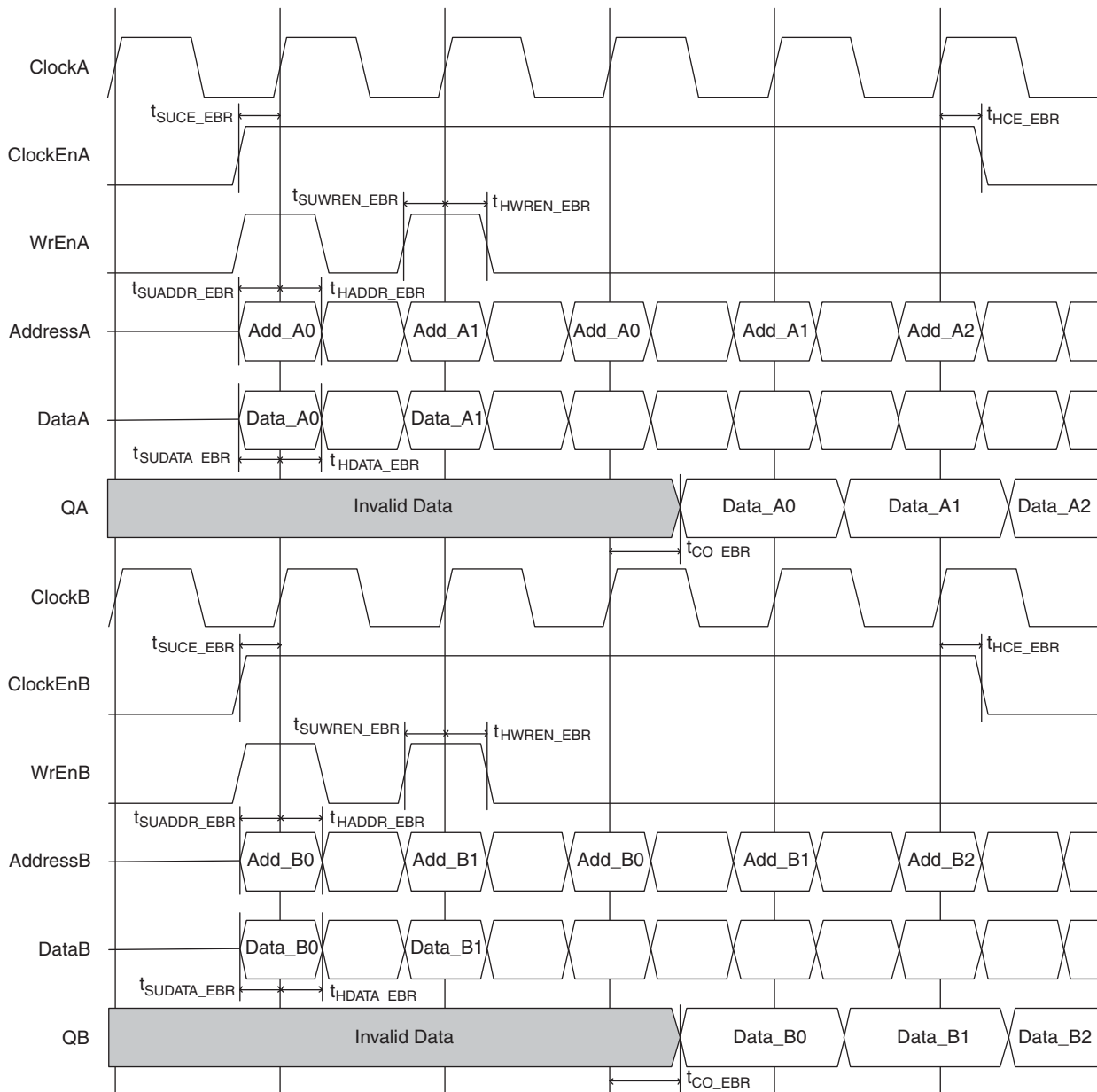
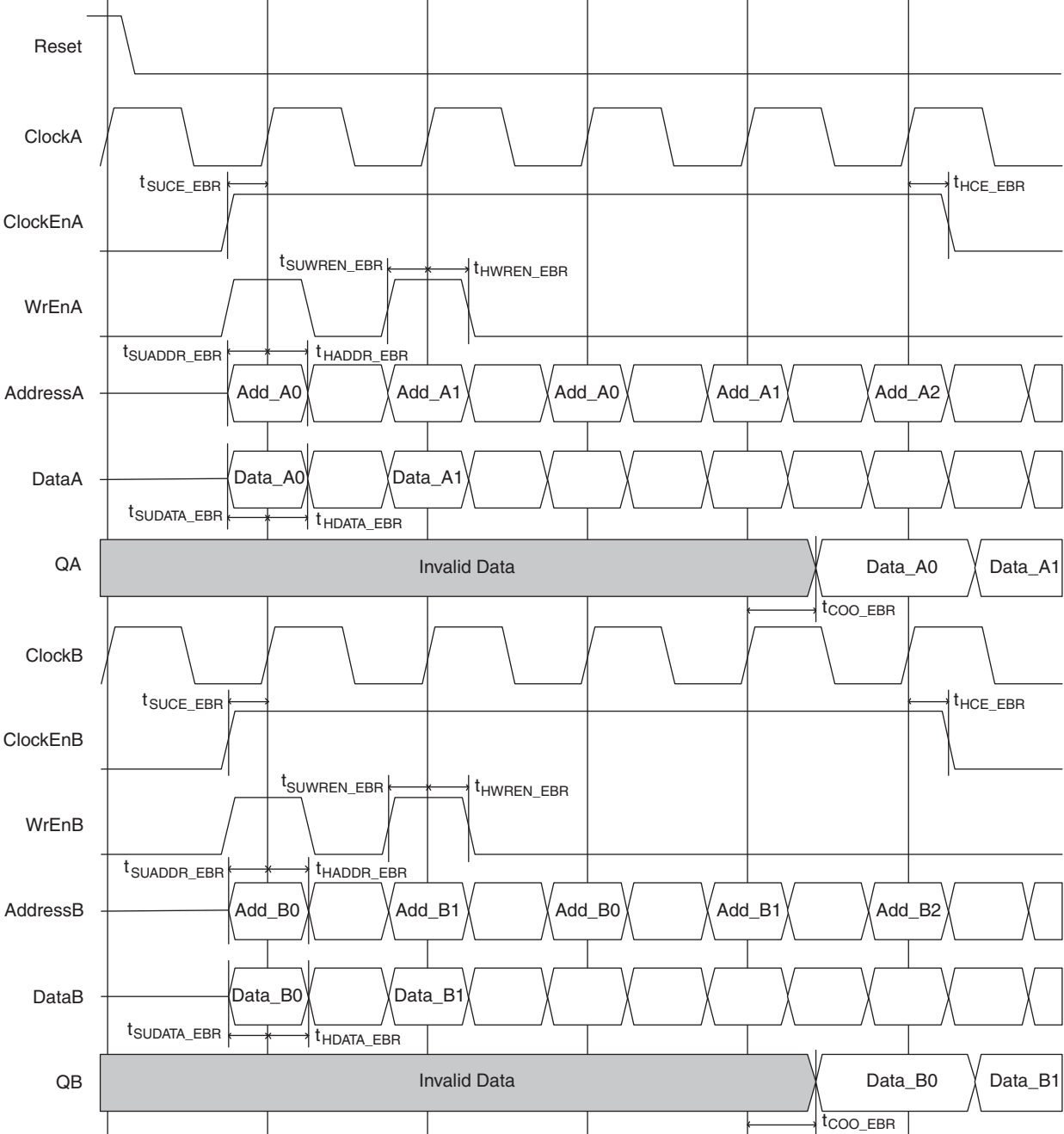
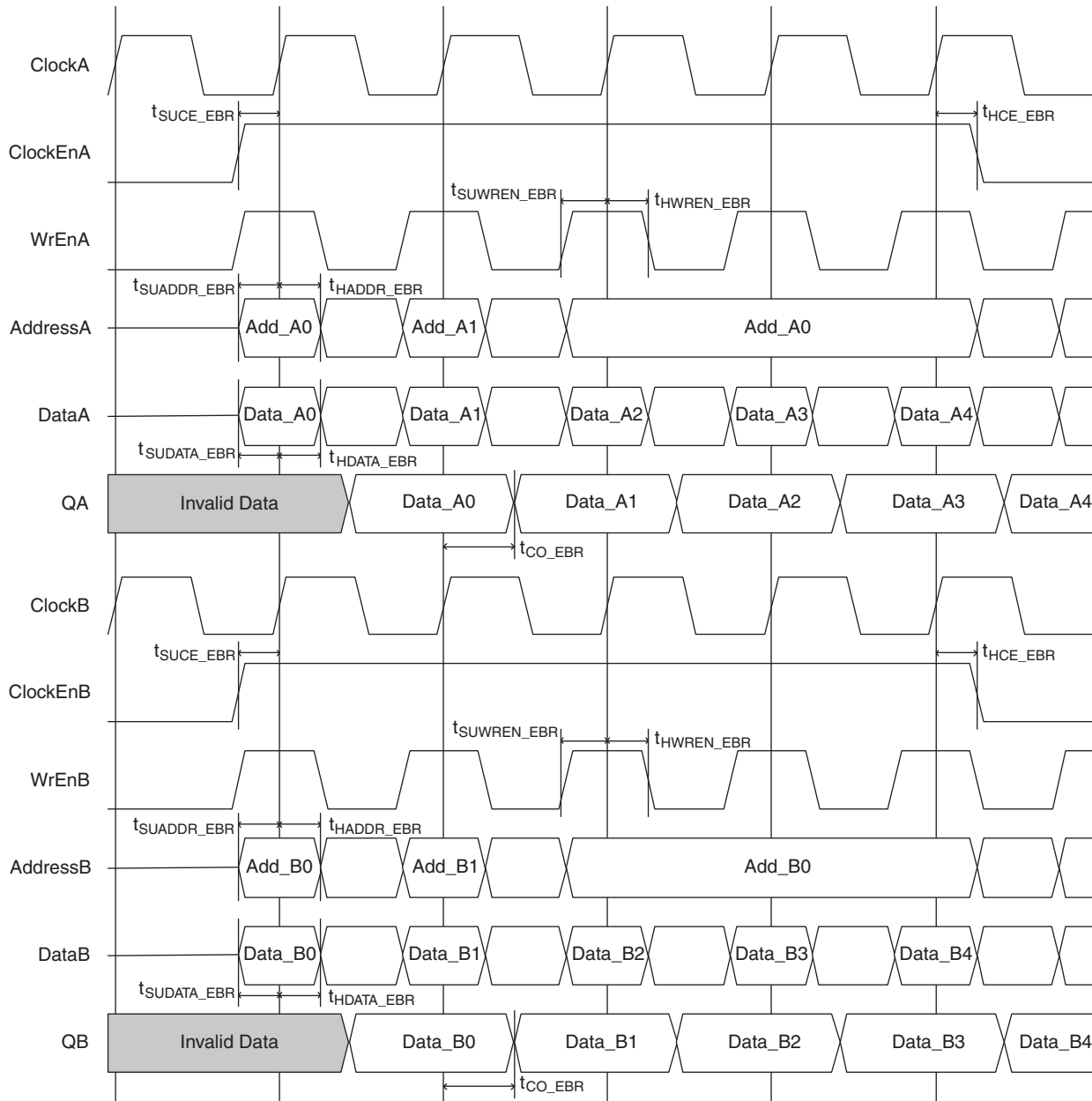




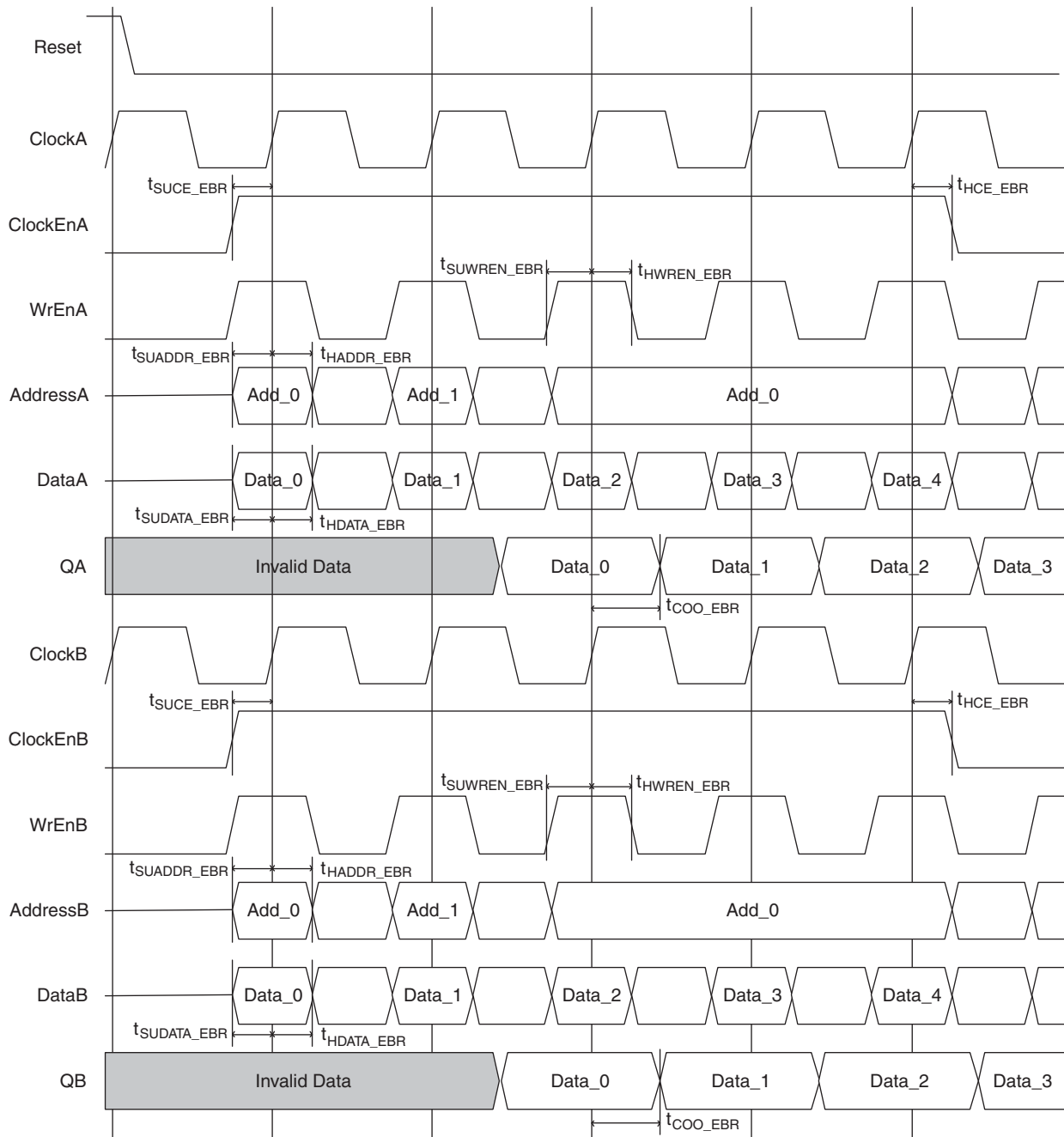
Figure 11-13. True Dual Port RAM Timing Waveform in Normal (NORMAL) Mode with Output Registers



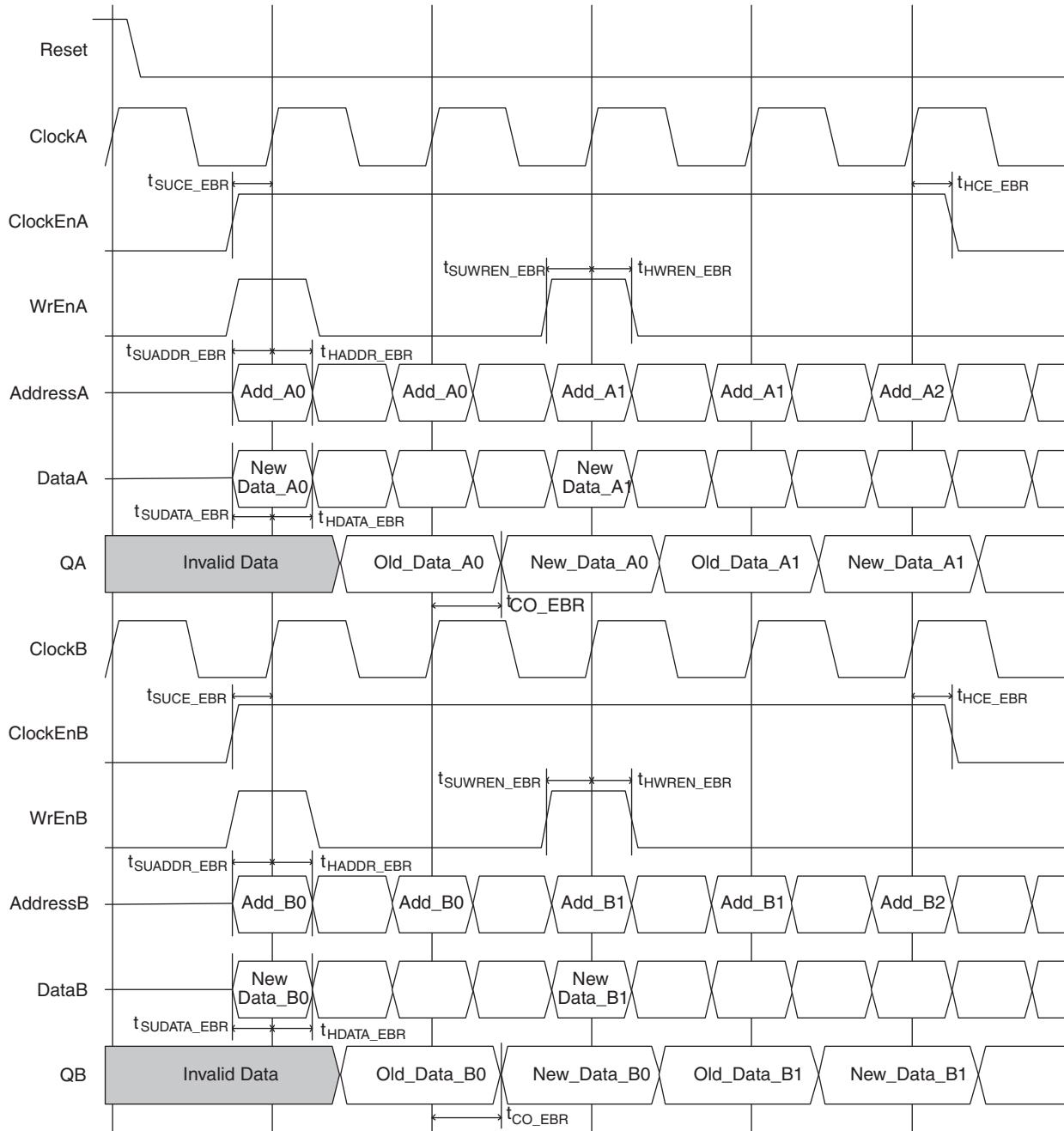
**Figure 11-14. True Dual Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode, without Output Registers**



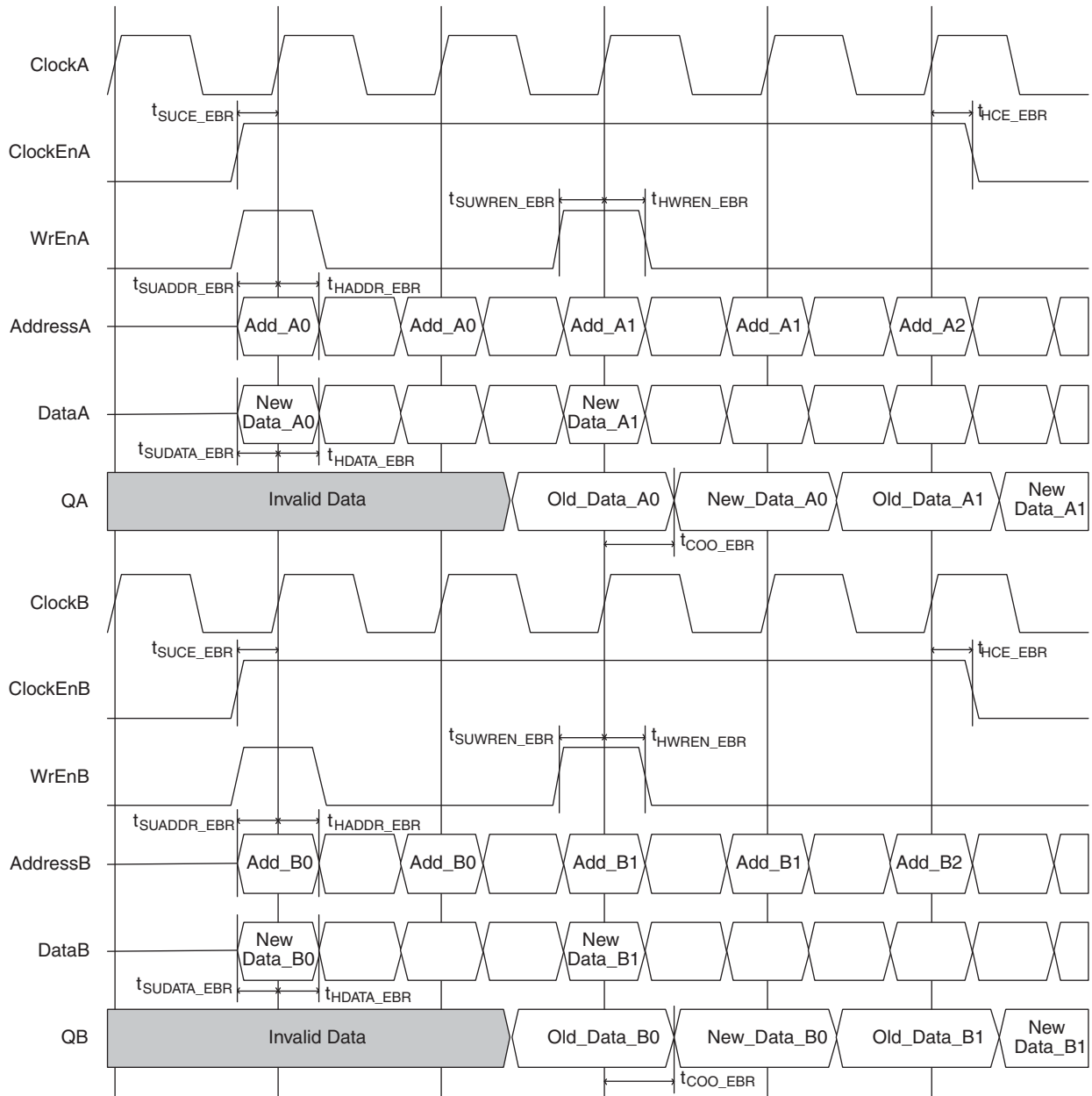
**Figure 11-15. True Dual Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode, with Output Registers**



**Figure 11-16. True Dual Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode, without Output Registers**



**Figure 11-17. True Dual Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode, with Output Registers**

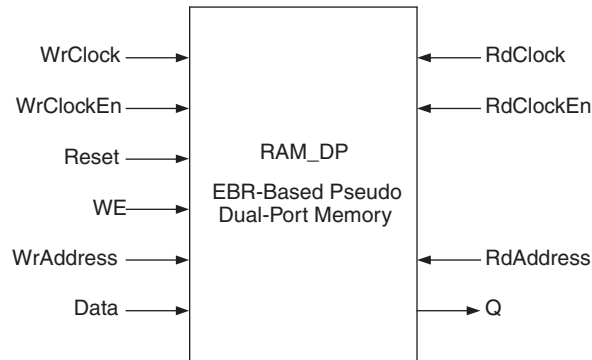


### Pseudo Dual-Port RAM (RAM\_DP) – EBR Based

The EBR blocks in LatticeECP3 devices can be configured as Pseudo-Dual Port RAM or RAM\_DP. IPexpress allows users to generate the Verilog-HDL or VHDL along with EDIF netlists for the memory size as per design requirements.

IPexpress generates the memory module shown in Figure 11-18.

**Figure 11-18. Pseudo Dual-Port Memory Module Generated by IPexpress**



The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than one EBR block, the module will be created in one EBR block. Where the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width as required to create these sizes.

In Pseudo Dual-Port RAM mode the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions for Pseudo Dual-Port memory are listed in Table 11-7. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM\_DP primitive.

**Table 11-7. EBR-Based Pseudo Dual-Port Memory Port Definitions**

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
RdAddress	ADR[x:0]	Read Address
WrAddress	ADW[x:0]	Write Address
RdClock	CLKR	Read Clock
WrClock	CLKW	Write Clock
RdClockEn	CER	Read Clock Enable
WrClockEn	CEW	Write Clock Enable
Q	DO[y:0]	Read Data
Data	DI[y:0]	Write Data
WE	WE	Write Enable
Reset	RST	Reset
—	CS[2:0]	Chip Select

Reset (RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

CS, or Chip Select, is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade eight memories easily. However, if the memory size specified by the user requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU external to the EBR blocks.

Each EBR block consists of 18,432 bits of RAM. The values for x (address) and y (data) of each EBR block are listed in Table 11-8.

**Table 11-8. Pseudo Dual-Port Memory Sizes for 18K Memory in LatticeECP3 Devices**

Pseudo Dual-Port Memory Size	Input Data Port B	Output Data Port A	Read Address Port A [MSB:LSB]	Write Address Port B [MSB:LSB]
16,384 x 1	DIB	DOA	RAD[13:0]	WAD[13:0]
8,192 x 2	DIB[1:0]	DOA[1:0]	RAD[12:0]	WAD[12:0]
4,096 x 4	DIB[3:0]	DOA[3:0]	RAD[11:0]	WAD[11:0]
2,048 x 9	DIB[8:0]	DOA[8:0]	RAD[10:0]	WAD[10:0]
1,024 x 18	DIB[17:0]	DOA[17:0]	RAD[9:0]	WAD[9:0]
512 x 36	DIB[35:0]	DOA[35:0]	RAD[8:0]	WAD[8:0]

Table 11-9 shows the various attributes available for the Pseudo Dual-Port Memory (RAM\_DP). Some of these attributes are user-selectable through the IPexpress GUI.

**Table 11-9. Pseudo Dual-Port RAM Attributes for LatticeECP3 Devices**

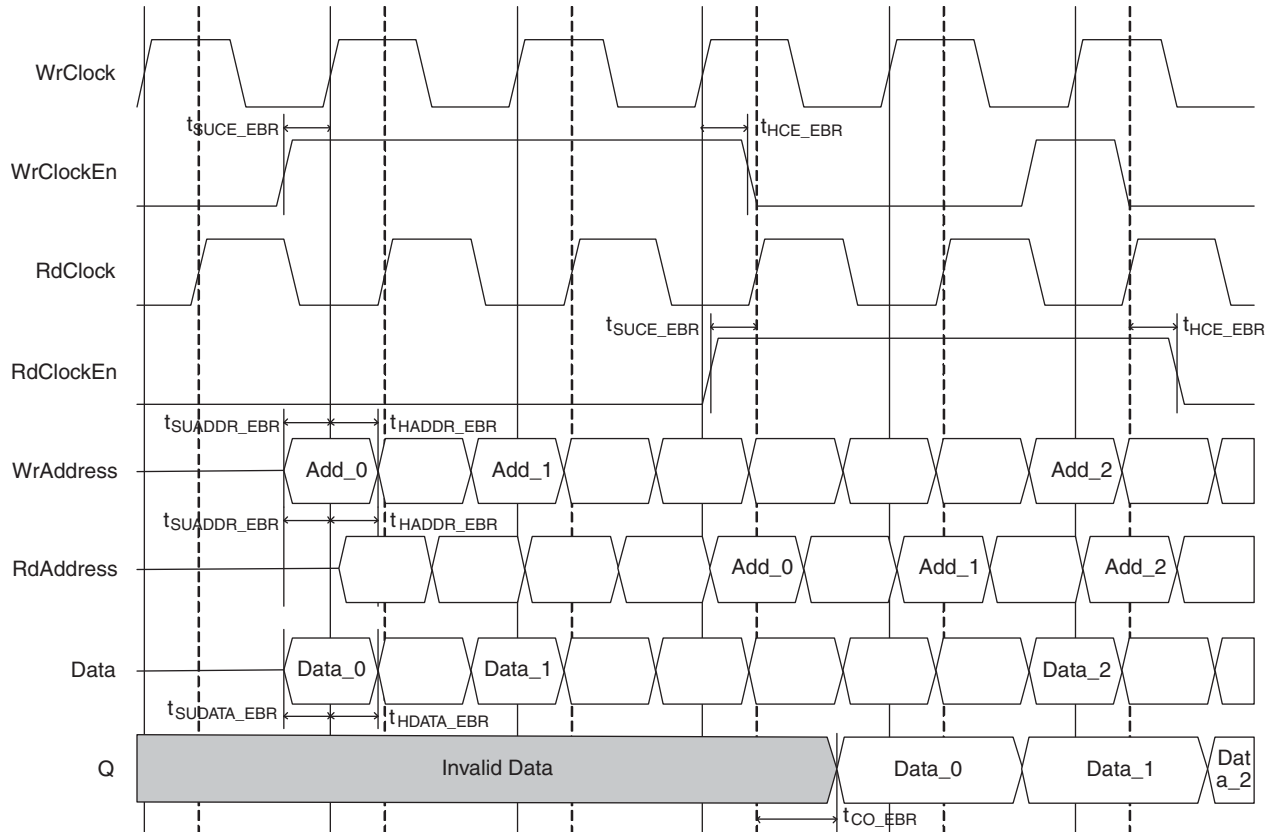
Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Read Port Address Depth	Address Depth Read Port	16K, 8K, 4K, 2K, 1K, 512		YES
Read Port Data Width	Data Word Width Read Port	1, 2, 4, 9, 18, 36	1	YES
Write Port Address Depth	Address Depth Write Port	16K, 8K, 4K, 2K, 1K		YES
Write Port Data Width	Data Word Width Write Port	1, 2, 4, 9, 18, 36	1	YES
Write Port Enable Output Registers	Register Mode (Pipelining) for Write Port	NOREG, OUTREG	NOREG	YES
Enable GSR	Enables Global Set Reset	ENABLE, DISABLE	ENABLE	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Memory File Format		BINARY, HEX, ADDRESSED HEX		YES
Read Port Write Mode	Read / Write Mode for Read Port	NORMAL	NORMAL	YES
Write Port Write Mode	Read / Write Mode for Write Port	NORMAL	NORMAL	YES
Chip Select Decode for Read Port	Chip Select Decode for Read Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Chip Select Decode for Write Port	Chip Select Decode for Write Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO
Init Value	Initialization value	0x00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 0.....0xFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFF	0x000000000000 000000000000 000000000000 000000000000 000000000000 000000000000 000000000000 000000000000	NO

Users have the option to enable the output registers for Pseudo-Dual Port RAM (RAM\_DP). Figures 11-19 and 11-20 show the internal timing waveforms for Pseudo-Dual Port RAM (RAM\_DP) with these options.

It is important that no setup and hold time violations occur on the address registers (RdAddress and WrAddress). Failing to meet these requirements can result in corruption of memory contents. This applies to both read and write operations.

A Post Place and Route timing report in Lattice Diamond or ispLEVER design software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

**Figure 11-19. PSEUDO DUAL PORT RAM Timing Diagram - without Output Registers**







The various ports and their definitions are listed in Table 11-10. The table lists the corresponding ports for the module generated by IPexpress and for the ROM primitive.

**Table 11-10. EBR-Based ROM Port Definitions**

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description
Address	AD[x:0]	Read Address
OutClock	CLK	Clock
OutClockEn	CE	Clock Enable
Reset	RST	Reset
—	CS[2:0]	Chip Select

Reset (RST) resets only the input and output registers of the RAM. It does not reset the contents of the memory.

Chip Select (CS) is a useful port when multiple cascaded EBR blocks are required by the memory. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. Since CS is a 3-bit bus, it can cascade eight memories easily. However, if the memory size specified by the user requires more than eight EBR blocks, the ispLEVER software automatically generates the additional address decoding logic, which is implemented in the PFU external to the EBR blocks.

When generating ROM using IPexpress, the designer must provide the initialization file to pre-initialize the contents of the ROM. These files are the \*.mem files and they can be of binary, hex or addressed hex formats. The initialization files are discussed in detail in the Initializing Memory section of this document.

Users have the option of enabling the output registers for Read Only Memory (ROM). Figures 11-19 and 11-20 show the internal timing waveforms for the Read Only Memory (ROM) with these options.

Each EBR block consists of 18,432 bits of RAM. The values for x's (for address) and y's (data) for each EBR block for the devices included in Table 11-11.

**Table 11-11. ROM Memory Sizes for 16K Memory for LatticeECP3**

ROM	Output Data	Address Port [MSB:LSB]
16K x 1	DOA	WAD[13:0]
8K x 2	DOA[1:0]	WAD[12:0]
4K x 4	DOA[3:0]	WAD[11:0]
2K x 9	DOA[8:0]	WAD[10:0]
1K x 18	DOA[17:0]	WAD[9:0]
512 x 36	DOA[35:0]	WAD[8:0]

LatticeECP3 FPGAs have Embedded block RAMs (EBR) which can be configured in Single-Port (RAM\_DQ), Pseudo Dual-Port (RAM\_DP) and True Dual-Port (RAM\_DP\_TRUE) RAMs. The FIFOs can be emulated to be built around these RAMs. The IPexpress point tool in ispLEVER allows users to build a FIFO and FIFO\_DC around Pseudo Dual-Port RAM (or DP\_RAM).

Table 11-12 shows the various attributes available for the Read Only Memory (ROM). Some of these attributes are user-selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

**Table 11-12. ROM Attributes for LatticeECP3**

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Address depth	Address Depth Read Port	16K, 8K, 4K, 2K, 1K, 512		YES
Data Width	Data Word Width Read Port	1, 2, 4, 9, 18, 36	1	YES
Enable Output Registers	Register Mode (Pipelining) for Write Port	NOREG, OUTREG	NOREG	YES
Enable GSR	Enables Global Set Reset	ENABLE, DISABLE	ENABLE	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Memory File Format		BINARY, HEX, ADDRESSED HEX		YES
Chip Select Decode	Chip Select Decode for Read Port	0b000, 0b001, 0b010, 0b011, 0b100, 0b101, 0b110, 0b111	0b000	NO

Users have the option to enable the output registers for Read Only Memory (ROM). Figures 11-22 and 11-23 show the internal timing waveforms for ROM with these options.

It is important that no setup and hold time violations occur on the address registers (Address). Failing to meet these requirements can result in corruption of memory contents. This applies to both read operations in this case.

A Post Place and Route timing report in Lattice Diamond or ispLEVER design software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

**Figure 11-22. ROM Timing Waveform - without Output Registers**

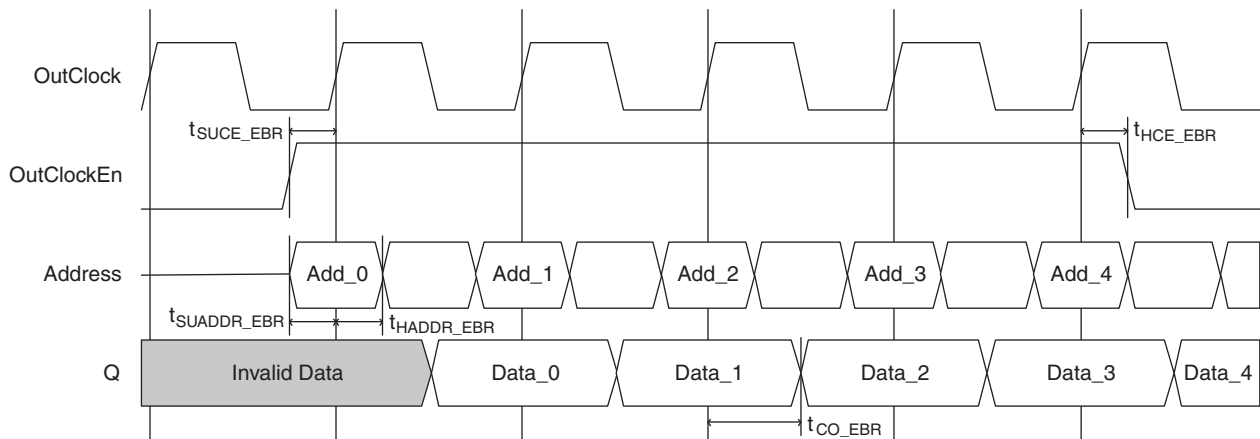
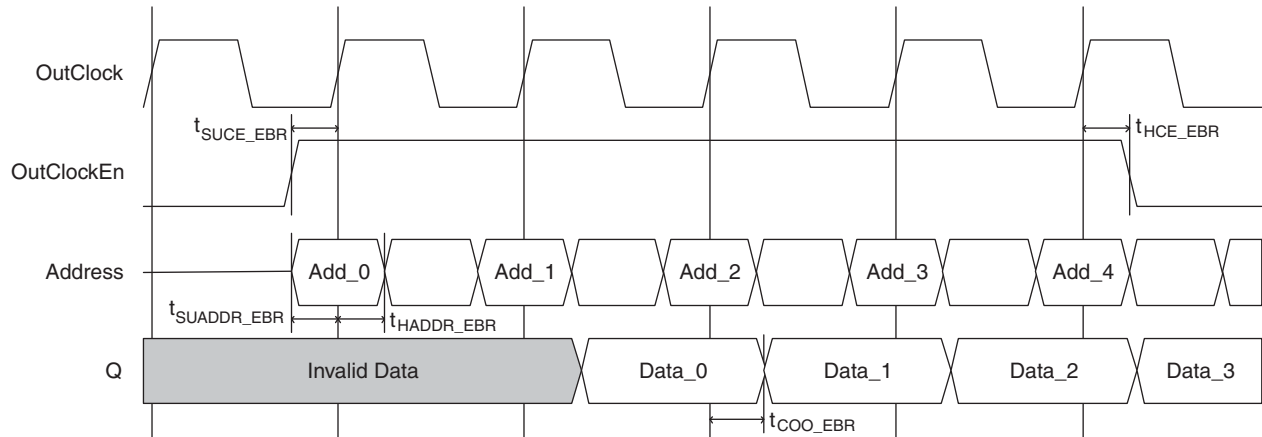


Figure 11-23. ROM Timing Waveform - with Output Registers



## First In First Out (FIFO) Memory

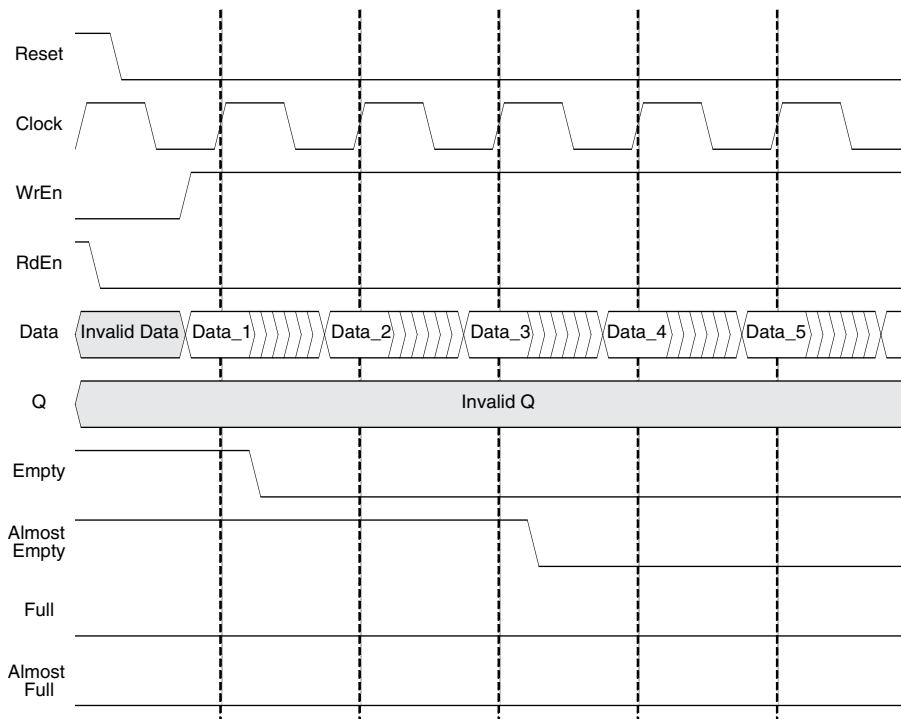
The FIFO, or the single clock FIFO, is an emulated FIFO. The address logic and flag logic is implemented in the FPGA fabric around the RAM.

The ports available on the FIFO are:

- Reset
- Clock
- WrEn
- RdEn
- Data
- Q
- Full Flag
- Almost Full Flag
- Empty Flag
- Almost Empty Flag

Let us first discuss the non-pipelined or the FIFO without output registers. Figure 11-24 shows the operation of the FIFO when it is empty and the data begins to be written into it.

**Figure 11-24. FIFO Without Output Registers, Start of data Write Cycle**

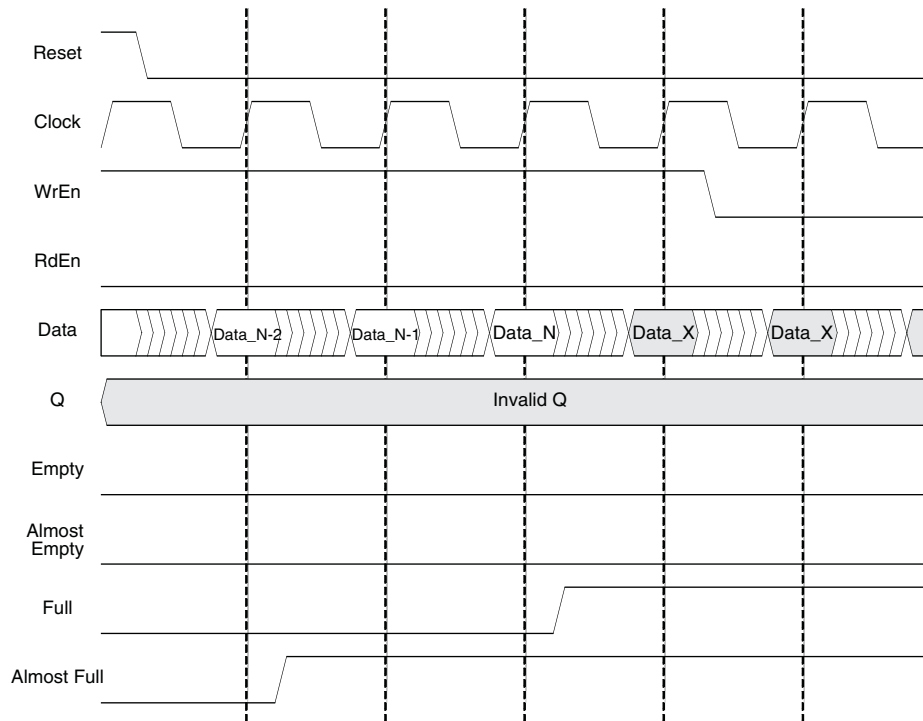


The WrEn signal must be high to start writing into the FIFO. The Empty and Almost Empty flags are high to begin and Full and Almost Full are low.

When the first data is written into the FIFO, the Empty flag de-asserts (or goes low) since the FIFO is no longer empty. In this figure we assume that the Almost Empty flag setting is 3 (address location 3). So, the Almost Empty flag is de-asserted when the third address location is filled.

Assume that we continue to write into the FIFO to fill it. When the FIFO is filled, the Almost Full and Full flags are asserted. Figure 11-25 shows the behavior of these flags. In this figure we assume that the FIFO depth is 'N'.

**Figure 11-25. FIFO Without Output Registers, End of Data Write Cycle**

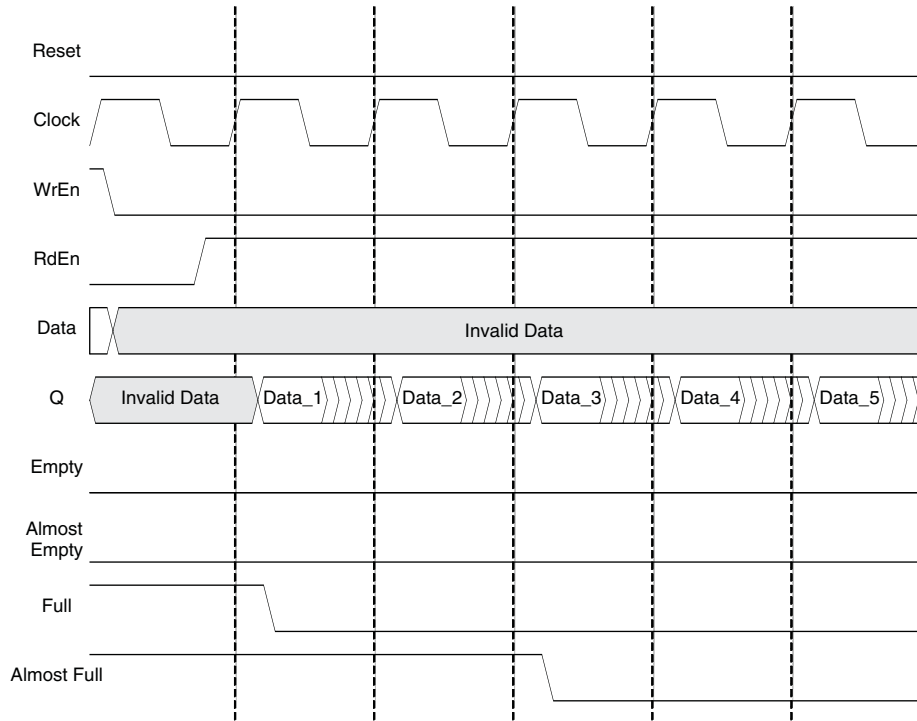


In Figure 11-25, the Almost Full flag is two locations before the FIFO is filled. The Almost Full flag is asserted when the N-2 location is written, and the Full flag is asserted when the last word is written into the FIFO.

Data\_X data inputs are not written since the FIFO is full (Full flag is high).

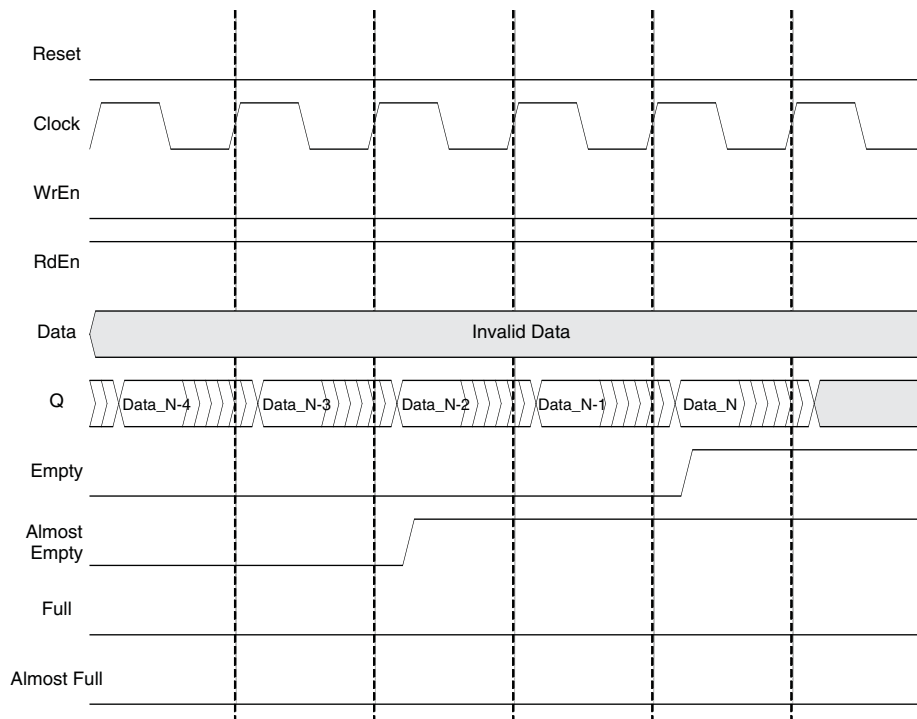
Now let us look at the waveforms when the contents of the FIFO are read out. Figure 11-26 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags are de-asserted, as shown.

**Figure 11-26. FIFO Without Output Registers, Start of Data Read Cycle**



Similarly, as the data is read out and FIFO is emptied, the Almost Empty and Empty flags are asserted (see Figure 11-27).

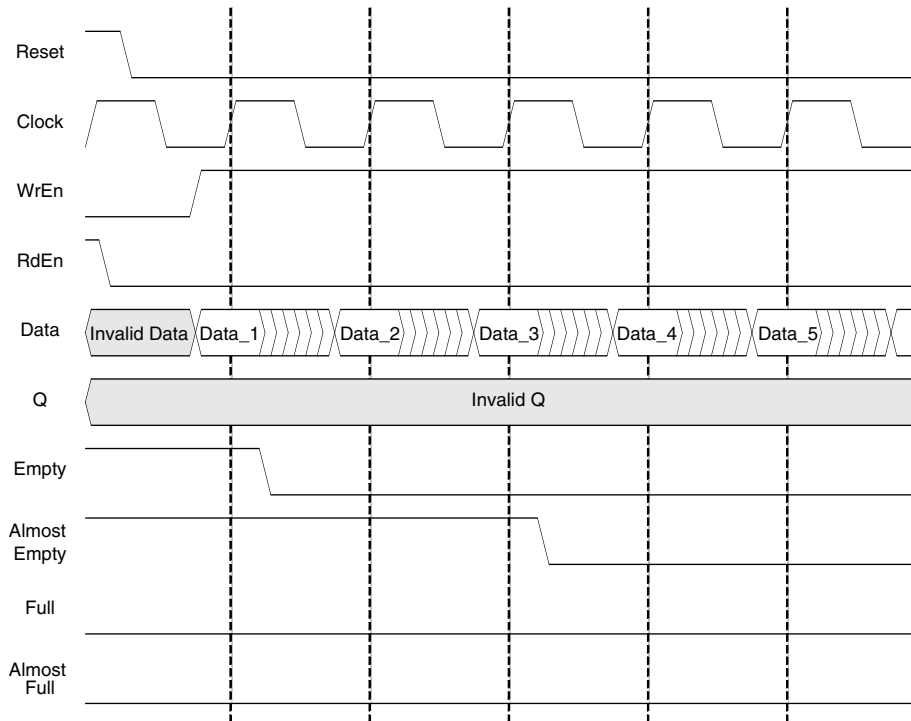
**Figure 11-27. FIFO Without Output Registers, End of Data Read Cycle**



Figures 11-24 to 11-27 show the behavior of non-pipelined FIFO or FIFO without output registers. When the registers are pipelined, the output data is delayed by one clock cycle. There is also an option for output registers to be enabled by the RdEn signal.

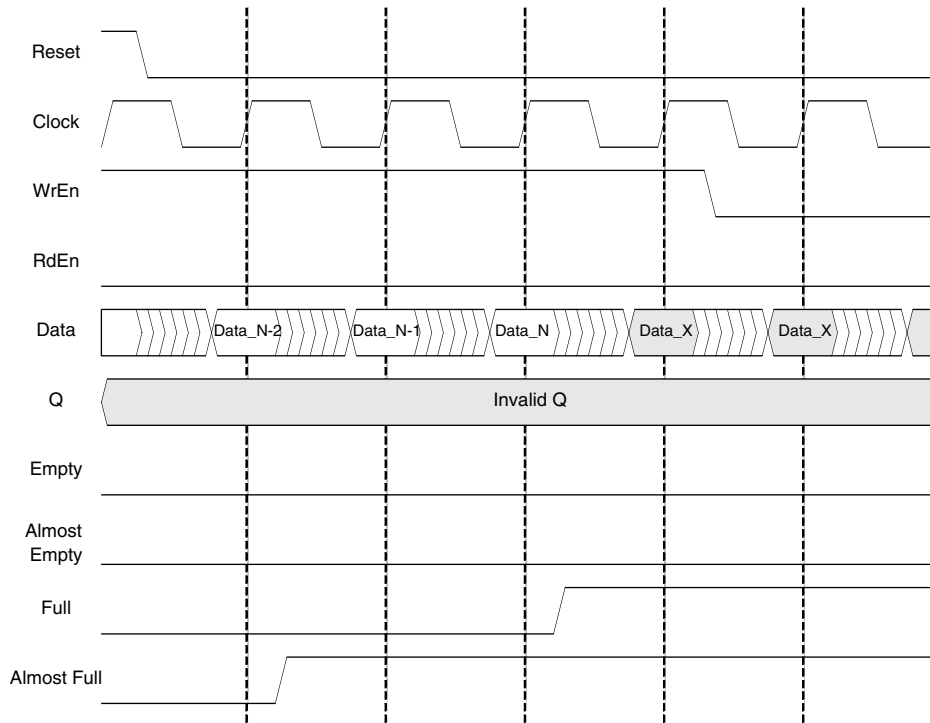
Figures 11-28 to 11-31 show similar waveforms for the FIFO with an output register and an output register enable with RdEn. Note that flags are asserted and de-asserted with similar timing to the FIFO without output registers. Only the data out 'Q' gets delayed by one clock cycle.

**Figure 11-28. FIFO with Output Registers, Start of Data Write Cycle**

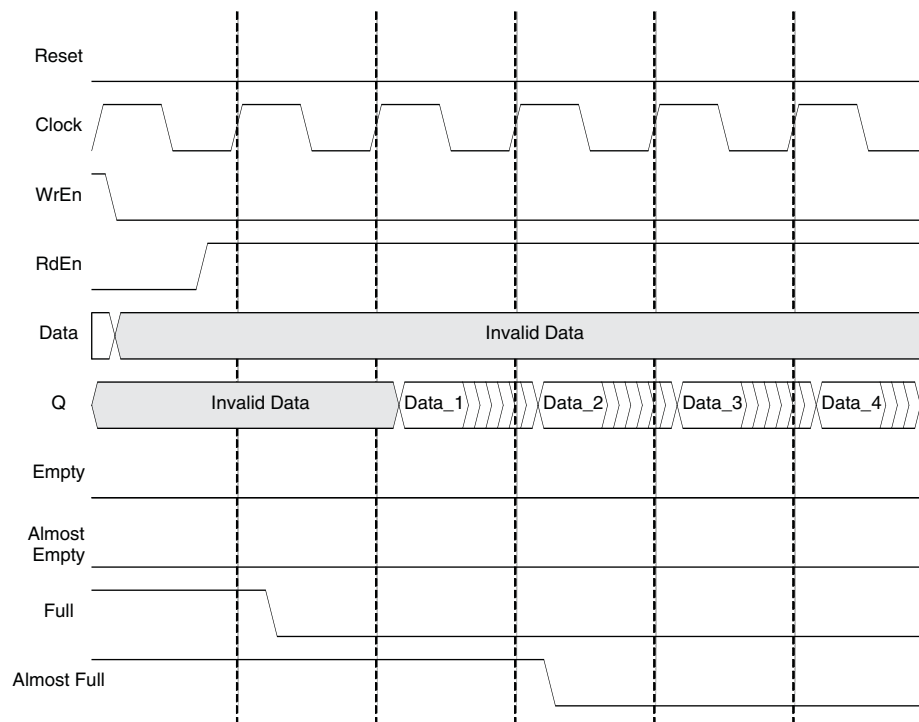




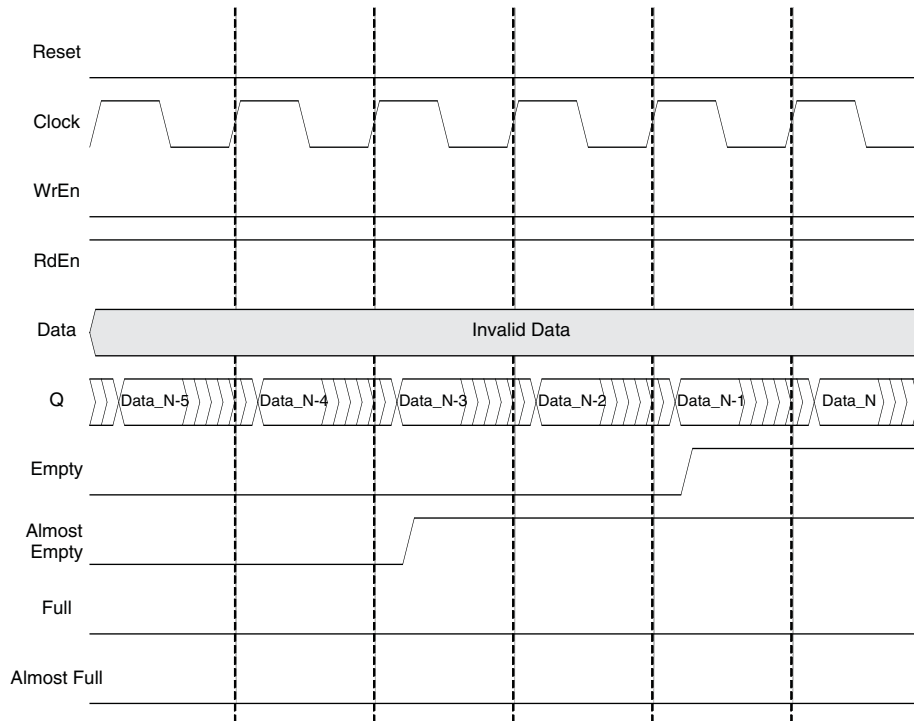
**Figure 11-29. FIFO with Output Registers, End of Data Write Cycle**



**Figure 11-30. FIFO with Output Registers, Start of Data Read Cycle**

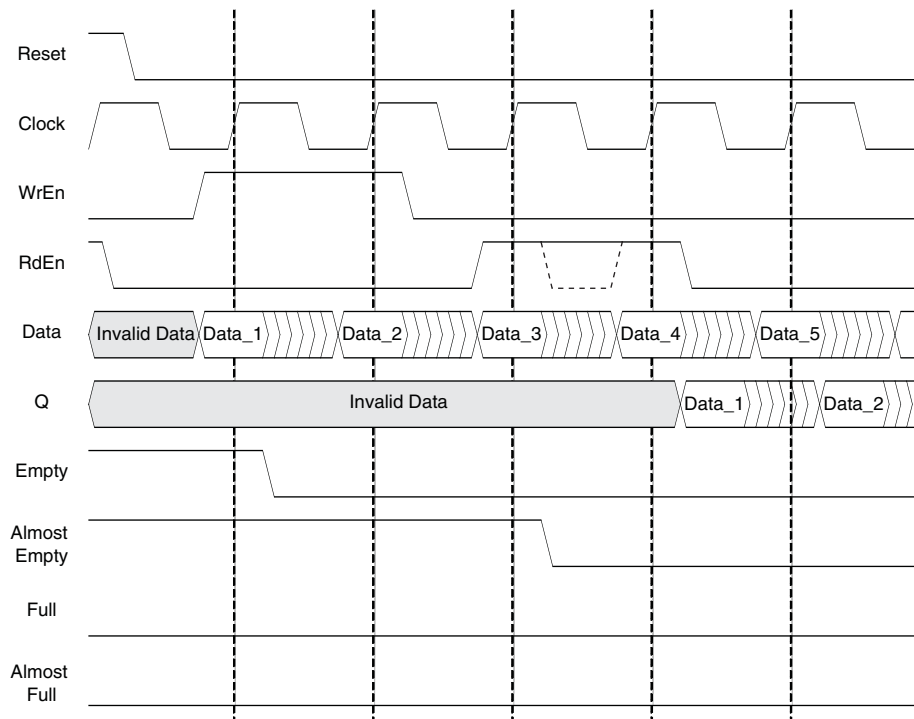


**Figure 11-31. FIFO with Output Registers, End of Data Read Cycle**



If the option enable output register with RdEn is selected, it still delays the data out by one clock cycle (as compared to the non-pipelined FIFO). The RdEn should also be high during that clock cycle, otherwise the data takes an extra clock cycle when the RdEn goes true.

**Figure 11-32. FIFO with Output Registers and RdEn on Output Registers**



---

## Dual Clock First-In-First-Out (FIFO\_DC) Memory

The FIFO\_DC, or dual clock FIFO, is also an emulated FIFO. The address logic and flag logic are implemented in the FPGA fabric around the RAM.

The ports available on the FIFO\_DC include:

- Reset
- RPRreset
- WrClock
- RdClock
- WrEn
- RdEn
- Data
- Q
- Full Flag
- Almost Full Flag
- Empty Flag
- Almost Empty Flag

### FIFO\_DC Flags

As an emulated FIFO, FIFO\_DC requires the flags to be implemented in the FPGA logic around the block RAM. Because of the two clocks, the flags are required to change clock domains from read clock to write clock and vice versa. This adds latency to the flags either during assertion or de-assertion. Latency can be avoided only in one of the cases (either assertion or de-assertion) or distributed among the two.

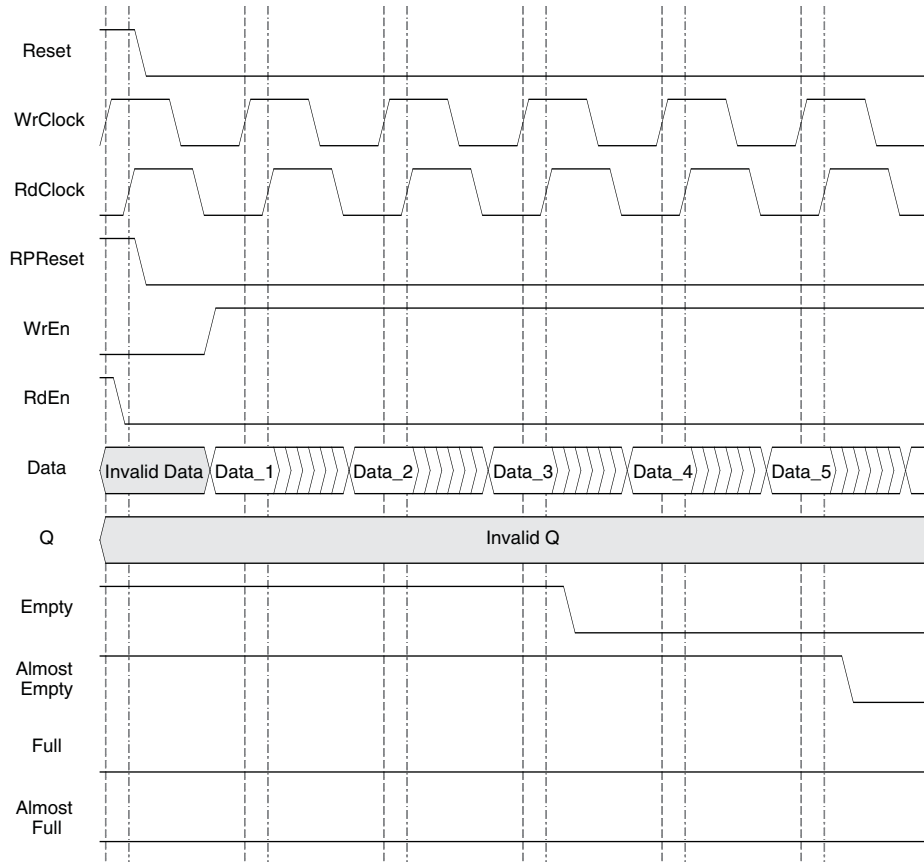
In the emulated FIFO\_DC, there is no latency during assertion of the flags. Thus, when the flag goes true, there is no latency. However, due to the design of the flag logic running on two clock domains, there is latency during de-assertion.

Let us assume that we start to write into the FIFO\_DC to fill it. The write operation is controlled by WrClock and WrEn. However, it takes extra RdClock cycles for the de-assertion of the Empty and Almost Empty flags.

De-assertion of Full and Almost Full although result of reading out the data from the FIFO\_DC, takes extra WrClock cycles after reading the data for these flags to come out.

With this in mind, let us look at the waveforms for FIFO\_DC without output registers. Figure 11-33 shows the operation of the FIFO\_DC when it is empty and the data begins to be written into it.

**Figure 11-33. FIFO\_DC Without Output Registers, Start of Data Write Cycle**

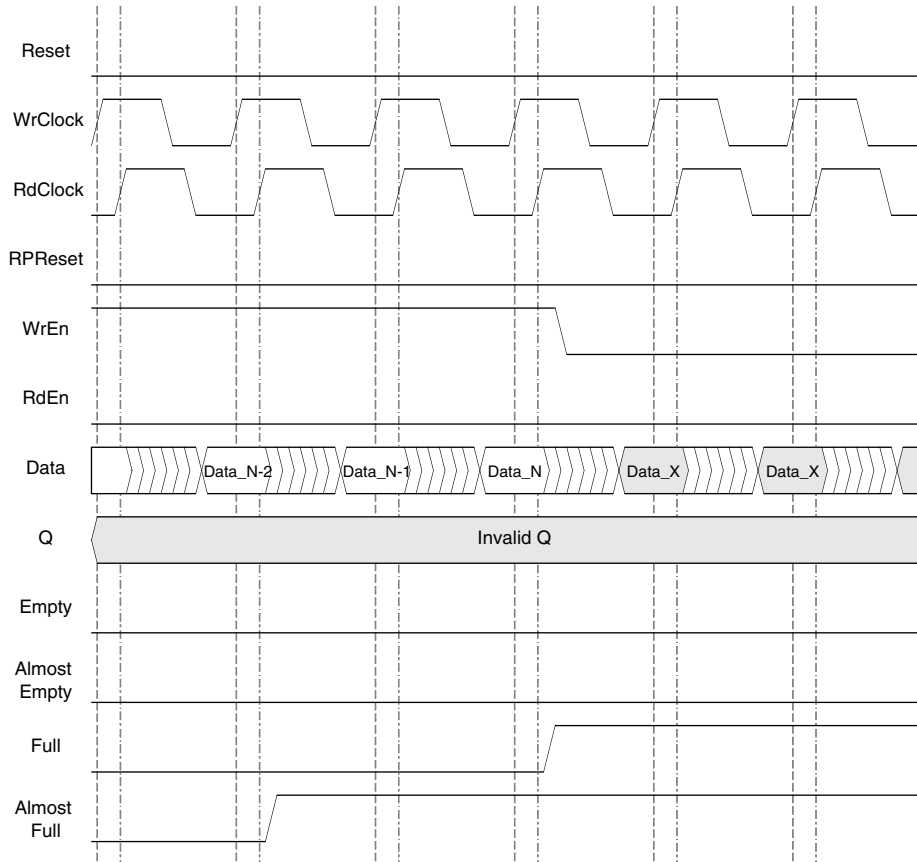


The WrEn signal has to be high to start writing into the FIFO\_DC. The Empty and Almost Empty flags are high to begin and Full and Almost Full are low.

When the first data is written into the FIFO\_DC, the Empty flag de-asserts (or goes low), as the FIFO\_DC is no longer empty. In this figure we assume that the Almost Empty setting flag setting is 3 (address location 3). The Almost Empty flag is de-asserted when the third address location is filled.

Now let us assume that we continue to write into the FIFO\_DC to fill it. When the FIFO\_DC is filled, the Almost Full and Full Flags are asserted. Figure 11-34 shows the behavior of these flags. In this figure we assume that FIFO\_DC depth is 'N'.

**Figure 11-34. FIFO\_DC Without Output Registers, End of Data Write Cycle**



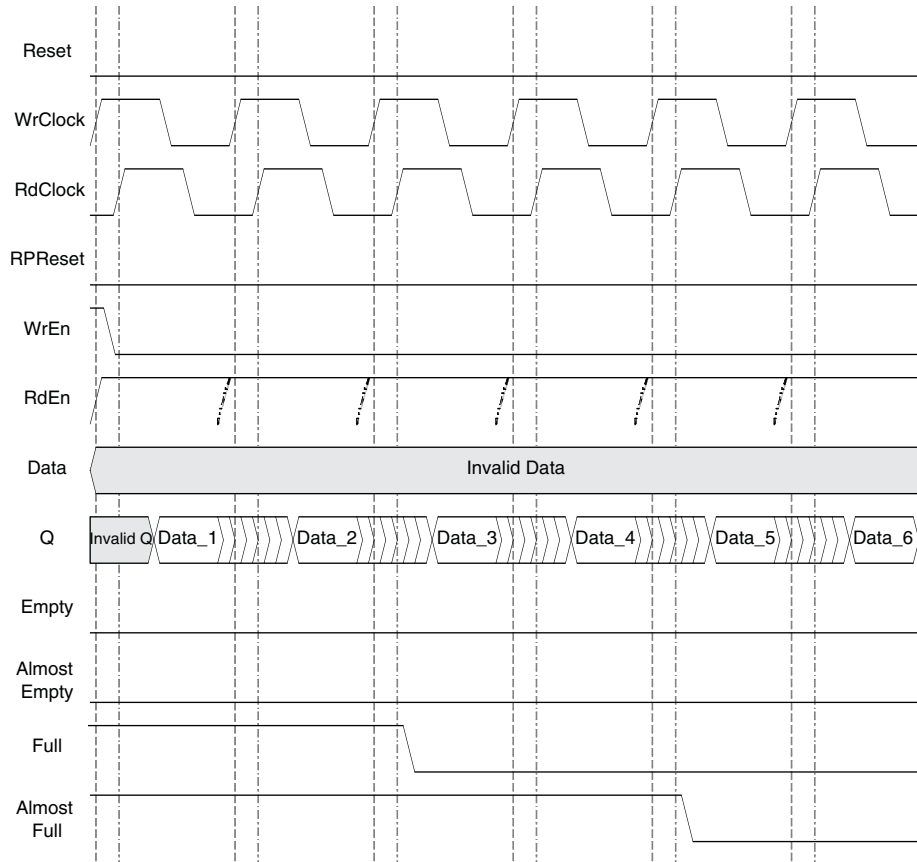
In Figure 11-34, the Almost Full flag is two locations before the FIFO\_DC is filled. The Almost Full flag is asserted when the N-2 location is written, and the Full flag is asserted when the last word is written into the FIFO\_DC.

Data\_X data inputs are not written since the FIFO\_DC is full (Full flag is high).

Note that the assertion of these flags is immediate and there is no latency when they go true.

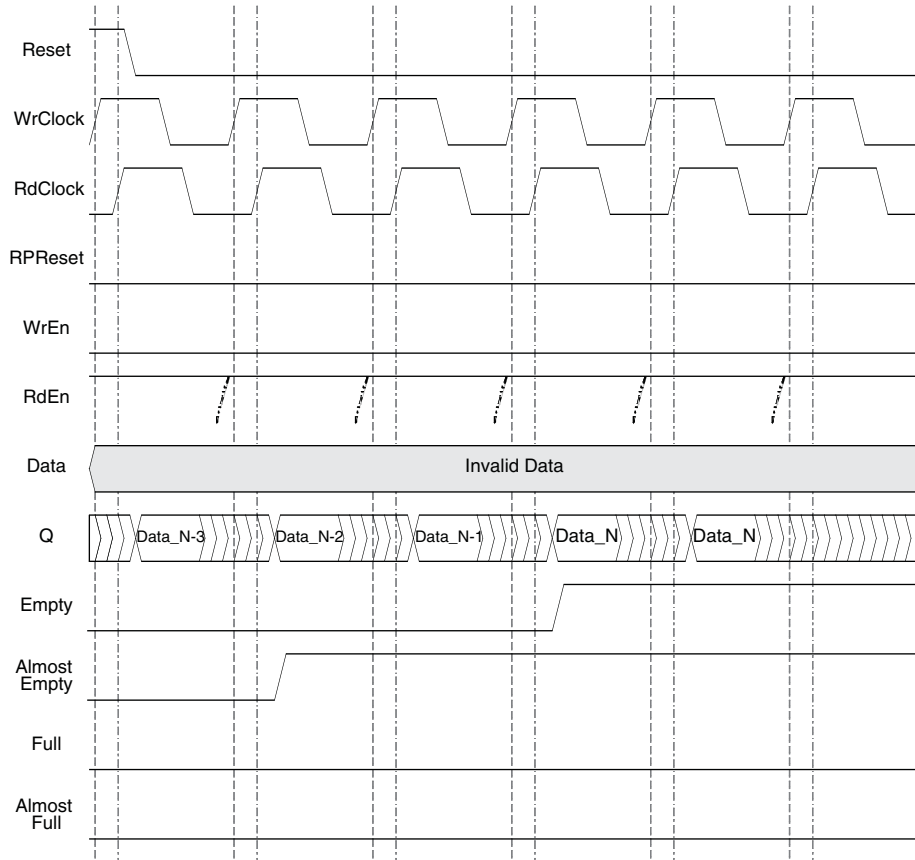
Now let us look at the waveforms when the contents of the FIFO\_DC are read out. Figure 11-35 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags are de-asserted as shown. Note that the de-assertion is delayed by two clock cycles.

Figure 11-35. FIFO\_DC Without Output Registers, Start of Data Read Cycle



Similarly, as the data is read out and FIFO\_DC is emptied, the Almost Empty and Empty flags are asserted (see Figure 11-36).

Figure 11-36. FIFO\_DC Without Output Registers, End of Data Read Cycle



Figures 11-33 to 11-36 show the behavior of the non-pipelined FIFO\_DC or FIFO\_DC without output registers. When we pipeline the registers, the output data is delayed by one clock cycle. There is an extra option for output registers to be enabled by the RdEn signal.

Figures 11-37 to 11-40 show similar waveforms for the FIFO\_DC with output register and without output register enable with RdEn. Note that flags are asserted and de-asserted with timing similar to the FIFO\_DC without output registers. However, only the data out 'Q' is delayed by one clock cycle.

Figure 11-37. FIFO\_DC with Output Registers, Start of Data Write Cycle

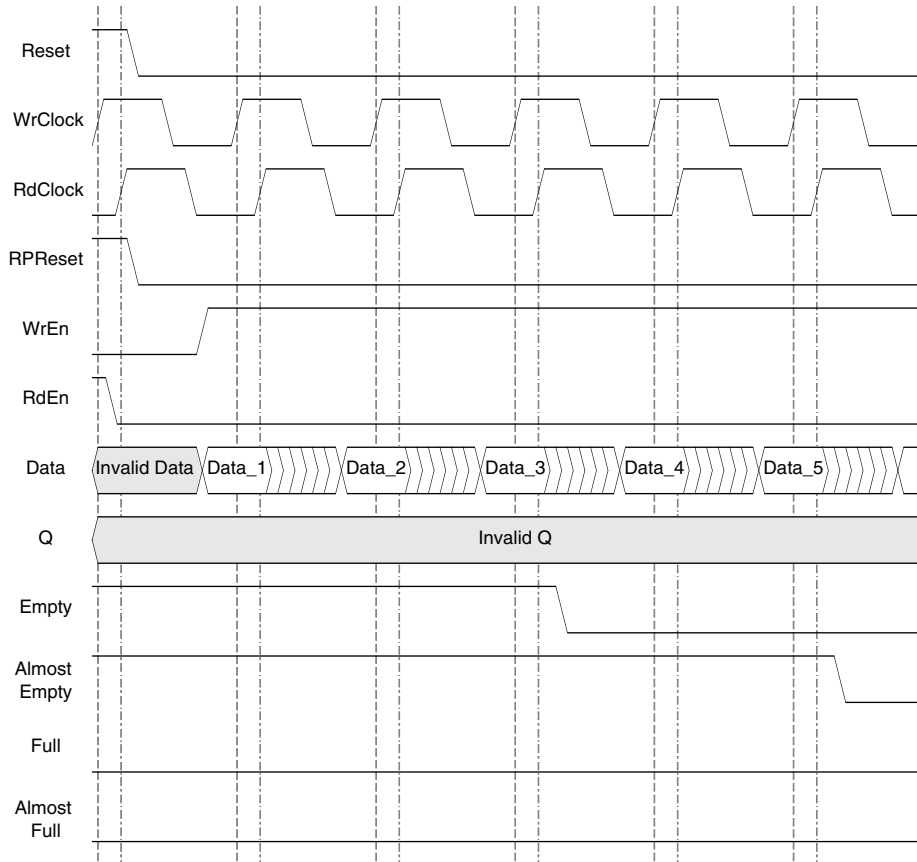




Figure 11-38. FIFO\_DC with Output Registers, End of Data Write Cycle

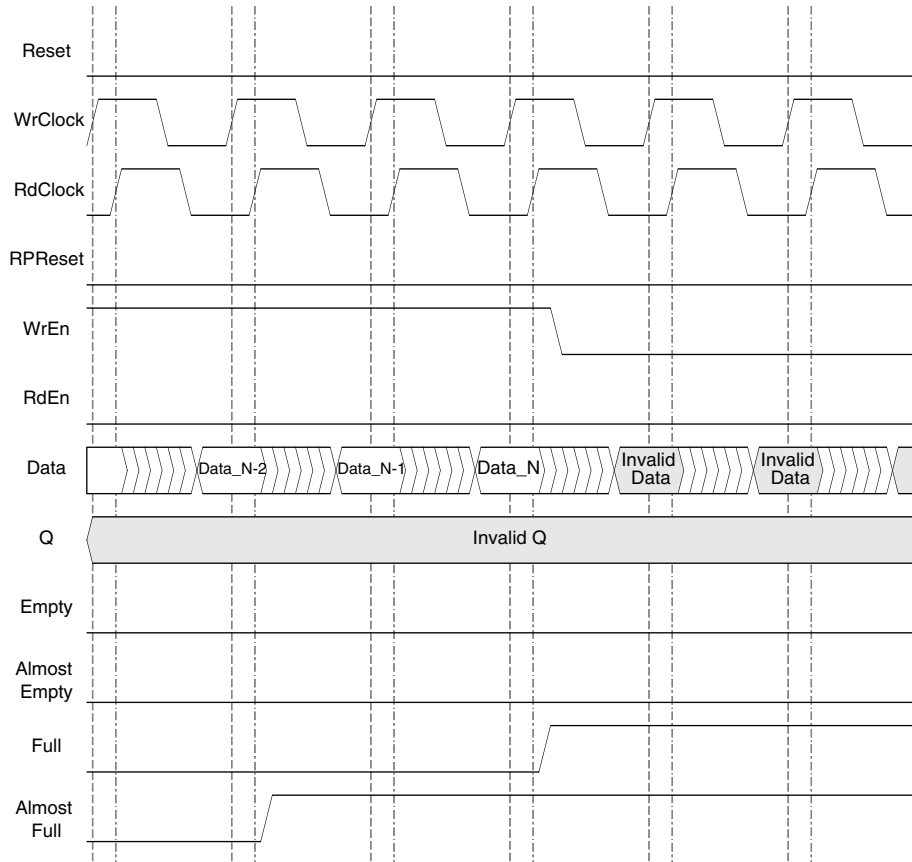


Figure 11-39. FIFO\_DC with Output Registers, Start of Data Read Cycle

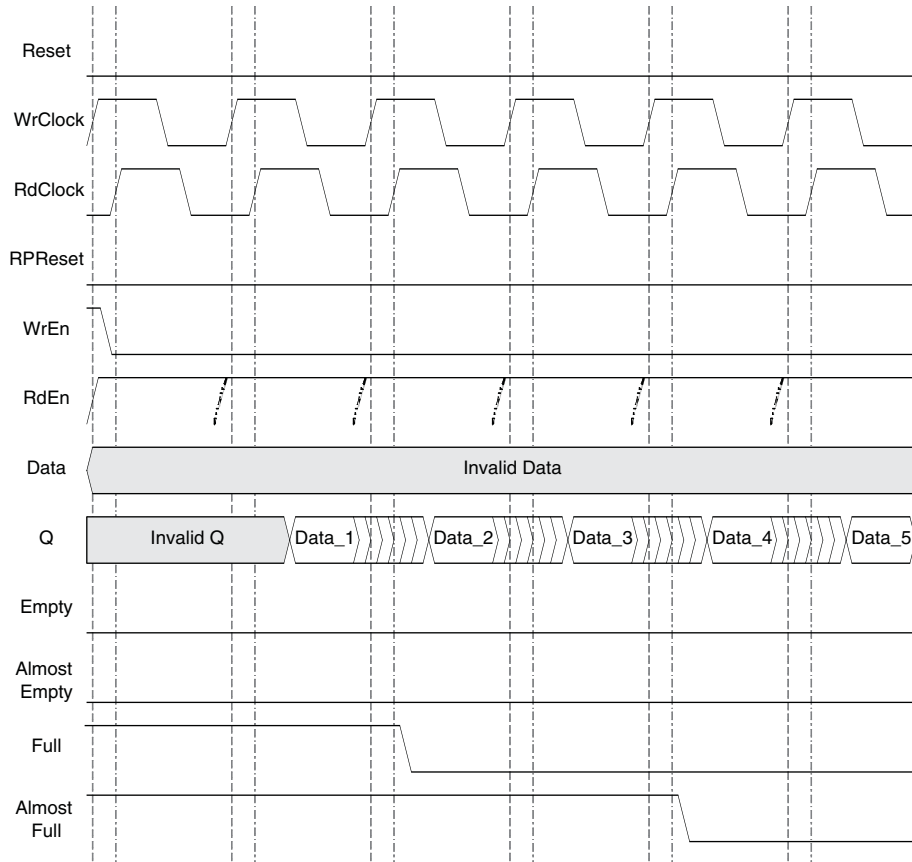
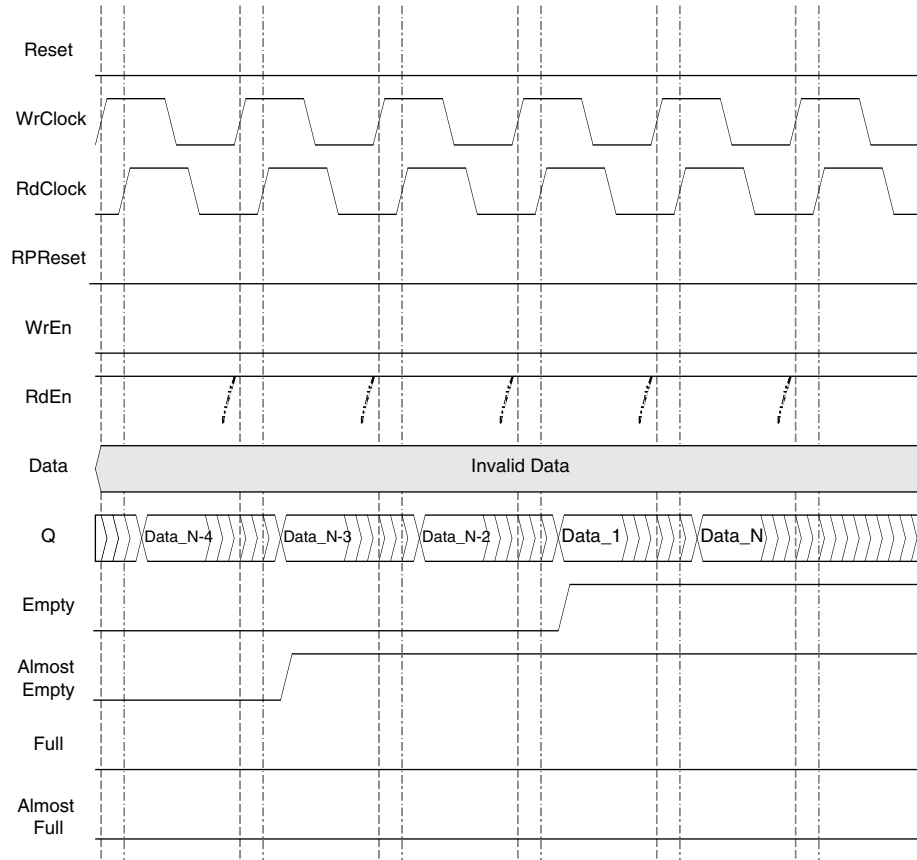
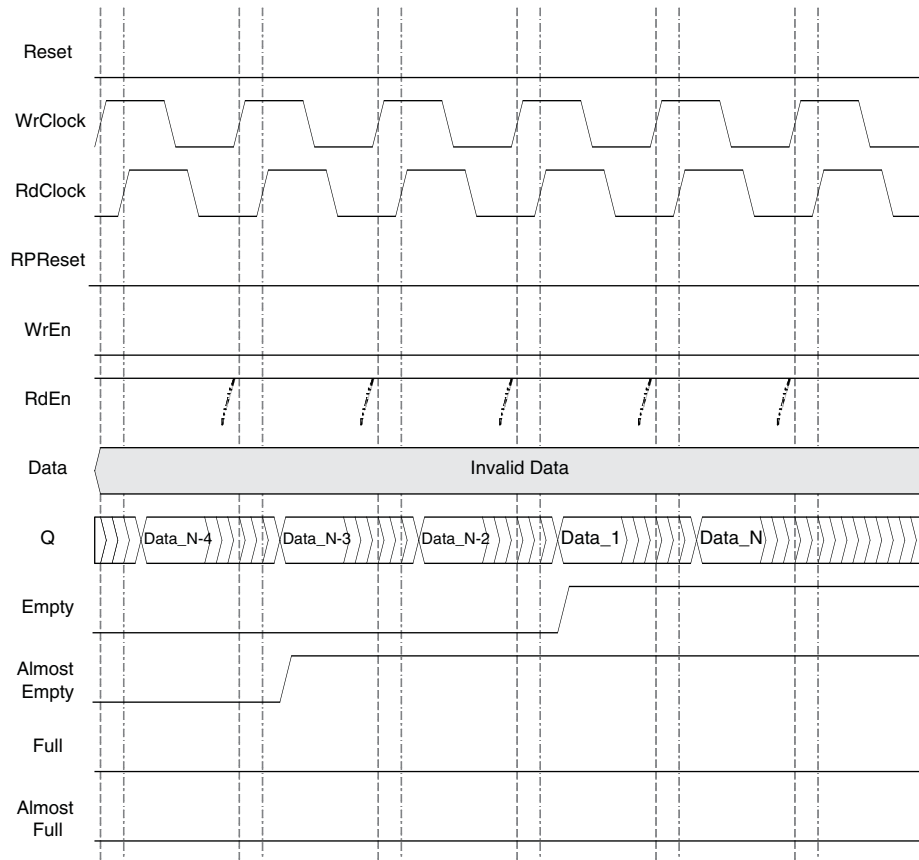


Figure 11-40. FIFO\_DC with Output Registers, End of Data Read Cycle



If the designer selects the option enable output register with RdEn, it still delays the data out by one clock cycle (as compared to the non-pipelined FIFO\_DC). RdEn should also be high during that clock cycle, otherwise the data takes an extra clock cycle when the RdEn is goes true.

Figure 11-41. FIFO\_DC with Output Registers and RdEn on Output Registers

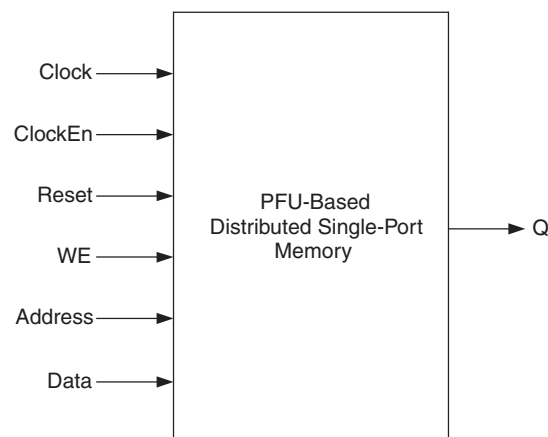


## Distributed Single-Port RAM (Distributed\_SPRAM) – PFU-Based

PFU-based Distributed Single-Port RAM is created using the 4-input LUTs available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 11-42 shows the Distributed Single-Port RAM module as generated by IPexpress.

Figure 11-42. Distributed Single-Port RAM Module Generated by IPexpress



The generated module makes use of the 4-input LUTs available in the PFU. Additional logic such as a clock or reset is generated by utilizing the resources available in the PFU.

Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants to enable the output registers in the IPexpress configuration.

The various ports and their definitions listed in Table 11-13. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

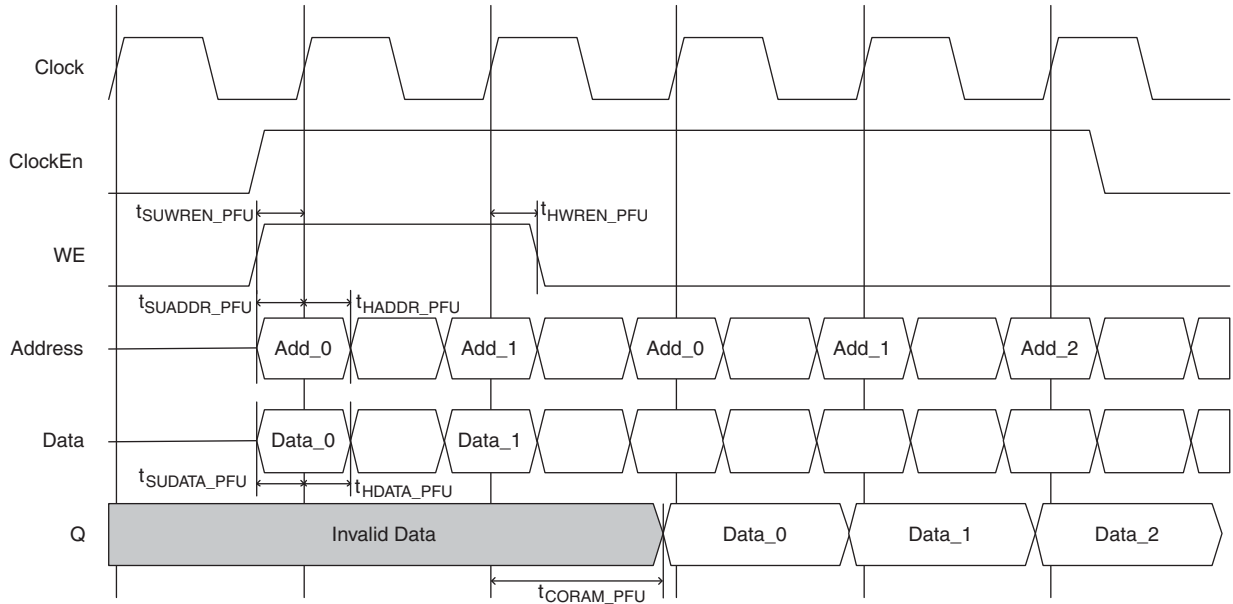
**Table 11-13. PFU-Based Distributed Single Port RAM Port Definitions**

Port Name in Generated Module	Port Name in the EBR block Primitive	Description
Clock	CK	Clock
ClockEn	—	Clock Enable
Reset	—	Reset
WE	WRE	Write Enable
Address	AD[3:0]	Address
Data	DI[1:0]	Data In
Q	DO[1:0]	Data Out

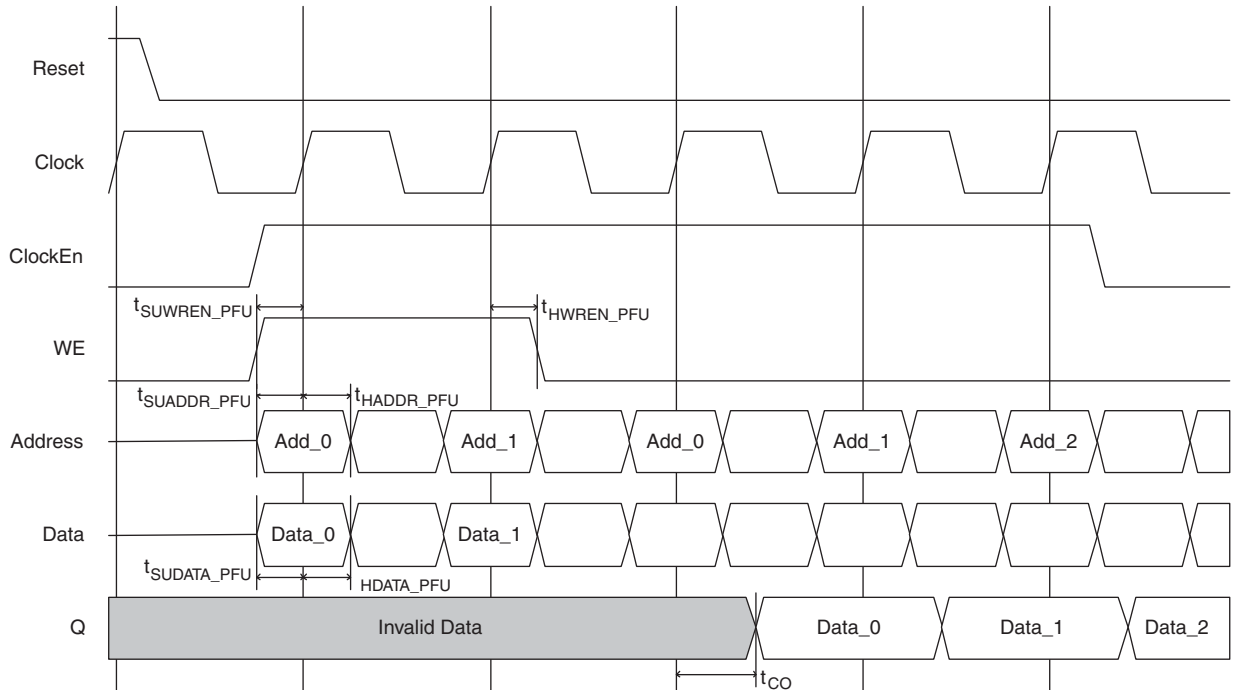
Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn), are not available in the hardware primitive. These are generated by IPexpress when the user wants to enable the output registers in their IPexpress configuration.

The various ports and their definitions for the memory are included in Table 11-11. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

**Figure 11-43. PFU Based Distributed Single Port RAM Timing Waveform - without Output Registers**



**Figure 11-44. PFU Based Distributed Single Port RAM Timing Waveform - with Output Registers**

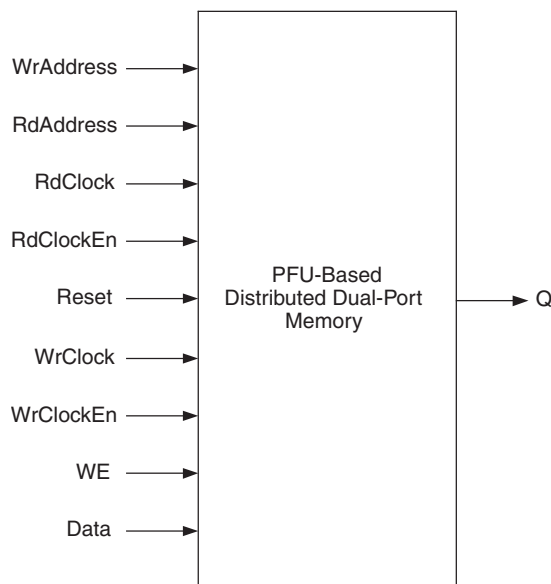


### Distributed Dual-Port RAM (Distributed DPRAM) – PFU-Based

PFU-based Distributed Dual-Port RAM is also created using the 4-input LUTs available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 11-45 shows the Distributed Single-Port RAM module as generated by IPexpress.

**Figure 11-45. Distributed Dual-Port RAM Module Generated by IPexpress**



The generated module makes use of the 4-input LUTs available in the PFU. Additional logic such as a clock or reset is generated by utilizing the resources available in the PFU.

Ports such as the Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants the to enable the output registers in the IPexpress configuration.

The various ports and their definitions are listed in Table 11-14. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

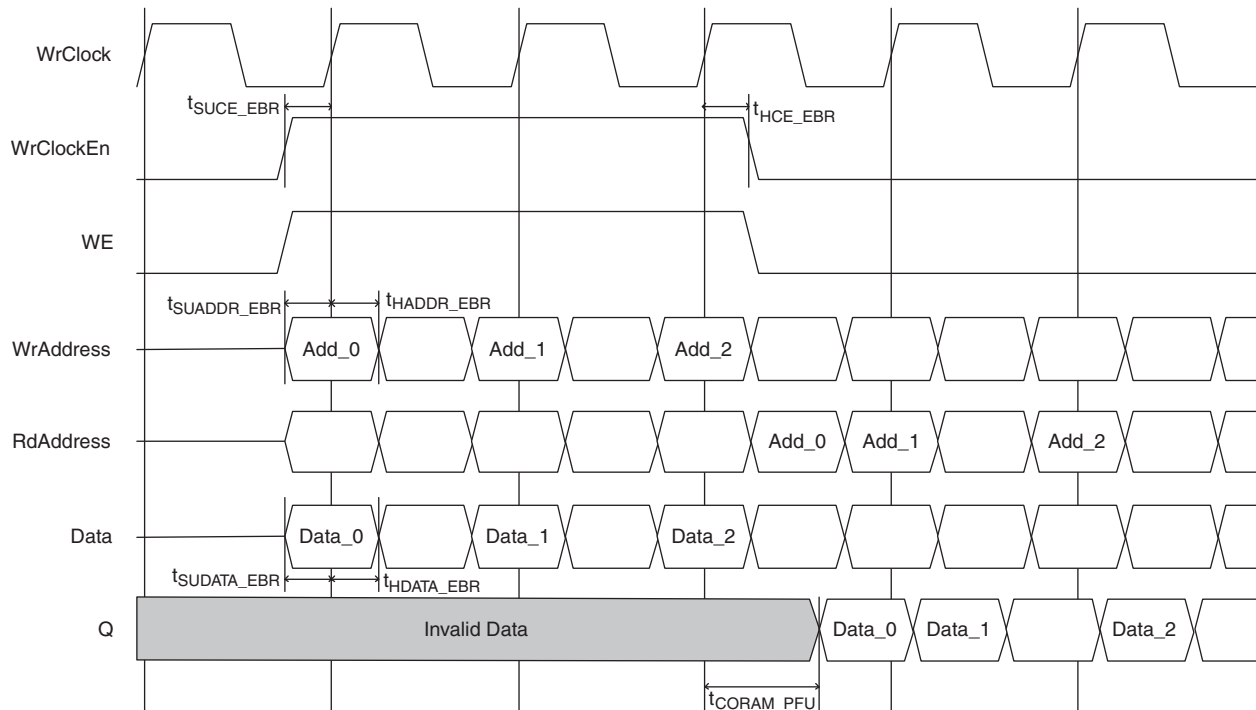
**Table 11-14. PFU-Based Distributed Dual-Port RAM Port Definitions**

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
WrAddress	WAD[23:0]	Write Address
RdAddress	RAD[3:0]	Read Address
RdClock	—	Read Clock
RdClockEn	—	Read Clock Enable
WrClock	WCK	Write Clock
WrClockEn	—	Write Clock Enable
WE	WRE	Write Enable
Data	DI[1:0]	Data Input
Q	RDO[1:0]	Data Out

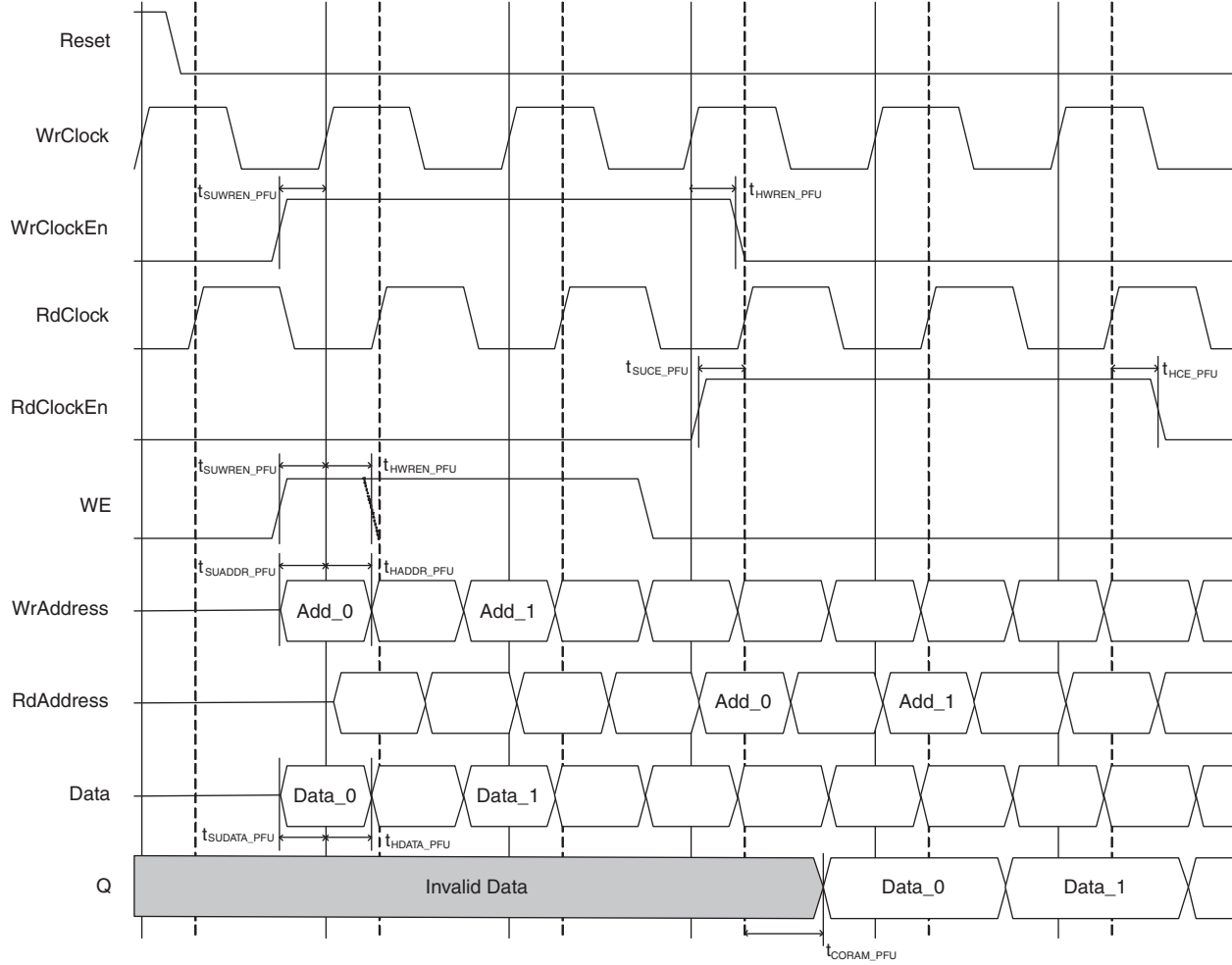
Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants the to enable the output registers in the IPexpress configuration.

Users have the option of enabling the output registers for Distributed Dual Port RAM (Distributed\_DPRAM). Figures 11-46 and 11-47 show the internal timing waveforms for the Distributed Dual Port RAM (Distributed\_DPRAM) with these options.

**Figure 11-46. PFU Based Distributed Dual Port RAM Timing Waveform - without Output Registers**



**Figure 11-47. PFU Based Distributed Dual Port RAM Timing Waveform - with Output Registers**

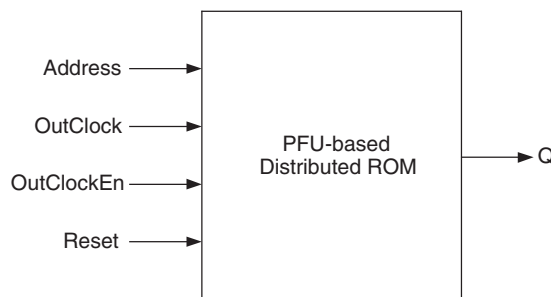


### Distributed ROM (Distributed\_ROM) – PFU-Based

PFU-based Distributed ROM is also created using the 4-input LUTs available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 11-48 shows the Distributed ROM module generated by IPexpress.

**Figure 11-48. Distributed ROM Generated by IPexpress**



The generated module makes use of the 4-input LUTs available in the PFU. Additional logic such as a clock or reset is generated by utilizing the resources available in the PFU.



Ports such as Out Clock (OutClock) and Out Clock Enable (OutClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants to enable the output registers in the IPexpress configuration.

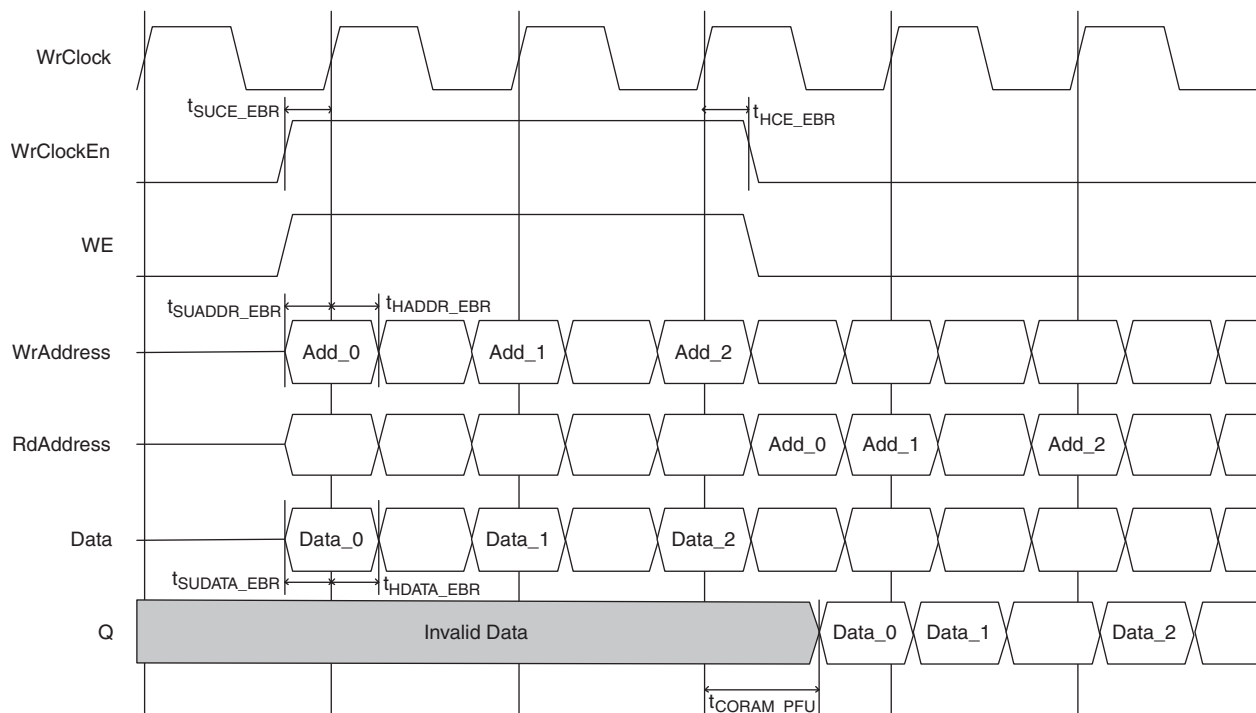
The various ports and their definitions are listed in Table 11-15. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

**Table 11-15. PFU-Based Distributed ROM Port Definitions**

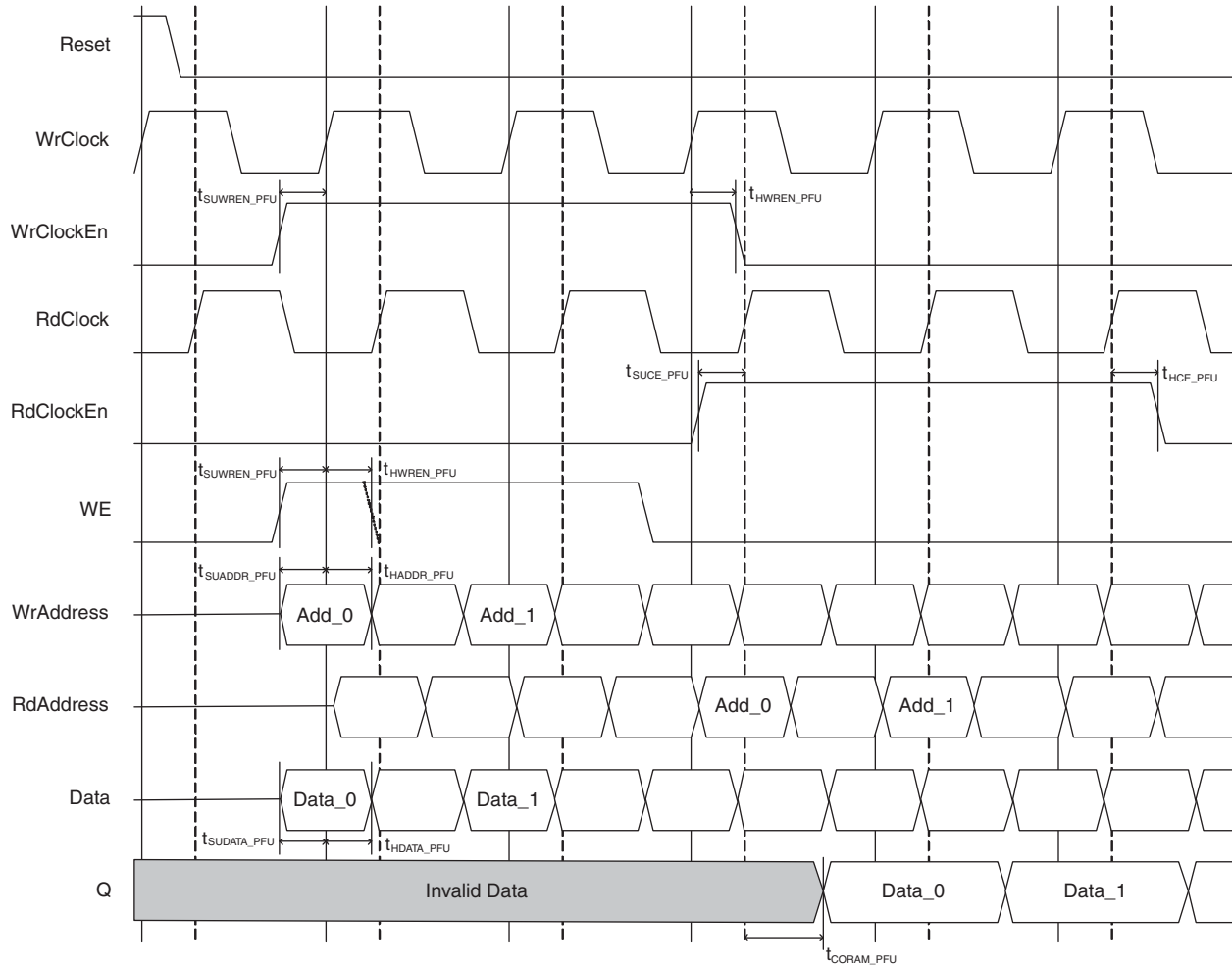
Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
Address	AD[3:0]	Address
OutClock	—	Out Clock
OutClockEn	—	Out Clock Enable
Reset	—	Reset
Q	DO	Data Out

Users have the option to enable the output registers for Distributed ROM (Distributed\_ROM). Figures 11-49 and 11-50 show the internal timing waveforms for the Distributed ROM with these options.

**Figure 11-49. PFU Based Distributed Dual Port RAM Timing Waveform - without Output Registers**



**Figure 11-50. PFU Based Distributed Dual Port RAM Timing Waveform - with Output Registers**



## Initializing Memory

In each memory mode, it is possible to specify the power-on state of each bit in the memory array. This allows the memory to be used as ROM if desired. Each bit in the memory array can have a value of 0 or 1.

### Initialization File Formats

The initialization file is an ASCII file, which the designer can create or edit using any ASCII editor. IPexpress supports three memory file formats:

1. Binary file
2. Hex File
3. Addressed Hex

The file name for the memory initialization file is \*.mem (<file\_name>.mem). Each row includes the value to be stored in a particular memory location. The number of characters (or the number of columns) represents the number of bits for each address (or the width of the memory module).

The initialization file is primarily used for configuring the ROMs. RAMs can also use the initialization file to preload memory contents.

---

**Binary File**

The binary file is a text file of 0s and 1s. The rows indicate the number of words and the columns indicate the width of the memory.

Memory Size 20x32

```
00100000010000000010000001000000
00000001000000010000000100000001
00000010000000100000001000000010
00000011000000110000001100000011
00000100000001000000010000000100
00000101000001010000010100000101
00000110000001100000011000000110
00000111000001110000011100000111
00001000010010000000100001001000
000010010100100100000100101001001
00001010010010100000101001001010
00001011010010110000101101001011
00001100000011000000110000001100
00001101001011010000110100101101
00001110001111100000111000111110
00001111001111110000111100111111
00010000000100000001000000010000
00010001000100010001000100010001
00010010000100100001001000010010
00010011000100110001001100010011
```

**Hex File**

The hex file is a text file of hexadecimal characters arranged in a similar row-column arrangement. The number of rows in the file is the same as the number of address locations, with each row indicating the content of the memory location.

Memory Size 8x16

```
A001
0B03
1004
CE06
0007
040A
0017
02A4
```

**Addressed Hex**

Addressed hex consists of lines of addresses and data. Each line starts with an address, followed by a colon, and any number of data. The format of the file is “address: data data data data” where the address and data are hexadecimal numbers.

```
A0 : 03 F3 3E 4F
B2 : 3B 9F
```

In the example above, the first line shows 03 at address A0, F3 at address A1, 3E at address A2, and 4F at address A3. The second line shows 3B at address B2 and 9F at address B3.

There is no limitation on the address and data values. The value range is automatically checked based on the values of `addr_width` and `data_width`. If there is an error in an address or data value, an error message is printed. It is

not necessary to specify data at all address locations. If data is not specified at a certain address, the data at that location is initialized to 0. SCUBA makes memory initialization possible both through the synthesis and simulation flows.

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
May 2009	01.1	References to 9K memories corrected to read 18K throughout the document.
June 2009	01.2	Number of bits of RAM contained in each EBR block corrected to read 18,432.
June 2009	01.3	Text and screenshots were updated to match the latest ispLEVER software.
		Added updated timing waveforms for EBR and Distributed RAMs.
		Removed RAM-Based Shift Register.
November 2009	01.4	Updated Reset Mode based on device support.
January 2010	01.5	Added Read Before Write mode for Single Port and True Dual Port RAM modules for LatticeECP3-xxEA devices.
July 2011	01.6	Added the setup and hold requirements for addresses to EBR-based memories.
February 2012	01.7	Updated document with new corporate logo.
July 2013	01.8	Updated Technical Support Assistance information.

---

## Appendix A. Attribute Definitions

### DATA\_WIDTH

Data width is associated with the RAM and FIFO elements. The DATA\_WIDTH attribute defines the number of bits in each word. It uses the values defined in the RAM size tables in each memory module.

### REGMODE

REGMODE, or the Register mode attribute, is used to enable pipelining in the memory. This attribute is associated with the RAM and FIFO elements. The REGMODE attribute takes the NOREG or OUTREG mode parameter that disables and enables the output pipeline registers.

### CSDECODE

CSDECODE or the Chip Select Decode attributes are associated with block RAM elements. CS, or Chip Select, is the port available in the EBR primitive that is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade eight memories easily.

CSDECODE takes the following parameters: "000", "001", "010", "011", "100", "101", "110", and "111". CSDECODE values determine the decoding value of CS[2:0]. CSDECODE\_W is chip select decode for write and CSDECODE\_R is chip select decode for read for Pseudo Dual Port RAM. CSDECODE\_A and CSDECODE\_B are used for True Dual-Port RAM elements and refer to the A and B ports.

### WRITEMODE

The WRITEMODE attribute is associated with the block RAM elements. It takes the NORMAL, WRITETHROUGH, and READBEFOREWRITE mode parameters.

In NORMAL mode, the output data does not change or get updated, during the write operation. This mode is supported for all data widths.

In WRITETHROUGH mode, the output data is updated with the input data during the write cycle. This mode is supported for all data widths.

WRITEMODE\_A and WRITEMODE\_B are used for Dual-Port RAM elements and refer to the A and B ports in True Dual-Port RAM.

In READBEFOREWRITE mode, the output data port is updated with the existing data stored in the write address, during a write cycle. This mode is supported for x9, x18 and x36 data widths.

For all modes of the True Dual-Port module, simultaneous read access from one port and write access from the other port to the same memory address is not recommended. The read data may be unknown in this situation.

Also, simultaneous write access to the same address from both ports is not recommended. When this occurs, the data stored in the address becomes undetermined when one port tries to write a 'H' and the other tries to write a 'L'. It is recommended that control logic be implemented to identify this situation if it occurs and do one of the following:

1. Implement status signals to flag the read data as possibly invalid, or
2. Implement control logic to prevent simultaneous access from both ports.

## Introduction

LatticeECP3™ devices support high-speed I/O interfaces, including Double Data Rate (DDR) and Single Data Rate (SDR) interfaces, using the logic built into the Programmable I/O (PIO). SDR applications capture data on one edge of a clock while DDR interfaces capture data on both the rising and falling edges of the clock, thus doubling the performance. LatticeECP3 I/Os also have dedicated circuitry that is used along with the DDR I/O to support DDR, DDR2 and DDR3 SDRAM memory interfaces. Refer to the [LatticeECP3 Family Data Sheet](#) for a detailed description of the I/O logic architecture.

This document discusses how to utilize the capabilities of the LatticeECP3 “E” and “EA” devices to implement both the high-speed generic DDR interface and the DDR and DDR2 memory interfaces. Refer to the [Implementing DDR/DDR2/DDR3 Memory Interfaces](#) section of this document for more information.

There are some differences in high-speed interface architecture between the LatticeECP3 “EA” and “E” devices. The implementation differences between the two devices are indicated in the appropriate sections of this document.

This document assumes that version 8.0 of the ispLEVER® software is used for all interfaces. Please see exceptions under each section if you are using the ispLEVER 7.2 SP2.

### Steps to Design a High-Speed DDR Interface:

The following steps must be followed to successfully design a high-speed DDR interface using this document.

- **Step 1: Determine the type of interface to implement.**  
Based on the external interface determine the type of high-speed interface to be built. See the [Types of High-Speed DDR Interfaces](#) section for details.
- **Step 2: Use the ispLEVER IPexpress™ tool to build the interface.**  
Once you have determined the type of interface to be built, use IPexpress to build the interface. See the section [Using IPexpress to Build High-Speed DDR Interfaces](#).
- **Step 3: Understand design rules, clocking requirements for each interface.**  
Understand the architecture, interface rules and clocking requirements for each of the interfaces. See the [High-Speed DDR Interface Details](#) section for a detailed description of each interface. If multiple interfaces are used in one device, it is critical to follow these design rules to avoid resource conflicts between interfaces.
- **Step 4: Review pinout requirements for clock and data pins for each interface before making pin assignments.** It is critical that the interface clock and data pins follow the placement recommendations listed in the section [Placement Guidelines for High-Speed DDR Interfaces](#). If using interfaces requiring DQ-DQS grouping, follow the rules described in the section [DQ-DQS Grouping Rules](#).
- **Step 5: Assign timing preferences and clock preferences.**  
Follow the guidelines in the section [Timing Analysis for High-Speed DDR Interfaces](#) to assign timing preferences for the interfaces.
- **Step 6: Run software Place and Route without DRC and Trace.**  
It is important that the software Place and Route tool pass without any DRC errors. If it runs into DRC errors review the sections described above to make sure you are not violating any of the design/pinout rules for the interface. Run Trace to look at the static timing analysis results for all the timing preferences added to the design. Make design changes as necessary to assure that you are meeting your timing requirements.

---

### Steps to Generate a Valid Pinout for a High-Speed DDR Interface:

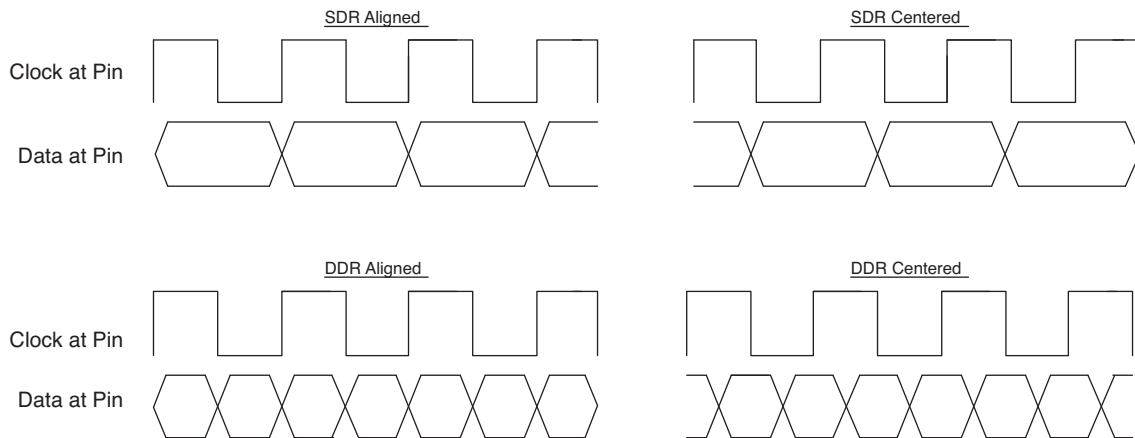
Due to the various design rules and pinout requirements for each interface, it is critical that the interfaces be created as described above and run through the software before designing the PCB. If it is necessary to determine pinouts for a PCB design before the complete design is in place, the user must follow the steps listed below to create the pinouts.

- **Step 1: Determine the type of interface to implement.**  
Based on the external interface, determine the type of high-speed interface to be built. See the section [Types of High-Speed DDR Interfaces](#) for details.
- **Step 2: Use the ispLEVER IPexpress Tool to build the interface.**  
Once you have determined the type of interface to be built, IPexpress should be used to build the interface. See the section [Using IPexpress to Build High-Speed DDR Interfaces](#).
- **Step 3: Build the complete I/O ring and clocking structure.**  
Some dummy logic may be required to assure that the data out of the DDR elements are not optimized out by the software. For example, if you are building an input and a corresponding output interface you may connect them using dummy register between the two interfaces. If you are building receive-only interfaces, the signals out of the receive interface will need to be used in dummy logic or assigned to outputs. Similarly, if building a transmit-only interface you must provide input signals that are used in the transmit interface.
- **Step 3: Understand the design rules and clocking requirements for each interface.**  
Understand the architecture, interface rules and clocking requirements for each of the interfaces. See the [High-Speed DDR Interface Details](#) section for a detailed description of each interface. If multiple interfaces are used in one device, it is critical to follow these design rules to avoid resource conflicts between interfaces.
- **Step 4: Make pin assignments for clock and data pins for each interface following the design and pinout rules.** Assign the input and output pins to specific sites, banks or DQS groups. Review the pinout requirements for the clock and data pins for each interface before making pin assignments. It is critical that the interface clock and data pins follow the placement recommendations listed in the section [Placement Guidelines for High-Speed DDR Interfaces](#). If using interfaces requiring a DQ-DQS grouping, you must follow the rules described in the section [DQ-DQS Grouping Rules](#).
- **Step 5: Run the design through ispLEVER Place and Route and pass without any DRC errors.**  
If you run into errors, change pin assignments as required to pass all the DRC checks. When the design passes Place and Route, the pin assignments listed in the PAD report can be used on the board.

### External Interface Description

This technical note uses two types of external interface definitions, centered and aligned. In a centered external interface, at the device pins, the clock is centered in the data opening. In an aligned external interface, at the device pins, the clock and data transition are aligned. This is also sometimes called “edge-on-edge”. Figure 12-1 shows the external interface waveform for SDR and DDR.

**Figure 12-1. External Interface Definition**



The interfaces described are referenced as centered or aligned interfaces. An aligned interface is needed to adjust the clock location to satisfy the capture flip-flop setup and hold times. A centered interface is needed to balance the clock and data delay to the first flip-flop to maintain the setup and hold already provided.

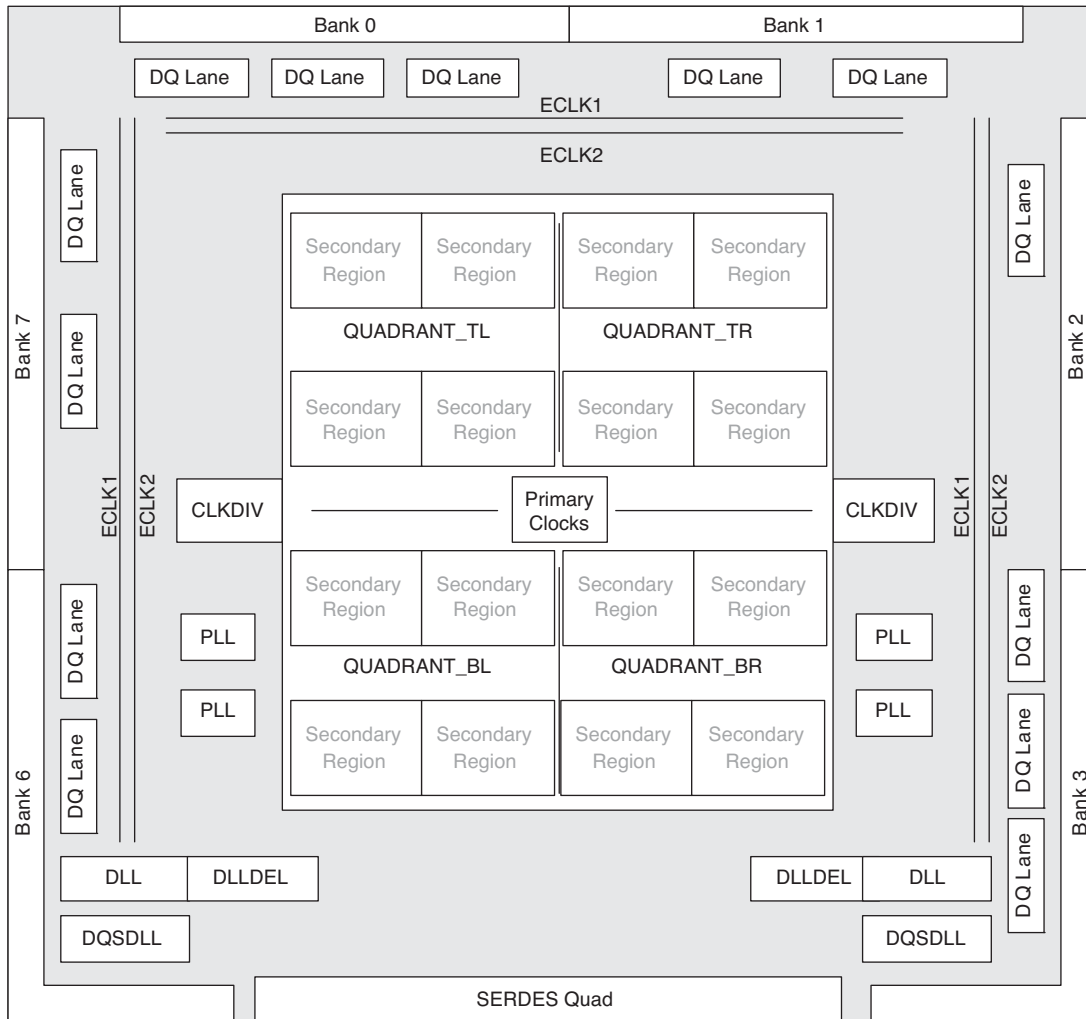
## High-Speed I/O Interface Building Blocks

The LatticeECP3 device contains dedicated functions for building high-speed interfaces. This section describes when and how to use these functions. A complete description of the library elements including descriptions and attributes is provided at the end of this document.

Figure 12-2 shows a high-level diagram of the clocking resources available in the “E” and “EA” devices for building high-speed I/O interfaces.



**Figure 12-2. LatticeECP3 Clocking Diagram (LatticeECP3-35 Shown)**



A complete description and of the LatticeECP3 clocking resources and clocking routing restrictions is available in TN1178, [LatticeECP3 sysCLOCK™ PLL/DLL Design and Usage Guide](#).

Below is a brief description for each of the major elements used for building various high-speed interfaces. The [DDR Software Primitives and Attributes](#) section describes the library elements for these components.

**ECLK**

Edge clocks are high-speed, low-skew I/O dedicated clocks. They are arranged in groups of two on the left, right, and top sides of the device.

**SCLK**

SCLK refers to the system clock of the design. SCLK can use either primary or secondary clocks.

**DQS Lane**

A DQS Lane borrows its name from memory interfaces, but can be used for general-purpose high-speed interfaces. Each DQS Lane provides a clock (DQS) and up to 10 data bits on that clock. The number of DQS Lanes on the device is different for each device size. LatticeECP3 devices support DQS signals on the top, left and right sides of the device.

## **PLL**

The PLL provides frequency synthesis as well as static and dynamic phase adjustment. Three output ports are provided: CLKOP, CLKOS and CLKOK. CLKOK provides a dedicated divide-by-2 function for use with the I/O logic gearing. The number of PLLs available on the device varies by device size from 2 to 10 PLLs per device.

## **DLL**

The general-purpose DLL provides clock injection delay removal as well as 90° delay compensation when used with the DLLDEL element. There are two DLLs, one on each side of all LatticeECP3 devices.

## **DQSDLL**

The DQSDLL is a dedicated DLL for creating a 90° clock delay. There are two DQSDLLs provided on all LatticeECP3 devices. There is one DQSDLL on each side of the device.

## **Input DDR (IDDR)**

The input DDR function can be used in either x1 or x2 gearing modes. The x1 mode is supported by the IDDRXD element. This library element inputs a single DDR data input and clock (DQS, ECLK or SCLK) and provides a 2-bit wide data synchronized to the SCLK (system clock) to the FPGA fabric.

In the x2 mode, the IDDRX2D element is used. This element is useful for interfaces with greater than a 200MHz clock. It supports a 4-bit wide interface to the FPGA fabric. The clock input to the IDDRX2D is the high-speed ECLK. The SCLK clock is divided down to half of the ECLK input clock.

## **Output DDR (ODDR)**

The output DDR function can also be supported in either x1 or x2 gearing modes. The output x1 DDR function is supported by the ODDRXD element. This library element provides a single DDR output and clock (SCLK) and accepts 2-bit wide data from the FPGA fabric.

The ODDR2XD element is used in the x2 mode and supports a 4-bit wide interface to the FPGA fabric. This element is useful for interfaces with greater than a 200MHz clock. The SCLK clock is divided down to half of the ECLK.

## **CLKDIV**

The CLKDIV element is used to divide the high-speed ECLK by 2 to generate the SCLK when using x2 input or output gearing modes.

## **DELAY**

There are three types of input data available. DELAYC provides a fixed value of delay to compensate for clock injection delay. DELAYC is used by default when configuring the interface in the software. Software will configure the DELAYC with delay values based on the interface used. DELAYB provides dynamic or user-defined delay. DELAYC is also used for SDR interfaces where it provides clock injection delay.

## **ECLK/SCLK vs. DQS Lanes**

ECLK and SCLK span the entire side of the device and is useful for creating wide input bus interfaces. The DQS Lanes cover only 10 bits of data width and work well for narrow input bus interfaces. Each DQS Lane is serviced by a DQSBUF to control clock access and delay. The DQS Lane is supported by the DQSDLL for 90° clock delay. There is only one DQSDLL per side of the device, but this DQSDLL can be used for all of the DQS Lanes on the side. If several narrow input bus interfaces are required it is best to use the DQS Lanes instead of the ECLK or SCLK.

Due to architectural difference between “E” and “EA” devices, wider buses are supported using ECLK in “E” devices and SCLK in “EA” devices in x1 mode.

## Building Generic High-Speed Interfaces

This section explains in detail how to build high-speed interfaces using the building blocks described above. The ispLEVER IPexpress tool builds these interfaces based on external interface requirements.

### Types of High-Speed DDR Interfaces

This section describes the different types of high-speed DDR interfaces available in the LatticeECP3 device. Table 12-1 lists these interfaces. A description of each interface in the table is provided below the table.

**Table 12-1. Generic High-Speed I/O DDR Interfaces**

Mode	Interface Name	Description	LatticeECP3 Device Support
RX SDR	GIREG_RX.SCLK	SDR Input register using SCLK	E, EA
RX DDRX1 Aligned	GDDR1_RX.SCLK.Aligned/ GDDR1_RX.SCLK.PLL.Aligned	DDR x1 Input using SCLK. Data is edge-to-edge with incoming clock.	EA
RX DDRX1 Aligned	GDDR1_RX.DQS.Aligned	DDR x1 Input using DQS. Data is edge-to-edge with incoming clock.	E, EA
RX DDRX1 Aligned	GDDR1_RX.ECLK.Aligned	DDR x1 Input using ECLK. Data is edge-to-edge with incoming clock.	E
RX DDRX2 Aligned	GDDR2_RX.ECLK.Aligned/ GDDR2_RX.ECLK.Aligned (no CLKDIV)	DDR x2 Input using ECLK. Data is edge-to-edge with incoming clock.	E, EA
RX DDRX2 Aligned	GDDR2_RX.DQS.Aligned	DDR x2 Input using DQS. Data is edge-to-edge with incoming clock.	E, EA
RX DDRX1 Centered	GDDR1_RX.DQS.Centered	DDR x1 Input using DQS. Clock is already centered in data window.	E, EA
RX DDRX1 Centered	GDDR1_RX.ECLK.Centered	DDR x1 Input using ECLK. Clock is already centered in data window.	E
RX DDRX2 Centered	GDDR2_RX.ECLK.Centered	DDR x2 Input using ECLK. Clock is already centered in data window.	E, EA
RX DDRX2 Centered	GDDR2_RX.DQS.Centered	DDR x2 Input using DQS. Clock is already centered in data window.	E, EA
RX DDRX2 Dynamic	GDDR2_RX.ECLK.Dynamic	DDR x2 Input with Dynamic Alignment using ECLK.	EA
RX DDRX2 Dynamic	GDDR2_RX.DQS.Dynamic	DDR x2 Input with Dynamic Alignment using DQS.	EA
RX DDRX2 Dynamic	GDDR2_RX.PLL.Dynamic	DDR x2 Input with Dynamic Alignment using ECLK.	EA
TX SDR	GOREG_TX.SCLK	SDR Output using SCLK. Clock is forwarded through ODDR.	E, EA
TX DDRX1 Centered	GDDR1_TX.SCLK.Centered	DDR x1 Output using SCLK. Clock is centered using PLL with different SCLK.	E, EA
TX DDRX1 Centered	GDDR1_TX.DQS.Centered	DDR x1 Output using DQS. Clock is centered using DQSDLL and ODDRQDS.	E, EA
TX DDRX1 Aligned	GDDR1_TX.SCLK.Aligned	DDR x1 Output using SCLK. Data is edge-on-edge using same clock through ODDR.	E, EA
TX DDRX2 Aligned	GDDR2_TX.Aligned	DDR x2 Output that is edge-on-edge.	EA
TX DDRX2 Centered	GDDR2_TX.DQSDLL.Centered	DDR x2 Output that is pre-centered using DQS-DLL	EA
TX DDRX2 Centered	GDDR2_TX.PLL.Centered	DDR x2 Output that is pre-centered using PLL	EA

The following describes the naming conventions used for each of the interfaces listed in Table 12-1.

- G – Generic
- IREG – SDR input I/O register
- OREG – SDR output I/O register
- DDRX1 – DDR x1 I/O register
- DDRX2 – DDR x2 I/O register
- \_RX – Receive interface
- \_TX – Transmit interface
- .ECLK – Uses ECLK (edge clock) clocking resource
- .SCLK – Uses SCLK (primary clock) clocking resource
- .DQS – Uses DQS clocking resource
- .Centered – Clock is centered to the data when coming into the device
- .Aligned – Clock is aligned edge-on-edge to the data when coming into the device

### Receive Interfaces

This section lists the receive interfaces can be implemented.

#### 1. Single Data Rate Interface (GIREG\_RX.SCLK)

This interface is used when a simple input register is required for the design. The clock input to the input register can be optionally inverted if required. These interfaces always use SCLK.

#### 2. Input DDR 1x Interfaces

Input DDR 1x interfaces are used for DDR interfaces running at or below 200MHz. The 1x interfaces can be further split into aligned and centered interfaces depending on the incoming clock-to-data relationship. In addition, for each of these interfaces the incoming data is delayed using the DELAYC element to compensate for the clock injection time.

##### a. Aligned Interfaces

These interfaces are used when the data and clock are aligned edge-on-edge when input to the device. The clock on the aligned interfaces is phase-shifted 90° using the on-chip DLL or DQSLL on each side of the device. These interfaces are further split into the following interfaces.

##### i. Input DDR 1x Aligned Interface using SCLK (GDDRX1\_RX.SCLK.Aligned/ GDDRX1\_RX.SCLK.PLL.Aligned)

This interface is used on “EA” devices when the clock and data are aligned edge-on-edge. The clock is shifted 90° using a DLL or PLL before it is used in the IDDRX1 element. The SCLK is used in this case to clock the IDDRX1 element on the “EA” device.

##### ii. Input DDR 1x Aligned Interface using ECLK (GDDRX1\_RX.ECLK.Aligned)

This interface is used on “E” devices when the clock and data are aligned edge-on-edge when coming into the device. ECLK is used to clock the IDDRX1 element on the “E” device. This clock is shifted 90° using DLL before it is routed to the ECLK.

##### iii. Input DDR 1x Aligned Interface using DQS (GDDRX1\_RX.DQS.Aligned)

This interface is used when the interface bus is narrow (<10 bits) and the clock and data are aligned edge-on-edge. In this case, the DQS Lanes are used for each interface and the DQS pin is used for the clock input. The 90° shift for the clock is generated using the DQS-DLL and the clock is delayed in the DQSBUF element. This delayed clock is used to clock the IDDRX1 element. This interface then uses SCLK to clock the data from the interface to the FPGA logic.

**b. Centered Interfaces**

These interfaces are used when the data and clock are centered when input to the device. Since the clock is centered to the data, it is not required to be phase-shifted for these interfaces. These interfaces are further split into the following interfaces.

**i. Input DDR 1x Centered Interface Using ECLK (GDDR1\_RX.ECLK.Centered)**

This interface is used on “E” devices when the clock and data are centered coming into the device. The ECLK is used to clock the input DDR element on the “E” device.

**ii. Input DDR 1x Centered Interface Using SCLK (GDDR1\_RX.SCLK.Centered)**

This interface is used on “EA” devices when the clock and data are centered coming into the device. The SCLK is used to clock the input DDR element on the “EA” device.

**iii. Input DDR 1x Centered Interface using DQS (GDDR1\_RX.DQS.Centered)**

This interface is used when the interface bus is narrow (<10 bits) and the clock and data are centered. In this case, the DQS Lanes are used for each interface and the DQS pin is used for the clock input. DQSDLL is held in reset for this case as no clock shift is required. This interface then uses SCLK to clock the data from the interface to the FPGA logic.

**3. Input DDR 2x Interfaces**

Input DDR 2x interfaces are used for DDR interfaces running higher than 200MHz. The 2x interfaces can be further split into aligned and centered interfaces depending on the incoming clock-to-data relationship. In addition, for each of these interfaces the incoming data is delayed using DELAYC element to compensate for the clock injection time.

**a. Aligned Interfaces**

These interfaces are used when the data and clock are aligned edge-on-edge when input to the device. The clock on the aligned interfaces is phase shifted 90° using the on-chip DLL or DQSL on each side of the device. These interfaces are further split into the following interfaces.

**i. Input DDR 2x Aligned Interface using ECLK (GDDR2\_RX.ECLK.Aligned/  
GDDR2\_RX.ECLK.Aligned (No CLKDIV))**

This interface is used when the clock and data are aligned edge-on-edge. The incoming clock is shifted 90° using the DLL. The output of the DLL is routed to the ECLK which is used to clock the IDDR2 element. The SCLK required for this interface is generated by dividing the ECLK by two in the CLKDIV module or in the DLL module.

**ii. Input DDR 2x Aligned Interface using DQS (GDDR2\_RX.DQS.Aligned)**

This interface is used when the interface bus is narrow (<10 bits) and the clock and data are aligned edge-on-edge. In this case, the DQS Lanes are used for each interface and DQS pin is used for the clock input. The 90° shift for clock is generated using the DQSDLL and the clock is delayed in the DQSBUF element. This delayed clock is used to clock the IDDR2 element. The SCLK required for this interface is generated by dividing the ECLK by two in the CLKDIV module.

**b. Centered Interfaces**

These interfaces are used when the data and clock are centered when input to the device. Since the clock is centered to the data, it does not required to be phase-shifted for these interfaces. These interfaces are further split into the following interfaces.

**i. Input DDR 2x Centered Interface using ECLK (GDDR2\_RX.ECLK.Centered)**

This interface is used when clock and data are centered coming into the device. The clock is connected directly to the ECLK which is used to clock the IDDR2 element. The SCLK required for this interface is generated by dividing the ECLK by two in the CLKDIV module.

- ii. **Input DDR 1x Centered Interface using DQS (GDDR2\_RX.DQS.Centered)**  
This interface is used when the interface bus is narrow (<10 bits) and the clock and data are centered. In this case, the DQS Lanes are used for each interface and the DQS pin is used for the clock input going to the DQSBUF module. DQSDLL is held in reset for this case as no clock shift is required. The clock output of the DQSBUF is used to clock the IDDRX2 element. A PLL is used to generate the SCLK for this interface.
- c. **Dynamic Interfaces (“EA” Devices Only)**  
The data delay input on the data input of the input DDR 2x centered interfaces can optionally be controlled dynamically by the user logic using the DELAYB element. For dynamic control of the clock or data delay, one of following dynamic interfaces can be used. The dynamic interfaces are only available on “EA” devices.
  - i. **Input DDR 2x Centered Interface Using ECLK and Dynamic Delay (GDDR2\_RX.ECLK.Dynamic)**  
This interface is similar to the GDDR2\_RX.ECLK.Centered interface described above but the input data delay is controlled by the user with the DELAYB element.
  - ii. **Input DDR 2x Centered Interface Using DQS and Dynamic Delay (GDDR2\_RX.DQS.Dynamic)**  
This interface is similar to the GDDR2\_RX.DQS.Centered interface described above but the input data delay is controlled by the user with the DELAYB element.
  - iii. **Input DDR 2x Centered Interface Using PLL and Dynamic Delay (GDDR2\_RX.PLL.Dynamic)**  
In this interface the PLL is used to dynamically shift the clock to adjust to the correct position to the data. This interface will generate a bus-based delay for the interface.
- d. **7:1 LVDS Interface**  
This interface should be used when implementing a 7:1 LVDS interface. The 7:1 LVDS interface requires that the input clock is multiplied 3.5x before input to the IDDRX2 element to demux the data. Refer to RD1030, [7:1 LVDS Video Interface Reference Design](#) for further information.

## Transmit Interfaces

This section lists the transmit interfaces can be implemented.

### 1. Single Data Rate Interface (GOREG\_TX.SCLK)

This interface is used for a SDR data output implementation with tight specifications on clock out to data out skew. This interface uses a simple output flip-flop for the data but forwards the clock using an ODDR register. The same clock is used for both data and clock generation.

### 2. Output DDR 1x Interfaces

Output DDR 1x interfaces are used for DDR interfaces running at or below 200MHz. The 1x interfaces can further be split into aligned and centered interfaces depending on the relationship between the forwarded clock and data. The following are the different 1x interfaces.

#### a. Aligned Interfaces

These interfaces are used to provide data and clocks that are aligned edge-on-edge when leaving the device. These interfaces are further split into the following.

##### i. Output DDR 1x Aligned Interface Using SCLK (GDDR1\_TX.SCLK.Aligned)

This interface uses SCLK to generate clock and data that are aligned edge on edge

#### b. Centered Interfaces

These interfaces are used to provide data and clocks that are centered when leaving the device. The clock output is phase-shifted 90° using the on-chip PLL so that it can be centered to the data. These interfaces are further split as follows.

- i. **Output DDR 1x Centered Interface Using SCLK (GDDR1\_TX.SCLK.Centered)**  
In this case, the SCLK is used to generate the data and clock output. SCLK used to generate the clock is shifted 90° using a PLL so that it can be pre-centered to the data output.
- ii. **Output DDR 1x Centered Interface Using DQS (GDDR1\_TX.DQS.Centered)**  
This interface is used when implementing interfaces that are <10 bits wide. The DQS Lanes are for data and clock assignments. The clock is pre-centered to the data using the DQDLL and the ODDRQSA blocks.

### 3. Output DDR 2x Interfaces

Output DDR 2x interfaces are used for DDR interfaces running higher than 200MHz. The 2x interfaces can further be split into aligned and centered interfaces depending on the relationship between the forwarded clock and data. Output DDR 2x interfaces are supported only on “EA” devices. The following are the different 2x interfaces.

#### a. Aligned Interfaces

These interfaces are used to provide data and clocks that are aligned edge-on-edge when leaving the device. These interfaces are further split into the following.

- i. **Output DDR 2x Aligned Interface (GDDR2\_TX.Aligned)**  
ODDRX2 is used generate the clock and data that are aligned in phase.

#### b. Centered Interfaces

These interfaces are used to provide data and clocks that are centered when leaving the device. The clock output is phase shifted 90° so that it can be centered to the data. These interfaces are further split into the following.

- i. **Output DDR 2x Centered Interface using DQSDLL (GDDR2\_TX.DQSDLL.Centered)**  
This interface is primarily used for interfaces that are <10 bits wide. In this case, a DQSDLL is used to generate the 90° shift required to pre-center the clock to the data output.
- ii. **Output DDR 2x Centered Interface using PLL (GDDR2\_TX.PLL.Centered)**  
This interface is primarily used for wider interfaces. Here, a PLL is used to generate the 90° shift required to pre-center the clock to the data output.

#### c. 7:1 LVDS Interface

This interface is used when implementing a 7:1 LVDS interface. The 7:1 LVDS interface requires that the clock output be multiplied 3.5x before going to the ODDRX2 module. Refer to RD1030, [7:1 LVDS Video Interface Reference Design](#) for further information.

## Using IPexpress to Build High-Speed DDR Interfaces

The IPexpress tool is used to configure and generate all the high-speed interfaces described above. IPexpress generates a complete HDL module including clocking requirements for each of the interfaces.

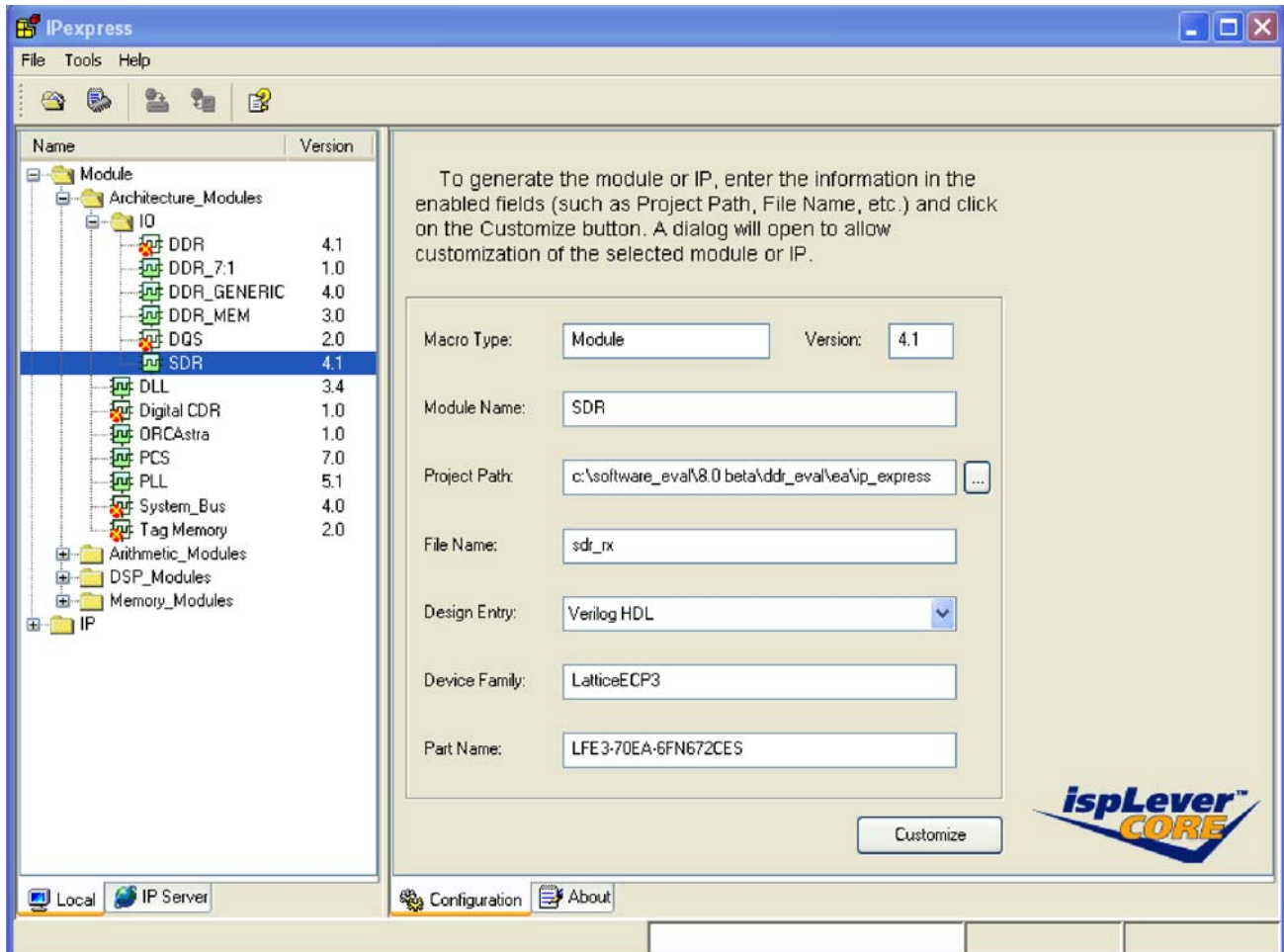
This section assumes that ispLEVER 8.0 is used for generation of the interfaces. If you are using ispLEVER 7.2 SP2, see [Appendix A. Building DDR Interfaces Using IPexpress in ispLEVER 7.2 SP2](#). If you are using Lattice Diamond® design software, see [Appendix B. Building SDR/DDR Interfaces Using IPexpress in Diamond](#).

For a detailed block diagram of each interface generated by IPexpress, see the [High-Speed DDR Interface Details](#) section.

IPexpress can be opened from the **Tools** menu in Project Navigator. All DDR modules are located under **Architecture Modules -> IO**. This section will cover SDR and DDR\_GENERIC. DDR\_MEM is discussed in the [Implementing DDR/DDR2/DDR3 Memory Interfaces](#) section.



Figure 12-3. IPexpress Main Window



Select the type of interface you would like to build and enter the name of the module. Figure 12-3 shows the type of interface selected as “SDR” and module name entered. Each module can then be configured by clicking the **Customize** button.

### Building SDR Modules

Choose interface type **SDR**, enter module name and click **Customize** to open the configuration tab.

Figure 12-4 shows the Configuration Tab for the SDR module in IPexpress. Table 12-2 lists the various configurations options available for SDR modules.



Figure 12-4. SDR Configuration Tab

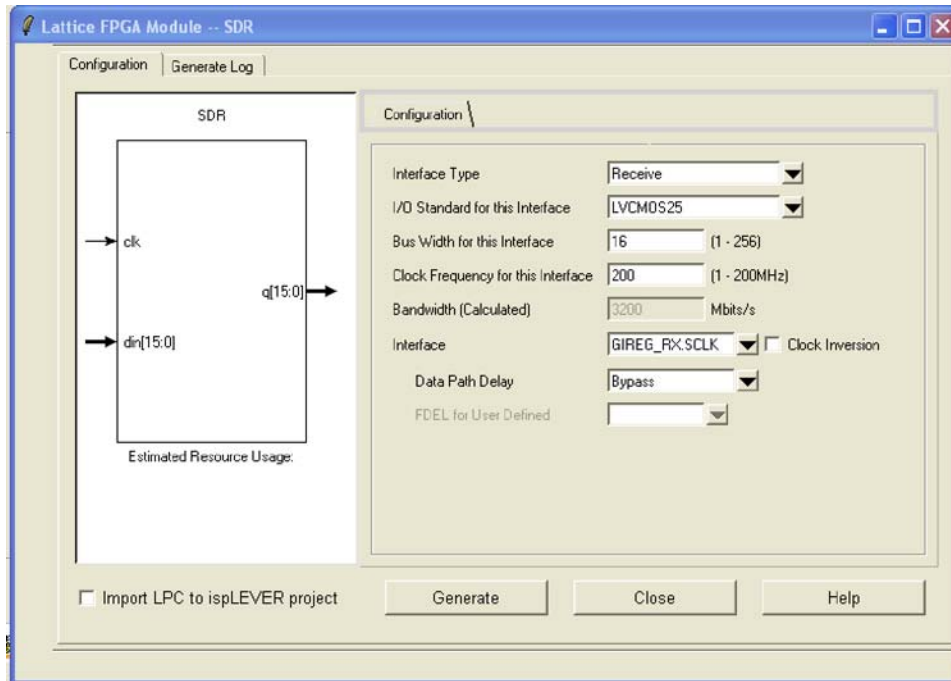


Table 12-2. SDR Configuration Parameters

GUI Option	Description	Values	Default
Interface Type	Type of interface (transmit or receive)	Transmit, Receive	Receive
I/O Standard for this Interface	I/O standard to be used for the interface.	Transmit and Receive: LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, LVCMOS33, LVCMOS33D, LVDS25, BLVDS25, MLVDS, LVPECL33, HSTL18_I, HSTL18_II, HSTL18D_I, HSTL18D_II, HSTL15_I, HSTL15D_I, SSTL33_I, SSTL33_II, SSTL33D_I, SSTL33D_II, SSTL25_I, SSTL25_II, SSTL25D_I, SSTL25D_II, SSTL18_I, SSTL18_II, SSTL18D_I, SSTL18D_II, SSTL15, SSTL15D, PCI33, LVTTTL33  Transmit only: RSDS, MINILVDS, PPLVDS, LVDS25E, RSDSE	LVCMOS25
Bus Width for this Interface	Bus size for the interface.	1 - 256	16
Clock Frequency for this Interface	Speed at which the interface will run.	1 - 200	200
Bandwidth (Calculated)	Calculated from the clock frequency entered.	(Calculated)	(Calculated)
Interface	Interface selected based on previous entries.	Transmit: GOREG_TX.SCLK Receive: GIREG_RX.SCLK (default)	GIREG_RX.SCLK
Clock Inversion	Option to invert the clock input to the I/O register.	DISABLED, ENABLED	DISABLED
Data Path Delay	Data input can be optionally delayed using the DELAY block.	Bypass, Dynamic <sup>1</sup> , User Defined	Bypass

Table 12-2. SDR Configuration Parameters (Continued)

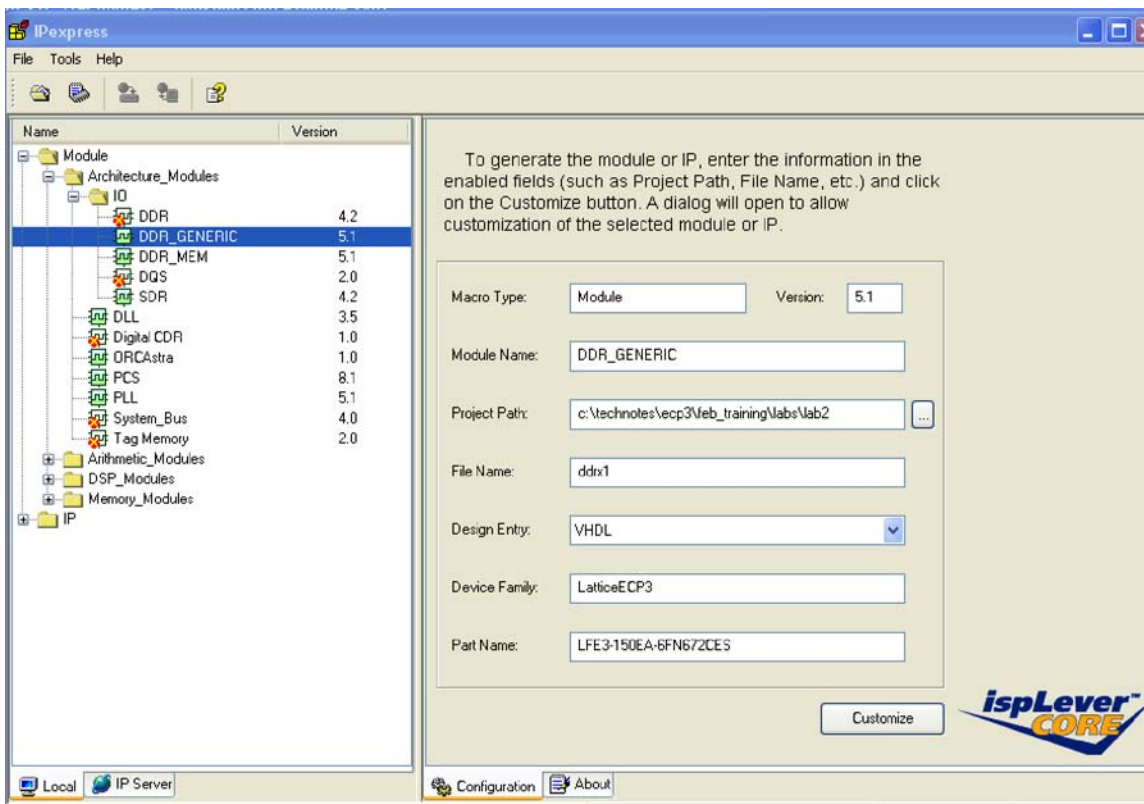
GUI Option	Description	Values	Default
FDEL for User Defined	If Delay type selected above is user defined, delay values can be entered with this parameter.	0 to 15 <sup>2</sup>	0

1. When Delay type Dynamic is selected, the 16-step delay values must be controlled from the user's design.
2. A FDEL is a fine-delay value that is additive. The delay value for a FDEL can be found in the [LatticeECP3 Family Data Sheet](#).

## Building DDR Generic Modules

Choose interface type **DDR\_GENERIC**, enter module name and click **Customize** to open the configuration tab.

Figure 12-5. “DDR\_Generic” Selected in Main IPexpress Window



When clicking **Customize**, DDR modules have a Pre-Configuration Tab and a Configuration” Tab. The Pre-Configuration Tab allows users to enter information about the type of interface to be built. Based on the entries in the Pre-configuration Tab, the Configuration Tab will be populated with the best interface selection. The user can also, if necessary, override the selection made for the interface in the Configuration Tab and customize the interface based on design requirements.

Figure 12-6 shows the Pre-Configuration Tab for DDR generic interfaces. Table 12-3 lists the various parameters in the tab.

Figure 12-6. DDR Generic Pre-Configuration Tab

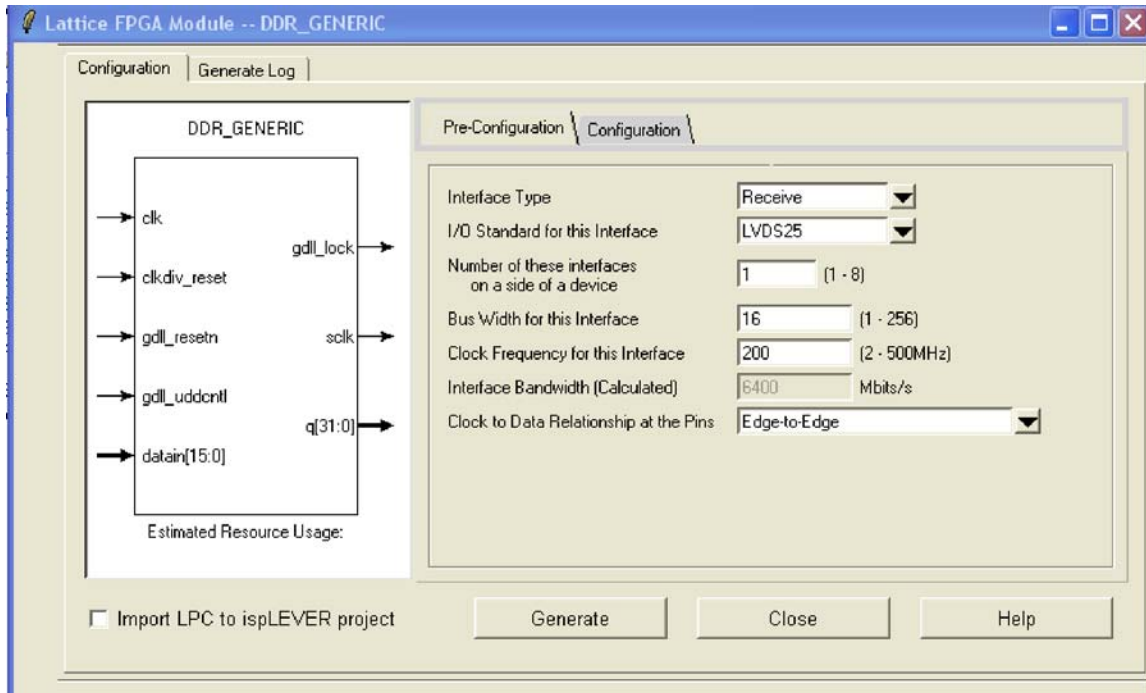


Table 12-3. Pre-Configuration Tab Settings

GUI Option	Description	Values
Interface Type (Transmit or Receive)	Type of interface (Receive or Transmit)	Transmit, Receive
I/O Standard for this Interface	I/O Standard used for the interface	Transmit and Receive: LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, LVCMOS33, LVCMOS33D, LVDS25, BLVDS25, MLVDS, LVPECL33, HSTL18_I, HSTL18_II, HSTL18D_I, HSTL18D_II, HSTL15_I, HSTL15D_I, SSTL33_I, SSTL33_II, SSTL33D_I, SSTL33D_II, SSTL25_I, SSTL25_II, SSTL25D_I, SSTL25D_II, SSTL18_I, SSTL18_II, SSTL18D_I, SSTL18D_II, SSTL15, SSTL15D, PCI33, LVTTTL33  Transmit only: RSDS, MINILVDS, PPLVDS, LVDS25E, RSDSE
Number of interfaces on a side of a device	Number of interfaces to be implemented per side. This is used primarily for narrow bus width interfaces (<10). Otherwise it is recommended to leave this at 1.	1 to 8
Bus Width for this Interface	Bus width for each interface. If the number of interfaces per side is >1 then the bus width per interface is limited to 10. If number of interfaces per side is >1 and if using differential I/O standards then bus width is limited to 5.	1-256
Clock Frequency for this Interface	Interface speed	2 - 500 MHz

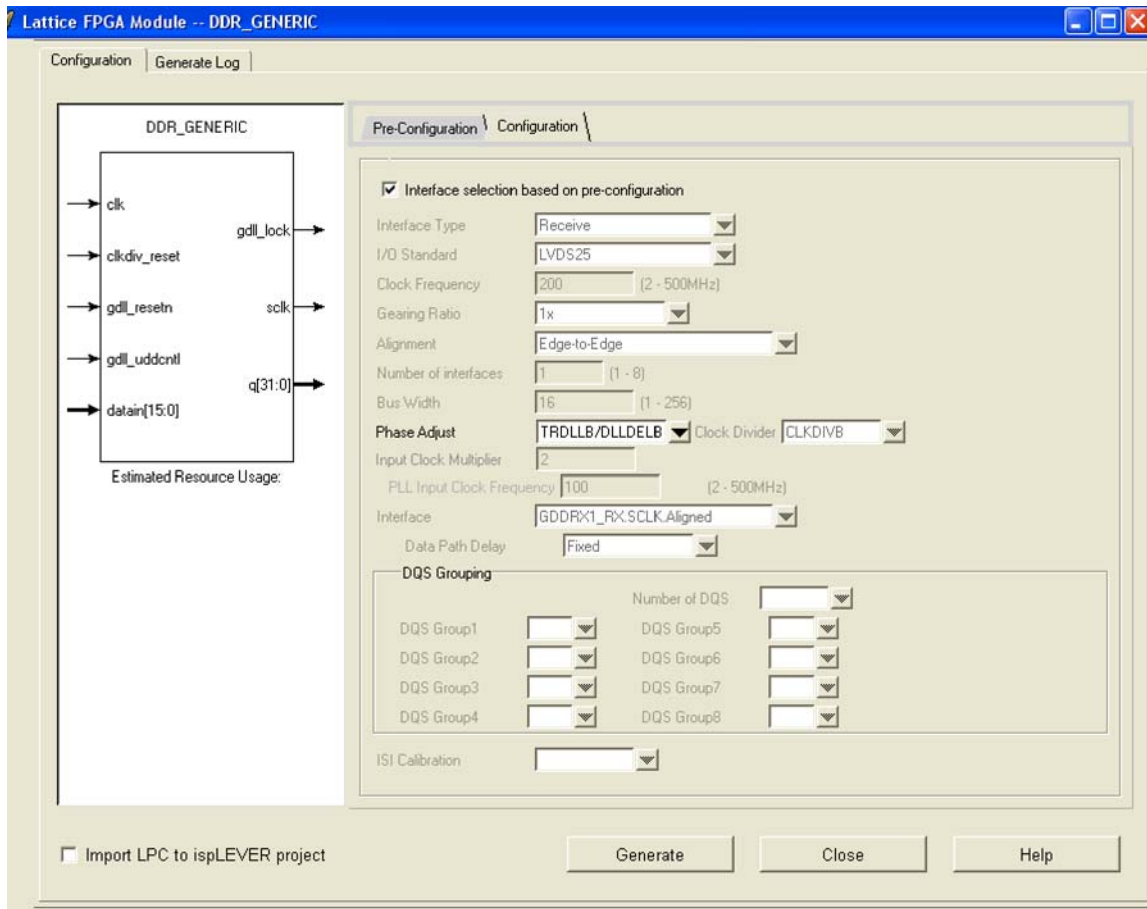
**Table 12-3. Pre-Configuration Tab Settings (Continued)**

GUI Option	Description	Values
Interface Bandwidth (Calculated)	Bandwidth is calculated from the clock frequency.	Calculated
Clock to Data Relationship at the Pins	Relationship between clock and data.	Edge-to-Edge, Centered, Dynamic Data Phase Alignment Required <sup>1</sup> , Dynamic Clock Phase Alignment Required

1. Dynamic Phase Alignment is only available for x2 interfaces (i.e, when the clock frequency is higher than 200 MHz).

Based on the selections made in the Pre-Configuration Tab, the Configuration Tab is populated with the selections. Figure 12-7 shows the Configuration Tab for the selections made in the Pre-Configuration Tab.

**Figure 12-7. DDR Generic Configuration Tab**



The checkbox at the top of this tab indicates that the interface is selected based on entries in the Pre-Configuration Tab. The user can choose to change these values by disabling this entry. Note that IPexpress chooses the most suitable interface based on selections made in the Pre-Configuration Tab.

Table 12-4 lists the various parameters in the Configuration Tab.

**Table 12-4. Configuration Tab Settings**

GUI Option	Description	Values	Default Value
Interface Selection Based on Pre-configuration	Indicates interface is selected based on selection made in the Pre-configuration tab. Disabling this checkbox allows users to make changes if needed.	ENABLED, DISABLED	ENABLED
Interface Type	Type of interface (receive or transmit)	Transmit, Receive	Receive
I/O Standard	I/O standard used for the interface	All the ones listed in the Pre-configuration tab	LVC MOS25
Clock Frequency	Speed of the interface	2 to 500 MHz	200 MHz
Gearing Ratio	DDR register gearing ratio (1x or 2x)	1x, 2x	1x
Alignment	Clock to data alignment	Edge-to-Edge, Centered, Dynamic Data Phase Alignment Required, Dynamic Clock Phase Alignment Required	Edge-to-Edge
Number of Interfaces	Number of interfaces to be implemented per side. This is primarily used for narrow bus width interfaces (<10), otherwise it is recommended to leave this at 1.	1 to 8	1
Bus Width	Bus width for each interface. If the number of interfaces per side is >1 then the bus width per interface is limited to 10. If the number of interfaces per side is >1 and if using differential I/O standards then the bus width is limited to 5.	1 to 256	10
Phase Adjust	Module used for phase shifting input clock.	TRDLLB/DLLDELB, PLL <sup>1</sup>	TRDLLB/DLLDELB
Clock Divider	Module used for generation of SCLK from ECLK.	CLKDIVB, TRDLLB <sup>2</sup>	CLKDIVB
Interface	Shows list of all valid high-speed interfaces for a given configuration.	See Table 12-5 for interfaces available for a given configuration.	GDDR1_RX.SCLK. Aligned (EA devices); GDDR1_RX.ECLK. Aligned (E devices)
Data Path Delay	Data input can be optionally delayed using the DELAY block. Value is selected based on Interface Type.	Bypass, Fixed, Dynamic <sup>3</sup>	Fixed
Number of DQS Groups	Enabled when a DQS interface is selected in the Interface selection.	1 to 8	
Number of DQ: DQS Group1 to DQS Group8	This option can be used to change the number of DQ assigned to each DQS lane. Each DQS lane can support up to 10 DQ.	1 to 10	

1. Only available when using GDDR2\_RX.ECLK.Aligned interface.

2. Only available when using GDDR2\_RX.SCLK Aligned interface.

3. When Dynamic Delay is selected, the 16-step delay values must be controlled from the user's design.

Table 12-5 shows how the interfaces are selected by IPexpress based on the selections made in the Pre-Configuration Tab.

**Table 12-5. IPexpress Interface Selection**

Device Selected	Interface Type	Gearing Ratio <sup>1</sup>	Alignment	Number of Interfaces	Interface
EA	Receive	1x	Edge-to-Edge	1	GDDR1_RX.SCLK.Aligned
EA	Receive	1x	Centered	1	GDDR1_RX.SCLK.Centered
E	Receive	1x	Edge-to-Edge	1	GDDR1_RX.ECLK.Aligned
E	Receive	1x	Centered	1	GDDR1_RX.ECLK.Centered
E, EA	Receive	1x	Edge-to-Edge	>1	GDDR1_RX.DQS.Aligned
E, EA	Receive	1x	Centered	>1	GDDR1_RX.DQS.Centered
E, EA	Receive	2x	Edge-to-Edge	1	GDDR2_RX.ECLK.Aligned
E, EA	Receive	2x	Centered	1	GDDR2_RX.ECLK.Centered
E, EA	Receive	2x	Edge-to-Edge	>1	GDDR2_RX.DQS.Aligned
E, EA	Receive	2x	Centered	>1	GDDR2_RX.DQS.Centered
EA	Receive	2x	Dynamic	1	GDDR2_RX.ECLK.Dynamic (Default)
EA					GDDR2_RX.DQS.Dynamic <sup>2</sup>
EA					GDDR2_RX.PLL.Dynamic <sup>2</sup>
E, EA	Transmit	1x	Centered	1	GDDR1_TX.SCLK.Centered
E, EA	Transmit	1x	Edge-to-Edge	1	GDDR1_TX.SCLK.Aligned
E, EA	Transmit	1x	Centered	>1	GDDR1_TX.DQS.Centered
EA	Transmit	2x	Edge-to-Edge	1	GDDR2_TX.Aligned
EA	Transmit	2x	Centered	>1	GDDR2_TX.DQSDLL.Centered
EA	Transmit	2x	Centered	1	GDDR2_TX.PLL.Centered

1. Gearing Ratio of 1x is selected for clock frequencies less than 200MHz. Gearing ratio of 2x is selected for frequencies above 200 MHz.

2. These interfaces can only be selected in the Configuration Tab.

The implementation for several of the interfaces described above differs between the “E” and “EA” devices. Refer to the [High-Speed DDR Interface Details](#) section to see implementation details for “E” and “EA” devices.

The Data Delay setting for each interface is predetermined and cannot be changed by the user. User can only control Data Delay values when using a dynamic interface.

*Note: Some modules generated by IPexpress have a SCLK and ECLK output port. If present, this port must be used to drive logic outside the interface driven by the same signal. In these modules, the input buffer for the clock is inside the IPexpress module and therefore cannot be used to drive other logic in the top level.*

## High-Speed DDR Interface Details

This section describes each of the generic high-speed interfaces in detail including the clocking to be used for each interface. For detailed information about the LatticeECP3 clocking structure, refer to TN1178, [LatticeECP3 sys-CLOCK PLL/DLL Design and Usage Guide](#). The various interface rules and preferences listed under each interface should be followed to build these interfaces successfully. Each of the interfaces for the “EA” devices has an ID label associated with the interface. The interface ID will be set when generating the interface using IPexpress. This ID is entered using an attribute called IDDRAPPS or ODDRAPPS. The software uses this ID to set appropriate the data delay for the DELAYC element used in each of the interfaces. It is required that every interface on the “EA” devices use this attribute to achieve the correct timing results in the software Trace report. Refer to the [Timing Analysis for High-Speed DDR Interfaces](#) section for more information about the timing analysis on these interfaces. It is also necessary to follow the interface rules and preferences listed for each of the interface descriptions below for the interfaces to work as described.

In order to achieve higher speeds, the guidelines described in the [Placement Guidelines for High-Speed DDR Interfaces](#) section should be strictly followed.

All the interfaces described below are supported using ispLEVER 8.0 software. Some of these interfaces will not work in versions of ispLEVER prior to version 8.0. Refer to the [Generic DDR Design Guidelines](#) section to see all other design guidelines.

## GIREG\_RX.SCLK

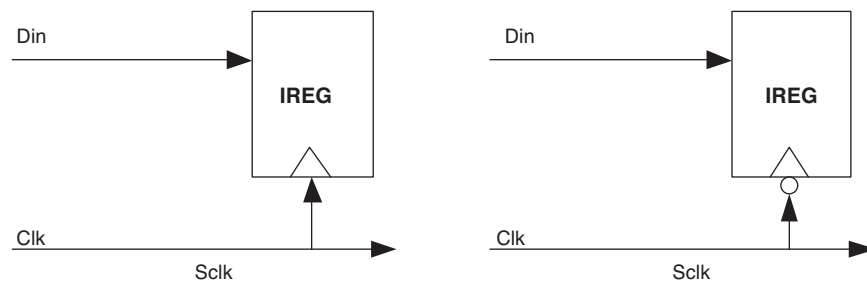
Generic SDR Receive Interface using SCLK.

Device Support: “E” and “EA” devices

### Description

This is a generic interface for single data rate (SDR) data. An optional inverter can be used to center the clock for aligned inputs. A PLL or DLL can be used to remove the clock injection delay or adjust the setup and hold times. There are a limited number of DLLs in the architecture and these should be saved for high-speed interfaces when necessary. This interface can either be built using IPexpress or inferred during synthesis.

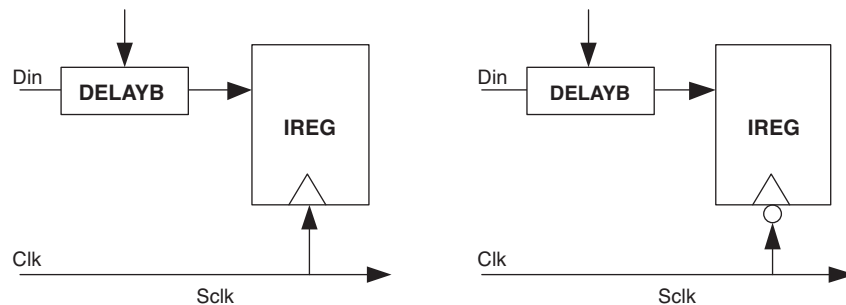
**Figure 12-8. GIREG\_RX Interface (“E” and “EA” Devices)**



This interface also supports data delay on the data input. This delay value is set using a DELAYB or a DELAYC element. A DELAYC element provides a fixed delay to match the SCLK injection time.

DELAYB is used when user chooses to update the delay dynamically or use user-defined static values. Figure 12-9 shows the DELAYB element connected to this interface.

**Figure 12-9. GIREG\_RX.SCLK with Delay Interface (“E” and “EA” Devices)**



### Interface Rules

- The input clock must use a dedicated clock (PCLK) input pin.

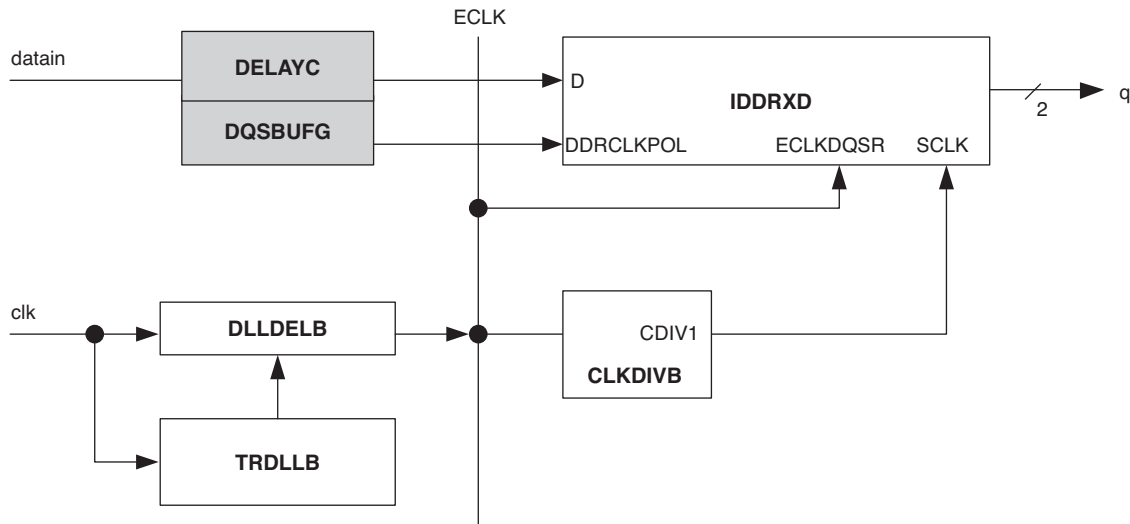
### GDDR1\_RX.ECLK.Aligned

Generic DDR Receive Interface using ECLK with Aligned External Interface  
Device Support: “E” devices only

#### Description

This DDR interface uses the ECLK and the TRDLLB to provide a 90° clock shift to center the clock at the IDDRXD. DLLDELB is used to delay the incoming clock by 90°. CLKDIV is used to generate a divide-by-1 clock that is connected to the SCLK. Since this interface uses the ECLK it can be extended to support large data bus sizes for the entire side of the device.

Figure 12-10. GDDR1\_RX.ECLK.Aligned Interface (“E” Devices Only)



#### Interface Rules

- The input clock must use a GPLLT\_IN or GDLLT\_IN pin. All data for the interface must all be on the same ECLK (same side).
- Since there is only one DLLDELB and one CLKDIVB per left and right sides of the device, users can implement only one such interface per side, or two total on the device.
- The clock net connected to SCLK should be on a primary clock net. The user must assign the “USE PRIMARY NET<clk>” preference to assign the SCLK clock net to a primary clock.
- The pin assignments for data and clocks will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.

### GDDR1\_RX.ECLK.Centered

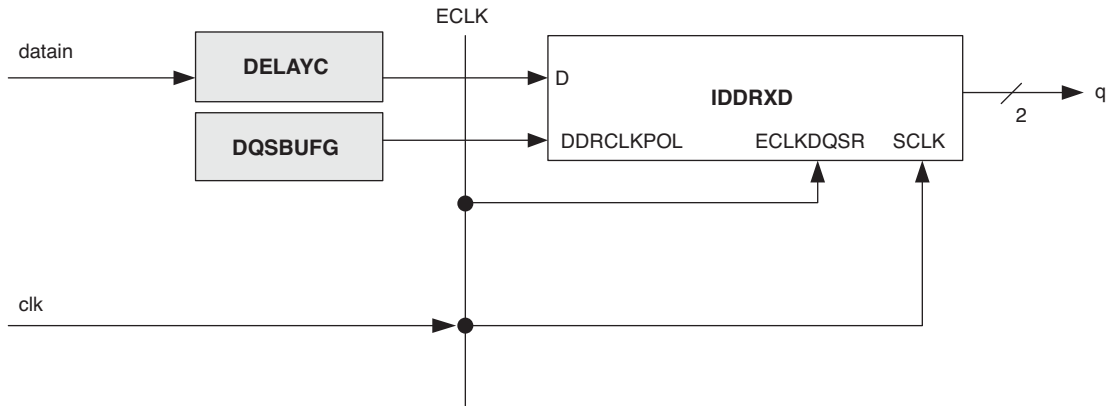
Generic DDR Receive Interface using ECLK with Centered External Interface.  
Device Support: “E” devices only

#### Description

This DDR interface uses the ECLK and DELAYC to match clock and data delay at the IDDRXD. Since this interface uses the ECLK it can be extended to support large data bus sizes for the entire side of the device.



**Figure 12-11. GDDR1\_RX.ECLK.Centered Interface (“E” Devices Only)**



**Interface Rules**

- The input clock port must use a dedicated clock (PCLK) input pin. All data for the interface must be on the same ECLK (same side).
- The clock net connected to SCLK must be routed on a primary clock net. It is the user’s responsibility to assign the SCLK clock net to a primary clock tree using the “USE PRIMARY NET<clk>” preference.
- The pin assignments for data and clock will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.

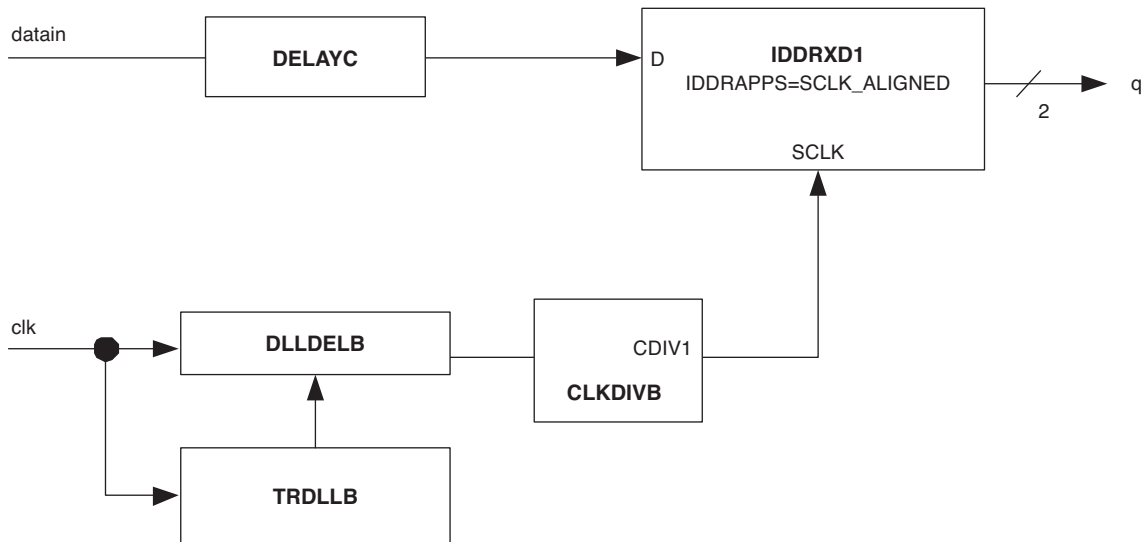
**GDDR1\_RX.SCLK.Aligned**

Generic DDR Receive Interface using SCLK with Aligned External Interface  
Device Support: “EA” devices only

**Description**

This DDR interface uses the SCLK and the TRDLLB to provide a 90° clock shift to center the clock at the IDDRXD1. This interface is useful for large data buses (>10 bits). A DELAYC is used to adjust data delay for the SCLK clock injection time. CLKDIV in divide-by-1 setting is used to generate the SCLK.

**Figure 12-12. GDDR1\_RX.SCLK.Aligned Interface (“EA” Devices Only)**



**Interface Rules**

- The clock input must use a dedicated GPLLT\_IN or GDLLT\_IN clock input pin (two pins per side). A dedicated PCLK pin on the top side can connect directly to the TRDLLB as well.
- There is only one DLLDELB per side of the device (left and right sides) which limits this interface to one clock rate per side or two per device.
- CLKDIV is required to generate SCLK.
- The clock net connected to SCLK should be on a primary clock net. The user must assign the “USE PRIMARY NET<clk>” preference to assign the SCLK clock net to a primary clock.

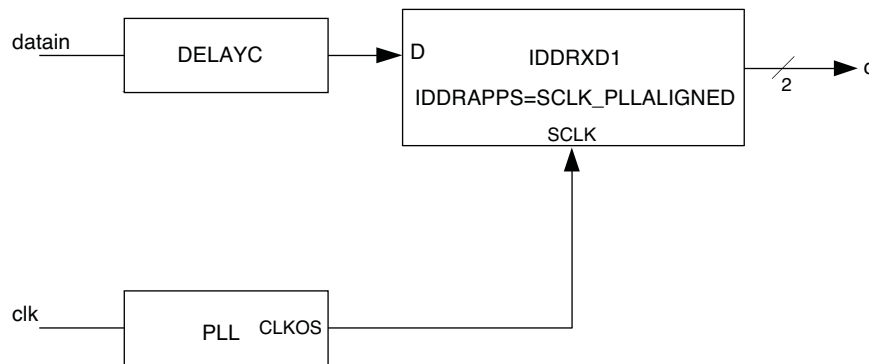
**GDDR1\_RX.SCLK.PLL.Aligned**

Generic DDR Receive Interface using SCLK with Aligned External Interface  
 Device Support: “EA” devices only

**Description**

This DDR interface uses the SCLK and a PLL to provide a 90° clock shift to center the clock at the IDDRXD1. This interface is useful for large data buses (>10 bits). A DELAYC is used to adjust data delay for the SCLK clock injection time. CLKDIV in divide by 1 setting is used to generate the SCLK.

**Figure 12-13. GDDR1\_RX.SCLK.PLL.Aligned Interface (“EA” Devices Only)**



**Interface Rules**

The clock input must use a dedicated GPLLT\_IN clock input pin (two pins per side).

CLKDIV is required to generate SCLK.

The clock net connected to SCLK should be on a primary clock net. The user must assign the “USE PRIMARY NET<clk>” preference to assign the SCLK clock net to a primary clock.

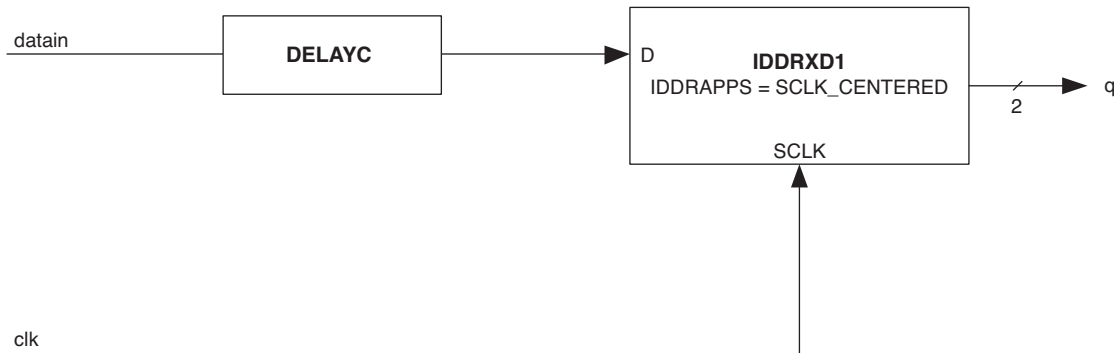
### GDDRX1\_RX.SCLK.Centered

Generic DDR Receive Interface using SCLK with Centered External Interface  
Device Support: “EA” devices only

#### Description

This DDR interface uses the SCLK and DELAYC to match clock and data delay at the IDDRXD. This interface is useful for large data buses (>10 bits).

**Figure 12-14. GDDRX1\_RX.SCLK.Centered Interface (“EA” Devices Only)**



#### Interface Rules

- The clock input must use a dedicated clock (PCLK) input pin. The output of a PLL clock in bypass mode can be connected to the SCLK as well.
- The clock connected to SCLK should be on a primary clock net. The user must assign the “USE PRIMARY NET<clk>” preference to assign the SCLK clock net to a primary clock.

### GDDRX1\_RX.DQS.Aligned

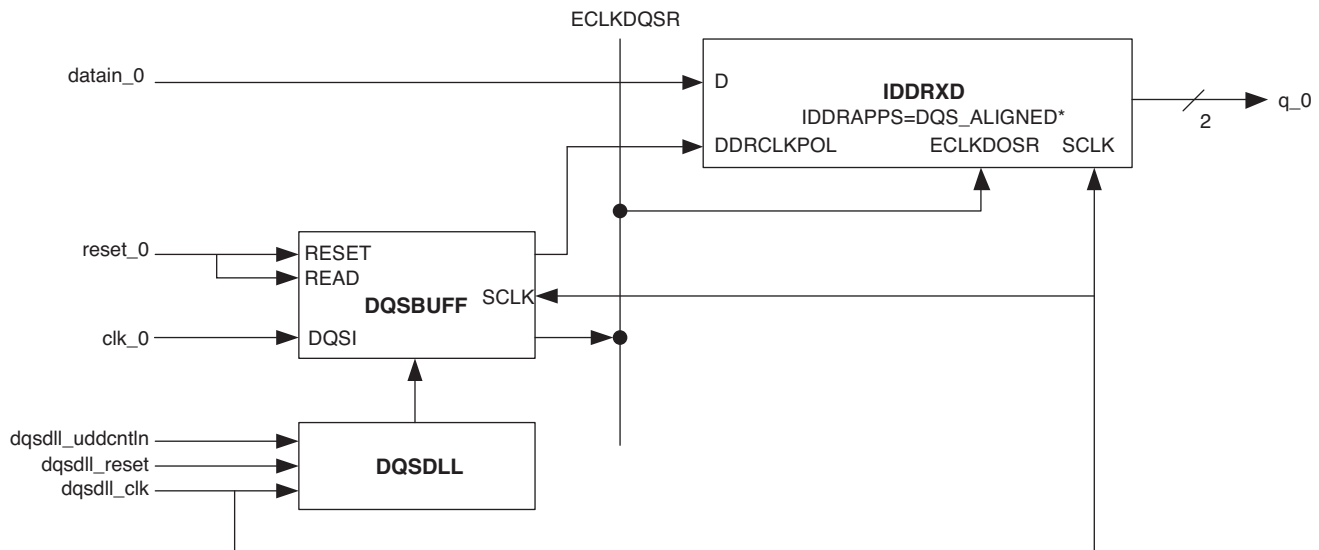
Generic DDR Receive Interface using DQS Lane with Aligned External Interface  
Device Support: “E” and “EA” devices

#### Description

This DDR interface uses the DQS and the DQSDLL to provide a 90° clock shift to center the clock at the IDDRXD. This interface is uses a DQS lane and is useful for small data buses (< 11 bits). DQSDLL provide the 90° delay to the DQSBUFF which is used to delay the incoming clock.

A reference clock input “dqsdll\_clk” running at the same frequency as DQS clock is required to be input to the DQSDLL to generate the delay.

Figure 12-15. GDDR1\_RX.DQS.Aligned Interface (“E” and “EA” Devices)



\* IDDRAPPS required for “EA” devices only.

Any frequency-locked clock or the local clock from the input pin can be used for SCLK. The timing transfer between the ECLKDQSR and SCLK is handled in the hardware through the DDRCLKPOL.

### Interface Rules

- The input clock must use a DQS input pin and all data inputs must be in the same DQS lane.
- The VREF1 for the selected DQS lane must be powered on the board.
- There is only one DQSDLL per side of the device which limits sharing of this interface on a side unless all are running at the same rate.
- The following sequence must be followed when resetting the interface:
  - Assert DQSDLL\_RESET to DQSDLL and RESET to the modules
  - Deassert DQSDLL\_RESET to DQSDLL first
  - Wait for DQSDLL lock to go high
  - Assert DQSDLL\_UDDCNTLN input of DQSDLL for at least four SCLK cycles. See the section “DQSDLLB” on page 89 for the detailed requirements for the DQSDLL\_UDDCNTLN input of DQSDLLB
  - Deassert DQSDLL\_UDDCNTLN
  - Wait for four SCLK cycles, then deassert RESET to the other modules
- “dqsdl\_clk” and clock net connected to SCLK must use the primary clock tree. It is the user’s responsibility to assign the “USE PRIMARY NET<clk>” preference on these nets to assign it to the primary clock.
- The pin assignments for data and clock will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.

### GDDR1\_RX.DQS.Centered

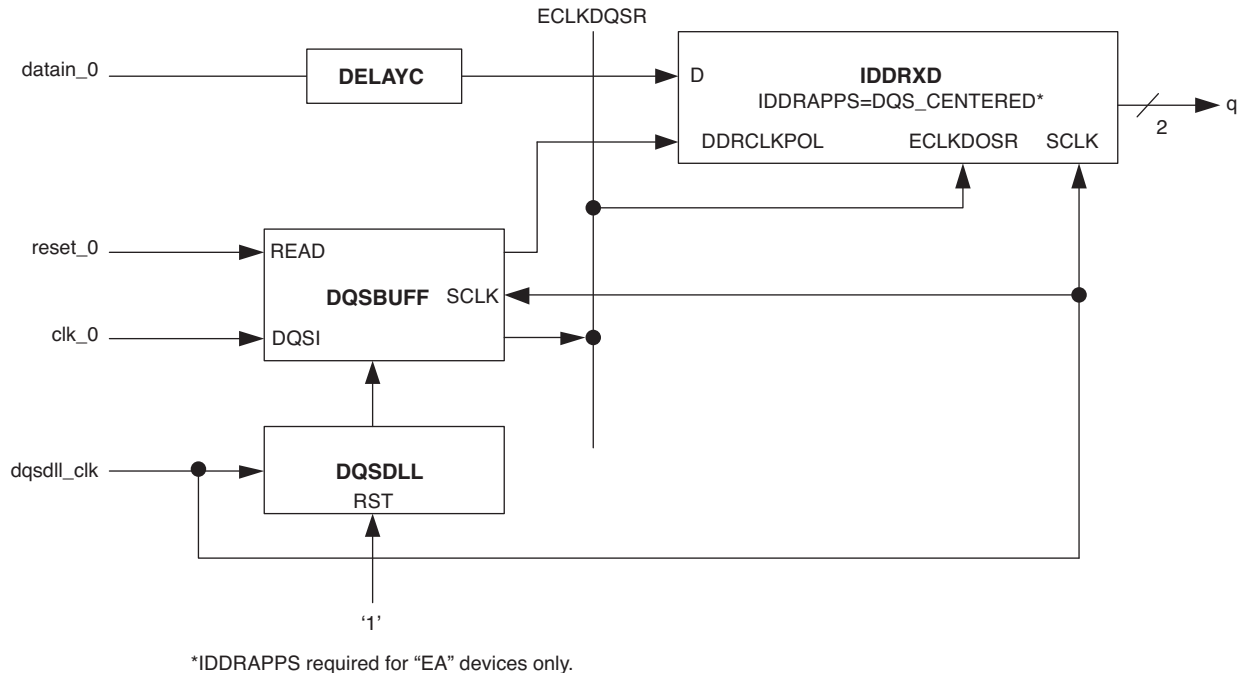
Generic DDR Receive Interface using DQS Lane with Centered External Interface  
Device Support: “E” and “EA” devices

#### Description

This DDR interface uses the DQS and DELAYC to match clock and data delay at the IDDRXD. This interface is useful for small data buses (<11 bits). Since a 90° shift is not required, the DQSDLL is held in reset for this interface.

The user can use any frequency-locked clock for SCLK or the local clock from the input pin. The timing transfer between the ECLKDQSR and SCLK is handled in the hardware through the DDRCLKPOL

**Figure 12-16. GDDR1\_RX.DQS.Centered Interface (“E” and “EA” Devices)**



**Notes:**

1. The DELAYC is only applicable for “EA” devices. For “E” devices, change the DELAYC to DELAYB with a delay value of 7.
2. When retargeting from “E” to “EA” devices, manually update DELYAB to DELAYC, otherwise the software will error out.

**Interface Rules**

- The input clock must use a DQS input pin and all data inputs must be in the same DQS lane.
- The VREF1 for the selected DQS lane must be powered on the board.
- There is only one DQSDLL per side of the device which limits sharing of this interface on a side unless all are running at the same rate.
- The clock net connected to SCLK must use the primary clock tree. It is the user’s responsibility to assign “USE PRIMARY NET<clk>” preference on this SCLK clock net to assign it to the primary clock.
- The pin assignments for data and clock will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.

**GDDR2\_RX.ECLK.Aligned**

Generic DDR Receive Interface with x2 Gearing using ECLK with Aligned External Interface  
Device Support: “E” and “EA” devices

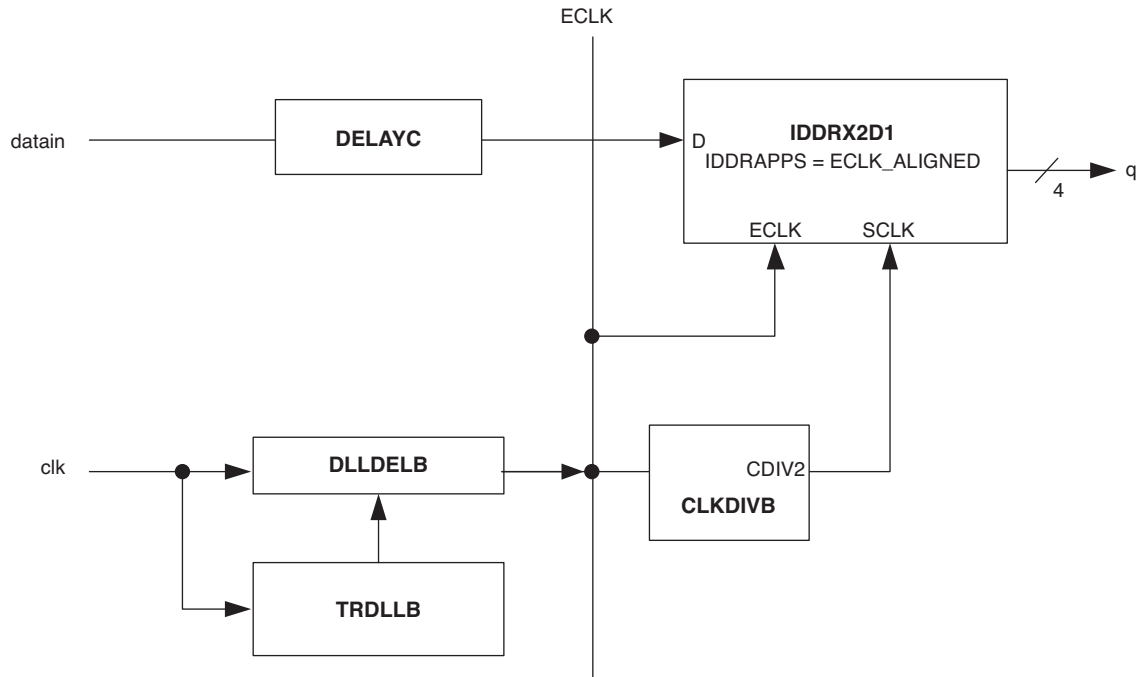
**Description**

This DDR x2 interface uses the ECLK and the TRDLLB to provide a 90° clock shift to center the clock at the IDDRX2D. DELAYC is used to delay data to match the ECLK injection delay. Since this interface uses the ECLK it can be extended to support large data bus sizes for the entire side of the device.

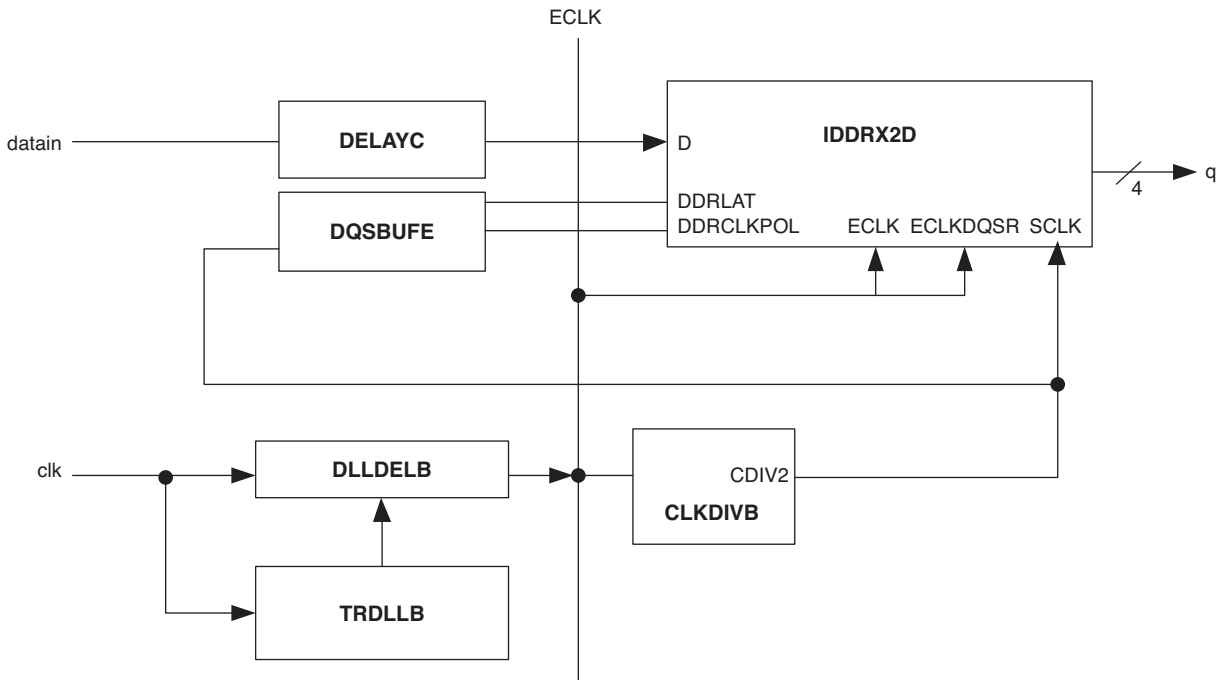
This interface uses x2 gearing with the IDDRX2D element. This requires the use of a CLKDIVB to provide the SCLK which is half the frequency of the ECLK and ECLKDQSR.

Note the difference in interface implementation between the “E” and “EA” devices. “E” devices require the use of a DQSBUFE which is not required on the “EA” devices.

**Figure 12-17. GDDR2\_RX.ECLK.Aligned Interface (“EA” Devices)**



**Figure 12-18. GDDR2\_RX.ECLK.Aligned Interface (“E” Devices)**



**Interface Rules**

- It is recommended that a dedicated GDLL T\_IN is used for the clock input. In “EA” devices, a GPLLT\_IN or PCLK from the top side can also be used for the clock input.
- The clock net output of the CLKDIVB module is connected to SCLK and it must be routed on a primary clock. The “USE PRIMARY NET<clk>” preference must be used on this clock net as well.
- There is only one DLLDELB and one CLKDIV per side of the device (left and right sides) which limits this interface to one clock rate per side or two per device.
- The data/clock pin assignments for “E” devices require following the DQ-DQS group pinout guidelines. See the [Generic DDR Design Guidelines](#) section for details. This is not required for “EA” devices.

**GDDR2\_RX.ECLK.Aligned (No CLKDIV)**

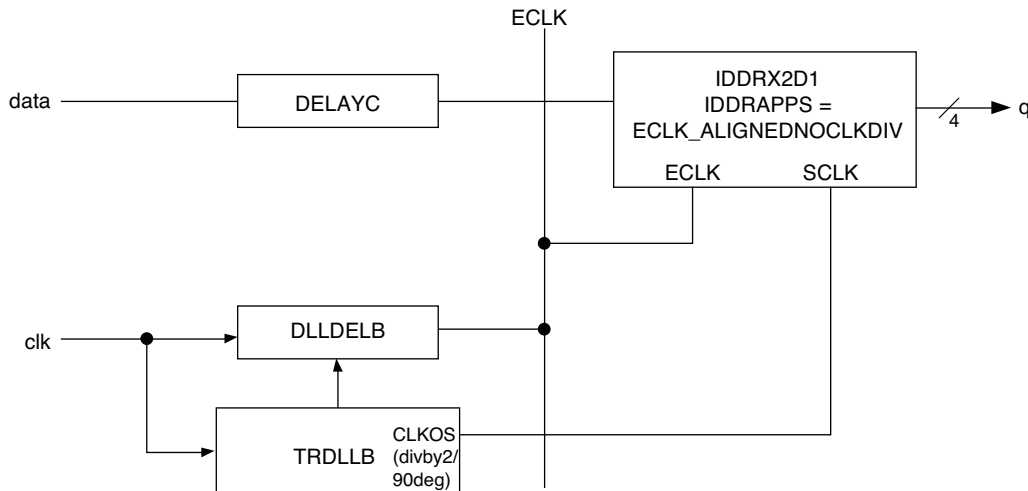
Generic DDR Receive Interface with x2 Gearing using ECLK with Aligned External Interface Device Support: “EA” devices

**Description**

This DDR x2 interface uses the ECLK and the TRDLLB to provide a 90° clock shift to center the clock at the IDDRX2D. DELAYC is used to delay data to match the ECLK injection delay. Since this interface uses the ECLK it can be extended to support large data bus sizes for the entire side of the device.

This interface uses x2 gearing with the IDDRX2D element. The CLKOS output of the TRDLLB can be set up to divide ECLK by 2 to generate the SCLK.

**Figure 12-19. GDDR2\_RX.ECLK.Aligned (No CLKDIV) Interface (“EA” Devices)**



**Interface Rules**

It is recommended that a dedicated TGDLL\_IN pin be used for the clock input. In an “EA” device a GPLLT\_IN or PCLK from the top side can also be used for the clock input.

The clock CLKOS output of the TRDLLB module is connected to SCLK of IDDRX2D1 and it must be routed on a primary clock. The “USE PRIMARY NET<clk>” preference must be used on this clock net as well.

There is only one DLLDELB per side of the device (left and right sides) which limits this interface to one clock rate per side or two per device.

### GDDR2\_RX.ECLK.Centered

Generic DDR Receive Interface with x2 Gearing using ECLK with Centered External Interface  
Device Support: “E” and “EA” devices

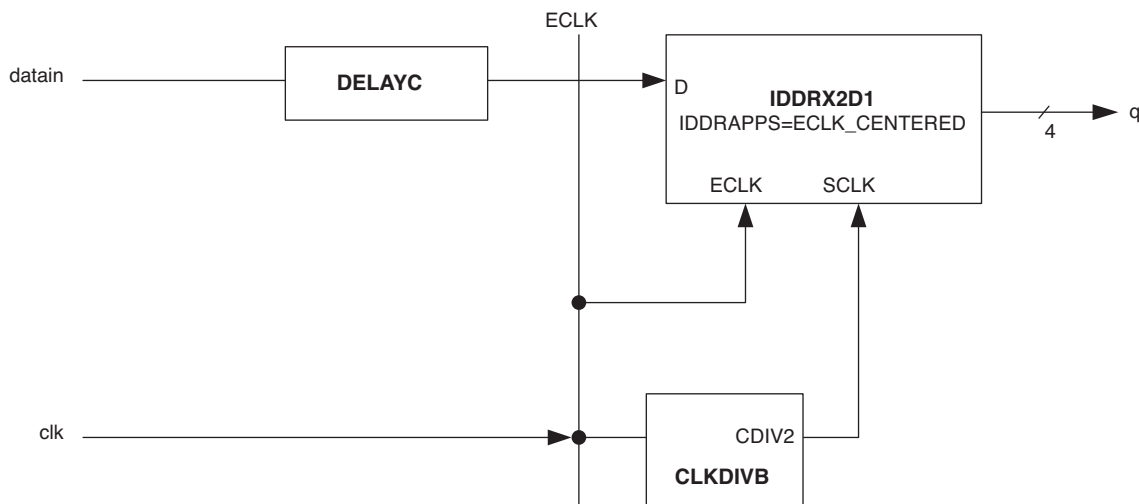
#### Description

This DDR x2 interface uses the ECLK and DELAYC to match clock and data delay at the IDDRX2D. Since this interface uses the ECLK it can be extended to support large data bus sizes for the entire side of the device.

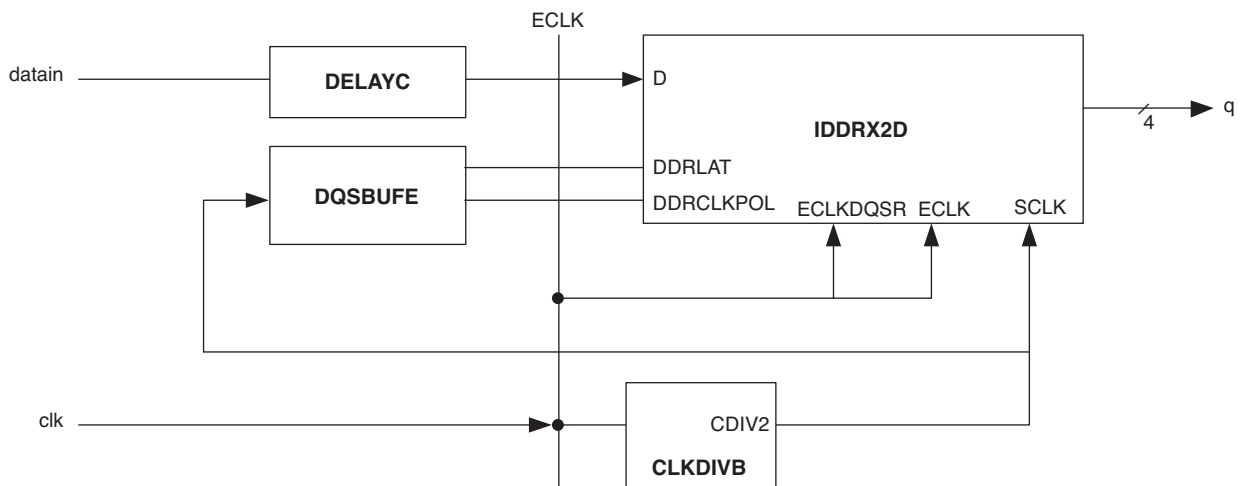
This interface uses x2 gearing with the IDDRX2D element. This requires the use of a CLKDIVB to provide the SCLK which is half the frequency of the ECLK and ECLKDQSR.

Note the difference in interface implementation between the “E” and “EA” devices. “E” devices require the use of a DQSBUFE which is not required on the “EA” devices.

**Figure 12-20. GDDR2\_RX.ECLK.Centered Interface (“EA” Devices)**



**Figure 12-21. GDDR2\_RX.ECLK.Centered Interface (“E” Devices)**



#### Interface Rules

- Input clock port must use a dedicated clock (PCLK) input pin. All the data for the interface must be on the same ECLK tree (same side of the device). In “EA” devices, PLL output in Bypass mode can be connected to the ECLK



as well.

- The clock net connected to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.
- There is only one CLKDIVB per side of the device, so the interface is limited to one per side.
- The data/clock pin assignments for “E” devices require following the DQ-DQS group pinout guidelines. See the [Generic DDR Design Guidelines](#) section for details. This is not required for “EA” devices.

### GDDR2\_RX.DQS.Aligned

Generic DDR Receive Interface with x2 Gearing using a DQS lane with Aligned External Interface  
Device Support: “E” and “EA” devices

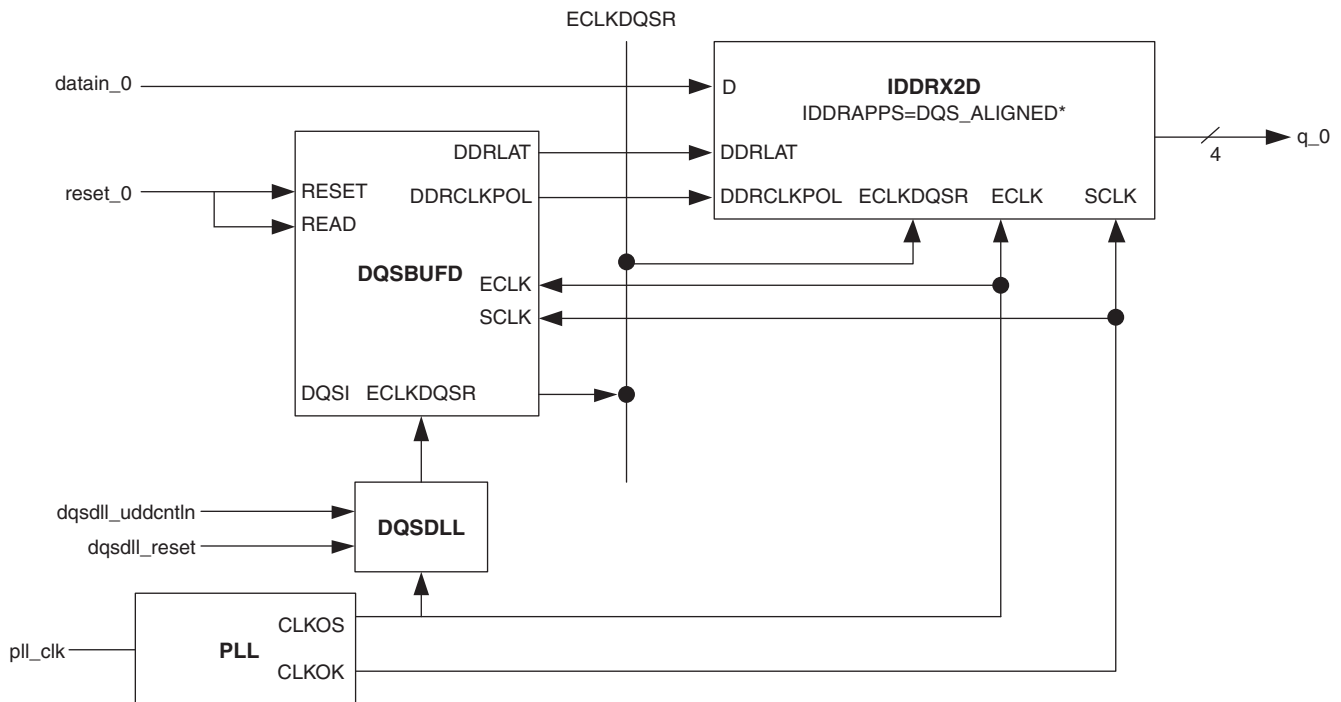
#### Description

This DDR x2 interface uses the DQS and the DQSDLL to provide a 90° clock shift to center the clock at the IDDRXD. This interface uses a DQS lane and is useful for small data buses (<11 bits). There is only one DQSDLL per side of the device which limits sharing of this interface on a side unless all are running at the same rate.

This interface uses x2 gearing with the IDDRX2D element. This requires the use of a CLKDIVB to provide the SCLK which is half the frequency of the ECLK and ECLKDQSR.

This interface requires the use of a secondary input pll\_clk for the PLL input. The PLL output must run at the same rate as the DQSI clk input, but can be arbitrary phase.

**Figure 12-22. GDDR2\_RX.DQS.Aligned Interface (“E” and “EA” Devices)**



\*IDDRAPPS is only required for “EA” devices.

#### Interface Rules

- The input clock must use a DQS input pin and all data inputs must be in the same DQS lane.

- The input pll\_clk must use a GPLLT\_IN or GDLLT\_IN input pin or from a primary clock tree.
- The clock net connected to SCLK must also be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.
- There is only one DQSDLL per side of the device which limits sharing of this interface on a side unless all are running at the same rate.
- The following sequence must be followed when resetting the interface:
  - Assert DQSDLL\_RESET to DQSDLL and RESET to the modules
  - Deassert DQSDLL\_RESET to DQSDLL first
  - Wait for DQSDLL lock to go high
  - Assert DQSDLL\_UDDCNTLN input of DQSDLL for at least four SCLK cycles. See the section [“DQSDLLB” on page 89](#) for the detailed requirements for the DQSDLL\_UDDCNTLN input of DQSDLLB
  - Deassert DQSDLL\_UDDCNTLN
  - Wait for four SCLK cycles, then deassert RESET to the other modules
- The pin assignments for data and clock will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.
- The VREF1 for the selected DQS lane must be powered on the board.

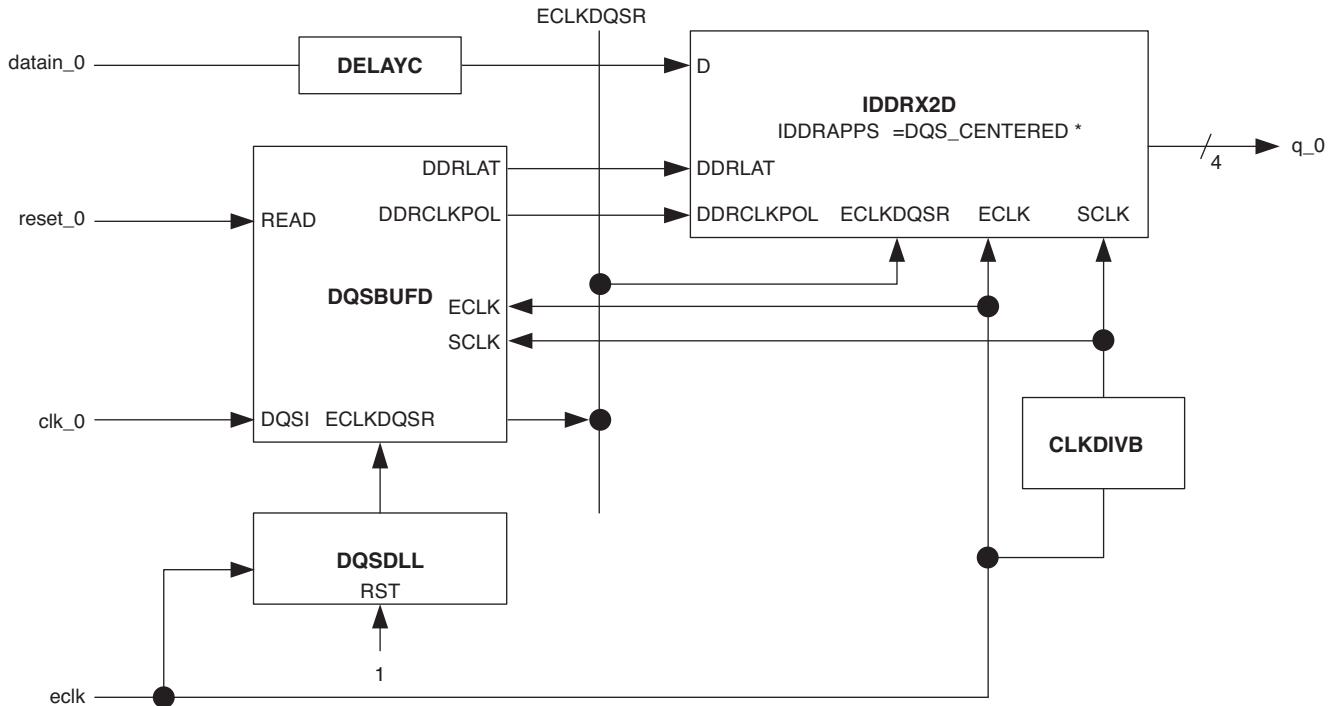
### **GDDR2\_RX.DQS.Centered**

Generic DDR Receive Interface with x2 Gearing using a DQS lane with Centered External Interface  
Device Support: “E” and “EA” devices

#### **Description**

This DDR x2 interface uses the DQS and DELAYC to match clock and data delay at the IDDRX2D. This interface is useful for small data buses (<11 bits). Since a 90° shift is not required, the DQSDLL is held in reset for this interface. An additional refclk input running at the same as the clk input to DQSI is provided to the DQSDLL, ECLK input of the IDDRX2D. It is also used in the CLKDIV module to generate the SCLK which is half the frequency as the input clock.

Figure 12-23. GDDR2\_RX.DQS.Centered Interface (“E” and “EA” Devices)



\*IDDRAPPS is only required for “EA” devices.

Notes:

1. The DELAYC is only applicable for “EA” devices. For “E” devices, change DELAYC to DELAYB and set the value to 7.
2. When retargeting from “E” to “EA”, manually update DELAYB to DELAYC, otherwise the software will error out.

**Interface Rules**

- The input clock must use a DQS input pin and all data inputs must be in the same DQS lane.
- The ECLK clock input must use a dedicated GPLLT\_IN input or PCLK input pin or from a primary clock tree. This second synchronous clock input is used for the DQSDLL, ECLK, and CLKDIV.
- There is only one DLLDELB and one CLKDIV per side of the device (left and right sides) which limits this interface to one clock rate per side or two per device.
- The clock net connected to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.
- The pin assignments for data and clock will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.
- The VREF1 for the selected DQS lane must be powered on the board.

**GDDR2\_RX.ECLK.Dynamic**

Generic DDR Receive Interface with x2 Gearing using ECLK with Centered External Interface and Dynamic Data Delay Control

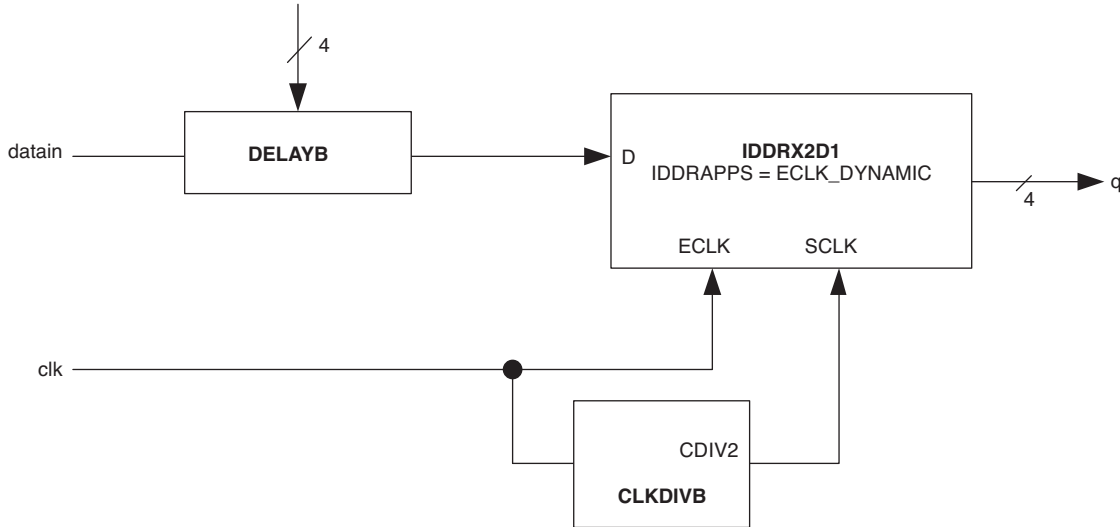
Device Support: “EA” devices

**Description**

This interface uses a DELAYB and ECLK for bit-level control of the alignment. User logic will control the inputs of the DELAYB delay module. The CLKDIV module is used to generate the SCLK which is half the frequency of

ECLK. This interface should only be used when the input clock is centered to the data as this interface does not have phase-shift capability on the clock. This interface is similar to the GDDR2\_RX.ECLK.Centered, but in this version the data delay is controlled dynamically by the user.

**Figure 12-24. GDDR2\_RX.ECLK.Dynamic (“EA” Devices)**



**Interface Rules**

- Input clock port must use a dedicated clock (PCLK) input pin. All the data for the interface must be on the same ECLK tree (same side of the device).
- The clock net connected to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.
- There is only one CLKDIVB per side of the device, so the interface is limited to one per side.

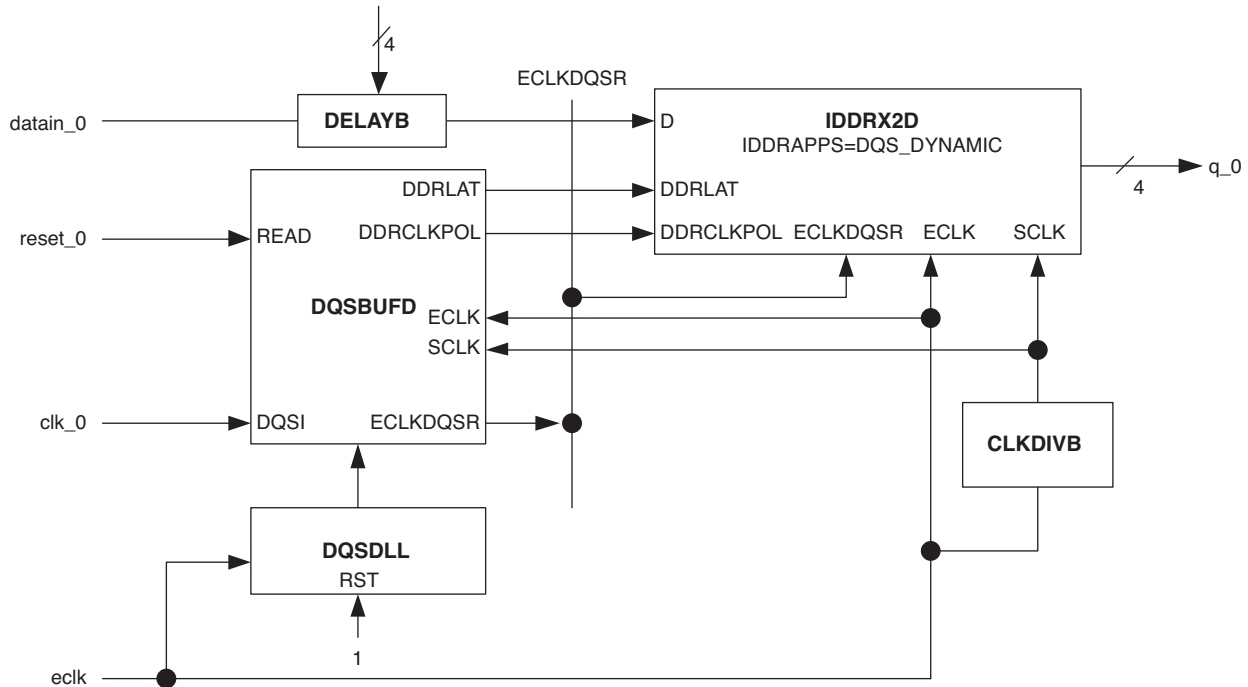
**GDDR2\_RX.DQS.Dynamic**

Generic DDR Receive Interface with x2 Gearing using a DQS lane with Centered External Interface and Dynamic Data Delay Control  
Device Support: “EA” devices

**Description**

This interface uses a DELAYB and DQS lane for bit-level control of the alignment. User logic will control the inputs of the DELAYB delay module. CLKDIV module is used to generate the SCLK which is half the frequency of ECLK. This interface should only be used when the input clock is centered to the data as this interface does not have phase-shift capability on the clock. This interface is similar to the GDDR2\_RX.DQS.Centered, but in this version the data delay is controlled dynamically by the user.

Figure 12-25. GDDR2\_RX.DQS.Dynamic (“EA” Devices)



### Interface Rules

- The input clock must use a DQS input pin and all data inputs must be in the same DQS lane.
- The eclk clock input must use a dedicated GPLLT\_IN input or PCLK input pin. This second synchronous “eclk” input is used for the DQSDLL, ECLK, and CLKDIV.
- The clock net connected to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.
- There is only one DLLDELB and one CLKDIV per side of the device (left and right sides) which limits this interface to one clock rate per side or two per device.
- The pin assignments for data and clock will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.

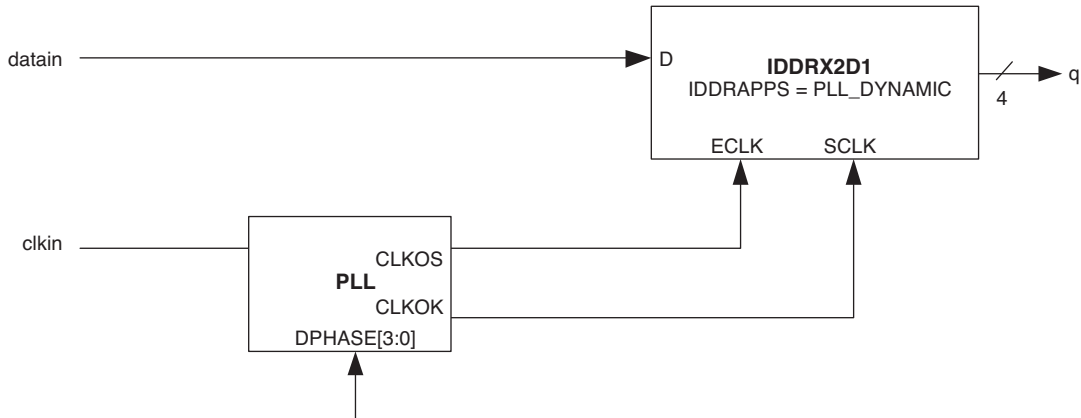
### GDDR2\_RX.PLL.Dynamic

Generic DDR Receive Interface with x2 Gearing using ECLK with Dynamic control on ECLK phase using PLL  
Device Support: EA devices

#### Description

This interface uses a PLL to delay the ECLK for bus-level control of the alignment. The benefit of the PLL is that an entire period of delay is provided. User logic will control the DPHASE input to the PLL.

Figure 12-26. GDDR2\_RX.PLL.Dynamic (“EA” Devices)



**Interface Rules**

- The input clock must use a dedicated GPLLT\_IN clock input pin. The clock net connected to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.

**GOREG\_TX.SCLK**

Generic SDR Transmit Interface using SCLK  
Device Support: “E” and “EA” devices

**Description**

This is a generic interface for SDR data and a forwarded clock. The ODDR used for the clock balances the clock path to match the data path. A PLL can also be used to clock the ODDRXD to phase shift the clock to provide a precise clock-to-data output.

On “E” devices, the sides (left and right) need to pass through a DQSBUFG before going to the ODDRXD element. The top does not require the DQSBUFG and can take SCLK directly.

The “EA” device does not require a DQSBUFG block.

Figure 12-27. GOREG\_TX.SCLK Interface (“EA” Devices)

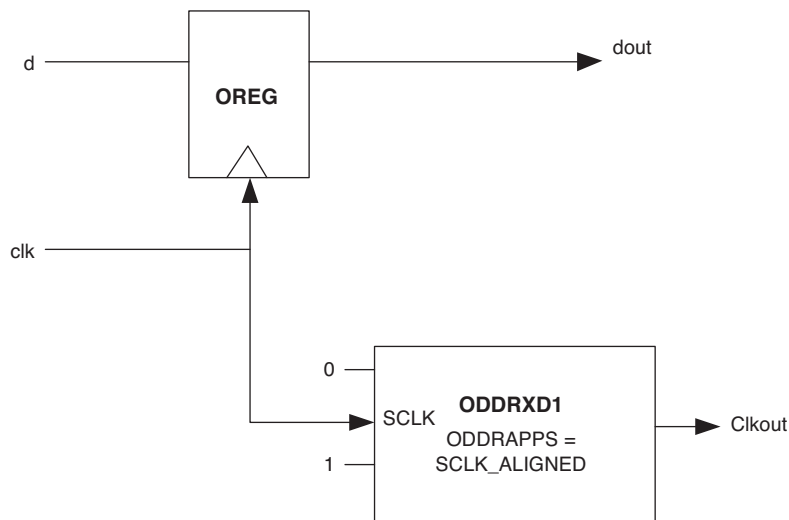


Figure 12-28. GOREG\_TX.SCLK Interface (“E” Devices, Top)

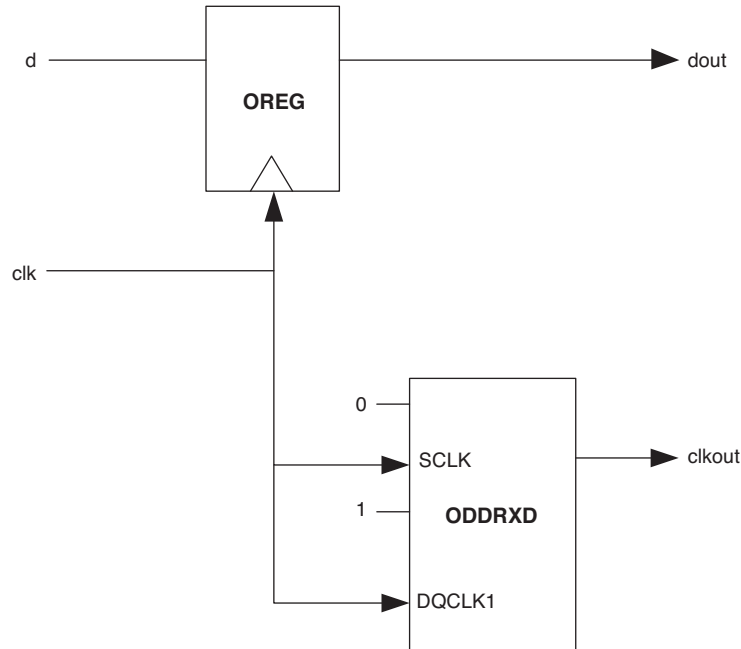
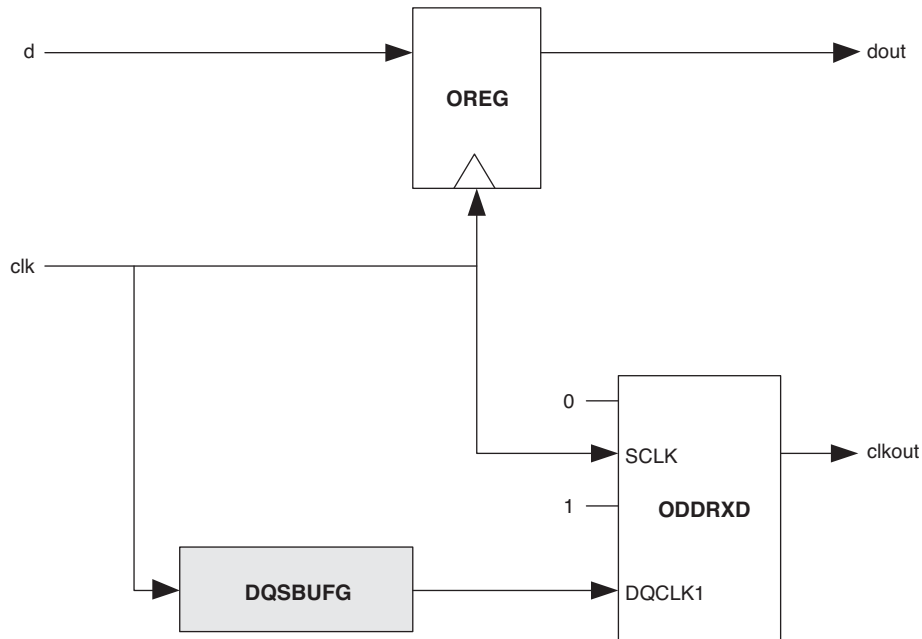


Figure 12-29. GOREG\_TX.SCLK Interface (“E” Devices, Left/Right)



**Interface Rules**

- SCLK must be routed on either primary or secondary clock resources using the USE PRIMARY NET<clk> or USE SECONDARY preferences.

**GDDR1\_TX.SCLK.Centered**

Generic DDR Transmit Interface using SCLK with Centered External Interface  
Device Support: “E” and “EA” devices

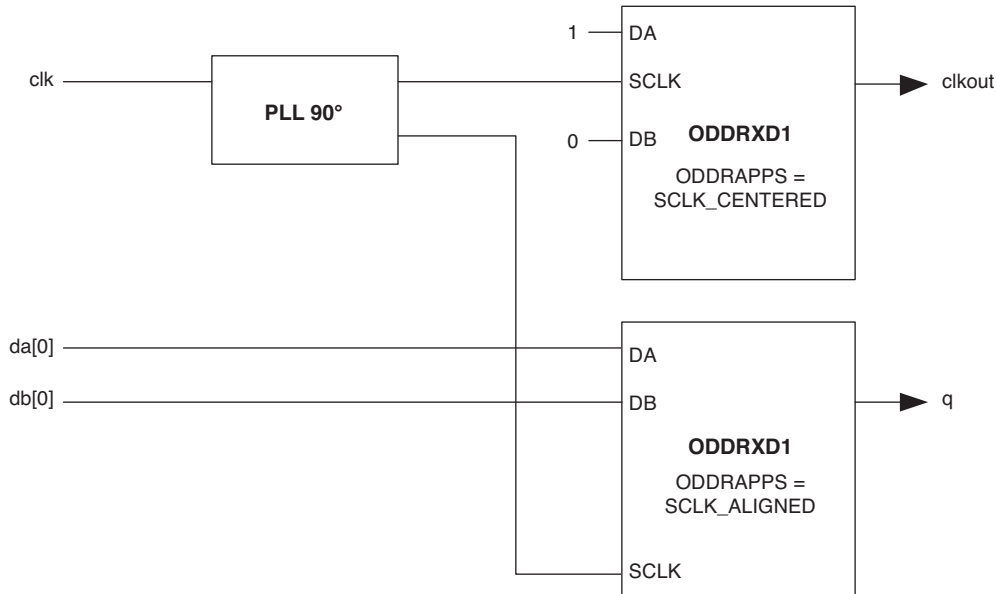
**Description**

This output DDR interface provides clock and data that are pre-centered using a PLL and two SCLKs.

On “E” devices, the left and right sides need to pass through a DQSBUFG before going to the ODDRXD element. The top side of the “E” device does not require the DQSBUFG and can take SCLK directly. When using the DQSBUFG the clock output will need to be in a different DQS lane than the data since there is only one DQSBUFG per lane.

The “EA” device does not require the DQSBUFG.

**Figure 12-30. GDDR1\_TX.SCLK.Centered Interface (“EA” Devices)**



**Figure 12-31. GDDR1\_TX.SCLK.Centered Interface (“E” Devices, Top)**

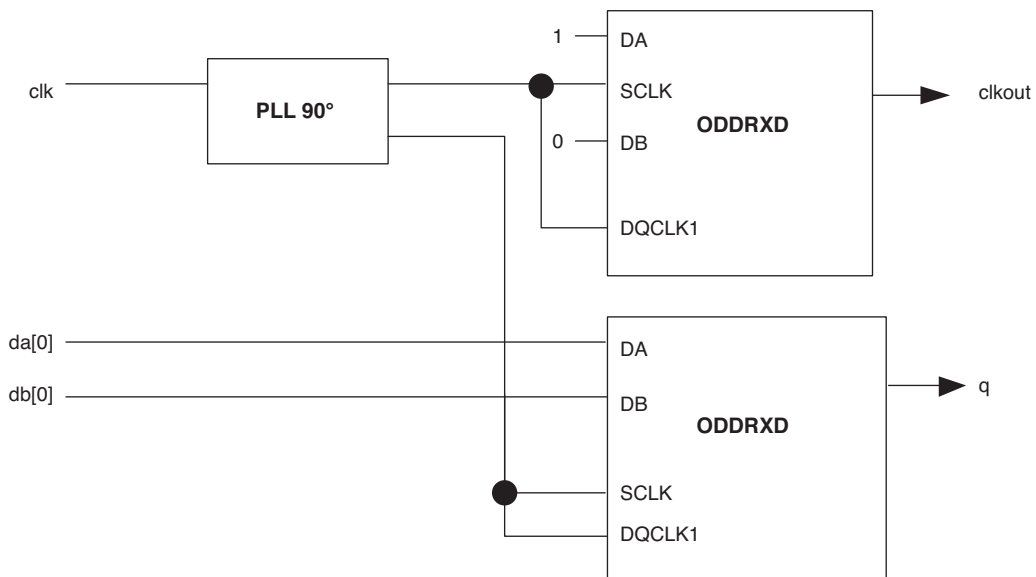
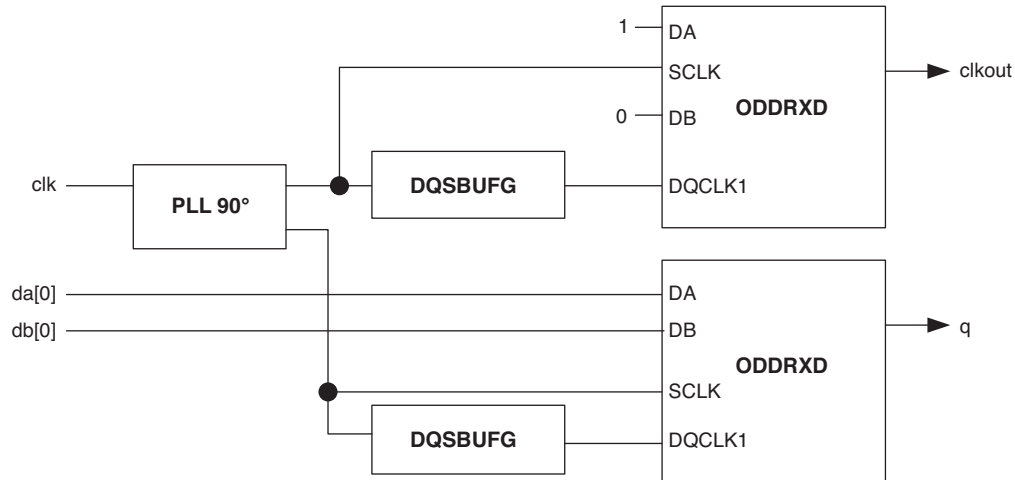




Figure 12-32. GDDR1\_TX.SCLK.Centered Interface (“E” Devices, Left/Right)



**Interface Rules**

- On “E” devices, the clock and data outputs need to be in different DQS lanes on the left and right sides since there is only one DQSBUF per lane. Clock and data outputs can use the same DQS lane on top. Clock and data outputs cannot use the DQS site. They must use the DQ site.
- SCLK and 90° shifted SCLK should be assigned to a primary clock pin using the “USE PRIMARY NET<clk>” preference.
- The pin assignments for data and clock will require following the DQ-DQS lane assignments for the “E” device. Refer to the [Generic DDR Design Guidelines](#) section for details. “EA” devices do not have this requirement. The clock pin to the PLL path must be routed on a dedicated clock route. A dedicated GPLLT\_IN pin must be used for input of this clock.

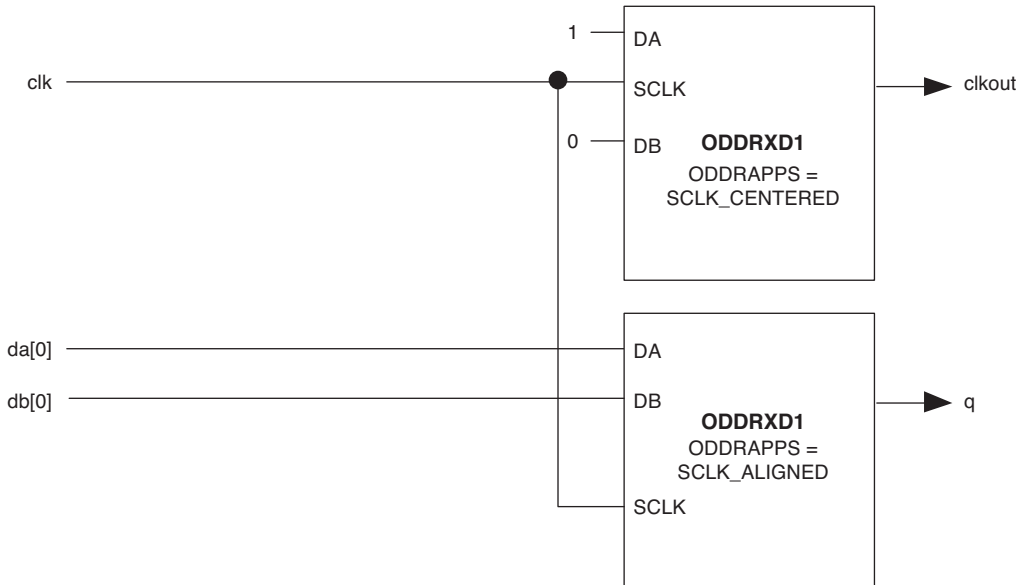
**GDDR1\_TX.SCLK.Aligned**

Generic DDR Transmit Interface using SCLK with Aligned External Interface  
 Device Support: “E” and “EA” devices

**Description**

This output DDR interface provides clock and data that are aligned using a single SCLK.

Figure 12-33. GDDR1\_TX.SCLK.Aligned Interface (“EA” Devices)



“E” devices require the use of DQSBUFG on the left and right sides of the device. If the clock and data bus can fit in the same DQS lane then a single DQSBUFG is all that is needed (<10 bits). For a wider data bus (>0 bits) it is required to use a DQSBUFG for clock and data and assign the clock to a different DQS lane.

Figure 12-34. GDDR1\_TX.SCLK.Aligned Interface (“E” Devices, Top Side)

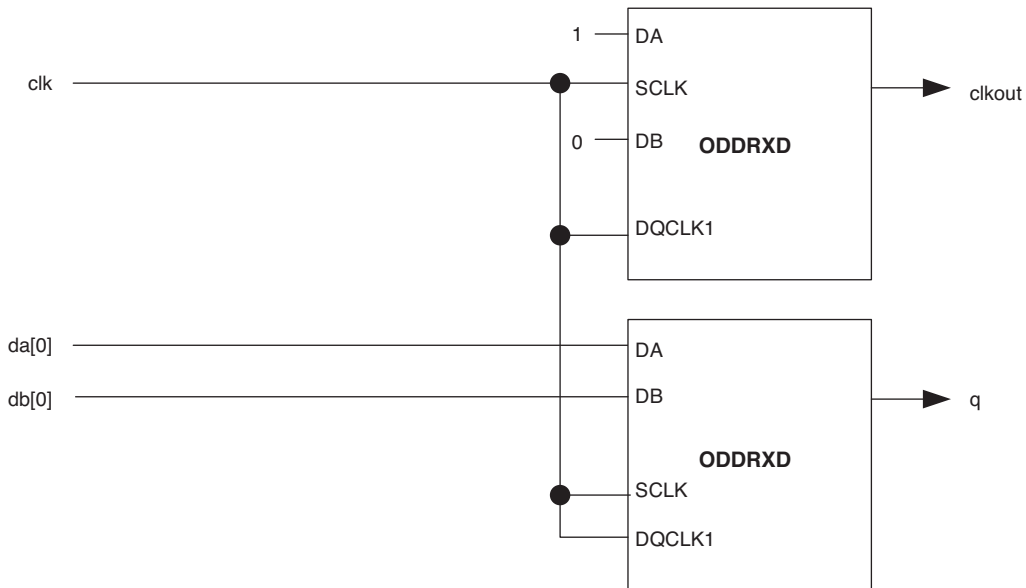


Figure 12-35. GDDR1\_TX.SCLK.Aligned Interface (“E” Devices, Left/Right Sides) < 10 Bits

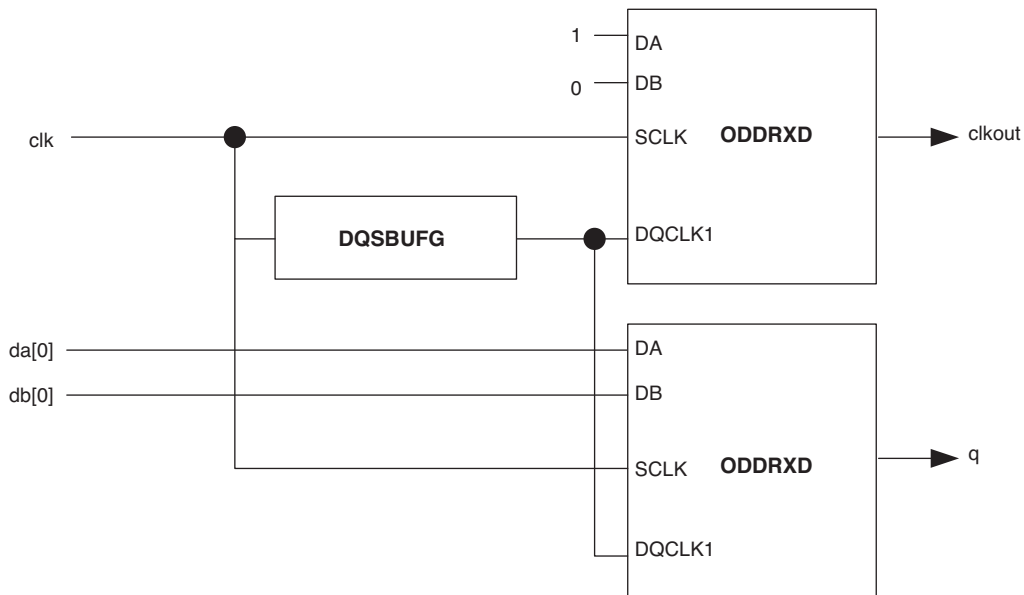
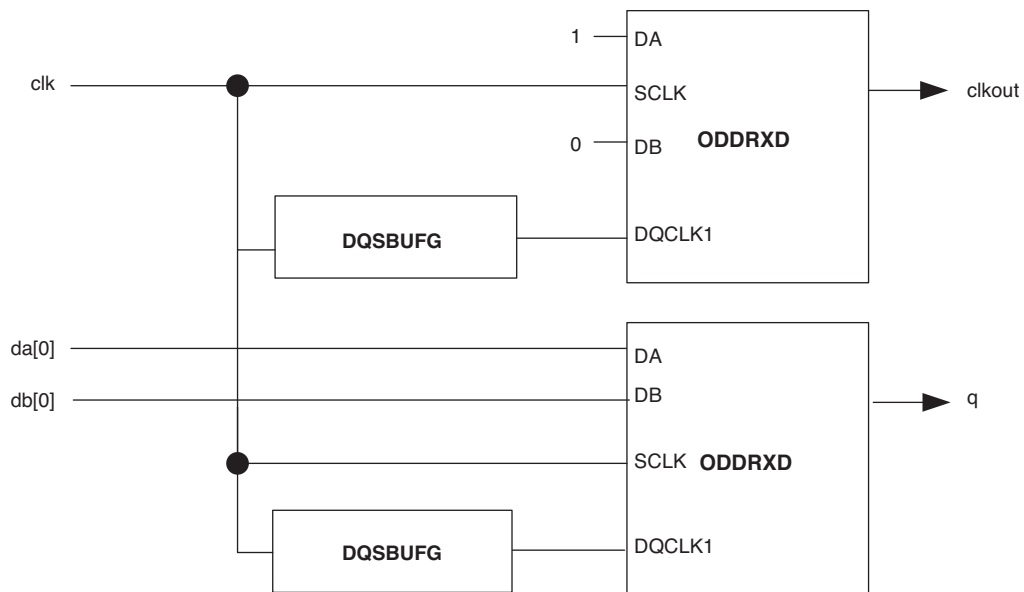


Figure 12-36. GDDR1\_TX.SCLK.Aligned Interface (“E” Devices, Left/Right Sides) > 10 Bits



**Interface Rules**

- On “E” devices, the clock and data outputs need to be in different DQS lanes on the left and right sides since there is only one DQSBUFG per lane. Clock and data outputs can use the same DQS lane on top. Clock and data outputs cannot use the DQS site. They must use the DQ site.
- The clock to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.
- The pin assignments for data and clock will require following the DQ-DQS lane assignments for “E” devices. Refer to the [Generic DDR Design Guidelines](#) section for details. “EA” devices do not have this requirement.

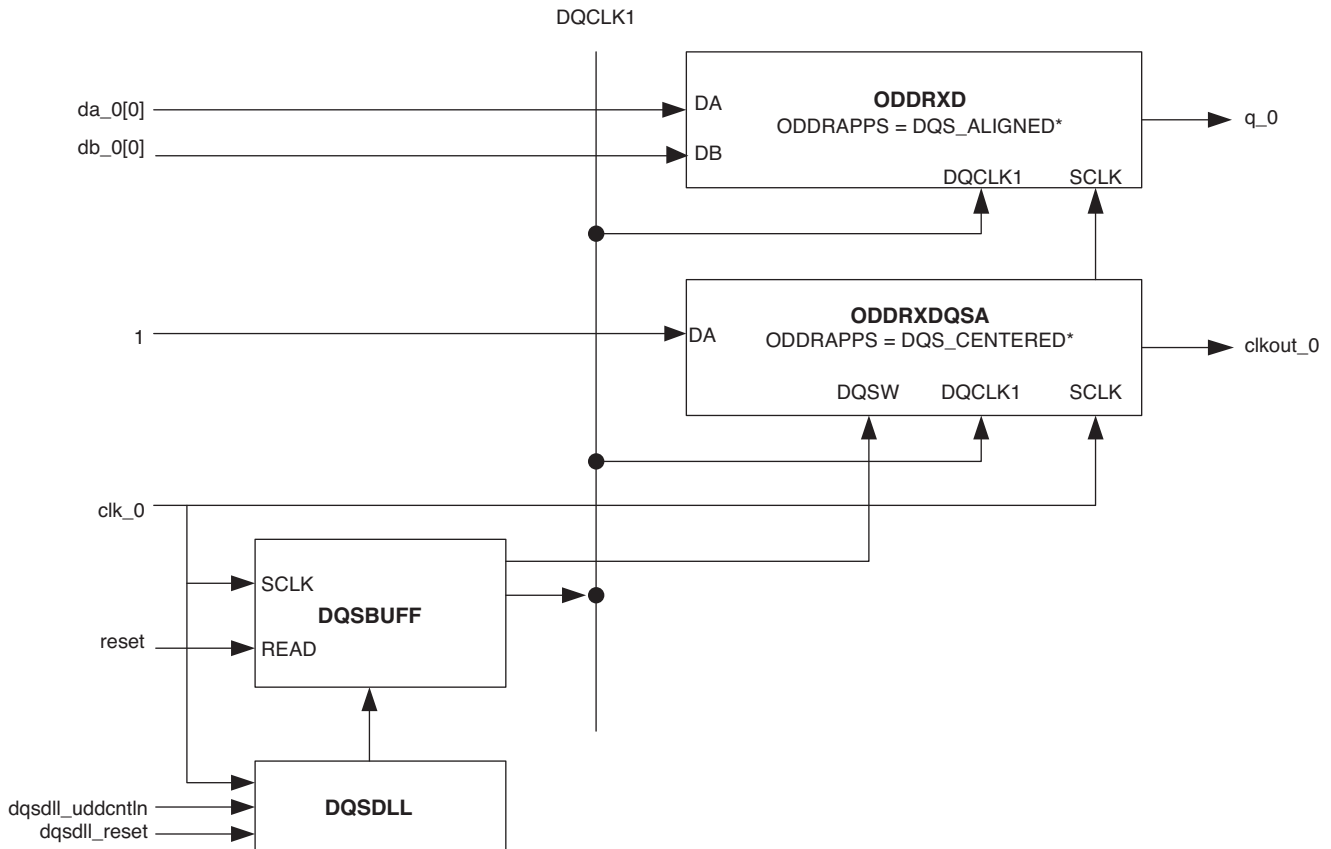
## GDDR\_X1\_TX.DQS.Centered

Generic DDR Transmit Interface using DQS Lane with Centered External Interface  
Device Support: “E” and “EA” devices

### Description

This output DDR x1 interface provides clock and data that is pre-centered using a DQSDLL and ODDRXDQSA. This is the same as the GDDR\_X1\_TX.SCLK.Centered, but does not require a PLL. This interface can also be used multiple times using the same DQSDLL. This interface should be used for narrow data buses (<11 bits wide)

Figure 12-37. GDDR\_X1\_TX.DQS.Centered Interface (“E” and “EA” Devices)



\*ODDRAPPS required only for “EA” devices.

### Interface Rules

- The pin assignments for data and clock will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.
- There is only one DQSDLL per side of the device. One of these interfaces can be placed in each DQS group, but they all need to run at the same rate for that side.
- The following sequence must be followed when resetting the interface:
  - Assert DQSDLL\_RESET to DQSDLL and RESET to the modules
  - Deassert DQSDLL\_RESET to DQSDLL first
  - Wait for DQSDLL lock to go high
  - Assert DQSDLL\_UDDCNTLN input of DQSDLL for at least four SCLK cycles. See the section “[DQSDLLB](#)” on page 89 for the detailed requirements for the DQSDLL\_UDDCNTLN input of DQSDLLB

- Deassert DQSDLL\_UDDCNTLN
- Wait for four SCLK cycles, then deassert RESET to the other modules
- “clk\_0” must use the primary clock tree. It is the user’s responsibility to assign “USE PRIMARY NET<clk>” preference on this net to assign it to the primary clock.
- This interface cannot use the LVDS25 IO\_TYPE for the clkout from the DQSI.

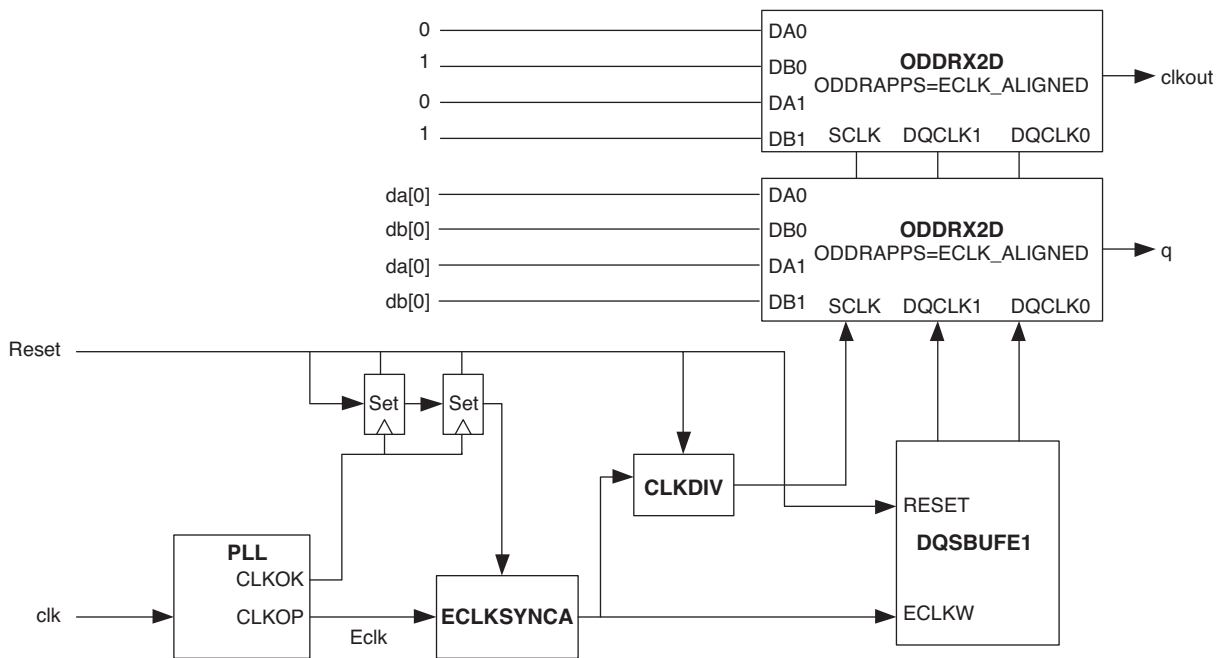
### GDDR2\_TX.Aligned

Generic DDR x2 Transmit Interface with Aligned External Interface  
Device Support: “EA” devices only

#### Description

This output DDR x2 interface provides clock and data that are aligned. A PLL is used to generate ECLK. A CLKDIV is use to generate the SCLK which is half the frequency of the ECLK. The PLL CLKOK can also be used to generate the SCLK. Additional soft logic is required for this interface to work as expected. This logic is used to control the ECLKSYNCA to create the proper relationship between ECLK and SCLK.

**Figure 12-38. GDDR2\_TX.Aligned Interface (“EA” Devices)**



#### Interface Rules

- Clock input must use a dedicated GPLLT\_IN input pin or from a primary clock tree.
- The pin assignments for data and clock will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.
- The additional soft logic required for this interface is included in the module generated using IPexpress.
- Clock to the reset flops should be at least half the speed or slower than the ECLK.
- The clock to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.

### GDDR2\_TX.DQSDLL.Centered

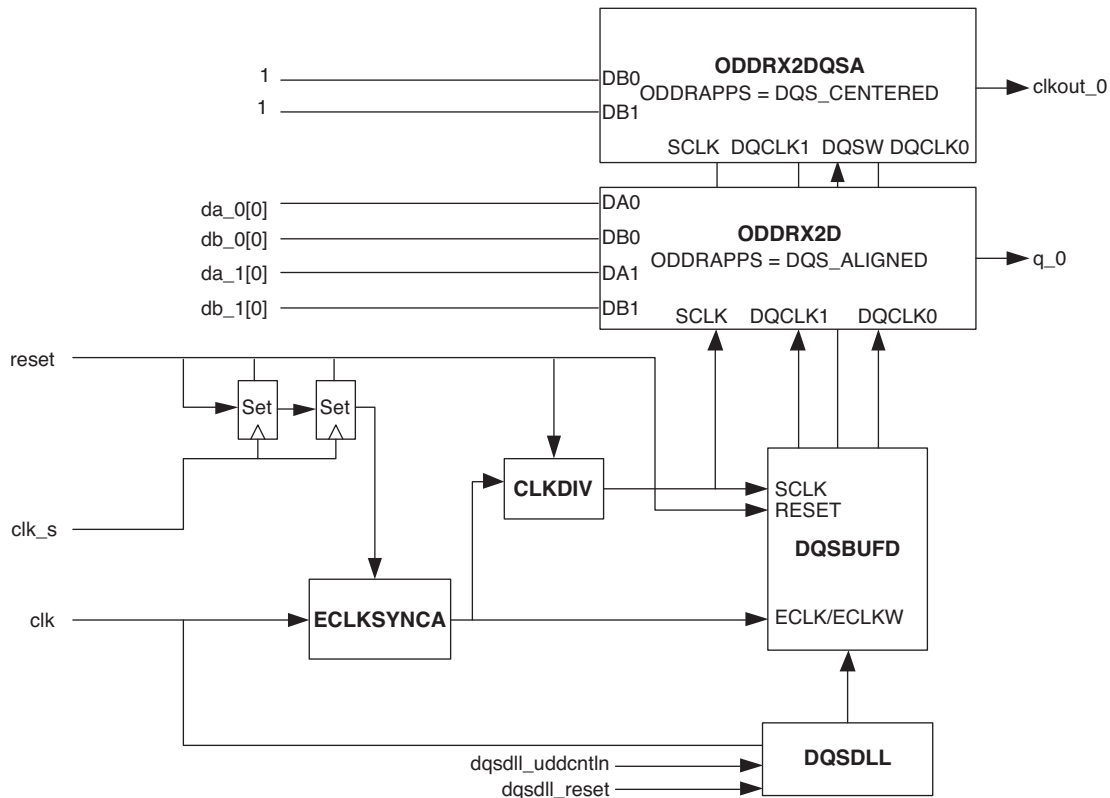
Generic DDR x2 Transmit Interface with Centered External Interface using DQSDLL

Device Support: “EA” devices

**Description**

This output DDR x2 interface provides a clock and data that are centered. This interface uses a DQSDLL along with the DQSBUFD to generate the 90° delayed clock used to generate the clock output. ODDR2DQSA module is used for clock generation. A CLKDIV is use to generate SCLK which is half the frequency of the ECLK. Additional logic is needed to control the ECLKSYNCA to create the proper relationship between ECLK and SCLK.

**Figure 12-39. GDDR2\_TX.DQSDLL.Centered Interface (“EA” Devices)**



**Interface Rules**

- Clock inputs must come in from a primary clock tree.
- The pin assignments for data and clock will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.
- The additional logic required for this interface is included in the module generated using IPexpress.
- Clock to the reset flops should be at least half the speed or slower than the ECLK.
- There is only one DQSDLL per side of the device. One of these interfaces can be placed in each DQS group, but they all need to run at the same rate for that side.
- The following sequence must be followed when resetting the interface:
  - Assert DQSDLL\_RESET to DQSDLL and RESET to the modules
  - Deassert DQSDLL\_RESET to DQSDLL first
  - Wait for DQSDLL lock to go high
  - Assert DQSDLL\_UDDCNTLN input of DQSDLL for at least four SCLK cycles. See the section “[DQSDLLB](#)” on page 89 for the detailed requirements for the DQSDLL\_UDDCNTLN input of DQSDLLB
  - Deassert DQSDLL\_UDDCNTLN
  - Wait for four SCLK cycles, then deassert RESET to the other modules

- The clock to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.
- CLKOUT must be assigned to a DQS pin. The DQS pins do not support True LVDS outputs, hence CLKOUT cannot use LVDS IO\_TYPE.

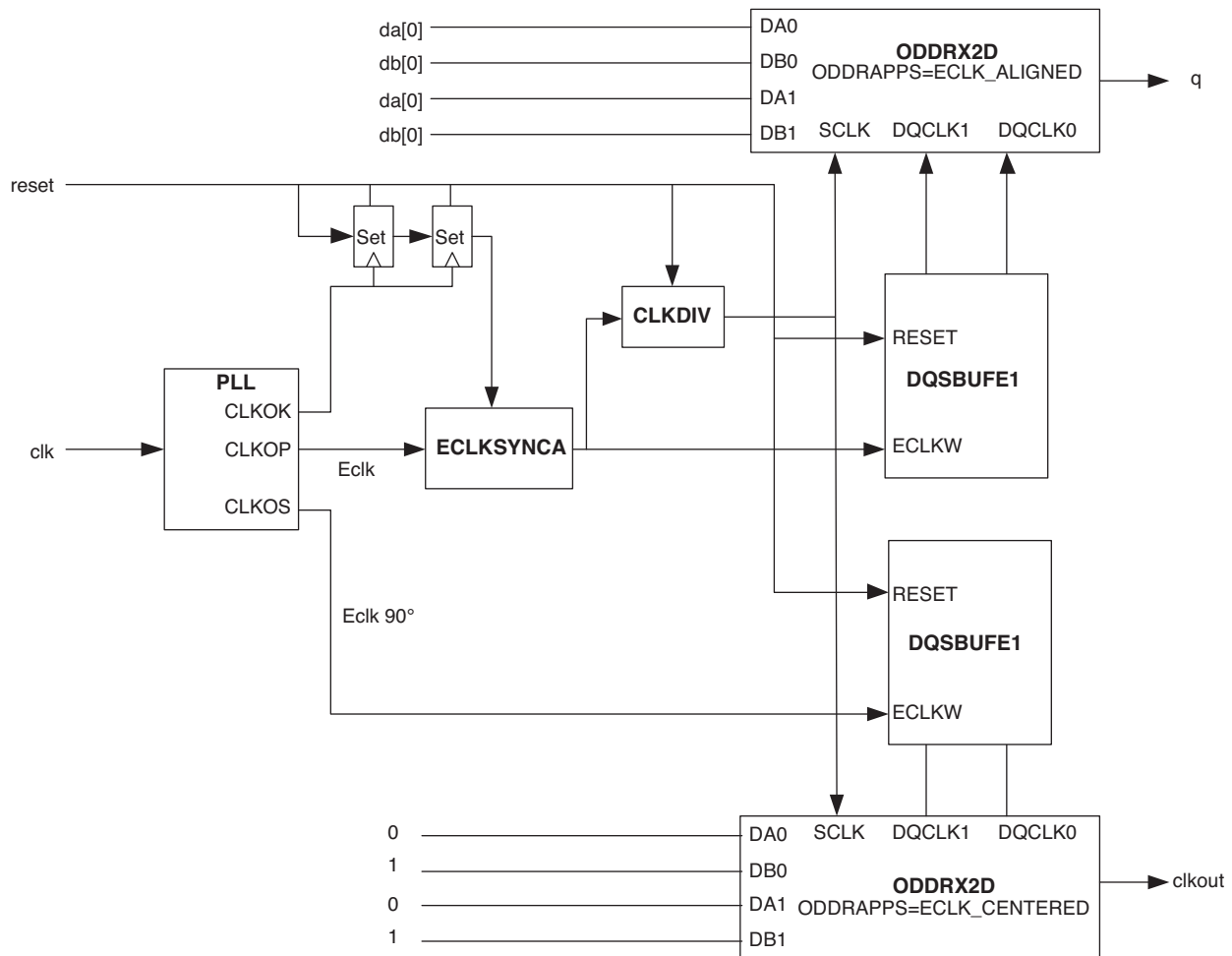
### GDDR2\_TX.PLL.Centered

Generic DDR x2 Transmit Interface with Centered External Interface using PLL  
Device Support: “EA” devices only

#### Description

This output DDR x2 interface provides a clock and data that are centered. This interface uses a PLL to generate the 90° phase shift required for the clock. A CLKDIV is used to generate SCLK which is half the frequency of the ECLK. Additional logic is required to control the ECLKSYNCA to create the proper relationship between ECLK and SCLK.

**Figure 12-40. GDDR2\_TX.PLL.Centered (“EA” Devices)**



#### Interface Rules

- Clock input must use a dedicated GPLLT\_IN input pin or from a primary clock tree.
- The pin assignments for data and clock will require following the DQ-DQS lane assignments. Refer to the [Generic DDR Design Guidelines](#) section for details.

- The clock to SCLK must be routed on a primary routing resource using the “USE PRIMARY NET<clk>” preference.
- The additional soft logic required for this interface is included in the module generated using IPexpress.
- Clock to the reset flops should be at least half the speed or slower than the ECLK.

### 7:1 LVDS Implementation

It is recommended that the [7:1 LVDS Video Interface Reference Design](#) provided on the lattice web site be used to implement all 7:1 LVDS designs.

## Generic DDR Design Guidelines

This section describes the various design guidelines used for building generic high-speed DDR interfaces in LatticeECP3 FPGAs. In addition to these guidelines, it is also necessary to follow the interface rules for each interface type as described above.

### Placement Guidelines for High-Speed DDR Interfaces

The following clock and data placement guidelines should be followed for high-speed design requirements. The software will place the clock and data as specified by the user, but in order to achieve higher speeds the user must follow the placement rules listed in Table 12-6 for each interface type.

It is required that all clocks used to clock the DDR Interfaces use a dedicated clock path. No general routing should be used to route the clock pin. General routing used for a clock path will cause duty cycle distortion as well as limit the operational frequency of the interface. It is the responsibility of the user to assign clock inputs to dedicated clock pins and use preferences such as “USE PRIMARY NET<clk>” to route clock nets on dedicated clock paths.

Table 12-6 lists the various high-speed interfaces and the placement required for the clock and data.

**Table 12-6. Pin Placement Guidelines for High-Speed Interfaces**

DDR Interface	LatticeECP3 “EA” Devices			LatticeECP3 “E” Devices		
	DATA	CLK	CLK PIN	DATA	CLK	CLK PIN
GDDR1_RX.SCLK.Aligned	L/R/T	L/R	GDLLT_IN	N/A	N/A	N/A
	L/R/T	L/R	GPLL_T_IN	N/A	N/A	N/A
	L/R/T	T	PCLK	N/A	N/A	N/A
GDDR1_RX.SCLK.Centered	L/R/T	L/R/T	PCLK	N/A	N/A	N/A
	L/R/T	L/R/T	GPLL_T_IN	N/A	N/A	N/A
GDDR1_RX.ECLK.Aligned	N/A	N/A	N/A	L/R	L/R	GDLLT_IN
GDDR1_RX.ECLK.Centered	N/A	N/A	N/A	L/R	L/R	PCLK
GDDR1_RX.DQS.Aligned	L/R/T	L/R/T	DQS	L/R	L/R	DQS
GDDR1_RX.DQS.Centered	L/R/T	L/R/T	DQS	L/R	L/R	DQS
GDDR2_RX.ECLK.Aligned	L/R/T	L/R	GDLLT_IN	L/R	L/R	GDLLT_IN
	L/R/T	L/R	GPLL_T_IN	N/A	N/A	N/A
	L/R/T	T	PCLK	N/A	N/A	N/A
GDDR2_RX.ECLK.Centered	L/R	L/R	PCLK	L/R	L/R	PCLK
	L/R/T	L/R	GPLL_T_IN	N/A	N/A	N/A
GDDR2_RX.DQS.Aligned	L/R	L/R	DQS	L/R	L/R	DQS
GDDR2_RX.DQS.Centered	L/R	L/R	DQS	L/R	L/R	DQS
GDDR2_RX.ECLK.Dynamic	L/R	L/R	PCLK	N/A	N/A	N/A
GDDR2_RX.DQS.Dynamic	L/R	L/R	DQS	N/A	N/A	N/A



**Table 12-6. Pin Placement Guidelines for High-Speed Interfaces (Continued)**

DDR Interface	LatticeECP3 “EA” Devices			LatticeECP3 “E” Devices		
	DATA	CLK	CLK PIN	DATA	CLK	CLK PIN
GDDR2_RX.PLL.Dynamic	L/R	L/R	GPLL_T_IN	N/A	N/A	N/A
	L/R/T	L	High Speed Bridge <sup>2,3</sup>	N/A	N/A	N/A
GDDR2_RX.ECLK.DR	L/R	L/R	GPLL_T_IN	N/A	N/A	N/A
GDDR1_TX.SCLK.Centered	L/R/T	L/R/T	ANY	L/R	L/R	ANY
GDDR1_TX.SCLK.Aligned	L/R/T	L/R/T	ANY	L/R	L/R	ANY
GDDR1_TX.DQS.Centered	L/R/T	L/R/T	DQS	L/R	L/R	DQS
GDDR2_TX.ECLK.Aligned	L/R	L/R	ANY	L/R	L/R	ANY
GDDR2_TX.Centered (DQS)	L/R	L/R	DQS	L/R	L/R	DQS
GDDR2_TX.Centered (PLL)	L/R	L/R	ANY	L/R	L/R	ANY

1. L, R and T refer to “Left”, “Right” and “Top” sides of the device.
2. The high-speed clock bridge can be accessed by using the “USE EDGE2EDGE < clk> “software preference. For preference details please see “ispLEVER Help” in the software.
3. High-speed bridge is only available on “EA” devices.
4. Top-side DDR is not supported on “E” devices.

### High-Speed Clock Bridge (“EA” Devices)

The high-speed clock bridge is available only on “EA” devices on the GDDR2.RX.PLL\_Dynamic interface. The bridge enables a clock to route to a single edge clock or multiple edge clocks on the device using a three-way (left/right/top) bridge. It can only be used on clocks coming in from the left side dedicated GPLL\_T pin or PLL output. The software preference EDGE2EDGE is used to enable this route.

For example, USE EDGE2EDGE < pllin\_c>; where “pllin\_c” is the clock net coming from the dedicated GPLL\_T pin.

When this preference is placed on the clock coming in on the left side GPLL\_T pin, this clock will be connected to the one of the ECLK on the left, right and top sides using a dedicated route. This ECLK on all three sides cannot be used for any other ECLK function. User will have to make the following changes to the GDDR2.RX.PLL\_Dynamic module generated by IPExpress to incorporate the ECLKBRIDGE.

- A CLKDIV should be used to generate SCLK instead of using PLL CLKOK. The CLKOS output of the PLL should be connected to the input of the CLKDIV module. Output of CLKDIV is used as SCLK.
- User will have to instantiate 2 ECLKSYNC modules to be used on either side of the device. Both of them should be connected to the CLKOS of the PLL used as ECLK.
- An EDGE2EDGE preference should be assigned to the CLKOS output of the PLL to be used as ECLKBRIDGE.

### DQ-DQS Grouping Rules

Due to differences in architecture between the LatticeECP3 “E” and “EA” devices, the DQ-DQS grouping that is required for some interfaces in “E” devices is not required for “EA” devices. It is necessary to use the DQS grouping structure to group pins when either of the DQSBUF/DQSBUFF/DQSBUFG modules is used in an interface. Refer to the section [High-Speed DDR Interface Details](#) to see the requirements for each device.

Below are some of the rules to be followed when locking DQS groups. ispLEVER will check for these rules during MAP and Place and Route.

- Each DQS pin has a DQSBUF block which spans across 12 pins including the DQS and DQS# pins. Each DQS-BUF sends out control logic to these pins.
- The DQS# I/O logic registers cannot be used to implement DDR registers. DQS pins can be used for IDDR implementation only. ODDR cannot be assigned to a DQS pin.

- The DQS and DQS# I/O logic registers can be used to implement SDR input and output registers. If the DQS-BUF of that DQS group is used then it cannot be used for SDR functions unless the same clock going to the DQSBUF is used to clock the SDR register at the DQS/DQS# site.
- An IDDRX element used for a generic DDR interface cannot be mixed in a DQS group with an ODDRX element used for implementing a DDR memory interface.
- Similarly, an ODDRX element used for a generic DDR interface cannot be mixed in a DQS group with an IDDRX element used for a DDR memory interface.
- The ODDRXD and ODDRX2D elements in a given DQS group cannot share the same DQSBUF and therefore cannot be placed together within the same DQS group.
- The upper left corner of the LatticeECP3 device has a DQS group without a DQS pin. This group of I/Os does not have a DQS function. It is recommended to use this group for pins in non-DQS interfaces.
- See Table 12-13 to see the availability on each side.

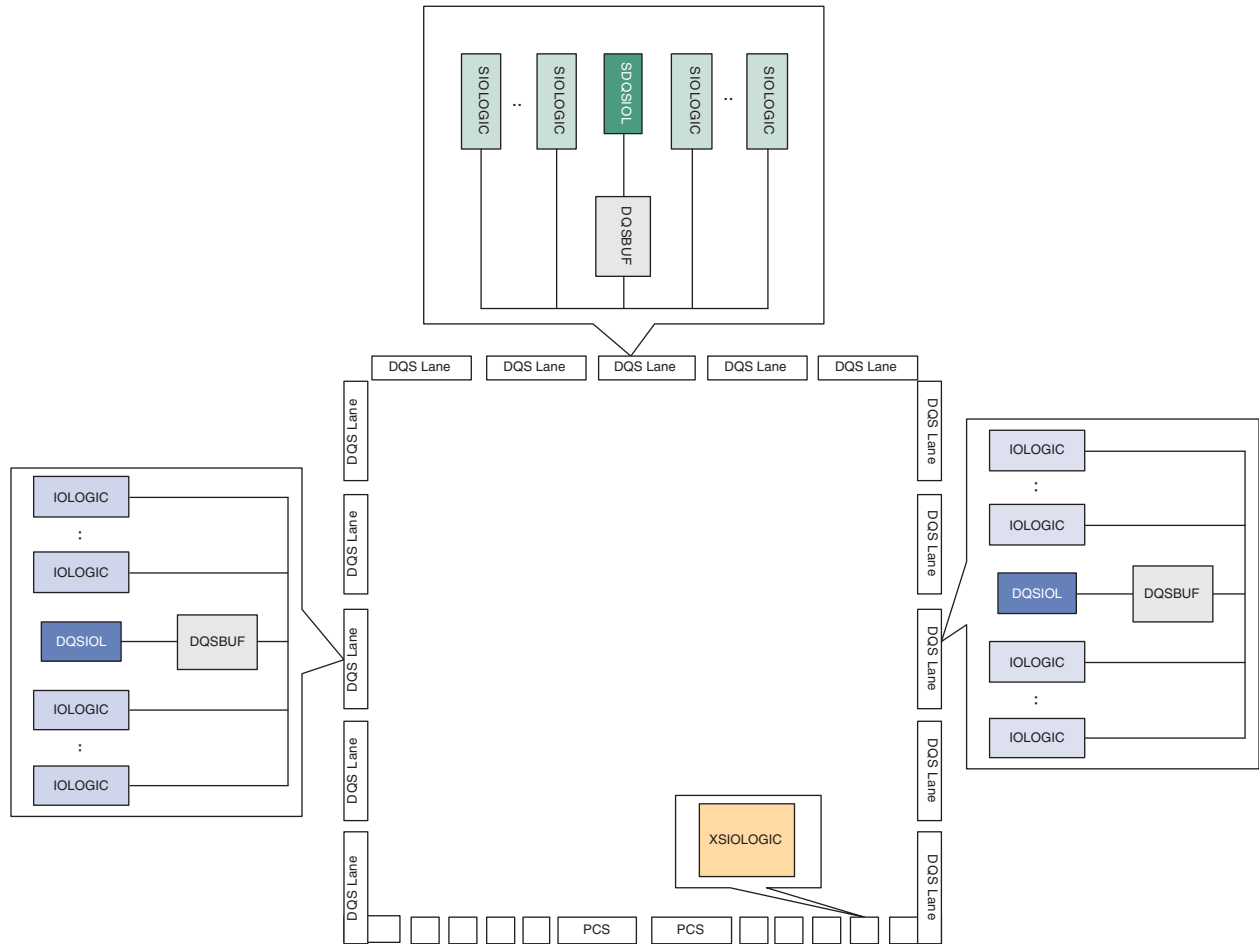
### **I/O Logic (IOL) Site Types/Names**

Based on the functions they support, I/O logic blocks are divided into the following site types.

- **IOLOGIC** – This site supports IREG, OREG, IDDRX, IDDRX2, ODDRX and ODDRX2 functions. These are the IOLs on the left and right sides of the device.
- **SIOLOGIC** – This site supports IREG, OREG, IDDRX, IDDRX2 and ODDRX functions. There is no ODDRX2 support in this site. These are mainly the IOLs on the top side of the device.
- **XSIOLOGIC** – This site supports IREG and OREG only. These are primarily the bottom side IOLs and DQS# IOLs.
- **DQSIOL** – These are the DQS IOLs. They support IREG, OREG, IDDRXD, IDDRX2, ODDRXDQSA and ODDRX2DQSA (left and right sides only) functions
- **SDQSIOL** – These are DQS IOLs with support for IREG, OREG and DQS ODDRXDQSA functions. Compared to DQSIOL, there is no ODDRX2DQSA support in this site. These are mostly the DQS IOLs on the top side of the device.

The software will issue an error message using these site names when an unsupported function is placed on one of these sites.

Figure 12-41. IOLOGIC Site Types



---

## Design Rules for Fitting Multiple Interfaces into One Device

### Rx Interfaces Running at High Speeds (>250 MHz)

Receive interfaces running at speeds higher than 250 MHz must use the x2 mode gearing DDR elements.

- To achieve high speeds these interfaces must be placed on the left and right sides of the device.
- If implementing a centered interface then the clock input must be locked to a primary clock (PCLK) input pin
- If implementing an aligned interface then the clock input must be locked to a dedicated GPLLT if interface is using a PLL GDLLT input pin if interface is using a DLL.
- Interfaces using the x2 gearing will need to use the edge clock resource. A single edge clock covers only one side of the device, hence all the data bits in the data bus should be assigned to one side of the device.
- There are two edge clocks per side. Two interfaces can be implemented per side of the device as long as they both do not require the DLL or CLKDIV module. See below:
  - There is only one CLKDIV and one DLL module available per side, hence two interfaces that use the CLKDIV and/or the DLL module cannot be on the same side.
  - When implementing an aligned interface using the DLL module (GDDR2\_RX.ECLK.Aligned) only one interface can be implemented per side as there is only one DLL and one CLKDIV per side of the device.
  - When implementing an aligned interface using a PLL (GDDR2\_RX.PLL.Dynamic), up to two of these interfaces can be implemented per side. It is recommended to allow the software to choose the best clock pins after locking the data inputs to the desired banks.
  - You can mix an aligned interface using a PLL (GDDR2\_RX.PLL.Dynamic) and a centered interface (GDDR2\_RX.ECLK.Centered) on the same side of the device. It is recommended to allow the software to choose the best clock pins after locking the data inputs to the desired banks.
  - You can mix an aligned interface using a DLL (GDDR2\_RX.ECLK.Aligned) and a centered interface (GDDR2\_RX.ECLK.Centered) on the same side of the device. It is recommended to allow the software to choose the best clock pins after locking the data inputs to the desired banks.
  - A GDDR2\_RX.ECLK.Aligned and GDDR2\_RX.PLL.Dynamic can be implemented on the same side of the device. In this case, the clock going to the PLL must be locked to a dedicated PLL clock input pin in the center of the device close to the DLL pin. For example, if the DLL pin is placed on the LUM0\_GDLLT\_IN\_A pin then the clock input to the PLL must be placed on the LUM0\_GPLLT\_IN\_A pin. It is recommended to allow the software to pick the best clock pins after locking the data inputs to the desired banks
  - For all interfaces using PLLs, refer to TN1178, [LatticeECP3 sysCLOCK PLL/DLL Design and Usage Guide](#) to see which two dedicated PLL clock outputs can feed the ECLK tree
- When multiple receive interfaces with multiple input clocks are required, and if the data widths for each of these interfaces is <10 bits wide, then the DQS clock can be used to implement these interfaces (GDDR2\_RX.DQS.Aligned and GDDR2\_RX.DQS.Centered)
  - For these interfaces it is required to connect the input clock to the DQS input pin
  - Data bits of the bus must be locked to the corresponding data pins
  - See the section “[DQ-DQS Grouping Rules](#)” for pin assignment rules.

### Rx Interfaces Running at Low Speeds (<250 MHz)

- Receive interfaces running at speeds lower than 250 MHz can use the x1 mode gearing DDR elements.
- If implementing a centered interface then the clock input must be locked to a primary clock (PCLK) input pin
- If implementing an aligned interface then the clock input must be locked to a dedicated “GPLLT” if interface is using a PLL “GDLLT” input pin if interface is using a DLL.
- Interfaces using the x1 gearing will use the primary clock resource. You can use as many interfaces as the number of primary clocks supported in the device.
  - If all interfaces are aligned and using a GDDR1\_RX.SCLK.Aligned interface that uses a DLL, then the number of interfaces is limited to two as there are two DLLs supported per device.
  - If using a GDDR1\_RX.SCLK.PLL.Aligned interface then the number of interfaces per device is limited by the number of PLLs available in that device.

---

### **Tx Interface Running at High Speeds (> 250 MHz)**

- Transmit Interfaces running at speeds higher than 250 MHz must use the x2 mode gearing DDR elements.
- To achieve high speeds these interfaces must be placed on the left and right sides of the device.
- Both edge clock and DQS clocks are used to implement the transmit side interfaces
- Data pins must be grouped into the DQS groups on these interfaces
- For interfaces requiring True LVDS outputs, the number of available True LVDS pins per DQS group must be calculated from the data sheet pinout tables since only a limited number of pins support True LVDS outputs. It may be required to modify the HDL generated from IPexpress to reduce the number of pins assigned to each DQS group
- All transmit interfaces using the x2 gearing mode use the CLKDIV module, hence only one TX interface can be implemented per side of the device.
- Inputs to the PLL used in the interface should be on primary clock routing or come in from a dedicated GPLLT input pin

### **Tx Interface Running at High Speeds (<250 MHz)**

- Transmit interfaces running at speeds lower than 250 MHz can use the x1 mode gearing DDR elements
- Interfaces using the x1 gearing will use the primary clock resource. You can as many interfaces as the number of primary clocks supported in the device.
- If all interfaces are centered, then the number of interfaces will depend on the number of available PLLs on the device.

### **Clocking Guidelines for Generic DDR Interfaces**

- Edge clock and primary clock resources are used when implementing a x2 receive or transmit interface
- Only the primary clock (PCLK) resources are used when implementing x1 receive or transmit interfaces
- Each edge clock can only span one side of the device, hence all the data bits of the in the x2 interface must be locked to one side of the device
- When implementing x1 interfaces, the bus can span any two sides as primary clocks can access DDR registers on all sides
- There are two edge clocks on each left, right of top side of the device. The bottom side does not support DDR registers and does not have any edge clocks. There are up to eight primary clocks available on a LatticeECP3 device.
- For high-speed interfaces it is recommended to use the edge clocks on the left and right sides of the device instead of the top side
- See [Design Rules for Fitting Multiple Interfaces into One Device](#) for details on implementing multiple interfaces on one side of a device
- The ECLK to DDR registers can be accessed through dedicated PCLK pins, GPLL outputs, DLL outputs, and GPLL input pins. See “LatticeECP3 sysCLOCK PLL/DLL Design & Usage Guide” for details
- Primary clock to DDR registers can be accessed through dedicated PCLK pins, GPLL outputs, DLL outputs & CLKDIV outputs. See “LatticeECP3 sysCLOCK PLL/DLL Design & Usage Guide” for details
- None of the clocks going to the DDR registers can come from internal general routing.
- DQS clocking is mainly used for DDR memory interface implementation.
- The DQS clock spans every 12 I/O's include the DQS pins, but only 10 of these can be used for generic DDR implementation

- DQS clocking can also be used for Generic DDR Receive interfaces when the bus size is 10 or less
- DQS clocking is also used when implementing all Generic DDR x2 Transmit interfaces
- Refer to the “DQ-DQS Grouping Rule” section for pinout assignment rules when using DQS clocking

## Common Software Errors and Solutions

- Placement error when implementing Transmit high speed True LVDS interface:

*“ERROR - par: DDR assignment finished unsuccessfully.”*

DQS grouping is used when implementing 2x gearing on the transmit side. This error occurs then there aren't enough true LVDS buffers in one DQS group. IPexpress generates DQS groups correctly for emulated buffers. Since the number of true LVDS pins vary per device and package, users need to update the module generated by IPexpress to correct the number of pins per DQS group.

- PAR Routing error when not using dedicated clock routing:

*“ERROR - par: netsanitycheck: the clock buf\_clk on comp Inst4\_DLLDELB port CLK1 is driven by general routing through comp clk”*

This error usually occurs when general routing is used on any of clocks routed to any of the DDR modules.

Check the following:

- If using a DLL or PLL, the clock input to the DLL or PLL should be on a dedicated GDLLT or GPLLT pin. A PCLK pin can be used but a USE PRIMARY preference must be assigned on this clock route from the PCLK pin to a DLL or PLL.
- If an interface requires the clock to be routed directly to the ECLK or PCLK tree, the PCLK pin should be used to input the clock. GPLLT or GDLLT pins do not have access to the PCLK or ECLK tree and cannot be used here.
- If the clocks going to any of the DDR registers are not used in any logic inside the FPGA (like a mux function) this will cause it to get on general route.
- Refer to [High-Speed DDR Interface Details](#) to see the clock placement requirements for each type of interface.

- PAR error when assigning a DDR function to a non-DDR I/O pin:

*ERROR - par: Cannot place PIO comp "datain\_2" on the proposed PIO site "PB11A / AE4" because the types of their IOLOGICs are incompatible: the associated IOLOGIC comp "datain\_2\_MGIOL" has been set to "IDDR\_OREG" mode (of type IDDRIOL), while the IOLOGIC site is of type XSILOGIC and supports FF only.*

In this case an IDDRX2D1 function is placed on the bottom side pin which does not support this function, hence the error.

See section [I/O Logic \(IOL\) Site Types/Names](#) to see the functions supported on each site type.

- Map Error on IDDRAPPS/ODDRAPPS function:

*ERROR - map: The 'IDDRAPPS' property is missing on IDDR instance 'Inst\_IDDRX2D1\_1\_2'. Each IDDR component targeted for this device needs the 'IDDRAPPS' property identifying the interface being implemented. Refer to DDR usage documentation for details.*

All LatticeECP3 “EA” designs require an IDDRAPPS attribute assigned to the input DDR module and an ODDRAPPS attribute assigned to the output DDR module. If these attributes are not assigned then MAP will error out with the error message above. To fix the error, regenerate the module in IPexpress. IPexpress will generate the module with all the required input and output DDR attributes.

## Timing Analysis for High-Speed DDR Interfaces

It is recommended that the user run Static Timing Analysis in the software for each of the high-speed interfaces. This section describes the timing preferences to use for each type of interface and the expected Trace results. The preferences can either be entered directly in the .lpf file or through the Design Planner graphical user interface.

The External Switching Characteristics section of the [LatticeECP3 Family Data Sheet](#) should be used along with this section. The data sheet specifies the actual values for these constraints for each of the interfaces.

### Frequency Constraints

Users must explicitly specify FREQUENCY (or PERIOD) PORT preferences to all input clocks in the design. This preference may not be required if the clock is generated out of a PLL or DLL or is input to a PLL or DLL.

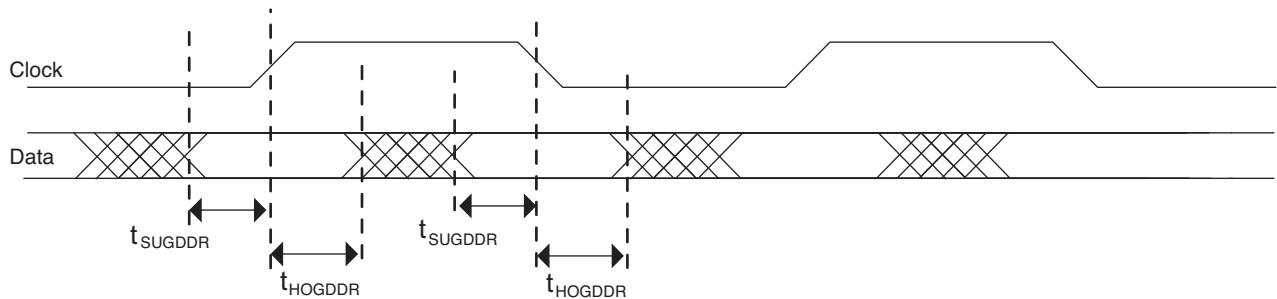
### DDR Input Setup and Hold Time Constraints

All of the receive (RX) interfaces, both x1 and x2, can be constrained with setup and hold preferences.

#### 1. Receive Centered Interface

Figure 12-42 shows the data and clock relationship for a Receive Centered Interface. The clock is centered to the data, so it comes into the devices with a setup and hold time.

**Figure 12-42. RX Centered Interface Timing**



Note:  $t_{SUGDDR}$  = Setup Time,  $t_{HOGDDR}$  = Hold Time

In this case the user must specify in the software preference the amount of setup and hold time available. These parameters are listed in the figure as  $t_{SUGDDR}$  and  $t_{HOGDDR}$ . These can be directly provided using the INPUT\_SETUP and HOLD preference as:

```
INPUT_SETUP PORT "Data" <tSUGDDR> ns HOLD <tHOGDDR> ns CLKPORT "Clock";
```

where:

Data = Input Data Port

Clock = Input Clock Port

The External Switching Characteristics section of the [LatticeECP3 Family Data Sheet](#) specifies the minimum setup and hold times required for each of the high-speed interfaces running at maximum speed. These values can be picked up from the data sheet if the interface is running at maximum speed.

Example:

For a GDDR2\_RX.ECLK.Centered interface on the left or right sides using a PCLK pin on the LatticeECP3-150EA-8 device when running at the maximum speed of 405MHz, the preference would be:

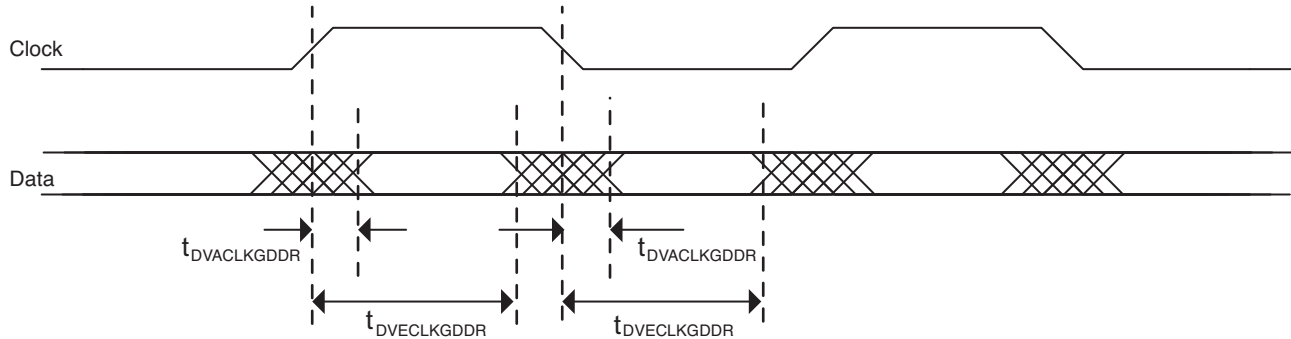
```
INPUT_SETUP PORT "Data" 0.321 ns HOLD 0.321 ns CLKPORT "Clock";
```

*Note: Please check the LatticeECP3 Family Data Sheet for the latest  $t_{SUGDDR}$  and  $t_{HOGDDR}$  numbers.*

## 2. Receive Aligned Interface

Figure 12-43 shows the data and clock relationship for a Receive Aligned Interface. The clock is aligned edge-to-edge with the data.

Figure 12-43. RX Aligned Interface Timing



Note:  $t_{DVACKGDDR}$  = Data Valid after CLK,  $t_{DVECKGDDR}$  = Data Hold After CLK

In this case, worst case data may occur after the clock edge resulting in a negative setup time when entering the device. The worst case setup is specified by the  $t_{DVACKGDDR}$  after the clock edge and the worst case hold time is specified as  $t_{DVECKGDDR}$ . The setup and hold time can be specified as:

```
INPUT_SETUP PORT "Data" <-tDVACKGDDR > ns HOLD < tDVECKGDDR> ns CLKPORT
"Clock";
```

where:

Data = Input Data Port  
Clock = Input Clock Port

Note: A negative number is used for SETUP time as the data occurs after the clock edge in this case.

The External Switching Characteristics section of the [LatticeECP3 Family Data Sheet](#) specifies the minimum  $t_{DVACKGDDR}$  and  $t_{DVECKGDDR}$  values required for each of the high-speed interfaces running at maximum speed. These values can be picked up from the data sheet if the interface is running at maximum speed. The data sheet numbers for this preference are listed in UI (Unit Interface). 1UI is equal to one-half the clock period. Therefore, these numbers will need to be calculated from the clock period used.

Preference Example:

For a GDDR2\_RX.ECLK.Aligned (no CLKDIV) interface on the left or right side using DLLCLKPIN for clock input on a LatticeECP3-150EA-8 device running at the maximum speed of 460MHz (UI = 1.09ns), the preference would be:

$$t_{DVACKGDDR} = 0.225UI = 0.25ns, t_{DVECKGDDR} = 0.775UI = 0.84ns$$

The preference for this case is:

```
INPUT_SETUP PORT "Data" -0.250000 ns HOLD 0.840000 ns CLKPORT "Clock";
```

Note: Please check the LatticeECP3 Family Data Sheet for the latest  $t_{DVACKGDDR}$  and  $t_{DVECKGDDR}$  numbers.

## 3. Receive Dynamic Interfaces

Static Timing Analysis will not show timing for all the dynamic interface cases as the either the clock or data delay will be dynamically updated at run time.

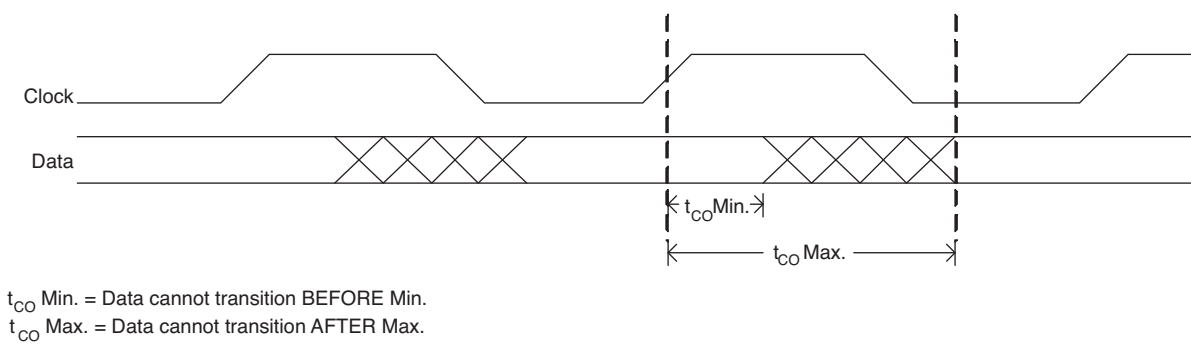


**DDR Clock to Out Constraints for Transmit Interfaces**

All of the transmit (TX) interfaces, both x1 and x2, can be constrained with clock-to-out constraints to detect the relationship between the clock and data when leaving the device.

Figure 12-44 shows how the clock-to-out is constrained in the software. Minimum  $t_{CO}$  is the minimum time after the clock edge transition that the data will not transition. So any data transition must occur between the  $t_{CO}$  minimum and maximum values.

**Figure 12-44.  $t_{CO}$  Minimum and Maximum Timing Analysis**



**1. Transmit Centered Interfaces**

In this case, the transmit clock is expected to be centered with the data when leaving the device. Figure 12-45 shows the timing for a centered transmit interface.

**Figure 12-45. Transmit Centered Interface Timing**

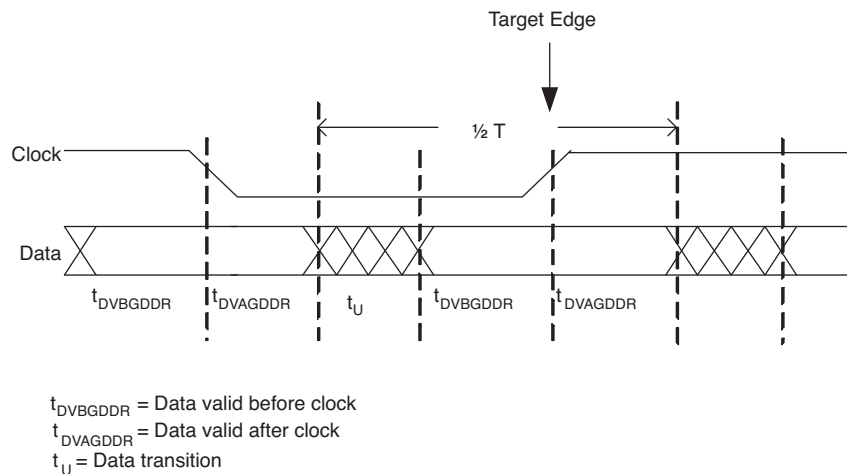


Figure 12-45 shows that the maximum value after which the data cannot transition is  $-t_{DVBCKGDDR}$ . The minimum value before which the data cannot transition is  $-(t_U + t_{VBCKGDDR})$ . A negative sign is used because in this particular case where clock is forwarded centered-aligned to the data, these two conditions occur before the clock edge.

The [LatticeECP3 Family Data Sheet](#) specifies the  $t_{DVBCKGDDR}$  and  $t_{DVACKGDDR}$  values at maximum speed. But we do not know the  $t_U$  value, so the minimum  $t_{CO}$  can be calculated using the following equation.

$$t_{CO} \text{ Min.} = -(t_{DVBGDDR} + t_U)$$

$$\frac{1}{2}T = t_{DVAGDDR} + t_{DVBGDDR} + t_U$$

$$-(t_{DVBGDDR} + t_U) = t_{DVAGDDR} - \frac{1}{2}T$$

$$t_{CO \text{ Min.}} = t_{DVAGDDR} - \frac{1}{2}T$$

The clock-to-out time in the software can be specified as:

```
CLOCK_TO_OUT PORT "Data" MAX <-tDVBGDDR> MIN <tDVAGDDR -1/2 Clock Period>
CLKPORT "clk" CLKOUT PORT "Clock";
```

where:

Data = Data Output Port

Clock = Forwarded Clock Output Port

clk = Input Clock Port

The values for  $t_{DVACKGDDR}$  and  $t_{DVACKGDDR}$  can be found in the External Switching Characteristics section of the [LatticeECP3 Family Data Sheet](#) for the maximum speed.

Preference Example:

For a GDDR<sub>X1\_TX</sub>.SCLK.Centered interface running on the LatticeECP3-150EA-8 device at 250MHz,  $t_{DVBGDDR} = t_{DVAGDDR} = 0.670\text{ns}$ , the preference would be:

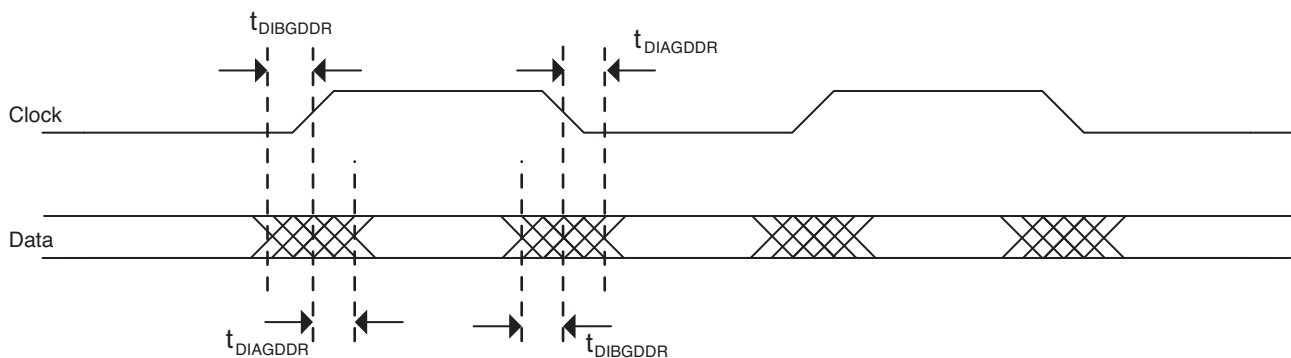
```
CLOCK_TO_OUT PORT "Data" MAX -0.670000 ns MIN -1.330000 ns CLKPORT "clk"
CLKOUT PORT "Clock";
```

*Note: Please check the LatticeECP3 Family Data Sheet for the latest  $t_{DVAGDDR}$  and  $t_{DVBGDDR}$  numbers.*

## 2. Transmit Aligned Interfaces

In this case, the clock and data are aligned when leaving the device. Figure 12-46 shows the timing diagram for this interface.

**Figure 12-46. Transmit Aligned Interface Timing**



$t_{DIA GDDR}$  = Data valid after clock.

$t_{DIB GDDR}$  = Data valid before clock.

Figure 12-46 shows that maximum value after which the data cannot transition is  $t_{DIA GDDR}$ . The minimum value before which the data cannot transition is  $-t_{DIB GDDR}$ . A negative sign is used for the minimum value because the minimum condition occurs before the clock edge.

The clock to out time in the software can be specified as:

```
CLOCK_TO_OUT PORT "Data" MAX <tDIAGDDR> MIN <-tDIBGDDR> CLKPORT "clk" CLK-  
OUT PORT "Clock";
```

where:

Data = Data Output Port

Clock = Forwarded Clock Output Port

clk = Input Clock Port

The  $t_{DIAGDDR}$  and  $t_{DIBGDDR}$  values are available in the External Switching Characteristics section of the [LatticeECP3 Family Data Sheet](#) for maximum speed.

Preference Example:

For a GDDR2\_TX.Aligned interface on the LatticeECP3-150EA-8 device running on the left or right sides at 500MHz,  $t_{DIAGDDR} = t_{DIBGDDR} = 0.200\text{ns}$ . The preference would be:

```
CLOCK_TO_OUT PORT "Data" MAX 0.200000 ns MIN -0.200000 ns CLKPORT "clk" CLKOUT  
PORT "Clock";
```

*Note: Please check the LatticeECP3 Family Data Sheet for the latest  $t_{DIAGDDR}$  and  $t_{DIBGDDR}$  numbers.*

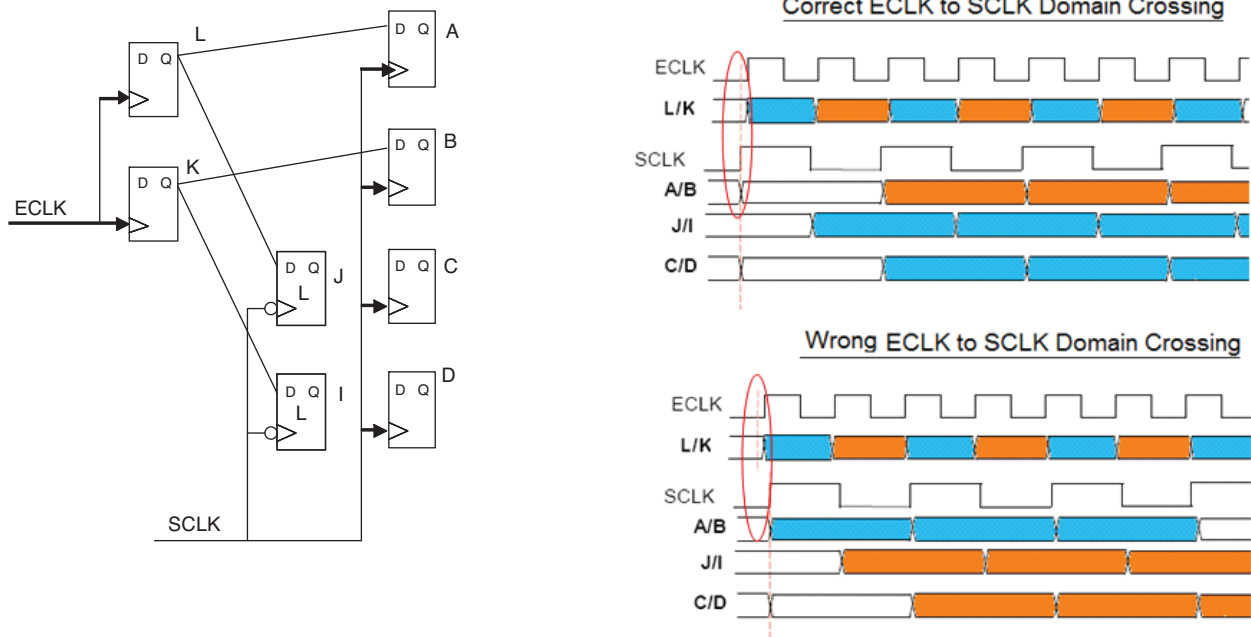
### Timing Rule Check for Clock Domain Transfers

Clock Domain Transfers within the IDDR and ODDR modules are checked by Trace automatically when these elements are used in a design. Most clock domain transfers occur in the IDDRX2 and ODDR2 modules where there are fast-speed and slow-speed clock inputs. For IDDRX2, there is a transfer from the fast-speed ECLK to the slow-speed SCLK. On the ODDR2, the transfer happens from the slow-speed SCLK to the fast-speed ECLK.

For ispLEVER 8.0, no special preferences are needed to run this check. The clock domain transfer checks are automatically done by the software and reported in the Trace report under section called "Timing Rule Check". The report lists the timing for the both the input and output DDR blocks where a clock domain transfer is done.

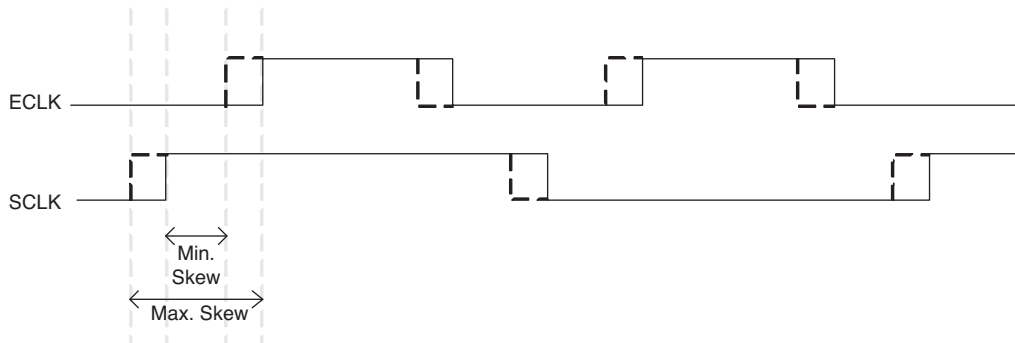
Figure 12-47 shows the transfer of data between the ECLK and SCLK for the IDDRX2 block. A cause of concern is the phase relationship between the ECLK and SCLK. As in the waveforms below, these two clocks have to maintain a certain amount of skew to transfer data successfully between the two clocks.

Figure 12-47. IDDRX2 ECLK to SCLK Transfer



The skew between the two clocks is specified in terms of “min.” skew and “max.” skew. Figure 12-48 shows how the “min.” and “max.” skew is measured for the IDDRX2 block. It is required that the “min.” and “max.” skew be within the specified “min.” and “max.” specs.

Figure 12-48. IDDRX2 ECLK to SCLK Skew Calculation



The following equations are used to check for valid skew between the ECLK and SCLK for an IDDRX2 block.

$$\text{Max. Skew} < - (0 + \text{Internal ECLK to SCLK Setup Time})$$

$$\text{Min. Skew} > - (\text{ECLK cycle} + \text{Internal ECLK to SCLK Hold Time})$$

The Trace report below shows an example IDDRX2 ECLK to SCLK Timing rule check.

Internal Preference: Timing Rule Check  
32 items scored, 0 timing errors detected.

-----  
This section of the Trace report will identify any inherent timing rule violations in the design. These rules may be independent of preferences.

Passed: din\_i\_13\_MGIOL meets ECLK to CLK skew range from -2.489ns to 0.006ns

Max skew of -1.367ns meets timing requirement of 0.006ns by 1.373ns

Max ECLK:

Name	Fanout	Delay (ns)	Site	Resource
PADI_DEL	---	0.369	M4.PAD to	M4.PADDI clk_i
ROUTE	33	0.509	M4.PADDI to	IOL_L43EA.ECLK iddr_inst/buf_clk (to sclk_o_c)

-----  
0.878 (42.0% logic, 58.0% route), 1 logic levels.

Min CLK:

Name	Fanout	Delay (ns)	Site	Resource
PADI_DEL	---	0.369	M4.PAD to	M4.PADDI clk_i
ROUTE	33	0.476	M4.PADDI to	*V_R61C15.CLKI iddr_inst/buf_clk
CLKOUT_DEL	---	0.353	*V_R61C15.CLKI to	*_R61C15.CDIV2 iddr_inst/Inst3_CLKDIVB
ROUTE	33	1.047	*_R61C15.CDIV2 to	IOL_L43EA.CLK sclk_o_c

-----  
2.245 (32.2% logic, 67.8% route), 2 logic levels.

Min skew of -1.537ns meets timing requirement of -2.489ns by 0.952ns

Min ECLK:

Name	Fanout	Delay (ns)	Site	Resource
PADI_DEL	---	0.423	M4.PAD to	M4.PADDI clk_i
ROUTE	33	0.476	M4.PADDI to	IOL_L43EA.ECLK iddr_inst/buf_clk (to sclk_o_c)

-----  
0.899 (47.1% logic, 52.9% route), 1 logic levels.

Max CLK:

Name	Fanout	Delay (ns)	Site	Resource
PADI_DEL	---	0.423	M4.PAD to	M4.PADDI clk_i
ROUTE	33	0.509	M4.PADDI to	*V_R61C15.CLKI iddr_inst/buf_clk
CLKOUT_DEL	---	0.353	*V_R61C15.CLKI to	*_R61C15.CDIV2 iddr_inst/Inst3_CLKDIVB
ROUTE	33	1.151	*_R61C15.CDIV2 to	IOL_L43EA.CLK sclk_o_c

-----  
2.436 (31.9% logic, 68.1% route), 2 logic levels

*Note: For paths that are common between the ECLK and CLK, the same delay will be used. For example, since PADI\_DEL is the shared path between the Max. ECLK and Min. CLK for "Max skew calculation" and between Min. ECLK and Max. CLK for "Min skew calculation" the value is the same in both cases.*

The "ECLK to CLK skew range from -2.510 ns to -0.019 ns" for this case is an allowable skew range between the two clocks. The skew between the two clocks must fall within this range or the Trace will fail on this preference. The allowable skew range is determined by the frequency at which the fast ECLK clock is running.

Similarly for the ODDR2D, there is an internal transfer from SCLK to DQCLK0/1 which will be listed in the Timing Rule Check section as well.

This internal rule check is required for normal data pins but is not required for the forwarded clock output itself where data inputs to the ODDR2D are constants. There is no internal data transfer in this case. Users can ignore Trace reported errors on this Internal Rule Check only for forwarded clocks. These errors may be blocked by a preference like:

```
BLOCK NET "sclk_c" COMP "clkout_MGIOL";
```

Where sclk\_c is the net on the CLK pin of the IOLOGIC component clkout\_MGIOL, where clkout is the forwarded clock. Trace violations of this rule check on corresponding data outputs need to be understood and resolved.

### Preferences for Specific Elements

1. **DLLDELB** – This element is used in to generate a 90° delay in all receive aligned interfaces. The 90° delay is calculated based on the input clock to the TRDLLB element. A frequency preference must be provided on the CLKI of the TRDLLB to allow trace to produce the 90° delay.
2. **DQSBUFx** – The DQSBUF element used in all the DQS interfaces will delay the DQSI input by 90°. The 90° delay is calculated from the input clock to the DQSDLL element connected via the DQSDEL signal. A frequency preference must be provided on the CLK input of the DQSDLL to allow trace to produce the 90° delay.

The [DDR Software Primitives and Attributes](#) section provides a detailed description of the DQSBUF elements.

*Note: Some device I/O pin names end with an “E\_A”, “E\_B”, “E\_C” or “E\_D” (for example, PR43E\_B). These pins are “Input Only” pins. These pins can be used only to implement receive interfaces. The clock delay to these pins is ~50ps longer than the other pins, so you will see some difference in timing between these and other I/O pins.*

### Valid Window Calculation

Users can calculate the available valid window using the transmitter device specifications to determine if it will be possible to meet the receiver timing requirements at the LatticeECP3 device.

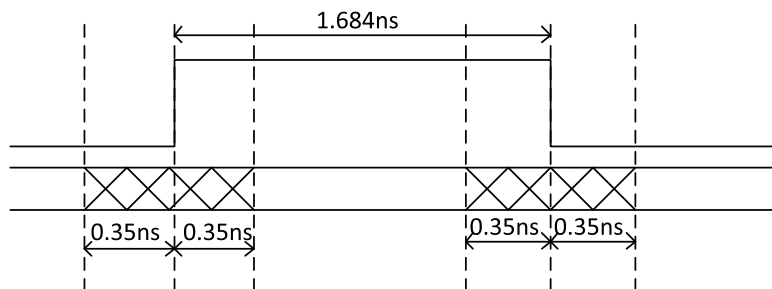
The data sheet numbers can be used to estimate the required valid window at the LatticeECP3 DDR inputs depending on the interface type.

For an Rx centered interface the total data valid window required =  $t_{SUGDDR} + t_{HGDDR}$

For an Rx aligned interface the total data valid window required =  $t_{DVECLKGDDR} - t_{DVACLKGDDR}$

For example, in the following case the data at the receiver looks like Figure 12-49.

**Figure 12-49. Data at Receiver (Tx Data)**



In this case, the data at the LatticeECP3 input is an aligned interface, with the following timing:

Data Rate = 594 Mbps

$f_{MAX} = 297$  MHz

Uncertainty around clock edges = 0.350ns

UI = 1.684ns

Since this is an aligned interface, we can calculate  $t_{DVECLKGDDR}$  and  $t_{DVACLKGDDR}$  to determine the available Data Valid Window.

$t_{DVECLKGDDR} = 0.350$  ns

$t_{DVACLKGDDR} = 1.684$  ns – 0.350 ns = 1.334 ns

Data Valid Window =  $t_{DVECLKGDDR} - t_{DVACLKGDDR} = 0.984$  ns

On the LatticeECP3 receive side, we must use the GDDR2\_RX.ECLK.Aligned interface given the frequency and data to clock alignment of this interface.

For the GDDR2\_RX.ECLK.Aligned for the -6 speed grade, here are the requirements from the data sheet at 297 MHz.

$t_{DVECLKGDDR} = 0.225$  UI = 0.379ns

$t_{DVACLKGDDR} = 0.775$  UI = 1.305ns

Data Valid Window =  $t_{DVECLKGDDR} - t_{DVACLKGDDR} = 0.926$ ns

Note: Please refer to the [LatticeECP3 Family Data Sheet](#) for the latest timing numbers.

The required data valid window for this example at the LatticeECP3 DDR input is 0.926 ns and the data valid window available on the interface is 0.984ns. This interface will just meet the timing requirements at the LatticeECP3 device.

To calculate the worst corner case, the software I/O Timing Analysis should be run to see if the worst case required timing will be met. I/O Timing Analysis will run the timing through all the speed grades of the device family and show the worst case results. The Trace Report will provide the worst case timing on the same speed grade.

## DDR/DDR2/DDR3 SDRAM Interfaces Overview

The DDR SDRAM interface transfers data at both the rising and falling edges of the clock. The DDR2 is the second generation of the DDR SDRAM memory, whereas DDR3 is the third generation.

The DDR, DDR2 and DDR3 SDRAM interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. The DDR SDRAM interface uses a single-ended DQS strobe signal and the DDR2 and DDR3 interfaces use a differential DQS strobe. The figures below show typical DDR and DDR2/DDR3 SDRAM interface signals. SDRAM interfaces are typically implemented with eight DQ data bits per DQS. So, a x16 interface will use two DQS signals, and each DQS is associated with eight DQ bits. Both the DQ and DQS are bi-directional ports and are used to read and write to the memory.

When reading data from the external memory device, data coming into the device is edge-aligned with respect to the DQS signal. This DQS strobe signal needs to be phase shifted 90° before the FPGA logic can sample the read data. When writing to a DDR/DDR2/DDR3 SDRAM, the memory controller (FPGA) must shift the DQS by 90° to center-align with the data signals (DQ). A clock signal is also provided to the memory. This clock is provided as a differential clock (CLKP and CLKN) to minimize duty cycle variations. The memory also uses these clock signals to generate the DQS signal during a read via a DLL inside the memory. The figures below show DQ and DQS timing relationships for read and write cycles.

During read, the DQS signal is low for some duration after it comes out of tristate. This state is called Preamble.

The state when the DQS is low before it goes into tristate is the Postamble state. This is the state after the last valid data transition.

DDR SDRAM also requires a Data Mask (DM) signal to mask data bits during write cycles. Note that the ratio of DQS to data bits is independent of the overall width of the memory. An 8-bit interface will have one strobe signal.

DDR SDRAM interfaces use the SSTL25 Class I/II I/O standards, DDR2 SDRAM interface uses the SSTL18 Class I/II and DDR3 SDRAM interface uses the SSTL15 I/O standards. Both the DDR2 and DDR3 SDRAM interfaces require differential DQS (DQS and DQS#). DDR2 has an option to use either single-ended or differential DQS.

In addition, the DDR3 memory module uses fly-by architecture for the data and strobe signals. This requires the memory controller to support read and write leveling to adjust for leveled delay on read and write data transfers.

**Table 12-7. DDR DDR2 and DDR3 Summary**

	DDR	DDR2	DDR3
Data Rate	200 to 400Mbps	250Mbps to 532Mbps	600 to 800Mbps
DQS	Single-Ended	Single-Ended /Differential	Differential
Interface	SSTL25	SSTL18	SSTL15
Leveling	No	No	Yes

**Figure 12-50. Typical DDR SDRAM Interface**

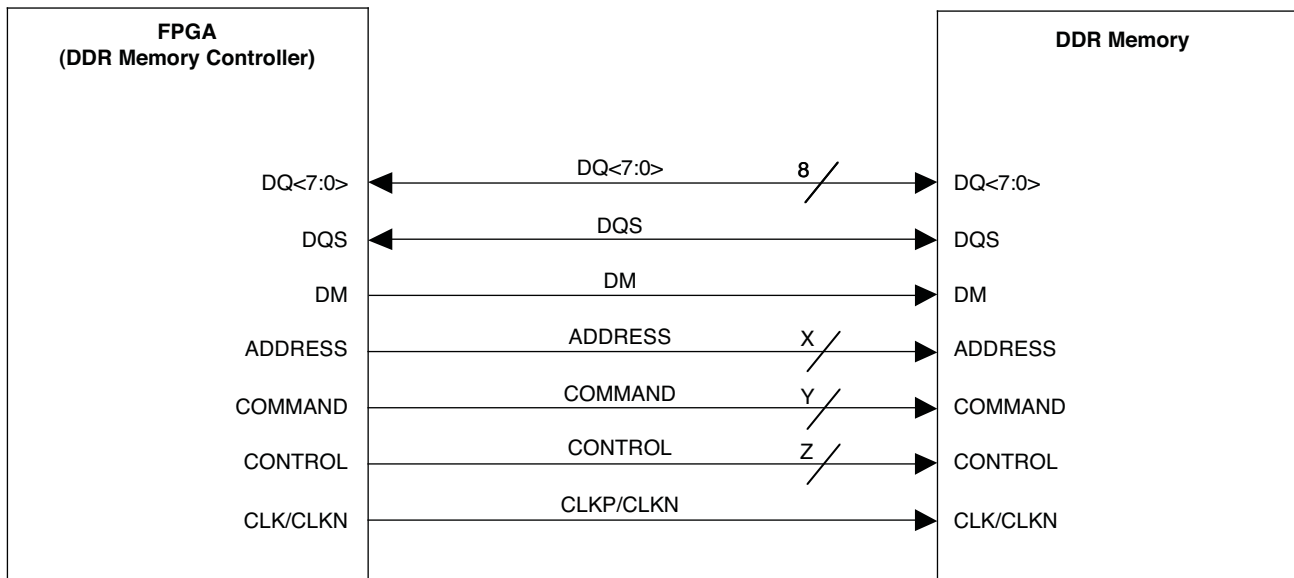
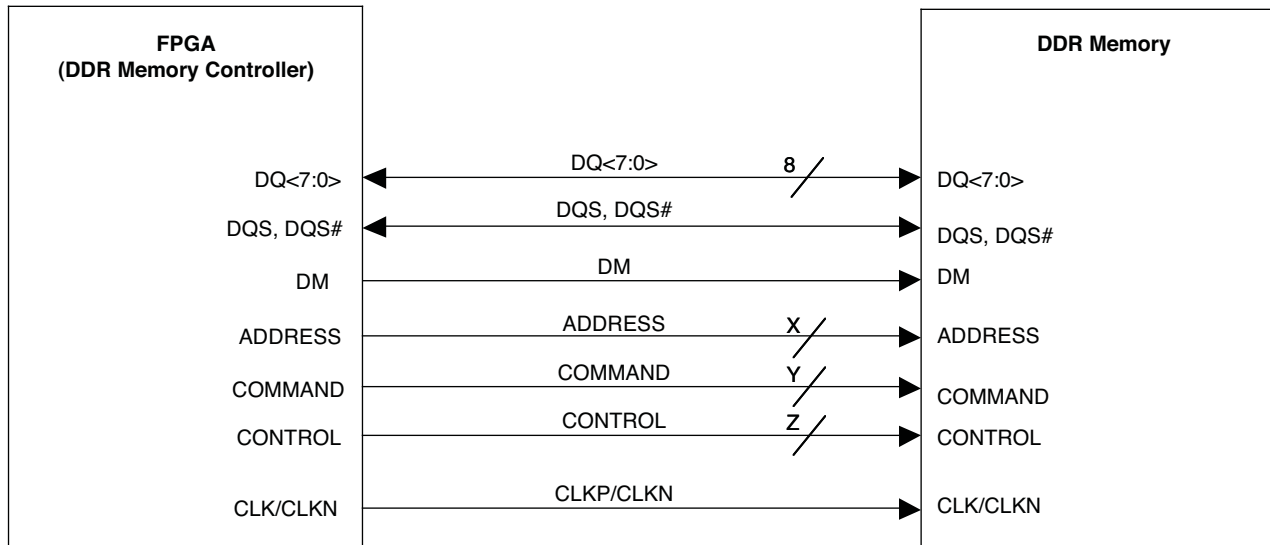




Figure 12-51. Typical DDR2 and DDR3 SDRAM Interface



Figures 12-52 and 12-53 show the DQ and DQS relationship for memory read and write interfaces.

Figure 12-52. DQ-DQS During Read

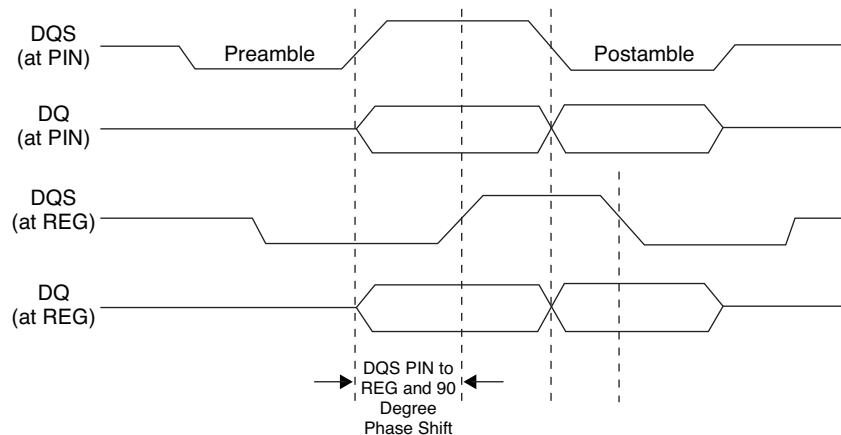
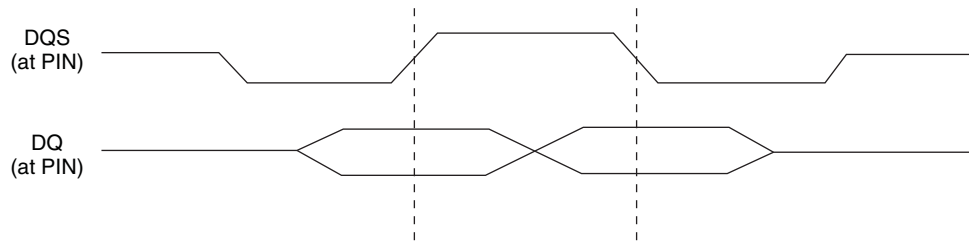


Figure 12-53. DQ-DQS During Write



## Implementing DDR/DDR2/DDR3 Memory Interfaces

As described in the [DDR/DDR2/DDR3 SDRAM Interfaces Overview](#) section, all the DDR SDRAM interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. When reading data from the external memory device, data coming into the LatticeECP3 device is edge-aligned with respect to the DQS signal. Therefore, the LatticeECP3 device needs to shift the DQS (90° phase shift) before using it to sample the read data. When writing

to a DDR SDRAM from the memory controller, the LatticeECP3 device must generate a DQS signal that is center-aligned with the DQ, the data signals. This is accomplished by ensuring a DQS strobe is 90° shifted relative to the DQ data.

For DDR3 memory, the memory controller also needs to handle the read and write leveling required by the interface due to the newer fly-by topology.

LatticeECP3 devices have dedicated DQS support circuitry for generating appropriate phase-shifting for DQS. The DQS phase shift circuit uses a frequency reference DLL to generate delay control signals associated with each of the dedicated DQS pins, and is designed to compensate for process, voltage and temperature (PVT) variations. The frequency reference is provided through one of the global clock pins. The dedicated DDR support circuit is also designed to provide comfortable and consistent margins for the data sampling window.

This section describes how to implement the read and write sections of a DDR memory interface. It also provides details of the DQ and DQS grouping rules associated with the LatticeECP3 devices.

Both the LatticeECP3 “E” and “EA” devices can be used to implement DDR and DDR2 memory interfaces. DDR3 memory interface can only be implemented on the “EA” devices. There are some differences in the DDR memory implementation between the “EA” and “E” devices. These differences between the devices are indicated in the appropriate sections below.

See the LatticeECP3 DDR3 Memory Controller IP to see how the DDR3 memory interface can be implemented on LatticeECP3 “EA” devices. DDR and DDR2 memory interface implementations are described in the sections below.

## DQS Grouping

Each DQS group generally consists of at least 10 I/Os (one DQS, eight DQ and one DM) for an 8-bit DDR/DDR2 memory interface or 11 I/Os (two DQS, eight DQ, one DM) to implement a complete 8-bit DDR3 memory interface. LatticeECP3 devices support DQS signals on the top, left and right sides of the device.

Each DQS signal spans across 12 I/Os. Any 10 (for DDR/DDR2) or 11 (for DDR2/DDR3) of these 12 I/Os spanned by the DQS can be used to implement an 8-bit DDR memory interface. In addition to the DQS grouping, the user must also assign one reference voltage VREF1 for a given I/O bank.

**Figure 12-54. DQ-DQS Grouping**

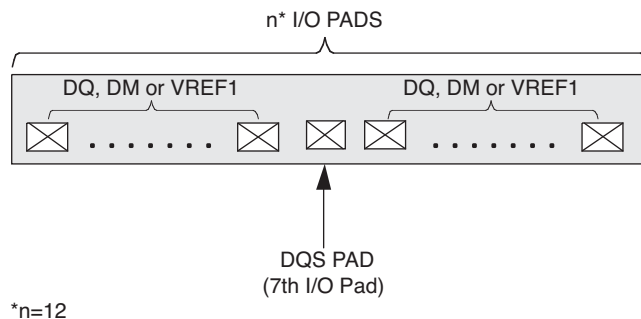


Figure 12-54 shows a typical DQ-DQS group for LatticeECP3 devices. The seventh I/O of this group of 12 I/Os is the dedicated DQS pin. All six pads before of the DQS and five pads after the DQS are covered by this DQS bus span. If using DDR2 or DDR3 memory then the DQS is differential and the eighth pad is used by the DQS# signal. The user can assign any eight of the other I/O pads to be DQ data pins. Therefore, to implement a 32-bit wide memory interface you would need to use four such DQ-DQS groups.

When not interfacing with the memory, the dedicated DQS pin can be used as a general-purpose I/O. Note that the DQS/DQS# pads cannot be used for other DDR functions like DQ, DM or generic DDR. Each of the dedicated DQS pins is internally connected to the DQS phase shift circuitry. The pinout information contained in the [LatticeECP3 Family Data Sheet](#) shows pin locations for the DQS pads and the corresponding DQ pads.

Some I/Os on the left, right and top sides will not support DQS grouping. See the [DDR Memory DQ/DQS Design Rules and Guidelines](#) section of this document for further information.

## Memory Read Implementation

The LatticeECP3 devices contain a variety of features to simplify implementation of the read portion of a DDR interface:

- DLL-compensated DQS delay elements
- DDR input registers
- Automatic DQS to system/edge clock domain transfer circuitry
- Data Valid Module

### DLL-Compensated DQS Delay Elements

The DQS from the memory is connected to the DQS Delay element. The DQS delay block receives a 7-bit delay control from the on-chip DQSDLL. LatticeECP3 devices support two DQSDLL, one on the left and the other on the right side of the device. The DQSDEL generated by the DQSDLL on the left side of the device is routed to all the DQS blocks on the left and top halves of the device. The delay generated by the DQSDLL on the right side of the device is distributed to all the DQS delay blocks on the right side and the top half of the device. There are no DQS pins on the bottom banks of the device. These digital delay control signals are used to delay the DQS from the memory by 90°.

The DQS received from the memory is delayed in each of the DQS delay blocks and this delay DQS is used to clock the first set stage DDR input registers.

### Automatic DQS to System/Edge Clock Domain Transfer Circuitry

In a typical DDR memory interface design, the phase relation between the incoming delayed DQS strobe and the internal system clock (during the read cycle) is unknown. Prior to the read operation in DDR memories DQS is in tristate. Coming out of tristate, the DDR memory device drives DQS low in the preamble state. The DQS Transition Detect block detects this transition and generates a signal indicating the required polarity for the FPGA system clock (DDRCLKPOL). This signal is used to control the polarity of the clock to the synchronizing registers. For the DDR3 memory interface, this block generates two signals, DDRCLKPOL and DDRLAT. DDRCLKPOL is used to transfer data from the DDR registers to the synchronization registers clocked by ECLK and the DDRLAT signal is used to transfer data from the synchronization registers to the clock transfer registers.

### Data Valid Module

The data valid module generates a DATAVALID signal. This signal indicates to the FPGA that valid data is transmitted out the input DDR registers to the FPGA core. Note that the DATAVALID signal indicates only the start of valid read data. It is the memory controller's responsibility to make the proper width of the read data valid signal.

### DDR I/O Register Implementation

The first set of DDR registers is used to de-mux the DDR data at the positive and negative edges of the phase-shifted DQS signal. The latch that captures the positive-edge data is followed by a negative-edge triggered register. This register transfers the positive edge data from the first register to the negative edge of DQS so that both the positive and negative portions of the data are now aligned to the negative edge of DQS.

For DDR and DDR2 memory interfaces, the second stage synchronization registers are clocked by the FPGA clock. The polarity of this clock is selected by the DDR Clock Polarity (DDRCLKPOL) signal generated by the DQS Transition Detect Block. The FPGA clock clocks an additional set of clock transfer registers at the end before the data enters the FPGA core.

The DDR3 memory interface uses the gearing feature of the input registers. The synchronization registers are clocked by the fast edge clock (ECLK) input and are then transferred to the clock transfer registers clocked by the slower FPGA clock. The polarity of the ECLK is set by the DDRCLKPOL signal and the FPGA clock (SCLK) polarity is set by the DDRLAT signal.

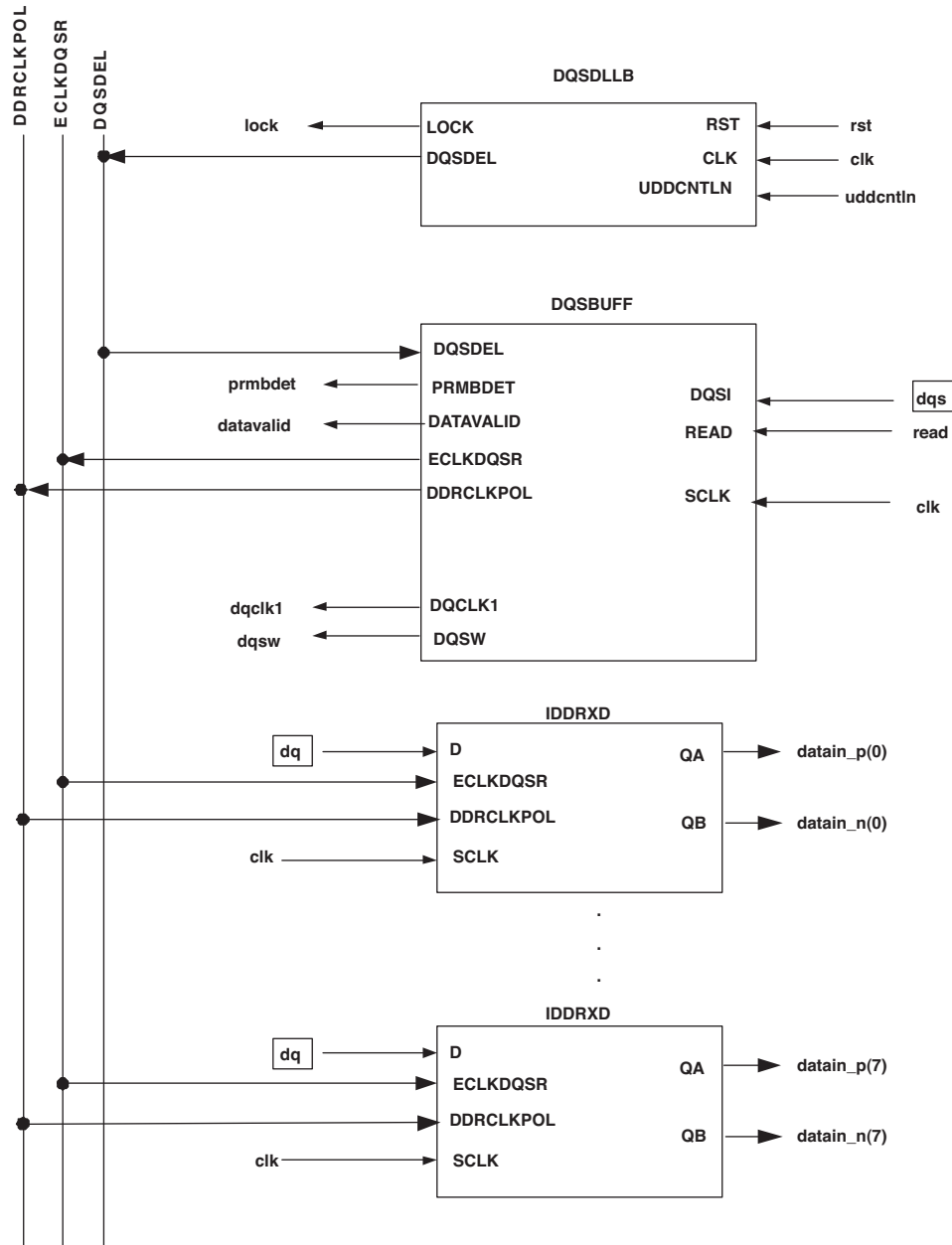
The [LatticeECP3 Family Data Sheet](#) describes each of these circuit elements in more depth.

**DDR/DDR2 Memory Read Implementation**

The following sections explain the DDR/DDR2 read-side implementation. LatticeECP3 devices support the DDR/DDR2 memory interface function using the DDR memory mode module generated through the IPexpress tool. Using IPexpress, a designer can generate the different modules required to read the data coming from the DDR/DDR2 memory. See the section [DDR Memory Interface Generation Using IPexpress](#) for details on the IPexpress interface. This section explains the read side module generated by IPexpress.

The DDR/DDR2 read side is implemented using the following three software elements. The DQS DLL represents the DLL used for calibration. The IDDRXD implements the input DDR registers. The DQSBUFF represents the DQS delay block, the clock polarity control logic and the Data Valid module. Figure 12-55 shows the read side implementation for both the “E” and “EA” devices. IPexpress should be used to generate this interface. See the section [DDR Memory Interface Generation Using IPexpress](#) for details.

Figure 12-55. DDR/DDR2 Read Side Implementation for “E” and “EA” Devices



### DDR3 Memory Read Implementation

The following sections explain the DDR3 read side implementation. LatticeECP3 devices support the DDR3 memory interface function using the DDR memory mode module generated through the IPexpress tool. Using IPexpress, a designer can generate the different modules required to read the data coming from the DDR3 memory. See the section [DDR Memory Interface Generation Using IPexpress](#) for details on the IPexpress interface. This section explains the read side module generated by IPexpress.

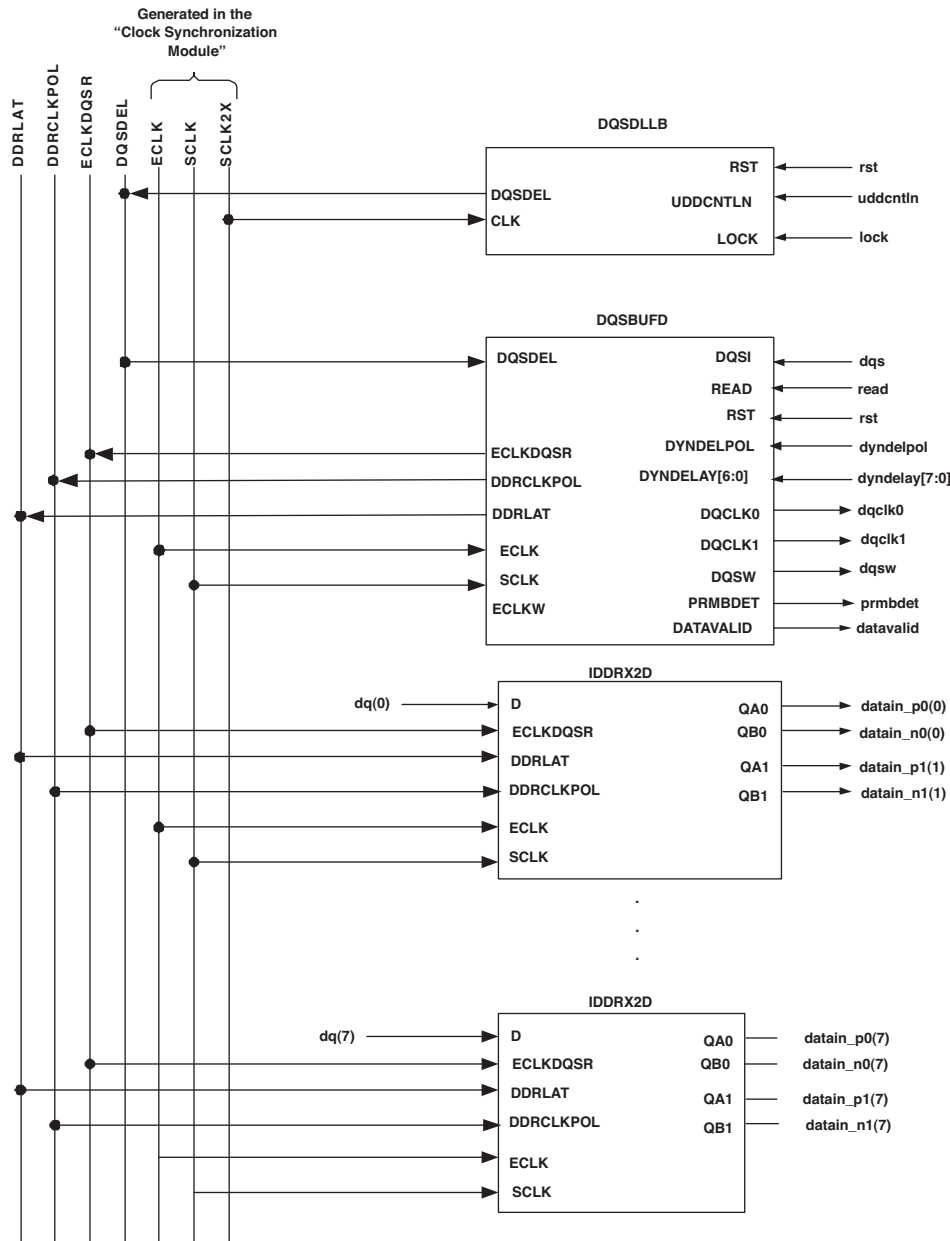
The DDR3 memory interface generated in IPexpress also includes a Clock Synchronization Module (CSM) that provides the clock synchronization and alignment among the required clocks for successful DDR3 functionality.

The DDR3 read side is implemented using three software elements. The DQS DLL represents the DLL used for calibration. The IDDRX2D implements the input DDR registers in x2 gearing mode required for DDR3 memory. The DQS BUF represents the DQS delay block, the clock polarity control logic and the Data Valid module.

The ECLK, SCLK2X and SCLK are generated in a clock synchronization module (CSM) that provides the clock synchronization and alignment among the required clocks for successful DDR3 functionality. See the [DDR3 Clock Synchronization Module](#) section for details on the Clock Synchronization Module and internal clock generation

Figure 12-56 shows the read side implementation. IPexpress should be used to generate this interface.

**Figure 12-56. DDR3 Read Side Implementation for “EA” Devices**



### Memory Write Implementation

To implement the write portion of a DDR memory interface, two streams of single data rate data must be multiplexed together with data transitioning on both edges of the clock. In addition, during a write cycle, DQS must arrive at the memory pins center-aligned with data, DQ. Along with the DQS strobe and DQ, CLKP, CLKN, Address/Command and Data Mask (DM) signals also need to be generated. IPexpress should be used to generate this interface. See the section [DDR Memory Interface Generation Using IPexpress](#) for details.

Challenges encountered during memory write:

- Differential CLK signals (CLKP and CLKN) need to be generated.
- Generate ADDR/CMD signal edge-aligned to CLKP falling edge.
- DQS needs to be edge-aligned to CLKP. In DDR3 interfaces where fly-by routing is used, write leveling should be used to compensate for skews between CLKP and DQS.
- DQ/DM needs to be center-aligned to DQS and therefore center-aligned to CLKP as well.
- The controller must meet the DDR interface specification for the  $t_{DSS}$  and  $t_{DSH}$  parameters, defined as DQS falling edge setup and hold time to/from CLKP rising edge, respectively. The skews, if caused by the fly-by topology, are compensated by write-leveling.
- The DDR output data must be muxed from two SDR streams into a single outgoing DDR data stream. For the DDR3 memory interface, this DDR output data is generated from four SDR streams.

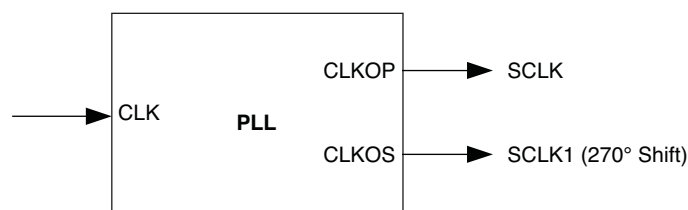
The DDR/DDR2 memory write interface is implemented using the following modules:

- **IDDRXD, ODDRXD, OFD1S3AX** – Used for DDR2 DQ input, output and tri-state control. DDR2 DM uses ODDRXD and OFD1S3AX only.
- **ODDRXDQSA, ODDRTDQSA** – Used for DDR2 DQS output and tri-state control.
- **DQSBUFF** – Dedicated for DDR2 memory interface application.
- **DQSBUFG** – Generic ODDRXD support for clocks on “E” devices.
- **ODDRXD** – Generic DDR mode for clocks on “E” devices.
- **ODDRXD1** – Generic DDR mode for clocks on “EA” devices.

### DDR/DDR2 Internal Clock Generation

LatticeECP3 devices require two clocks to implement a DDR/DDR2 memory interface. SCLK ( $k\_clk$ ) is used to generate the data and data strobe signals while a 270° shifted clock, SCLK1 ( $k1\_clk$ ), is needed to generate the memory clock and address/command signals. Figure 12-57 shows the generation of these clocks using a PLL. SCLK is also used for the read side implementation as described in Figure 12-55.

**Figure 12-57. DDR/DDR2 Internal Clock Generation**



### DDR/DDR2 Memory Write Implementation

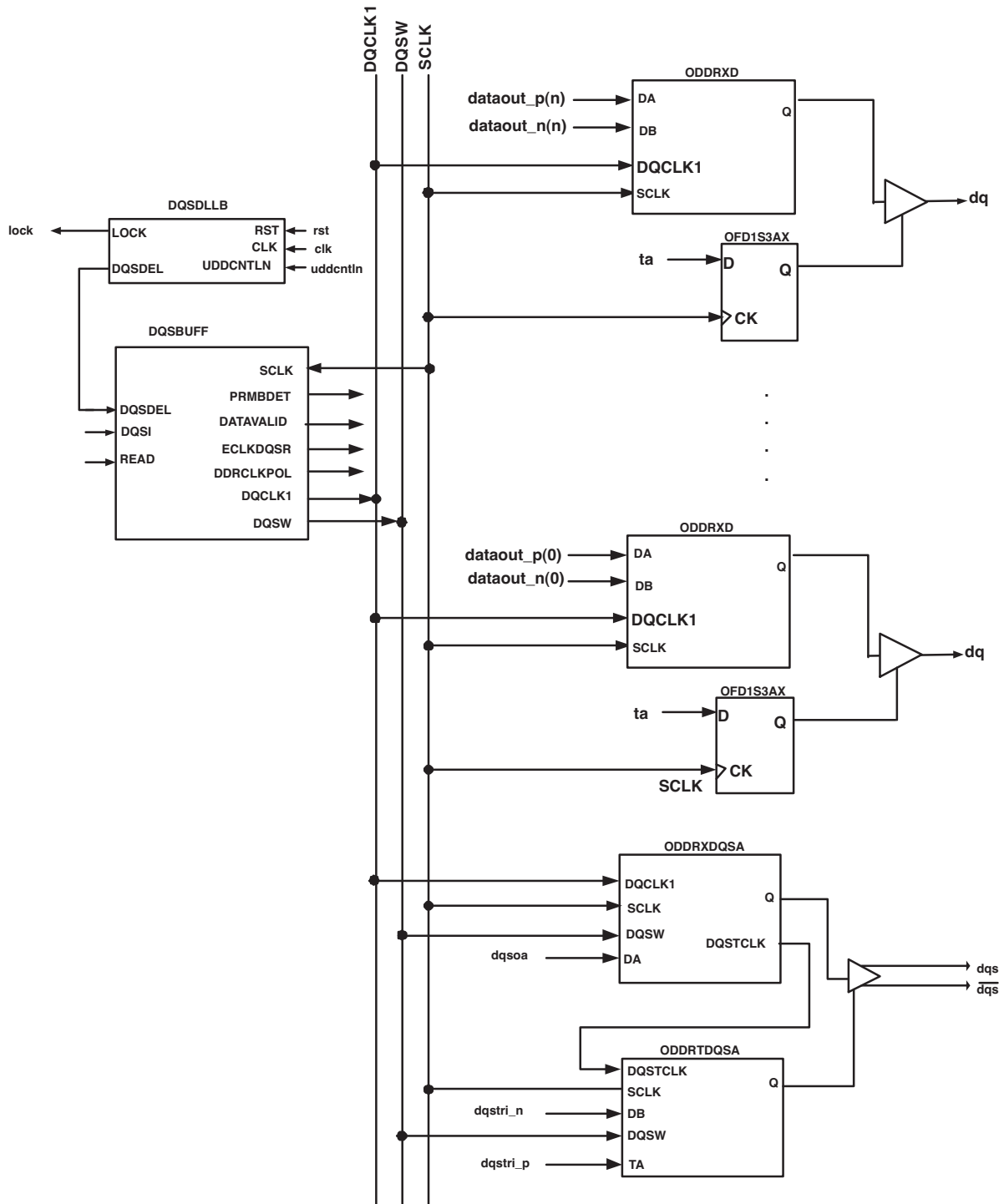
The following sections explain the DDR/DDR2 write side implementation. LatticeECP3 devices support the DDR/DDR2 memory interface function using the DDR memory mode module generated through the IPexpress tool. Using IPexpress, a designer may generate the Data (DQ), Strobe (DQS), Data Mask, Clock (CLKP/CLKN), Address/Command (ADDR/CMD) signals required when writing to the DDR memory. See the section [DDR Memory Interface Generation Using IPexpress](#) for details on the IPexpress Interface. This section explains the different Write Side modules generated by IPexpress.

#### DDR/DDR2 Write Side Data (DQ) and Strobe (DQS) Generation

Figure 12-58 shows the DDR2 write side for data and strobe generation. The DQCLK1 and DQSW are signals generated in the DQSBUFF module. DQCLK0 is not used for the DDR2 memory interface. For details on each element, refer to the [DDR Software Primitives and Attributes](#) section. The DDR2 memory interface does not require

write leveling, therefore the DYNDELAY [6:0] and DYNDELPOL are not used here. The DDR write side will use the same implementation except that the DQS signal in this case is single-ended.

**Figure 12-58. DDR/DDR2 Memory Write DQ and DQS Generation (“E” and “EA” Devices)**



**DDR/DDR2 Write Side Clock (CLKP/CLKN) Generation**

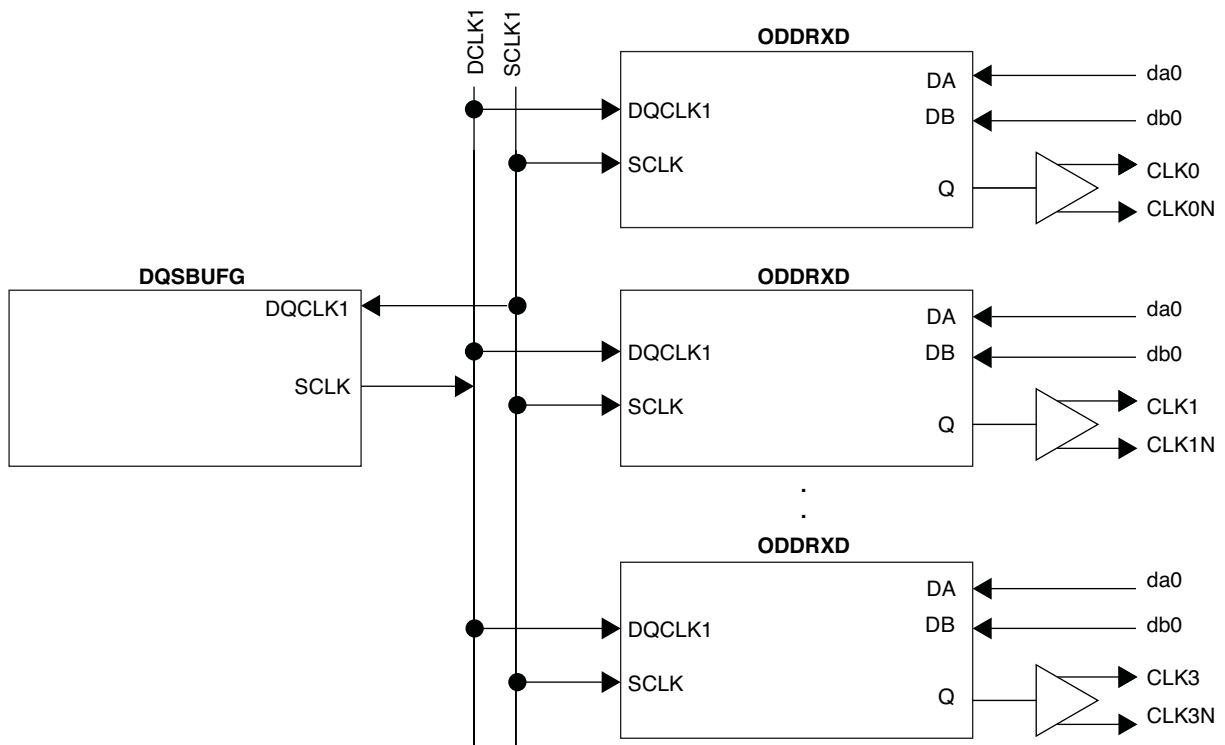
The figures below show the DDR clock output (CLKP/CLKN) generation. The 270° shifted SCLK1 is used to generate the DDR clock outputs. See the section [DDR/DDR2 Internal Clock Generation](#) for details. On the LatticeECP3



“E” devices the ODDRXD primitive is used to generate the clocks. The left and right sides of the “E” devices also require the DQSBUFG which is used to generate the DQCLK1 required in the ODDRXD module. The top side of the “E” device does not require the use of DQSBUFG and the DQCLK1 input of the ODDRXD can be tied to SCLK. The “EA” device uses the ODDRXD1 element instead. “EA” devices do not require the use of the DQSBUFG element.

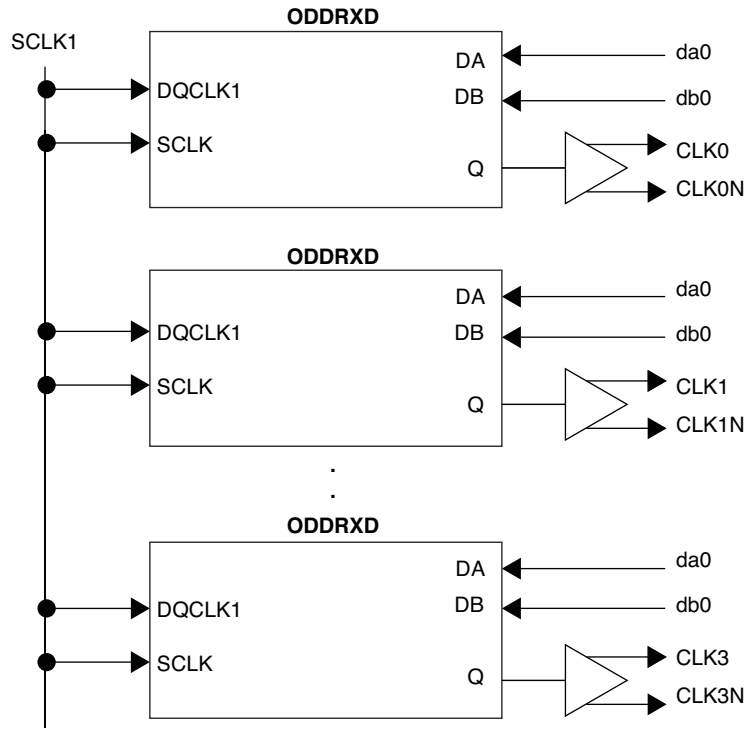
The inputs to the ODDRXD/ODDRXD1 are tied to constant values to generate a clock out of the ODDRXD/ODDRXD1. When interfacing to the DDR SDRAM memory, CLKP should be connected to the SSTL25D I/O Standard and when interfacing to DDR2 memory it should be connected to SSTL18D I/O Standard to generate the differential clock outputs. Generating the CLKN in this manner will prevent skew between the two signals.

**Figure 12-59. DDR/DDR2 Write Clock Generation (“E” Devices, Left/Right/Top Sides)**



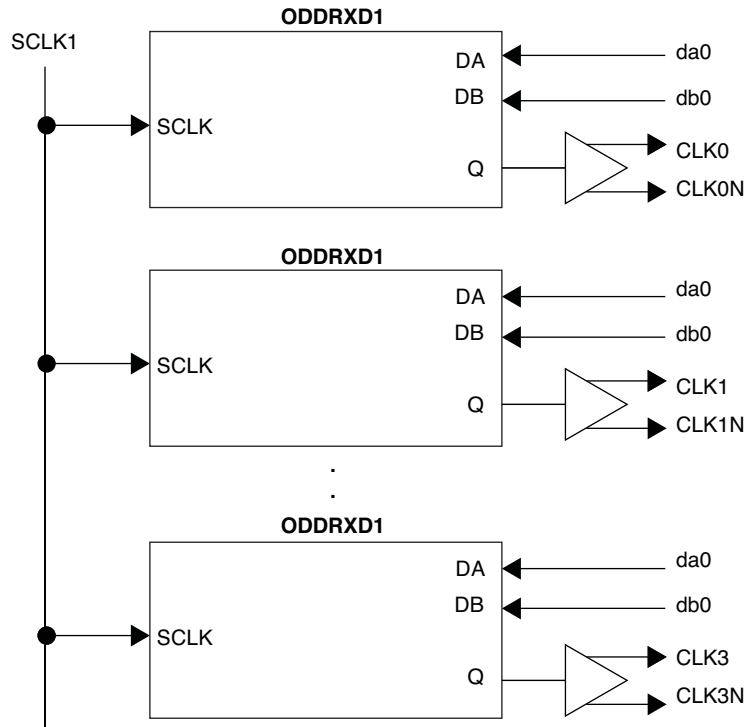
Note: (SCLK1=SCLK+270, DA0=0, DB0=1)

Figure 12-60. DDR/DDR2 Write Clock Generation (“E” Devices, Top Side)



Note: (SCLK1=SCLK+270, DA0=0, DB0=1)

Figure 12-61. DDR/DDR2 Write Clock Generation (“EA” Devices)

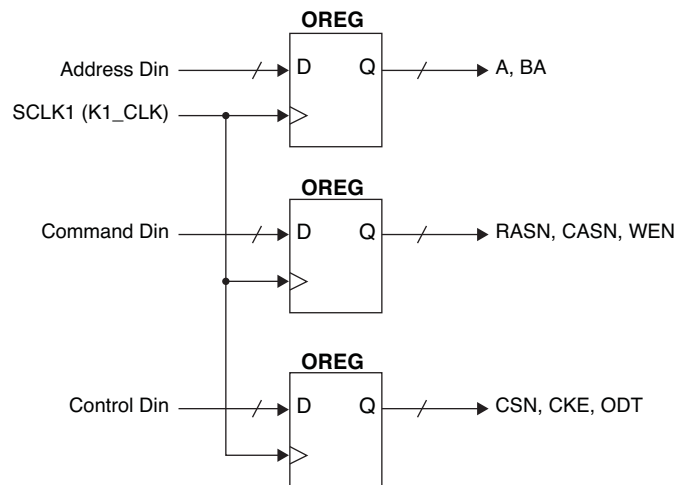


Note: (SCLK1=SCLK+270, DA0=0, DB0=1)

DDR/DDR2 Address/Command Generation

Figure 12-62 shows the address and command generation for the LatticeECP3 “E” and “EA” devices. The 270° shifted SCLK is used for address and command generation. See the section [DDR/DDR2 Internal Clock Generation](#) for details. SDR registers are used to generate the address and command signals.

Figure 12-62. DDR/DDR2 Address / Command Implementation (“E” and “EA” Devices)



Note: A: Address; BA: Bank Address; RASN: RAS; CASN: CAS; WEN: Write Enable; CSN: Chip Select; CKE: Clock Enable; ODT: On-Die Termination.

---

### DDR3 Memory Write Implementation

The following sections explain the DDR3 write side implementation. LatticeECP3 devices support the DDR3 memory interface function using the DDR memory mode module generated through the IPexpress tool. Using IPexpress, a designer may generate the Data (DQ), Strobe (DQS), Data Mask, Clock (CLKP/CLKN) and Address/Command (ADDR/CMD) signals required when writing to the DDR3 memory. See the section [DDR Memory Interface Generation Using IPexpress](#) for details on the IPexpress Interface. This section explains the different write side modules generated by IPexpress.

The DDR3 memory interface generated in IPexpress also includes a Clock Synchronization Module (CSM) that provides the clock synchronization and alignment among the required clocks for successful DDR3 functionality.

#### DDR3 Data (DQ) and Strobe (DQS) Generation

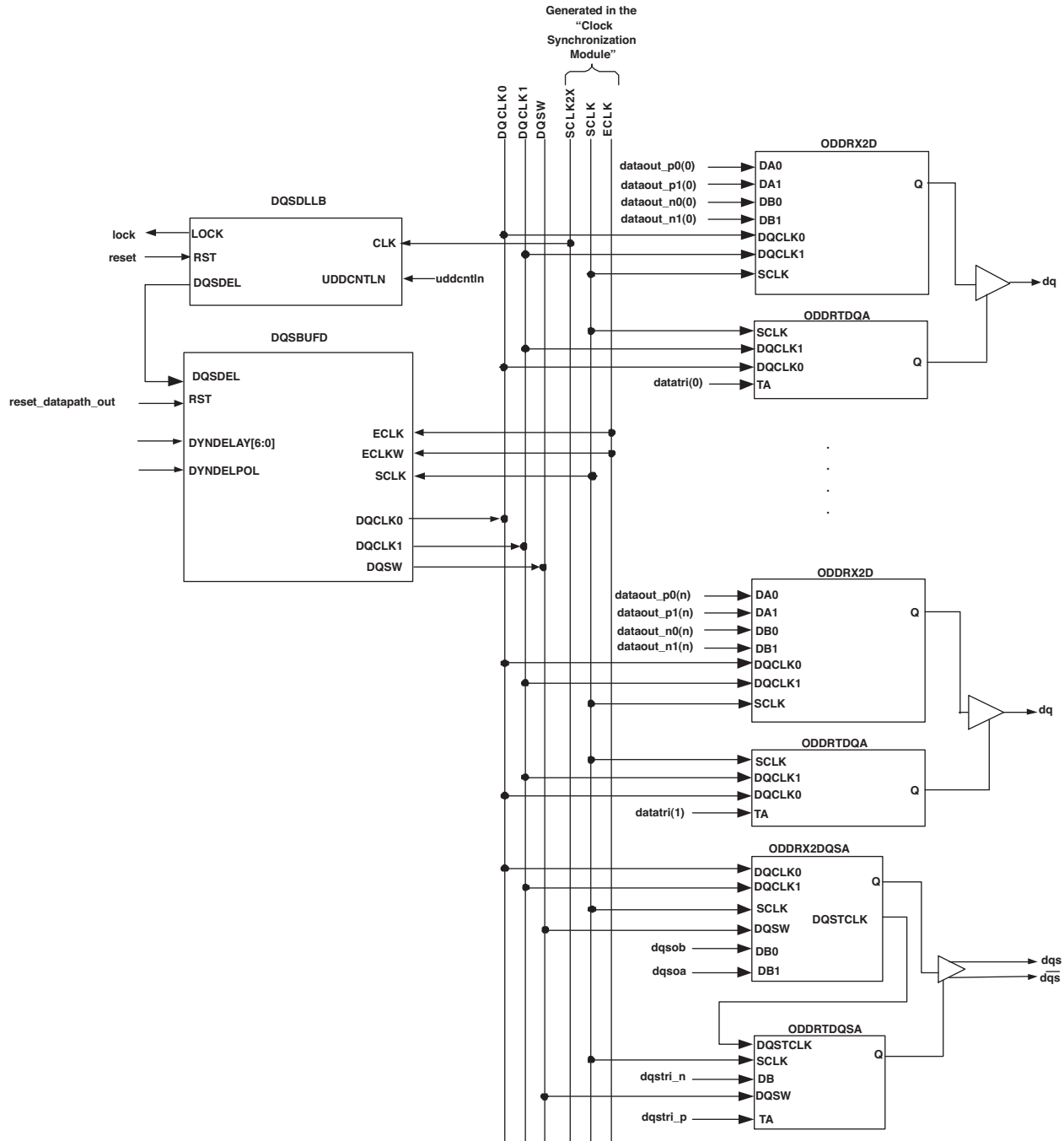
Figure 12-63 shows the DDR3 write side implementation for data (DQ) and strobe (DQS) generation. The DQCLK0, DQCLK1 and DQSW are signals generated in the DQSBUFD module. ODDR2DQSA implements the output DDR registers in x2 gearing mode required for DDR3 DQ generation. The ODDR2DQSA module is used to generate the DQS strobe output. For details on each element, refer to the [DDR Software Primitives and Attributes](#) section.

The DDR3 memory interface requires write leveling which is supported with the DYNDELAY [6:0] and DYNDEL-POL inputs of DQSBUFD. Users can provide the delays for each DQS group using these ports. They are available to the user as ports in the top-level module generated by IPexpress.

The ECLK, SCLK2X and SCLK are generated in a clock synchronization module (CSM) that provides the clock synchronization and alignment among the required clocks for successful DDR3 functionality. See the DDR3 Clock Synchronization Module section for details on the Clock Synchronization Module.

IPexpress is used to generate these signals. Figure 12-63 shows the module generated by IPexpress data (DQ) and strobe (DQS) generation.

Figure 12-63. DDR3 Write Side Implementation for DQ and DQS Generation



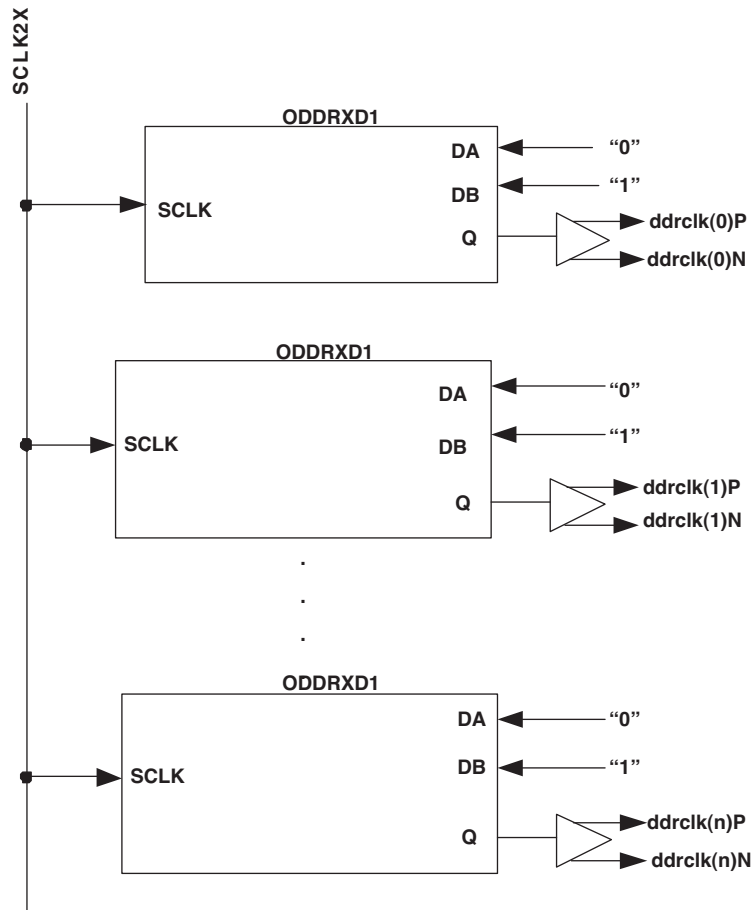
**DDR3 Write Side Clock (CLKP/CLKN) Generation**

The SCLK2X out of the Clock Synchronization Module is used to generate the DDR clock outputs. See the section DDR3 Clock Synchronization Module for details.

The ODDRXD1 element is used to generate the clock. The inputs to the ODDRXD1 are tied to constant values to generate a clock out of the ODDRXD1. When interfacing to the DDR3 SDRAM memory, CLKP (ddrclkP) should be connected to the SSTL15D I/O standard to generate the differential clock outputs. Generating the CLKN (ddrclkN) in this manner will prevent skew between the two signals

IPexpress is used to generate these signals. Figure 12-64 shows the module generated by IPexpress for DDR clock outputs (CLKP/CLKN).

**Figure 12-64. DDR3 Write Side Clock Generation**

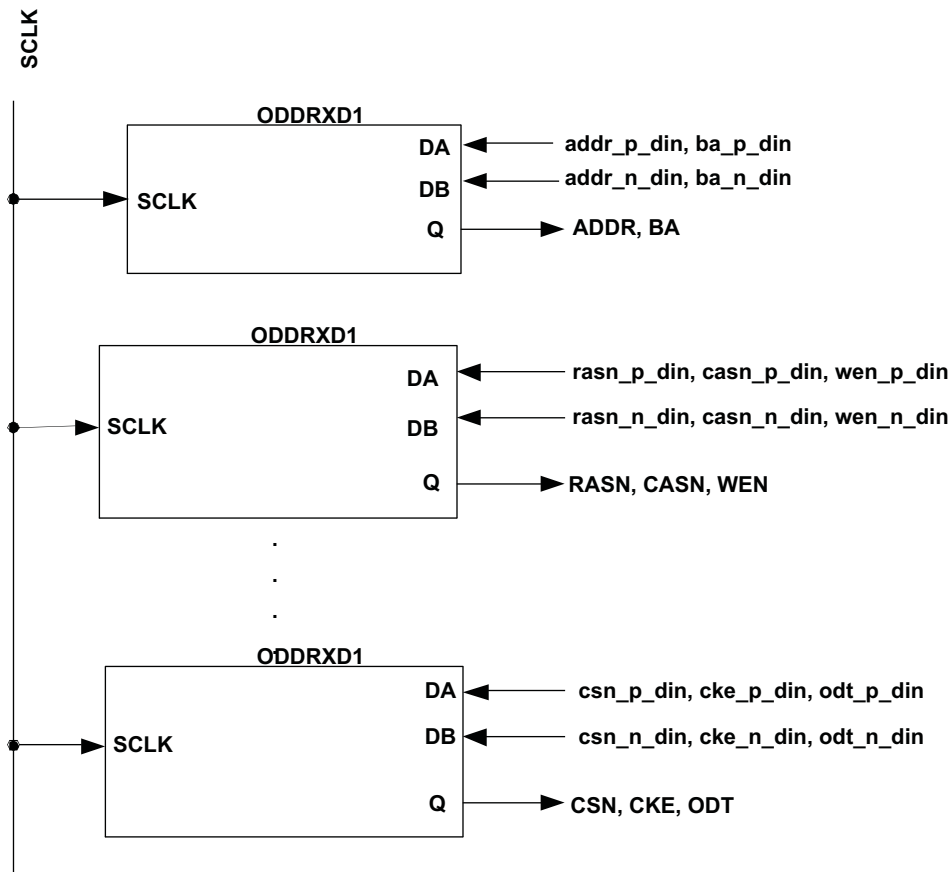


**DDR3 Write Side Address/Command (ADDR/CMD) Generation**

The SCLK output of the Clock Synchronization Module, along with the ODDRXD1 element, is used for address and command generation. See the section DDR3 Clock Synchronization Module for details on how the SCLK is generated.

IPexpress is used to generate these signals. Figure 12-65 shows the module generated by IPexpress for address and command signals.

Figure 12-65. DDR3 Write Side Address/Command Generation



### DDR3 Clock Synchronization Module

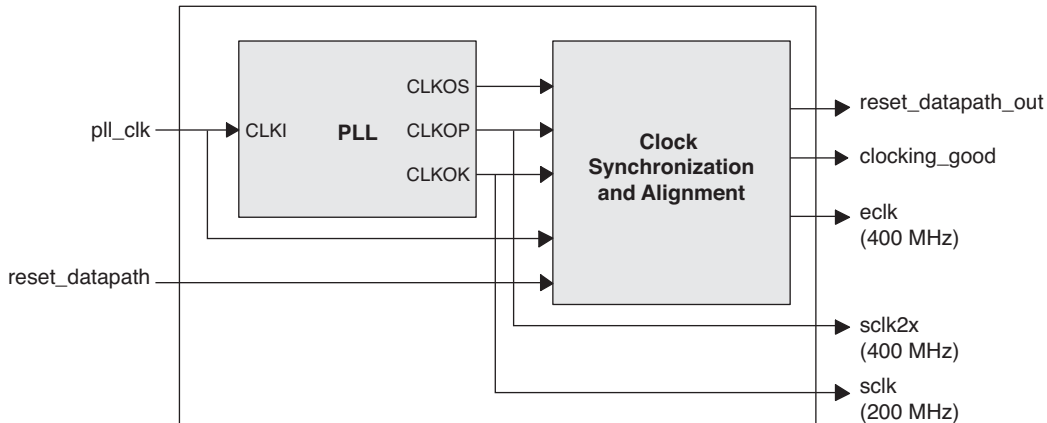
The DDR3 memory interface generated in IPexpress includes a clock synchronization module (CSM) that provides the clock synchronization and alignment among the required clocks for successful DDR3 functionality. A DDR3 memory interface that is implemented in LatticeECP3 must use the CSM block for correct read data transfer from the ECLK to SCLK domain. It is also used to provide the proper clock relationship between SCLK and DQCLK0/1 used for write data generation.

The CSM block generates the ECLK, SCLK2X and SCLK clocks to be used in the read and write side implementations. It generates a clocking\_good output to indicate that all the clocks for DDR3 operation have been synchronized and the device is ready to serve the DDR3 operation. The DDR3 memory interface and controller should wait until the clocking\_good signal is detected as high.

The DDR3 mode supports the write leveling feature by providing the dynamic delay control ports inputs (DYNDELAY) of DQSBUFD to users. In order for the CSM block to support write leveling, the reset\_datapath input port should be asserted high for at least one PLL input clock cycle (pll\_clk) to reset the DQSBUFD blocks through the reset\_datapath\_out signal after a write leveling operation is finished. The reset\_datapath\_out signal is directly connected to the reset (RST) port of all DQSBUFD.

Figure 12-66 shows the block diagram of the CSM block.

**Figure 12-66. DDR3 Clock Synchronization Module**



**PLL Settings for Clock Synchronization Module:** LatticeECP3 devices allow two dedicated PLL blocks per side to support the DDR3 memory interface applications. One of the four available PLLs (two PLLs in the left side and two PLLs in the right side) must be used to implement the CSM block. Since the PLL input clock is also used to control the synchronization logic, the use of a PLL input clock that is too fast may cause failures in PAR timing results. Therefore, it is recommended to use the 1:2:4 clock multiplication ratios for high-speed DDR3 memory interface applications as shown in the example of a 400MHz/800Mbps DDR3 memory interface below. A dedicated PLL input clock pad or a PCLK pad can be used to drive the selected PLL.

**Table 12-8. PLL Settings for 400 MHz DDR3 Operation**

	PLL Pin Name	Clock Output	Speed (400MHz)	Speed (300MHz)
Input Clock	CLKI	pll_clk	100 MHz	75 MHz
Output Clocks	CLKOP	sclk2x	400 MHz	300 MHz
	CLKOS	eclk	400 MHz	300 MHz
	CLKOK	sclk	200 MHz	150 MHz
PLL Divider Setting	CLKOK_DIV=2, CLKOP_DIV=2, CLKFB_DIV=4, CLKI_DIV=1			

**Timing Preferences for Clock Synchronization Module:** Since the CSM block includes a few timing-sensitive nets that affect the performance of DDR3 I/O functions, the route delays on these nets should be tightly controlled. To consistently guarantee successful routing results, these nets should be constrained by the MAXDELAY preferences. The nets shown in Table 12-9 include the net delay preferences to force the software Place and Route (PAR) tool to meet the required net delays.

**Table 12-9. MAXDELAY NET Preferences for Clock Synchronization Module**

Preference	Net Name	-8 Device	-7 Device	-6 Device
MAXDELAY NET	eclk	1.2 ns	1.3 ns	1.45 ns
MAXDELAY NET	stop	0.8 ns	0.85 ns	0.9 ns
MAXDELAY NET	clkos (PLL 1)	1.1 ns	1.2 ns	1.35 ns
MAXDELAY NET	clkos (PLL 2)	0.65 ns	0.75 ns	0.8 ns
MAXDELAY NET	dqclk1bar_ff	0.65 ns	0.7 ns	0.75 ns

The preferences for the constrained PAR targets are located in the HDL module generated by IPexpress. It is automatically transferred to the project preference file (.prf) to achieve the optimum timing results.

The example shown below includes the target preferences with the following conditions:



- DDR3 module's instance name is "u\_ddr3"
- Speed grade -8 device is targeted
- PLL location 1 is selected

```
MAXDELAY NET "[path]/eclk" 1.200000 nS;
MAXDELAY NET "[path]/u_ddr3/stop" 0.800000 nS;
MAXDELAY NET "[path]/u_ddr3/clkos" 1.100000 nS;
MAXDELAY NET "[path]/u_ddr3/Inst8_clk_phase/dqclk1bar_ff" 0.650000 nS;
```

**Locate Preferences for Clock Synchronization Module:** The CSM block also requires several manual locations for the clock resources in the module like PLL, ECLKSYNC etc. These modules are all generated by IPexpress and include an UGROUP attribute added to the modules. The corresponding PGROUP (physical grouping) should be locked to the specific sites as shown in Table 12-10.

**Table 12-10. DDR3 Clock and PGROUP Locations for CSM Support**

	LatticeECP3-150EA 1156				LatticeECP3-150EA 672			
	Left 1	Left 2	Right 1	Right 2	Left 1	Left 2	Right 1	Right 2
PLL input clock pad	U6	Y9	V34	Y28	M3	U4	T21	V20
PLL	R61C5	R79C5	R61C178	R79C178	R61C5	R79C5	R61C178	R79C178
ECLKSYNC	L2	L1	R2	R1	L2	L1	R2	R1
PGROUP clk_phase0	R50C5D	R78C5D	R50C178D	R78C178D	R50C5D	R78C5D	R50C178D	R78C178D
PGROUP clk_phase1a	R60C2D	R60C2D	R60C181D	R60C181D	R60C2D	R60C2D	R60C181D	R60C181D
PGROUP clk_phase1b	R60C2D	R60C2D	R60C181D	R60C181D	R60C2D	R60C2D	R60C181D	R60C181D
PGROUP clk_stop	R60C2D	R60C2D	R60C180D	R60C180D	R60C2D	R60C2D	R60C180D	R60C180D
	LatticeECP3-95/70EA 1156				LatticeECP3-95/70EA 672			
	Left 1	Left 2	Right 1	Right 2	Left 1	Left 2	Right 1	Right 2
PLL input clock pad	U6	Y9	V34	Y28	M3	U4	T21	V20
PLL	R43C5	R61C5	R43C142	R61C142	R43C5	R61C5	R43C142	R61C142
ECLKSYNC	L2	L1	R2	R1	L2	L1	R2	R1
PGROUP clk_phase0	R32C5D	R60C5D	R32C142D	R60C142D	R32C5D	R60C5D	R32C142D	R60C142D
PGROUP clk_phase1a	R42C2D	R42C2D	R42C145D	R42C145D	R42C2D	R42C2D	R42C145D	R42C145D
PGROUP clk_phase1b	R42C2D	R42C2D	R42C145D	R42C145D	R42C2D	R42C2D	R42C145D	R42C145D
PGROUP clk_stop	R42C2D	R42C2D	R42C144D	R42C144D	R42C2D	R42C2D	R42C144D	R42C144D
	LatticeECP3-95/70EA 484				LatticeECP3-35EA 672			
	Left 1	Left 2	Right 1	Right 2	Left 1	Left 2	Right 1	Right 2
PLL input clock pad	L5	T3	M18	R17	M3	U4	T21	V20
PLL	R43C5	R61C5	R43C142	R61C142	R35C5	R53C5	R35C70	R53C70
ECLKSYNC	L2	L1	R2	R1	L2	L1	R2	R1
PGROUP clk_phase0	R32C5D	R60C5D	R32C142D	R60C142D	R24C5D	R52C5D	R24C70D	R52C70D
PGROUP clk_phase1a	R42C2D	R42C2D	R42C145D	R42C145D	R34C2D	R34C2D	R34C73D	R34C73D
PGROUP clk_phase1b	R42C2D	R42C2D	R42C145D	R42C145D	R34C2D	R34C2D	R34C73D	R34C73D
PGROUP clk_stop	R42C2D	R42C2D	R42C144D	R42C144D	R34C2D	R34C2D	R34C72D	R34C72D
	LatticeECP3-35EA 484				LatticeECP3-35EA 256			
	Left 1	Left 2	Right 1	Right 2	Left 1	Left 2	Right 1	Right 2
PLL input clock pad	L5	T3	M18	R17	K3	P2	K14	T15
PLL	R35C5	R53C5	R35C70	R53C70	R35C5	R53C5	R35C70	R53C70
ECLKSYNC	L2	L1	R2	R1	L2	L1	R2	R1
PGROUP clk_phase0	R24C5D	R52C5D	R24C70D	R52C70D	R24C5D	R52C5D	R24C70D	R52C70D

**Table 12-10. DDR3 Clock and PGROUP Locations for CSM Support (Continued)**

PGROUP clk_phase1a	R34C2D	R34C2D	R34C73D	R34C73D	R34C2D	R34C2D	R34C73D	R34C73D
PGROUP clk_phase1b	R34C2D	R34C2D	R34C73D	R34C73D	R34C2D	R34C2D	R34C73D	R34C73D
PGROUP clk_stop	R34C2D	R34C2D	R34C72D	R34C72D	R34C2D	R34C2D	R34C72D	R34C72D
	<b>LatticeECP3-17EA 484</b>				<b>LatticeECP3-17EA 256</b>			
	Left 1	Left 2	Right 1	Right 2	Left 1	Left 2	Right 1	Right 2
PLL input clock pad	L5	NA	M18	NA	K3	NA	K14	NA
PLL	R26C5	NA	R26C52	NA	R26C5	NA	R26C52	NA
ECLKSYNC	L2	NA	R2	NA	L2	NA	R2	NA
PGROUP clk_phase0	R15C5D	NA	R15C52D	NA	R15C5D	NA	R15C52D	NA
PGROUP clk_phase1a	R25C2D	NA	R25C55D	NA	R25C2D	NA	R25C55D	NA
PGROUP clk_phase1b	R25C2D	NA	R25C55D	NA	R25C2D	NA	R25C55D	NA
PGROUP clk_stop	R25C2D	NA	R25C54D	NA	R25C2D	NA	R25C54D	NA

The PLL input pad, the corresponding PLL site and the legal ECLKSYNC site for the selected PLL are constrained in the user preference file (.lpf) as shown in the example below:

```
LOCATE COMP "clk_in" SITE "U6";
LOCATE COMP "[path]/u_ddr3/Inst1_EHXPLLF" SITE "PLL_R61C5";
LOCATE COMP "[path]/u_ddr3/Inst6_ECLKSYNCA" SITE "LECLKSYNC2";
```

Locating the defined PGROUPs is crucial for successful DDR3 operation. The PGROUPs should be manually located according to the table as shown below:

```
LOCATE PGROUP "clk_phase0" SITE "R59C3D";
LOCATE PGROUP "clk_phase1" SITE "R59C2D";
LOCATE PGROUP "clk_stop" SITE "R60C2D";
LOCATE PGROUP "rst_dp_out" SITE "R60C4D";
```

The following block paths should also be included to help avoid unnecessary domain crossing nets being reported in the trace report as false alarms.

```
[False paths for PAR and TRACE]
BLOCK PATH FROM CLKNET "pll_clk" TO CLKNET "sclk" ;
BLOCK PATH FROM CLKNET "pll_clk" TO CLKNET "**/clkos" ;
BLOCK PATH FROM CLKNET "sclk" TO CLKNET "pll_clk" ;
BLOCK PATH FROM CLKNET "**sclk2x" TO CLKNET "pll_clk" ;
BLOCK PATH FROM CLKNET "pll_clk" TO CLKNET "**eclk" ;
BLOCK PATH FROM CLKNET "**/clkos" TO CLKNET "**eclk" ;
BLOCK PATH FROM CLKNET "**/clkos" TO CLKNET "sclk" ;
BLOCK PATH FROM CLKNET "**sclk2x" TO CLKNET "**/clkos" ;
```

*Note: The clock net names and hierarchy path names may be changed after synthesis. If this happens, updates on the preferences to follow the changed names are required.*

## DDR Memory Interface Generation Using IPexpress

The IPexpress tool is used to configure and generate the DDR, DDR2 and DDR3 memory interfaces. This section assumes that ispLEVER 8.0 SP1 is used for generation of the interfaces. If you are using ispLEVER 7.2 SP2, see [Appendix A. Building DDR Interfaces Using IPexpress in ispLEVER 7.2 SP2](#). If you are using Lattice Diamond design software, see [Appendix B. Building SDR/DDR Interfaces Using IPexpress in Diamond](#).

To see the detailed block diagram for each interface generated by IPexpress see the [Memory Read Implementation](#) and [Memory Write Implementation](#) sections.

IPexpress can be opened from the **Tools** menu in Project Navigator. All the DDR modules are located under **Architecture Modules > IO**. DDR\_MEM is used to generate DDR memory interfaces.

**Figure 12-67. IPexpress Main Window**

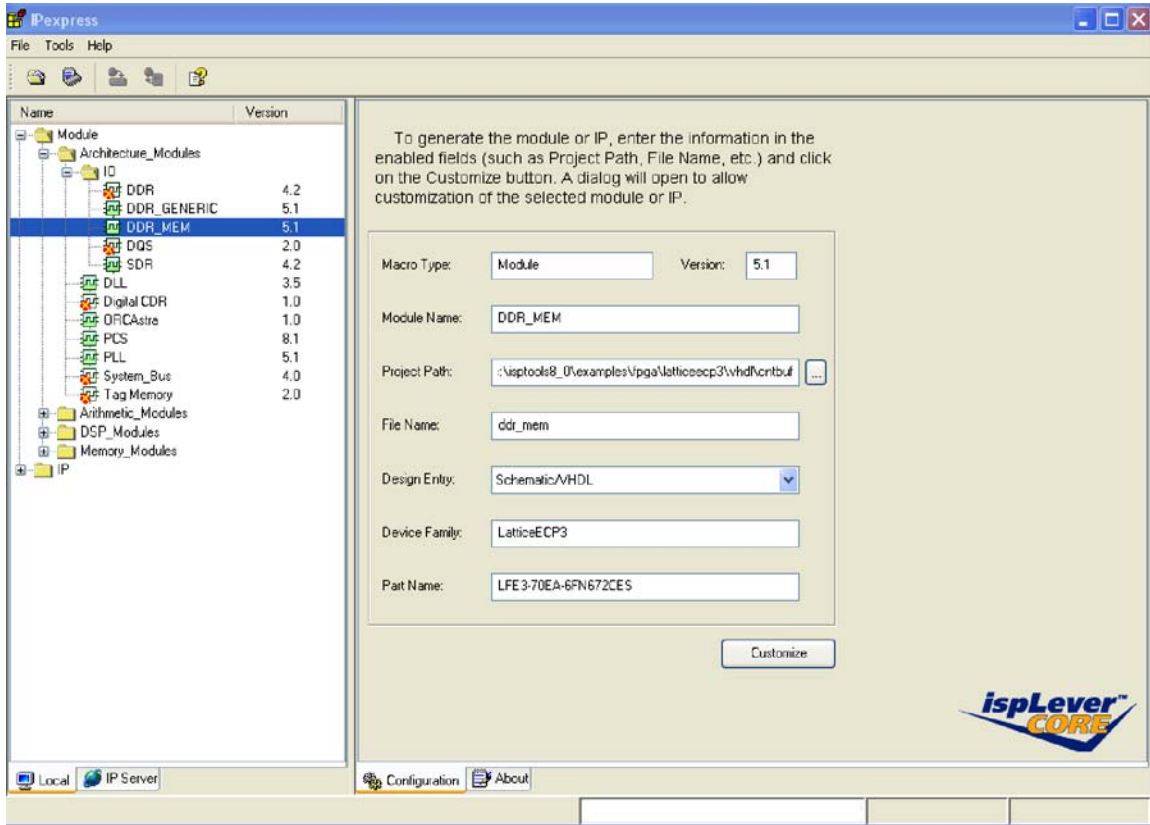


Figure 12-67 shows the IPexpress Main Window. To generate a DDR memory interface, select DDR\_MEM, assign a module name and click on **Customize** to see the Configuration Tab.

Figure 12-68 shows the Configuration Tab for the DDR\_MEM interface. You can choose to implement the DDR1\_MEM, DDR2\_MEM or DDR3\_MEM interface.

Figure 12-68. Configuration Tab for DDR\_MEM

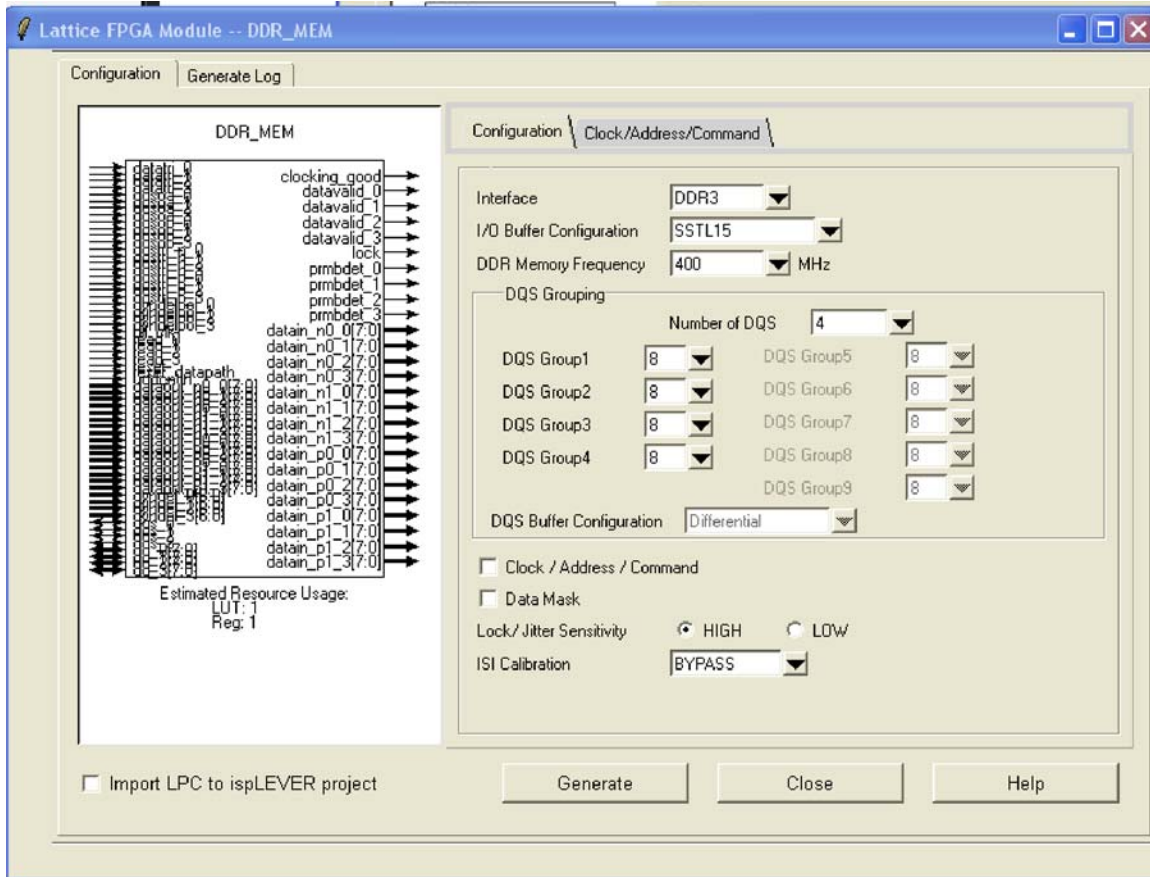


Table 12-11 describes the various settings shown in the Configuration Tab above.

Table 12-11. Configuration Tab Settings for DDR\_MEM

GUI Option	Description	Range	Default Value
Interface	DDR memory interface type	DDR, DDR2, DDR3	DDR2
I/O Buffer Configuration	I/O type configuration for DDR pins	SSTL25_I, SSTL25_II SSTL18_I, SSTL18_II, SSTL15	DDR – SSTL25_I DDR2 – SSTL18_I DDR3 – SSTL15
Number of DQS	Interface width (1 DQS per 8 bits of data)	1 to 9	4
DQS Group1 to DQS Group8	Number of DQ per DQS pin	1 to 8	8
DQS Buffer Configuration	DQS buffer type	DDR: Single-ended DDR2: Single-ended, Differential DDR3: Differential	DDR – Single-ended DDR2 – Single-ended DDR3 – Differential
Clock/Address/Command	Clock/address/command interface will be generated when this option is checked	ENABLED, DISABLED	DISABLED
Data Mask	Data mask signal will be generated when this option is checked	ENABLED, DISABLED	DISABLED
Lock/Jitter Sensitivity	Lock Sensitivity attribute for DQSDLL <sup>1</sup>	HIGH, LOW	HIGH

**Table 12-11. Configuration Tab Settings for DDR\_MEM (Continued)**

GUI Option	Description	Range	Default Value
DDR Memory Frequency	DDR Memory Interface Frequency	DDR – 87.5 MHz, 100 MHz, 133.33 MHz, 166.67 MHz, 200 MHz DDR2 – 125 MHz, 200 MHz, 266.67 MHz DDR3 – 300 MHz, 400 MHz	DDR – 200 MHz DDR2 – 200 MHz DDR3 – 400 MHz
ISI Calibration	ISI calibration is available for the DDR3 interface to adjust for inter-symbol interference adjustment per DQS group	BYPASS, DEL1, DEL2, DEL3, DEL4, DEL5, DEL6, DEL7	BYPASS

1. It is recommended to set Lock Sensitivity to HIGH for DDR Memory Frequency higher than 133 MHz.

If the user selects to generate the Clock/Address/Command signals using IPexpress, then the settings in the Clock/Address/Command Tab are active and can be set up as required. Figure 12-69 shows the Clock/Address/Command Tab in the IPexpress for DDR2 Memory.

**Figure 12-69. Clock/Address/Command Tab in the IPexpress for DDR\_MEM**

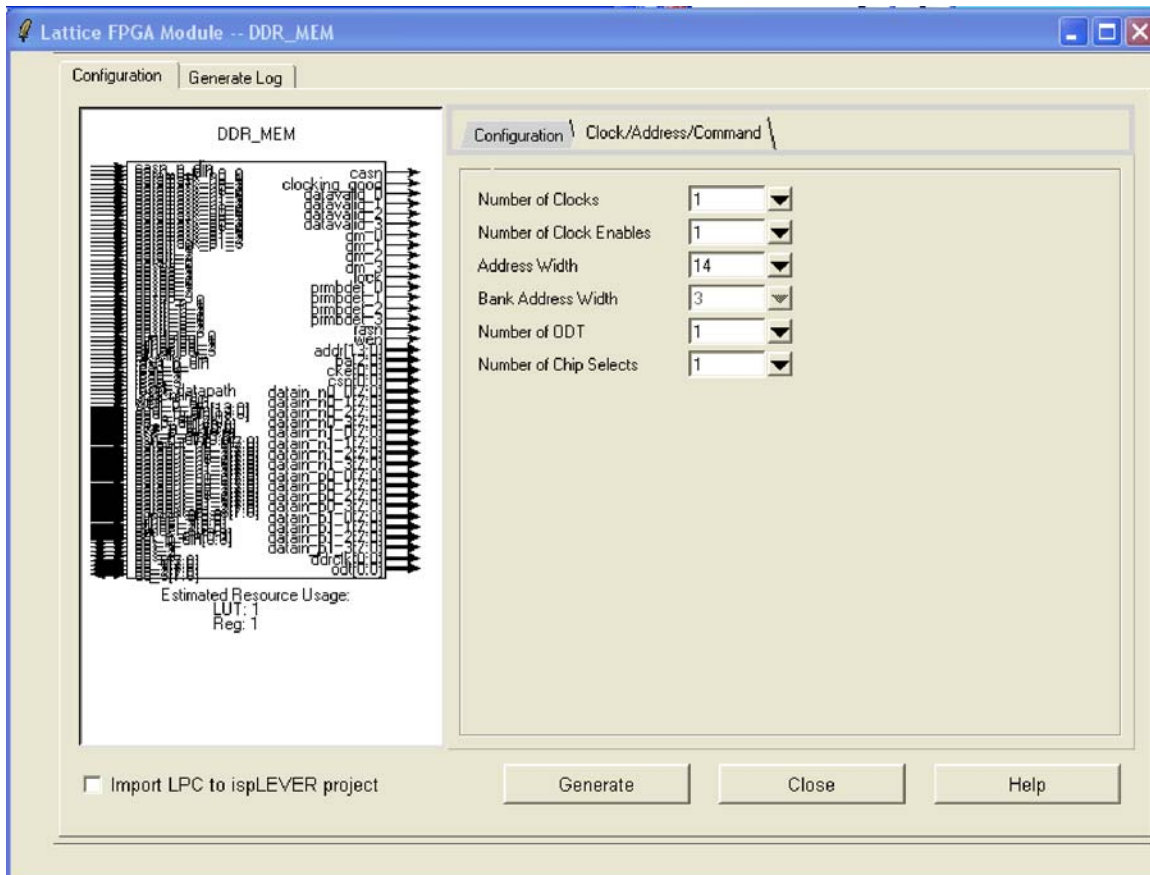


Table 12-12 lists the values that can be used for the Clock/Address/Command settings.

**Table 12-12. Clock/Address/Command Settings for DDR\_MEM**

GUI Option	Range	Default Value
Number of Clocks	1, 2, 4	1
Number of Clock Enables	1, 2, 4	1
Address Width	DDR: 12-14 DDR2: 13-16 DDR3: 13-16	DDR: 13 DDR2: 13 DDR3: 14
Bank Address Width	DDR: 2 DDR2: 2, 3 DDR3: 3	DDR: 2 DDR2: 2 DDR3: 3
Number of ODT	DDR: N/A DDR2: 1, 2, 4 DDR3: 1, 2, 4	DDR: N/A DDR2: 1 DDR3: 1
Number of Chip Selects	DDR: 1, 2, 4, 8 DDR2: 1, 2, 4 DDR3: 1, 2, 4	DDR: 1 DDR2: 1 DDR3: 1

## DDR Memory DQ/DQS Design Rules and Guidelines

Listed below are some rules and guidelines to keep in mind when implementing DDR memory interfaces in LatticeECP3 devices.

- LatticeECP3 devices have dedicated DQ-DQS banks. Please refer to the Logical Signal Connections tables in the [LatticeECP3 Family Data Sheet](#) before locking these pins.
- There are two DQSDLLs on the device, one for the left half and one for the right half of the device. Only one DQSDLL primitive should be instantiated for each half of the device. Since there is only one DQSDLL on each half, all the DDR memory interfaces on that half should run at the same frequency.
- Each DQSDLL will generate 90° digital delay bits for all the DQS delay blocks on that half of the device based on the reference clock input to the DLL.
- The clock to the PLL used in the write implementation to generate the clocks for the outputs must be locked to the correct dedicated PLL pin input.
- When implementing a DDR SDRAM interface, all interface signals should be connected to the SSTL25 I/O standard.
- For the DDR2 SDRAM interface, the interface signal should be connected to the SSTL18 I/O standard.
- For the DDR3 SDRAM interface, these signals should be connected to the SSTL15 standard.
- The DDR, DDR2 and DDR3 require a differential clock signal. For these standards, the differential clock signals should be connected to SSTL25D (DDR), SSTL18D (DDR2) or SSTL15D (DDR3).
- DDR3 also requires differential DQS signal. The use of differential DQS is optional for DDR2. If differential DQS is used it should be connected to SSTL18D for DDR2 and SSTL15D for DDR3.
- When implementing the DDR interface, the VREF1 of the bank is used to provide the reference voltage for the interface pins. VREF1 should not be connected with VREF2 of the bank when implementing DDR memory interfaces.
- There is no DQS strobe support on the bottom of the device, so memory interfaces cannot be implemented on this side.
- Within a DQS-12 group, the IOLOGIC in the group's DQS/DQS# pads cannot be used for DDR registers.
- If the register is implemented inside the FPGA fabric instead of the IOLOGIC, there is no restriction on using the DQS pad.



- The upper left corner of the LatticeECP3 device has a non-DQS DDR group. This group of I/Os does not have a DQS function. This group of I/Os can only be used for generic DDR implementations.
- IDDRX memory cannot be mixed in the same DQS group as an ODDRX generic implementation. Similarly, ODDRX memory cannot be mixed in the same DQS group with an IDDRX generic implementation.
- Generic DDR interfaces are not available on the top side of the LatticeECP3 “E” device only. The generic DDR required for DDR clock and control generation needs to be implemented on the left and right sides of the device. Generic DDR interfaces can be implemented on the top side of the “EA” devices, therefore this issue does not exist for “EA” devices.
- Table 12-13 summarizes what is available on each side of the LatticeECP3-70E, LatticeECP3-95E and LatticeECP3-150EA devices. Note that DDR registers are not available on the bottom of the device.

**Table 12-13. DDR Pin Limitations**

Left	Right	Top
DQS12 group available	DQS12 available	DQS 12 with some restrictions
Input x1 DDR Generic Output x1DDR Generic Input x2 DDR Generic Output x2 DDR Generic DDR Memory DDR2 Memory DDR3 Memory (“EA” only)	Input x1 DDR Generic Output x1DDR Generic Input x2 DDR Generic Output x2 DDR Generic DDR Memory DDR2 Memory DDR3 Memory (“EA” only)	Input x1 DDR Generic (“EA” only) Output x1DDR Generic (“EA” only) Input x2 DDR Generic (“EA” only) DDR Memory (“E” and “EA”) DDR2 Memory (“E” and “EA”)
Upper left side has DDR function without DQS.	Upper right side I/Os shared with sysCONFIG pins in bank 8. No DDR functions are available on these pins	Right part of top side has eight I/Os shared with the sysCONFIG pins. No DDR functions are available.

Note: See the [LatticeECP3 Family Data Sheet](#) Pinout tables to find DQS group assignments.

## DDR/DDR2 Pinout Guidelines

- The DQS-DQ association rule must be followed.
  - All associated DQs (8 or 4) to a DQS must be in the same DQS-12 group.
- The data mask (DM) must be part of the corresponding DQS-12 group.
  - Example: DM[0] must be in the DQS-12 group that has DQ[7:0], DQS[0].
- DQS pad must be allocated to a dedicated DQS pad.
  - DQS# pad is used when differential DQS is selected.
- Do not assign any signal to the DQS# pad if SSTL18D is applied to the DQS pad.
  - The software automatically places DQS# when SSTL18D is applied.
- DQS/DQS# pads cannot be used for other DDR functions.
  - The DQS IOLOGIC structure is not compatible with non-DQS DDR IOLOGIC.
- The clock to the PLL used to generate the outputs must be assigned to use dedicated clock routing.
- Data group signals (DQ, DQS, DM) can use either the left, right or top edge of LatticeECP3.
- Locating memory clock signals:
  - For the LatticeECP3 “E” device, it is highly recommended to have all clock pads within a memory controller be in one DQS-12 group to minimize pinout restrictions.
- The bottom-side pads in Bank 6 and Bank 3 are also good candidates for address/command/control signals. Save pins on the left and right sides for DDR.
- VREF1 of the bank where DQs are located must not be taken by any signal.

- VREF2 is OK to use.
- Do not connect VREF1 and VREF2 together.
- External termination to VTT is required for DDR and DDR2 interfaces. All DQ and DQS pins must be terminated to VTT using an external termination resistor.  $V_{TT} = \frac{1}{2} V_{CCIO}$  (0.9V for DDR2 and 1.25V for DDR). It is recommended that SI Simulation be run to determine the best termination value. If signal integrity simulation is not available, parallel termination of 75ohms to VTT should be used.
- It is required to provide a PCB connect resistor to the XRES pins. These pins cannot be used for other functions. See [TN1189, LatticeECP3 Hardware Checklist](#), for detailed requirements on the XRES pin.

## DDR3 Termination Guidelines

Proper termination of a DDR3 interface is an important part of implementation that ensures reliable data transactions at high speed. Below is the general termination guideline for the LatticeECP3 DDR3 interface.

### Termination for DQ, DQS and DM

- Do not locate any termination on the memory side. The memory side termination on DQ, DQS and DM is dynamically controlled by the DDR3 SDRAM's ODT function.
- External termination to VTT is required for DDR3 interfaces at the FPGA side. Each DQ and DQS pin should be terminated to VTT using an external termination resistor. The termination resistor location is important. See the [Layout Considerations for DDR3](#) section for the requirements.
- It is recommended that signal integrity (SI) simulation be run to determine the best termination value. If SI simulation is not available, parallel termination of 100 ohms to 120 ohms to VTT is recommended.
- Use of series termination resistors at the FPGA side is not recommended.

### Termination for CK

DDR3 memory clocks require differential termination because they use a differential signaling, SSTL15D, in DDR3 SDRAM applications. You can locate an effective 100-ohm termination resistance on the memory side to achieve the differential termination using the following guideline:

- Locate a 100-ohm resistor between the positive and negative clock signal, or
- Connect one end of an R<sub>tt</sub> resistor to the positive pin and one end of another R<sub>tt</sub> to the negative pin of a CK pair, then connect the other ends of two R<sub>tt</sub> resistors together and return to VDD through a C<sub>tt</sub> capacitance. This is a JEDEC CK termination scheme defined in the DIMM specifications. JEDEC uses 36-ohm for R<sub>tt</sub> with 0.1uF C<sub>tt</sub> for DIMM. 50-ohm R<sub>tt</sub> can also be used for non-DIMM applications.
- Use of series termination resistors at the FPGA side is not recommended.
- When fly-by wiring is used, the CK termination resistor should be located after the last DDR3 SDRAM device.

### Termination for Address, Commands and Controls

Parallel termination to VTT on address, command and control lines is typically required.

- Locate a 50-ohm parallel-to-VTT resistor (or a best known resistance obtained from your SI simulation) to each address, command and control line on the memory side.
- When fly-by wiring is used, the address, command and control termination resistors should be located after the last DDR3 SDRAM device.
- Series termination resistors can be optionally used on the address, command and control signals to suppress overshoot/undershoot and to help decrease overall SSO noise level. 22-ohm or 15-ohm series termination is recommended when used.



---

## Termination for DDR3 DIMM

The DDR3 SDRAM DIMMs incorporate internal termination following the requirements defined by the JEDEC DIMM specification. For this reason, the user termination requirement for the DDR3 DIMM is slightly different from that of DDR3 SDRAM devices.

- Do not locate any termination on the memory side. The memory side termination on DQ, DQS and DM is dynamically controlled by the DDR3 SDRAM's ODT function.
- Do not locate differential termination on CK at the memory side.
- Do not locate parallel termination to VTT on address, command and control signals at the memory side.
- Follow the termination for DQ, DQS and DM guideline above for the FPGA side termination.

## DDR3 Interface without Termination

When a wide DDR3 data bus is implemented and requires most of the pin resources in the assigned banks to be used as data lines, SSO impact usually becomes a designer's concern. While proper use of termination resistors provides optimized signal integrity results, removing them may also provide improved noise margin in some cases due to increased eye height.

### DQ, DQS and DM without Parallel VTT Termination

Although the external parallel VTT termination is typically required for read operations at the FPGA side, it can be removed if the following conditions are met:

- a. Point-to-point connection between FPGA and DDR3 SDRAM device
- b. PCB trace length is maintained shorter than 3.0"
- c. SI simulation confirms that there is no significant reflection or ringing noise due to unmatched line impedance

Either or both of the following considerations are suggested when you implement a DDR3 interface without external VTT termination:

- You can still keep the external VTT termination in your PCB layout without population if the board space allows. It will give you an opportunity to return to the external termination scheme without board re-spin work.
- You can connect the VTT input pads in the banks where the DDR3 interface's data bus is implemented to the external VTT source. This will allow you to use the LatticeECP3's internal on-chip termination on selective signals in case only a few have bad signal integrity results. This approach also allows you to use internal VTT termination only on DQS signals, which can be a useful termination option in some cases.

### Address, Command and Control Signals without Parallel VTT Termination

As described, parallel termination to VTT on address, command and control lines is typically required. However, the parallel VTT termination on them can also be optionally removed to increase the noise margin or to avoid layout difficulties at the memory side. When they are removed, use of series termination resistors at the driver side (FPGA) is recommended to suppress overshoot and undershoot noise. The same condition as specified above for DQ, DQS and DM is applied to remove the parallel termination.

## Layout Considerations for DDR3

- Placement of external discrete resistors or resistor packs (RPACKS) for parallel VTT termination is critical and must be placed within 0.6 inches (600-mil) of the FPGA ball.
  - 120 ohm BGA RPACKS (CTS RT2432B7 type) are recommended for the 64- and 32-bit interfaces due to better routing and density issues. Each RPACK contains 18 resistors in a very small BGA footprint. Note that only 120, 75 and 50 ohm values are available in this package type.
  - 4x1 RPACKS (CTS 741X083101JP type) can also be used in cases where a 100 ohm value is needed with-

out routing/density issues.

- The termination resistor stubs should be kept minimal.
- All traces should be matched to 50 ohms.
- For SODIMM and UDIMM designs, write leveling should be used and all traces from the FPGA to the DIMM should be matched in length on the PC board similar to DDR2.
- If using a discrete DDR3 device, fly-by routing can be used for traces. If fly-by routing is used, the Write Leveling option must be enabled. If not using fly-by routing, the Write Leveling option must be disabled.
- Refer to TN1189, [LatticeECP3 Hardware Checklist](#), DDR3 Interface Requirements section for complete DDR3 layout guidelines.

## DDR3 Pinout Guidelines

The LatticeECP3 device contains dedicated I/O functions for supporting DDR3 memory interfaces. The following pinout rules must be followed to properly use the dedicated I/O functions.

- The DQS-DQ association rule must be followed.
  - All associated DQs (8 or 4) to a DQS must be in the same DQS-12 group.
- A data mask (DM) must be part of the corresponding DQS-12 group.
  - Example: DM[0] must be in the DQS-12 group that has DQ[7:0], DQS[0].
- A DQS pad must be allocated to a dedicated DQS True (+) pad.
  - A DQS# pad is auto-placed when a differential SSTL type (SSTL15D) is selected.
- Do not assign any signal to a DQS# pad if SSTL15D is applied to the DQS pad.
  - The software automatically places DQS# when SSTL15D is applied.
- DQS/DQS# pads cannot be used for other DDR functions.
  - The DQS IOLOGIC structure is not compatible with non-DQS DDR IOLOGIC.
  - Do not use DQS pads for any DDR3 signals except DQS and RST#. They can be used for other user logic signals if a DDR function is not required.
- Data group signals (DQ, DQS, DM) must use the left and right sides of the LatticeECP3 device.
  - Top-side pads do not have 2x gearing.
- Address, command, control and CK signals must be located on generic DDR-capable pads (ODDRXD).
  - Place the CK/CK# outputs on the same side where the DQ and DQS pads are located. The top side is not recommended for the high-speed DDR CK function.
  - Place the address, command and control signals either on the same side as where the DQ and DQS pads are located or on the top side. The bottom side cannot be used.
- RST# can be located anywhere an output is available as long as LVCMOS15 is applicable.
  - All DDR3 signals except RST# use DDR functions.
- The DDR3 input reference clock to the PLL must be assigned to use dedicated clock routing.
  - The dedicated PLL input pads are recommended while PCLK inputs can also be used.
  - Two PLLs that have a direct connection to ECLK in each side can be used for a DDR3 function. (Note that LatticeECP3-17EA devices have one PLL that supports DDR3 in each side.)
- Do not use the bottom-side pads in Bank 6 and Bank 3 for address, command and control signals.
  - No generic DDR support on the bottom side of the device.
- VREF1 of the bank where the DQ, DQS and DM pads are located must not be taken by any signal.
  - Do not PROHIBIT VREF1 in the preference file.
  - VREF2 can be used for a general purpose I/O signal.
  - Do not connect VREF1 and VREF2 together.
- Leave VTT pins unconnected when external VTT termination is implemented.

- VTT pins can be optionally connected to the external VTT source when a DDR3 interface is implemented without external VTT termination. See the [DDR3 Interface without Termination](#) section for details.
  - VTT pins can be connected in series with a capacitor (around .01uf) to ground when the bank has LVDS internal termination to suppress externally generated common mode noise. Internal VTT termination cannot be used in this case.
- It is required to provide a PCB connect resistor to the XRES pins. These pins cannot be used for other functions. See [TN1189, LatticeECP3 Hardware Checklist](#), for detailed requirements on the XRES pin.

## Pin Placement Considerations for Improved Noise Immunity

In addition to the general pinout guidelines, there are additional pinout considerations for minimizing simultaneous switching noise (SSN) impact. The following considerations are necessary to control SSN within the required level:

1. Properly terminated interface
2. SSN optimized PCB layout
3. SSN considered I/O pad assignment
4. Use of pseudo power pads

The guidelines listed below address the I/O pad assignment and pseudo power pad usage. Unlike the pinout guidelines, they are not absolute requirements. However, it is recommended that the pin placement follow the guidelines as much as possible to increase the SSO/SSI noise immunity.

- Place the DQS groups for data implementation starting from the middle of the (right or left) edge of the LatticeECP3. Allow a corner DQS group to be used as a data group only when necessary to implement the required width.
- Locate a spacer DQS group between the data DQS groups if possible. A DQS group becomes a spacer DQS group if the I/O pads inside the group are not used as data pads (DQ, DQS or DM).
  - The pads in a spacer group can be used for address, command, control or CK pads as well as for user logic or the pseudo power pads.
  - No more than two consecutive-data DQS group placements is recommended in the middle of the edge. When a corner-side DQS group is used for data, locate a spacer DQS group right next to it.
  - If there is an incomplete DQS group (not the size of DQS-12) or enough space for more than 10 pads between two DQS groups, the following DQS group can be located next to the previous complete DQS group, both as data groups. The incomplete DQS groups, or the space between them, can be used as a spacer.
- It is recommended that you locate a few pseudo VCCIO/ground (GND) pads inside a spacer DQS group. An I/O pad becomes a pseudo power pad when it is configured to OUTPUT with its maximum driving strength (i.e., SSTL15, 10mA for DDR3) and connected to the external VCCIO or ground power source on the PCB.
  - Your design needs to drive the pseudo power I/O pads according to the external connection. (i.e., you assign them as OUTPUT and let your design drive '1' for pseudo VCCIO pads and '0' for pseudo GND pads in your RTL coding.)
  - Locating four pseudo power pads in a spacer DQS group should be sufficient to efficiently suppress the SSN impact. At least two pseudo pads should be implemented in a spacer DQS group if more pins are needed for a user design.
  - Good candidate pads are two pads in both ends (the first and the last ones in the group) and/or two DQS (positive and negative) pads in the middle.
- You may have one (DDR2/DDR3) or two (DDR/DDR2) remaining pads in a data DQS group which are not assigned as a data pad in a DDR memory interface. Assign them to pseudo VCCIO or pseudo GND. The preferred location is in the middle of the group (right next to a DQS pad pair). Note that you may not have this extra pad in DDR2/DDR3 if the DQS group includes a VREF pad for the bank.
- Avoid fast switching signals located close to the XRES pad of a LatticeECP3 device. XRES requires an external resistor which is used to create the bias currents for the I/O. Since this resistor is used for a calibration reference

for sensitive on-chip circuitry, careful pin assignment around the XRES pad is also necessary to produce less jittery PLL outputs for DDR memory interface operations.

The guidelines below are not as effective as the ones listed above. However, following them is still recommended to improve the SSN immunity further:

- Assign the DM (data mask) pad in a data DQS group close to the other side of DQS pads where a pseudo power pad is located. If the data DQS group includes VREF, locate DM to the other side of VREF with respect to DQS. It can be used as an isolator due to its almost static nature in most applications.
- Other DQS groups (neither data nor spacer group) can be used for accommodating DDR3 address, command, control and clock pads. It is recommended that you still assign all or most DQS pads (both positive and negative) in these groups to pseudo power. Since LatticeECP3 DQS pads have a dedicated DDR function that cannot be shared with other DDR3 signals, they are good pseudo power pad candidates.
- You can assign more unused I/O pads to pseudo power if you want to increase the SSN immunity. Note that the SSN immunity does not get increased at the same rate as the increased number of pseudo power pads. The first few pseudo power pad placements described above are more crucial. Keep the total pseudo power pad ratio (VCCIO vs. GND) between 2:1 to 3:1.
- Although not necessary, it is slightly more effective to locate a pseudo VCCIO to a positive pad (A) and GND to a negative pad (B) of a PIO pair if possible.
- If a bank includes unused input-only pads such as dedicated PLL input pads, you can also connect them to VCCIO on your PCB. They are not as efficient as the pseudo power pads but can still be used as isolators, and the connections on the board would provide good shielding. No extra consideration is necessary for these pins in your design.
- It is a good idea to shield the VREF1 pad by locating pseudo power pads around it if the VREF1 pad is not located in a data DQS group.

Table 12-14 shows the recommended examples of DQS group allocations following the guidelines. If you have enough pin resources, following the best examples would provide you with maximized SSN immunity results. It is more practical in most applications to follow the examples in the “Allowed” columns. It is expected that the SSN and ground bounce impacts are considerably less than those cases where you do not include any consideration.

**Table 12-14. Recommended Examples of DQS Group Allocation**

LatticeECP3-150EA													
DQS	150-1156 Left		150-1156 Right		150-672 Left		150-672 Right						
	Best	Allowed	Best	Allowed	Best	Allowed	Best	Allowed					
1		D		D		D		D*					
2	D		D	D*	D		D	D					
3		D				D							
4	D		D	D	D	D	D	D					
5		D		D			D*	D*					
6	D	D	D	D	D	D							
7				D	D*	D*							
8	D	D	D	D									
9		D											
10	D		D	D									
11		D											
12	D	D											
13													
Bus Size	48	64	40	56	32	40	24	32					

LatticeECP3-95/70EA													
DQS	95/70-1156 Left		95/70-1156 Right		95/70-672 Left		95/70-672 Right		95/70-484 Left		95/70-484 Right		
	Best	Allowed	Best	Allowed	Best	Allowed	Best	Allowed	Best	Allowed	Best	Allowed	
1		D		D		D		D*				D	
2	D		D		D		D	D	D	D	D		

**Table 12-14. Recommended Examples of DQS Group Allocation (Continued)**

3		D		D		D				D		D					
4	D	D	D	D	D	D	D	D	D		D	D					
5								D*	D*		D						
6	D	D	D	D	D	D				D*							
7		D	D*	D*	D*	D*											
8	D	D*															
9																	
DDR3 Bus	32	48	32	40	32	40	24	32	16	32	16	24					
<b>LatticeECP3-35EA</b>																	
<b>DQS</b>					<b>35-672 Left</b>		<b>35-672 Right</b>		<b>35-484 Left</b>		<b>35-484 Right</b>		<b>35-256 Left</b>		<b>35-256 Right</b>		
					<b>Best</b>	<b>Allowed</b>	<b>Best</b>	<b>Allowed</b>	<b>Best</b>	<b>Allowed</b>	<b>Best</b>	<b>Allowed</b>	<b>Best</b>	<b>Allowed</b>	<b>Best</b>	<b>Allowed</b>	
1						D		D		D		D		D*	D*	D*	D*
2					D		D		D		D			D*			
3						D		D		D		D					
4					D	D	D	D	D		D	D					
5										D							
6					D	D		D		D*							
7																	
DDR3 Bus					24	32	16	32	16	32	16	24	8	16	8	8	
<b>LatticeECP3-17EA</b>																	
<b>DQS</b>									<b>17-484 Left</b>		<b>17-484 Right</b>		<b>17-256 Left</b>		<b>17-256 Right</b>		
									<b>Best</b>	<b>Allowed</b>	<b>Best</b>	<b>Allowed</b>	<b>Best</b>	<b>Allowed</b>	<b>Best</b>	<b>Allowed</b>	
1										D				D*	D*	D*	
2									D		D	D	D*	D*			
3										D		D					
4									D	D							
5																	
DDR3 Bus									16	24	8	16	8	16	8	8	

Notes: DQS groups with a 'D' indicate the data DQS groups while the blank ones indicate the spacer DQS groups.  
Data DQS groups with an asterisk indicate that they have an incomplete DQS group or enough isolation in front.  
Shaded cells are not-applicable to the selected device.

## DDR Software Primitives and Attributes

This section describes the software primitives that can be used to implement all the DDR interfaces. These primitives are divided into ones that are used to implement the DDR data and ones for DDR Strobe signal or the Source Synchronous clock. The DQSBUF primitives are used to generate the signals required to correctly capture the data from the DDR memory.

**Table 12-15. DDR Software Primitives List**

Type	Primitive	Usage
Data Input	IDDRXD	E and EA Generic DDRX1 E and EA DDR/DDR2 Memory
	IDDRX1D	EA Generic DDRX1
	IDDRX2D	E Data Input Generic DDRX2 E and EA DDR3 Memory
	IDDRX2D1	EA Generic DDRX2
Data Output	ODDRXD	E and EA Generic DDRX1 E and EA DDR/DDR2/DDR3 Memory
	ODDRXD1	EA Generic DDRX1
	ODDRX2D	E and EA Generic DDRX2 E and EA DDR3 Memory

**Table 12-15. DDR Software Primitives List (Continued)**

Type	Primitive	Usage
Data Tristate	ODDRTDQA	E and EA Generic DDRX2 E and EA DDR3 Memory
	OFD1S3AX	E and EA Generic DDRX1 E and EA DDR/DDR2 Memory
DQS Output	ODDRXDQSA	E and EA DDR/DDR2 Memory
	ODDRX2DQSA	E and EA DDR3 Memory
DQS Tristate	ODDRTDQSA	E and EA DDR/DDR2 Memory E and EA DDR3 Memory
DQSBUF Logic	DQSBUFD	E and EA DDR3 Memory, E and EA Generic DDRX2 (for bus widths <10 bits)
	DQSBUFF	E and EA DDR/DDR2 Memory, E and EA Generic DDRX1 (for bus widths <10 bits)
	DQSBUFE	E Generic DDRX2
	DQSBUFE1	EA Generic DDRX2
	DQSBUFG	E Generic DDRX1
DQSDLL	DQSDLL	DLL for Generic DDRX1/DDR2 (for bus width <10 bits and multiple interfaces per side of the device) and DDR/DDR2/DD3 Memory
Input Delay	DELAYB	Delay block for Generic DDRX2 with Dynamic Control
	DELAYC	Delay block for Generic DDRX1/X2 with clock injection removal. The amount of Fixed Delay will vary by interface.
ECLK Stop	ECLKSYNCA	EA Generic DDRX2 Output EA ECLK Synchronization for DDR3 Memory

**DQSDLLB**

The DQSDLLB will generate the 90° phase shift required for the DQS signal. This primitive will implement the on-chip DQSDLL. Only one DQSDLL should be instantiated for all the DDR implementations on one-half of the device. The clock input to this DLL should be at the same frequency as the DDR interface. The DLL will generate the delay based on this clock frequency and the update control input to this block. The DLL will update the dynamic delay control to the DQS delay block when this update control (UDDCNTLN) input is asserted. Figure 12-70 shows the primitive symbol. The active low signal on UDDCNTLN updates the DQS phase alignment.

**Figure 12-70. DQSDLL Symbol**

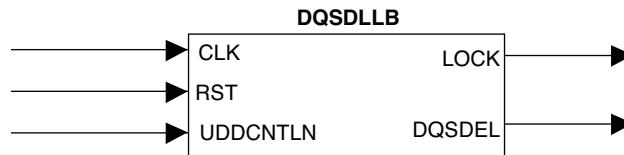


Table 12-16 provides a description of the ports.

**Table 12-16. DQSDLLB Ports**

Port Name	I/O	Definition
CLK	I	CLK should be at the frequency of the DDR interface.
RST	I	Resets the DQSDLLB.
UDDCNTLN	I	Provides update signal to the DLL that will update the dynamic delay.
LOCK	O	Indicates when the DLL is in phase.
DQSDEL	O	The digital delay generated by the DLL, should be connected to the DQSBUF primitive.

### DQSDLL Update Control

The DQS delay can be updated for PVT variation using the UDDCNTLN input. The DQSDEL is updated when the UDDCNTLN is held low. The DDR memory controller or user logic can update DQSDEL when variations are expected. It can be updated anytime except during a memory READ or WRITE operation.

It is important to understand that the UDDCNTLN signal is a synchronous input to the DQSDLL CLK domain. When using DDR in 2x gearing, it is required to use clock domain transfer logic first to transfer UDDCNTLN from slow clock domain to fast clock domain before it is input to DQSDLL. You can use two- or three-stage pipeline registers to safely transfer the DQSDEL update control input to the DQSDLL block. The first stage register uses the local domain clock while the second and third registers use the DQSDLL CLK domain clock. The second to third stage pipelining is desirable because it can eliminate the placement and routing issue due to the increased clock rate (2x) for the net and also avoids any meta-stability issues.

### DQSDLL Configuration

By default this DLL will generate a 90° phase shift for the DQS strobe based on the frequency of the input reference clock to the DLL.

### DQSBUF Logic Primitives for Generic DDR

The DQSBUF primitives (DQSBUFE and DQSBUFG for “E” devices) are used to generate the strobe logic and delay used in the input DDR modules to correctly demux the DDR data. The DQSBUFE is used for x2 interfaces and the DQSBUFG is used for x1 interfaces. The DQSBUFE1 is used only for output x2 interfaces in LatticeECP3 devices.

Figure 12-71 shows DQSBUF Generic DDR functions for the “E” and “EA” devices.

**Figure 12-71. DQSBUFE Function for Generic Input/Output DDR x2 Interfaces (“E” Devices)**

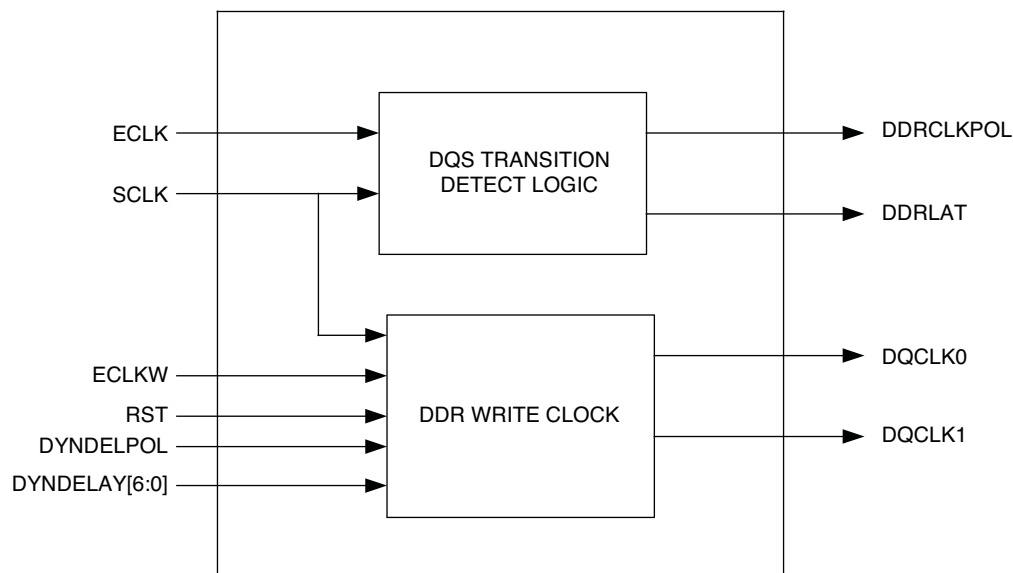




Figure 12-72. DQSBUFG Function for Generic Input/Output DDR x 1 Interface (“E” Devices)

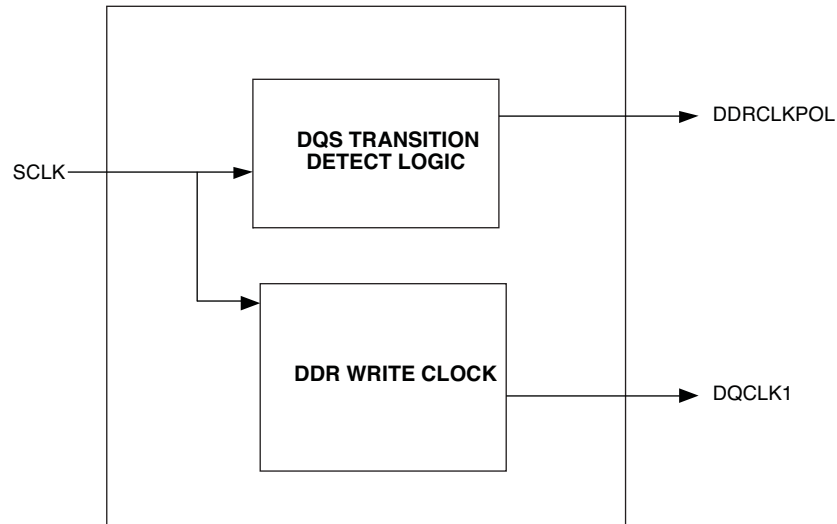
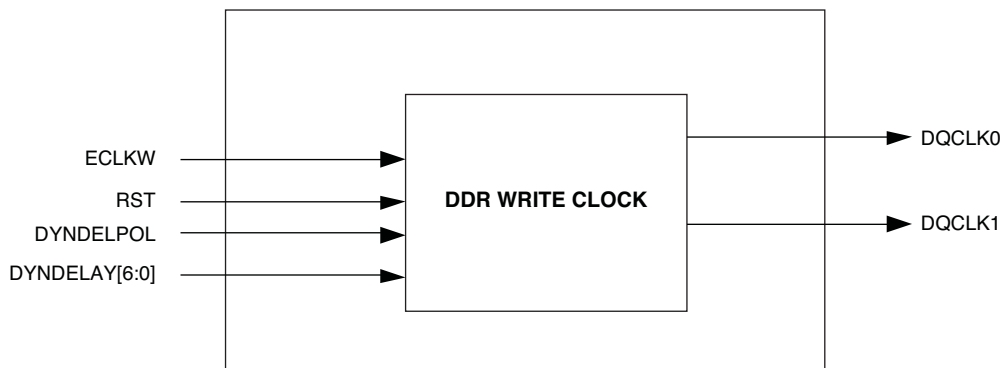


Figure 12-73. DQSBUFE1 Function for Generic Output DDR x2 Interfaces (“EA” Devices)



### DQS Transition Detect

The DQS Transition Detect block inputs the fast ECLK and the slower SCLK (=1/2 ECLK) inputs and generates the DDRCLKPOL and the DDRLAT signals. These signals are generated based on the phase of the FPGA SCLK at the first ECLK transition. The DDRLAT signal is used in generic DDRX2 mode to transfer data from the ECLK to SCLK domain. These are only required to implement generic DDR on the LatticeECP3 “E” devices. LatticeECP3 “EA” devices do not require these signals.

### DDR Write Clock

This block inputs the fast edge clock used for the write side and generates two control signals, DQCLK0 and DQCLK1. For Generic DDRX2 gearing, both DQCLK0 and DQCLK1 are generated. These control clocks run at a rate of one-half the fast edge clock, and DQCLK1 is offset delayed by 90° relative to DQCLK0. DQCLK1 output matches the phase of the SCLK input to the block. These two clocks toggle between different legs of a 4:1 mux in the output logic, which allows 4:1 gearing of data at twice the edge clock rate. When using generic DDRX1, instead of DDRX2 gearing, only the DQCLK1 is output, which toggles at the rate of FPGA clock provided by SCLK and matches the phase of the SCLK input.

The DDR write clock block also inputs DYNDLPOL and DYNDLAY [6:0] delay inputs. These inputs support the DDR3 memory interface to adjust delay among the various DQS groups. They can also be used for the generic DDR for the same purpose. The DYNDLAY [6:0] can input 128 possible delay step settings with each step generating approximately 25ps nominal delay. In addition, the DYNDLPOL can be used to invert the clock for a 180°



shift of the incoming clock. Each of the source synchronous clock outputs can be adjusted to account for the data skew using this delay. The DYNDELAY [6:0] and DYNDELPOL inputs should be generated using the FPGA core.

### DQSBUFE

This primitive provides the control logic for Generic DDRX2 interface in the LatticeECP3 “E” devices. Figure 12-74 shows the primitive symbol.

Figure 12-74. DQSBUFE Symbol

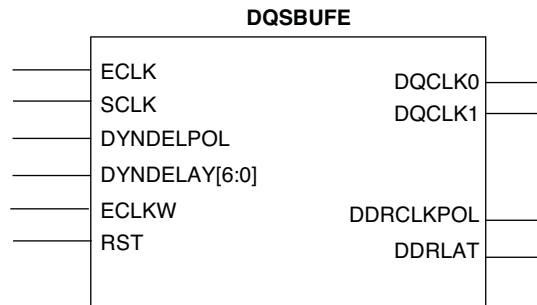


Table 12-17 provides a description of all the I/O ports associated with the DQSBUFE primitive.

Table 12-17. DQSBUFE Primitive Ports

Port Name	I/O	Definition
ECLK	I	Edge CLK
SCLK	I	System CLK
ECLKW	I	Edge CLK used the DDR write side
RST	I	Reset input
DYNDELPOL	I	Input from user logic used to invert the clock polarity
DYNDELAY [6:0]	I	Input from user logic used to delay the ECLK
DQCLK0	O	Clock output at frequency of SCLK used for output side gearing
DQCLK1	O	Clock output at frequency of SCLK and matches the phase of SCLK using for output side gearing. DQCLK1 is 90° shifted from DQCLK0.
DDRCLKPOL	O	DDR clock polarity signal
DDRLAT	O	DDR latch control signal

### DQSBUFG

This primitive implements the strobe logic for Generic DDRX1 interface for LatticeECP3 “E” devices. Figure 12-75 shows the primitive symbol.

Figure 12-75. DQSBUFG Symbol

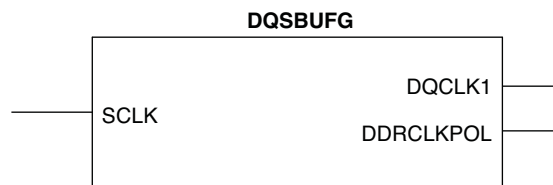


Table 12-18 provides a description of all the I/O ports associated with the DQSBUFG primitive.

**Table 12-18. DQSBUFG primitive Ports**

Port Name	I/O	Definition
SCLK	I	System CLK
DQCLK1	O	Clock output at frequency of SCLK, matches the phase of SCLK used for output gearing
DDRCLKPOL	O	DDR clock polarity signal

### DQSBUFE1

This primitive implements the strobe logic for Generic DDRX2 Output interfaces on the LatticeECP3 “EA” devices. Figure 12-76 shows the primitive symbol.

**Figure 12-76. DQSBUFE1 Symbol**

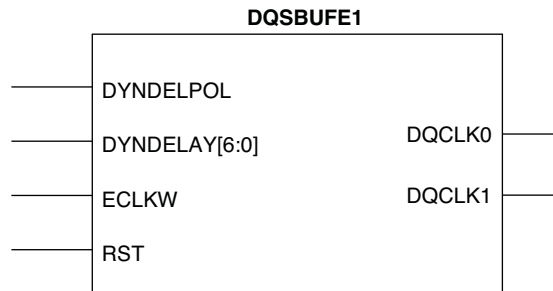


Table 12-19 provides a description of all the I/O ports associated with the DQSBUFE1 primitive.

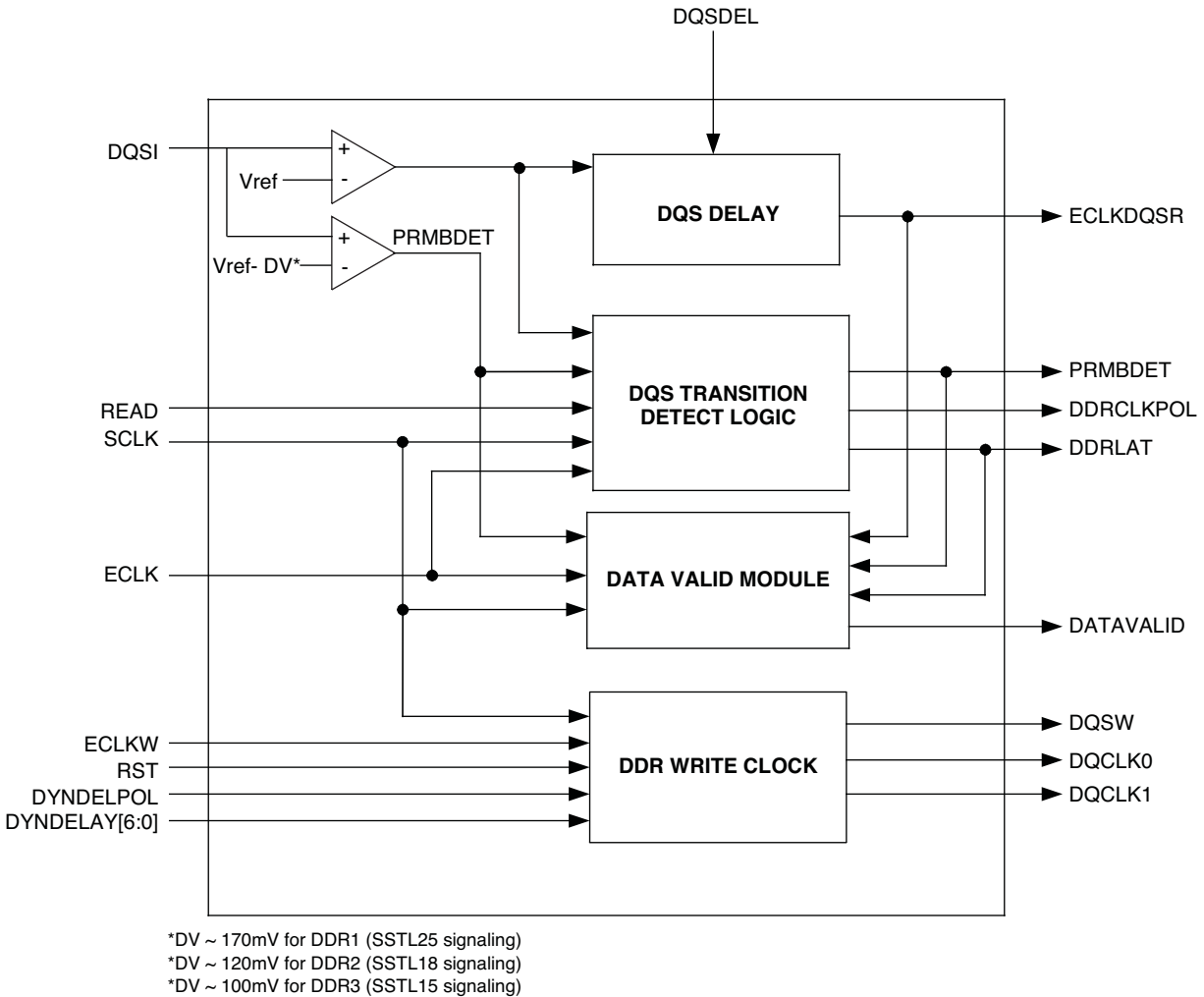
**Table 12-19. DQSBUFE1 Primitive Ports**

Port Name	I/O	Definition
ECLKW	I	Edge CLK used the DDR write side
RST	I	Reset input
DYNDLPOL	I	Input from user logic used to invert the clock polarity
DYNDelay [6:0]	I	Input from user logic used to delay the ECLK
DQCLK0	O	Clock output at frequency of SCLK used for output side gearing
DQCLK1	O	Clock output at frequency of SCLK, matches the phase of SCLK using for output side gearing. DQCLK1 is 90° shifted from DQCLK0.

### DQSBUF Logic Primitives for DDR Memory Interfaces

The DQSBUF primitives (DQSBUFD and DQSBUFF) are used to generate the DQS strobe logic and delay used in the input DDR modules to correctly demux the DDR data. The DQSBUF module used for the DDR memory interface is composed of DQS Delay, DQS Transition Detect, Data Valid Generation and the DDR Write Clock block, as shown in Figure 12-77.

Figure 12-77. DQSBUF Block for DDR Memory Interfaces



### DQS Delay Block

The DQS Delay block receives the digital control delay line (DQSDEL) coming from one of the two DQSDLL blocks. These control signals are used to delay the DQSI by 90°. ECLKDQSR is the delayed DQS and is connected to the clock input of the first set of input DDR registers.

### DQS Transition Detect

The DQS Transition Detect block generates the DDR Clock Polarity (DDRCLKPOL) and DDR Latch Control (DDR-LAT) signal based on the phase of the FPGA clock (SCLK) and edge clock signal (ECLK) at the first DQS transition. The DDR READ control signal and FPGA CLK inputs to this block come from the FPGA core. The DDRLAT signal is used when implementing DDRX2 output gearing to transfer data from the ECLK to SCLK domain.

### Data Valid Module

The data valid module generates a DATAVALID signal. This signal indicates to the FPGA that valid data is transmitted out of the input DDR registers to the FPGA core.

### DDR Write Clock

This block inputs the fast edge clock used for the write side and generates two control signals, DQCLK0 and DQCLK1, and the clock used to generate the DQS clock (DQSW). DQSW is generated by applying the DQSDEL from the DQSDLL to delay the ECLK (DDR3) or SCLK (DDR and DDR2) inputs. For the DDR3 memory interface both DQCLK0 and DQCLK1 are generated. These control clocks run at a rate of one-half the edge clock, and

DQCLK1 is offset delayed by 90° relative to DQCLK0. These two clocks toggle between different legs of a 4:1 mux allowing 4:1 gearing of data at twice the edge clock rate. When using output DDRX1 instead of DDRX2 gearing, only the DQCLK1 is output, which is the same as the FPGA clock.

The DDR write clock block also inputs DYNDLPOL and DYNDLAY [6:0] delay inputs. These are used to support the write leveling required for DDR3 memory interface. This delay can be used to adjust the delays among the DQS groups to account for any skew that may be introduced due to the DDR3 fly-by topology.

The DYNDLAY [6:0] can input 128 possible delay step settings with each step generating approximately 26ps nominal delay. In addition, the DYNDLPOL can be used to invert the clock for a 180° shift of the incoming clock. The DYNDLAY [6:0] and DYNDLPOL inputs should be generated in the memory controller.

## DQSBUFD

This primitive implements the strobe logic for DDR3 memory interface. Figure 12-78 shows the primitive symbol.

**Figure 12-78. DQSBUFD Symbol**

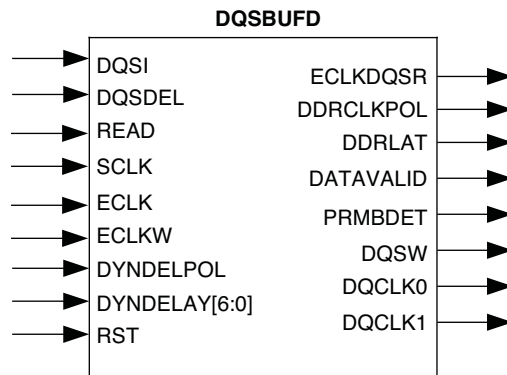


Figure 12-20 provides a description of all the I/O ports associated with the DQSBUFD primitive.

**Table 12-20. DQSBUFD Primitive Ports**

Port Name	I/O	Definition
DQSI	I	DQS strobe input from the memory
Read	I	Read signal generated from the FPGA core
ECLK	I	Edge CLK
SCLK	I	System CLK
DQSDEL	I	DQS delay signal from the DQSDLL module
ECLKW	I	Edge CLK used the DDR write side
RST	I	Reset input
DYNDLPOL	I	Input from user logic used to invert the clock polarity
DYNDLAY [6:0]	I	Input from user logic used to delay the ECLK
ECLKDQSR	O	Delay DQS used to capture the data
PRMBDET	O	Preamble detect signal, going to the FPGA core logic
DATAVALID	O	Signal indicating transmission of Valid data to the FPGA core
DDRCLKPOL	O	DDR Clock polarity signal
DDRLAT	O	DDR latch control signal
DQSW	O	Clock used to generate DQS on the write side
DQCLK0	O	Clock Output at frequency SCLK used for output gearing
DQCLK1	O	Clock output at frequency and phase of SCLK used for output gearing

## DQSBUFF

This primitive implements the strobe logic for DDR and DDR2 memory interface. Figure 12-79 shows the primitive symbol.

**Figure 12-79. DQSBUFF Symbol**

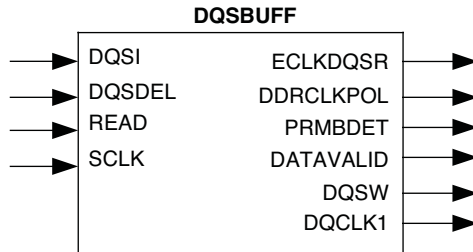


Table 12-21 provides a description of all the I/O ports associated with the DQSBUFF primitive.

**Table 12-21. DQSBUFF Primitive Ports**

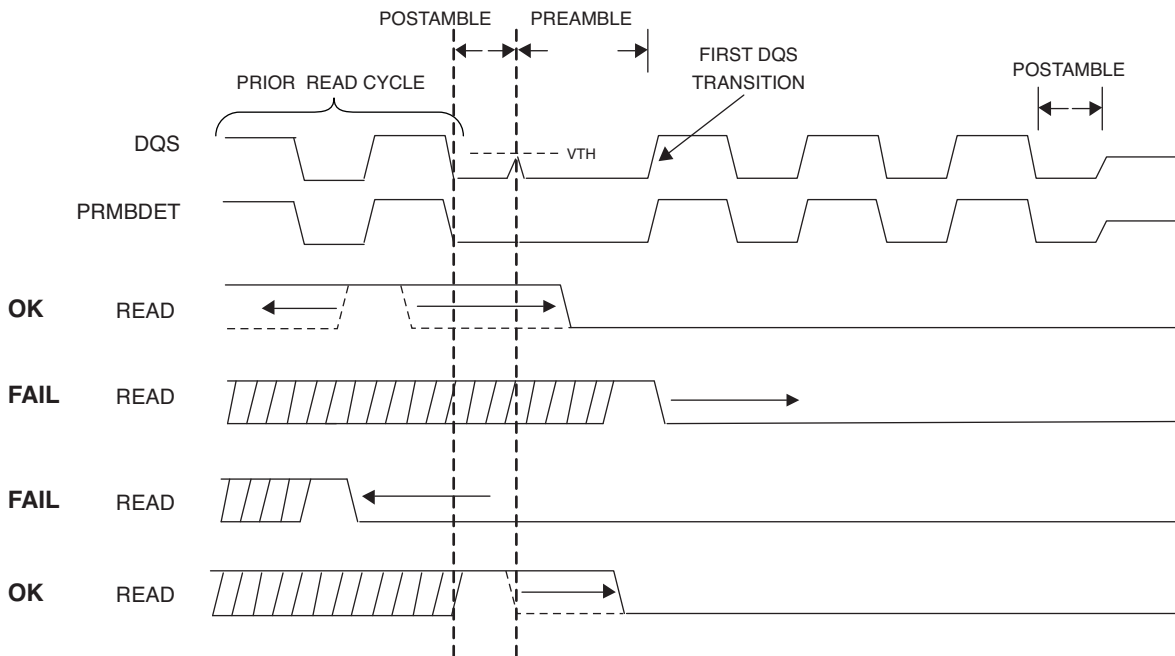
Port Name	I/O	Definition
DQSI	I	DQS strobe input from the memory
READ	I	Read signal generated from the FPGA core
SCLK	I	System CLK
DQSDEL	I	DQS delay signal from the DQSDLL module
ECLKDQSR	O	Delay DQS used to capture the data
PRMBDET	O	Preamble detect signal, going to the FPGA core logic
DATAVALID	O	Signal indicating transmission of valid data to the FPGA core
DDRCLKPOL	O	DDR Clock polarity signal
DQSW	O	Clock used to generate DQS on the write side
DQCLK1	O	Clock output at frequency and phase of SCLK used for output gearing

### READ Pulse Generation

The READ signal to the DQSBUFF block is internally generated in the FPGA core. The READ signal will go high after the READ command to control the DDR-SDRAM is initially asserted. This should normally precede the DQS preamble by one cycle yet may overlap the trailing bits of a prior read cycle. The DQS Detect circuitry requires the falling edge of the READ signal to be placed within the preamble stage.

Figure 12-80 shows a READ pulse timing example with respect to the PRMBDET signal.

Figure 12-80. READ Pulse Generation



### DQSBUF Attributes

Table 12-22 shows the attributes can be used with the DQSBUF primitives described above.

Table 12-22. DQSBUF Attributes

Attribute	Description	Values	Software Default	Used in DQSBUF
DYNDEL_TYPE	Type of Static Delay input to the write control block Normal: 0° phase shifted Shifted: 180° phase shifted using clock inversion	NORMAL, SHIFTED	NORMAL	DQSBUFD, DQSBUFE
DYNDEL_VAL	Value of Static Delay to the write control block	0-127	0	DQSBUFD, DQSBUFE
DYNDEL_CNTL	Attribute to enable Static or Dynamic DYNDEL	STATIC, DYNAMIC	DYNAMIC	DQSBUFD, DQSBUFE
NRZMODE <sup>1</sup>	Attribute used to select NRZMODE for DDR3 Memory	DISABLED ENABLED	DISABLED	DQSBUFD

1. NRZMODE is only used with the DDR3 memory interface. This attribute affects the read data valid signal. When enabled, the read data valid signal will toggle to indicate valid data.

### Input DDR Primitives

The input DDR primitives represent the input DDR module used to capture both the generic DDR data and the DDR data coming from a memory interface. There are two available modes for the DDR input registers, one is used to implement DDRX1 gearing and the other is for DDRX2 gearing. The signals connected to the inputs of the IDDR are different for the DDR memory interface.

#### IDDRXD

This primitive implements the input register block in x1 gearing mode. This mode is used to implement DDR/DDR2 memory interfaces in the LatticeECP3 “E” and “EA” devices. It is also used to capture the generic DDRX1 data on the LatticeECP3 “E” device.

DDR registers are designed to use edge clock routing on the I/O side and the primary clock on the FPGA side. The ECLK input is used to connect to the DQS strobe coming from the DQS delay block (DQSBUFF) when implement-

ing a DDR or DDR2 memory interface. For generic source synchronous DDR applications, this signal should connect to the edge clock input. The SCLK input should be connected to the system (FPGA) clock. DDRCLKPOL is an input from the DQS clock polarity tree. This signal is generated by the DQS Transition detect circuit in the corresponding DQSBUF block. The DDRCLKPOL signal is used to choose the polarity of the SCLK to the synchronization registers.

Figure 12-81 shows the primitive symbol and all the I/O ports.

**Figure 12-81. IDDRXD Symbol**

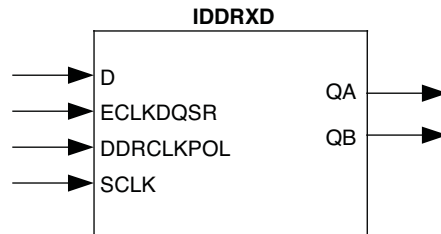


Table 12-23 provides a description of all I/O ports associated with the IDDRXD primitive.

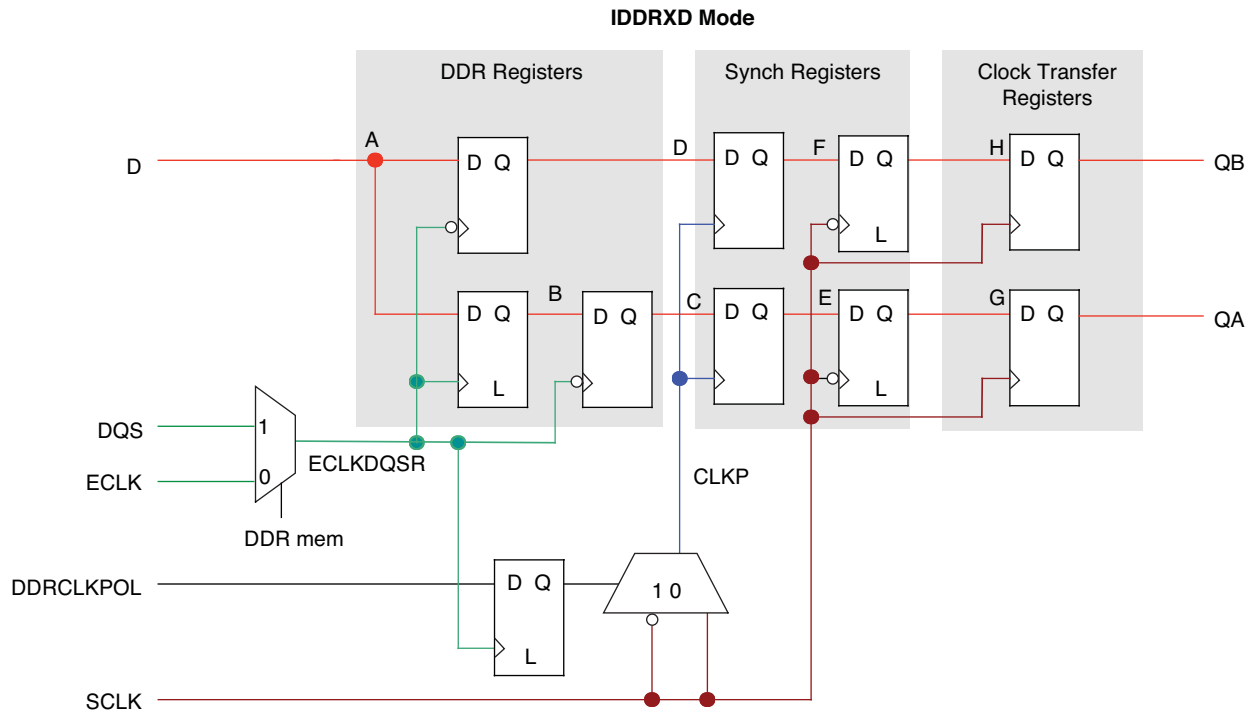
**Table 12-23. IDDRXD Ports**

Port Name	I/O	Definition
D	I	DDR data
ECLKDQSR	I	Phase-shifted DQS for DDR memory interfaces. ECLK for generic DDR interfaces.
SCLK	I	System clock
DDRCLKPOL	I	DDR clock polarity signal
QA	O	Data at positive edge of the clock
QB	O	Data at the negative edge of the clock

Note: The DDRCLKPOL input to IDDRXD should be connected to the DDRCLKPOL output of the DQSBUFF or DQFBUFG modules.

Figure 12-82 shows the Input Register Block configured in the IDDRXD mode.

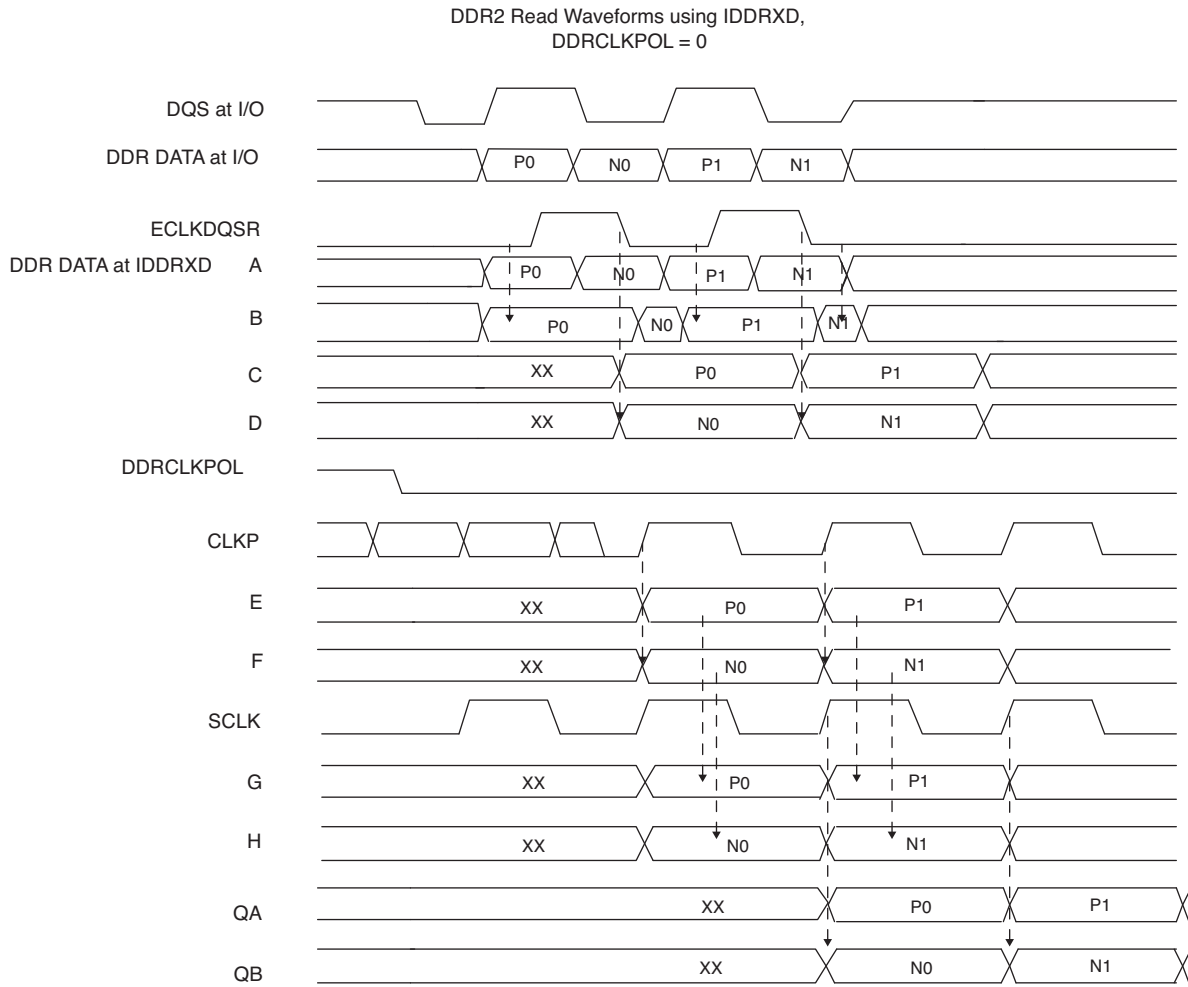
Figure 12-82. Input Register Block in IDDRXD Mode



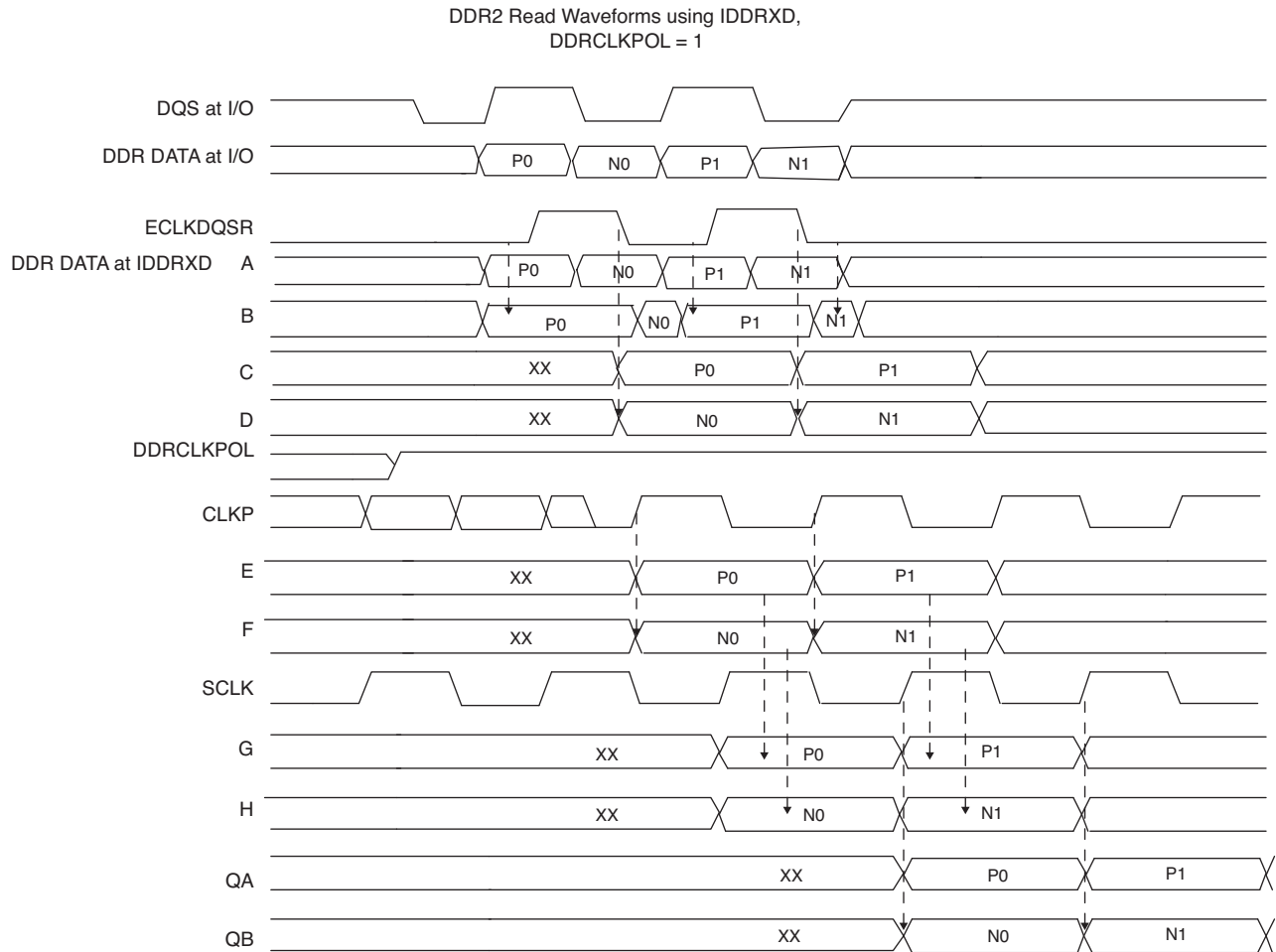
Note: Simplified diagram does not show CE, SET and RST details. All latches are transparent when low.



**Figure 12-83. IDDRXD Waveform DDRCLKPOL=0**



**Figure 12-84. IDDRXD Waveform DDRCLKPOL=1**



**IDDRXD1**

This primitive is a simplified version of IDDRXD without the DDRCLKPOL and ECLKDQSR signal for the “EA” devices. This will also implement the input register block in x1 gearing mode for generic DDRX1 interfaces.

On “EA” devices, the DDR registers use the primary clock (SCLK) only. The SCLK input should be connected to the system (FPGA) clock. “EA” devices do not require the control signals from the DQSBUF module in the IDDRXD1 element, making it more flexible for placement than the “E” device.

Figure 12-85 shows the primitive symbol and all the I/O ports.

**Figure 12-85. IDDRXD1 Symbol (“EA” Devices)**

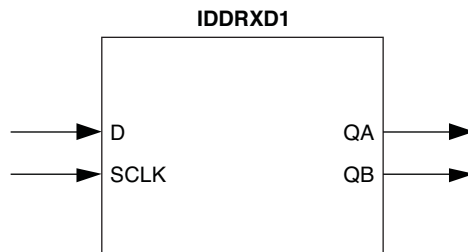


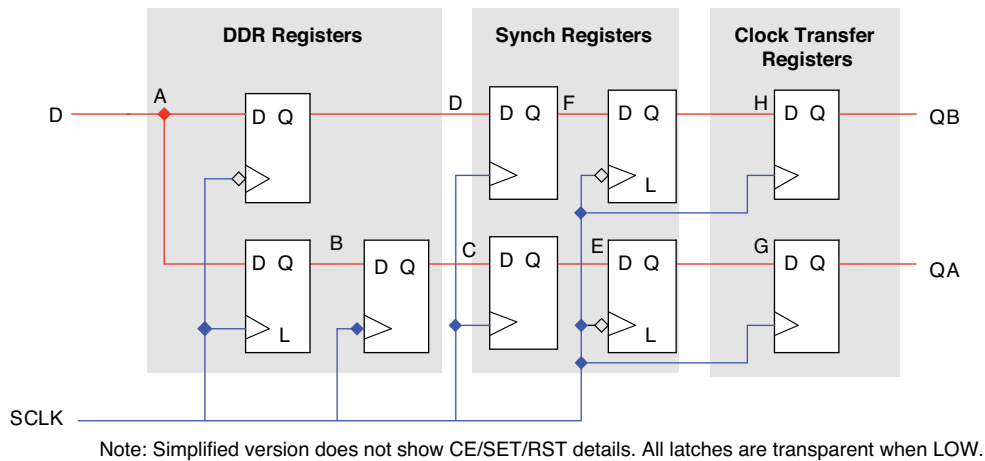
Table 12-24 provides a description of all I/O ports associated with the IDDRXD1 primitive.

**Table 12-24. IDDRXD1 Ports**

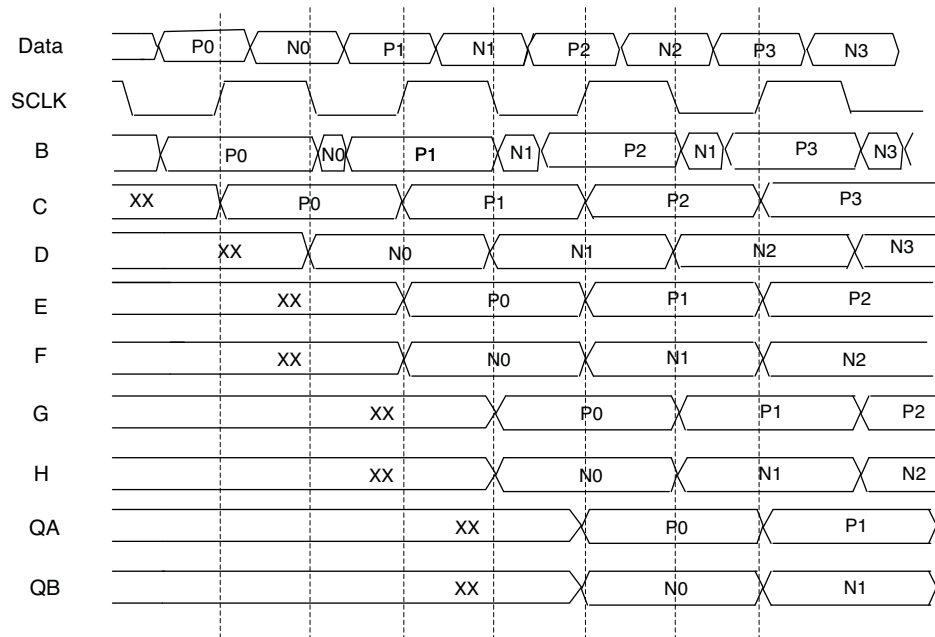
Port Name	I/O	Definition
D	I	DDR data
SCLK	I	System clock
QA	O	Data at the positive edge of the clock
QB	O	Data at the negative edge of the clock

Figure 12-86 shows the Input Register Block configured in the IDDRXD1 mode.

**Figure 12-86. Input Register Block in IDDRXD1 Mode (“EA” Devices)**



**Figure 12-87. IDDRXD1 Waveform**



**IDDRX2D**

This primitive will implement the input register block in x2 gearing mode. This mode is used to implement DDR3 memory interface on the LatticeECP3 “E” and “EA” devices. It is also used on “E” devices to capture the generic DDRX2 Input data.

Figure 12-88 shows the IDDRX2D primitive symbol and all the I/O ports.

**Figure 12-88. IDDRX2D Symbol**

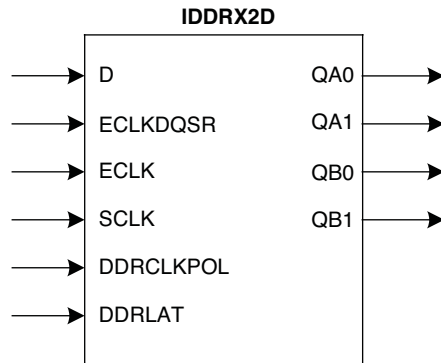


Table 12-25 provides a description of all I/O ports associated with the IDDRX2D primitive.

**Table 12-25. IDDRX2D Ports**

Port Name	I/O	Definition
D	I	DDR Data
ECLKDQSR	I	Phase-shifted DQS for DDR memory interfaces. Edge clock for generic DDR interfaces.
ECLK	I	Edge Clock. Should be connected to DQS strobe for DDR3 memory interfaces.
SCLK	I	System clock running at one-half the ECLK or DQS signal.
DDRCLKPOL	I	DDR clock polarity signal
DDRLAT	I	DDR latch control signal
QA0, QA1	O	Data at the positive edge of the clock.
QB0, QB1	O	Data at the negative edge of the clock.

Notes:

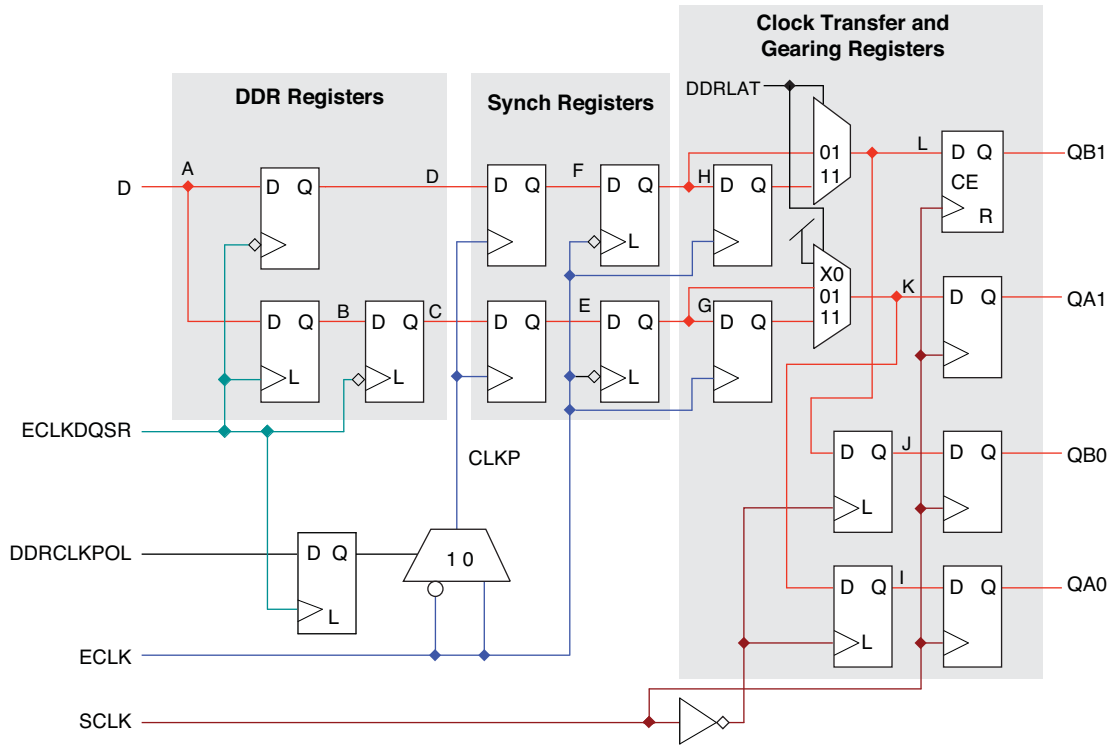
1. The DDRCLKPOL input to IDDRX2D should be connected to the DDRCLKPOL output of the DQSBUFD for DDR3 memory interfaces or the DDRCLKPOL output of the DQSBUFE for generic DDRX2 interfaces.
2. The DDRLAT input to the IDDRX2D should be connected to the DDRLAT output of the DQSBUFD for DDR3 memory interfaces or the DDRLAT output of the DQSBUFE for generic DDRX2 interfaces.

Figure 12-89 shows the LatticeECP3 Input Register Block configured to function in the IDDRX2D mode.

The ECLKDQSR input is used to connect to the DQS strobe coming from the DQS delay block (DQSBUFD) when implementing a DDR3 memory interface. For generic source synchronous DDR applications, this signal should be connected to the high-speed source synchronous edge clock input. The ECLK input is connected to the edge clock. The SCLK input should be connected to the system (FPGA) clock. The SCLK should run at half the frequency of ECLK.

The DDRCLKPOL and DDRLAT inputs are generated by the DQS transition detect circuit in the corresponding DQSBUF block. The DDRCLKPOL signal is used to choose the polarity of the ECLK to the synchronization registers. DDRLAT is used to transfer data from the ECLK to the SCLK in the Clock Transfer Register block.

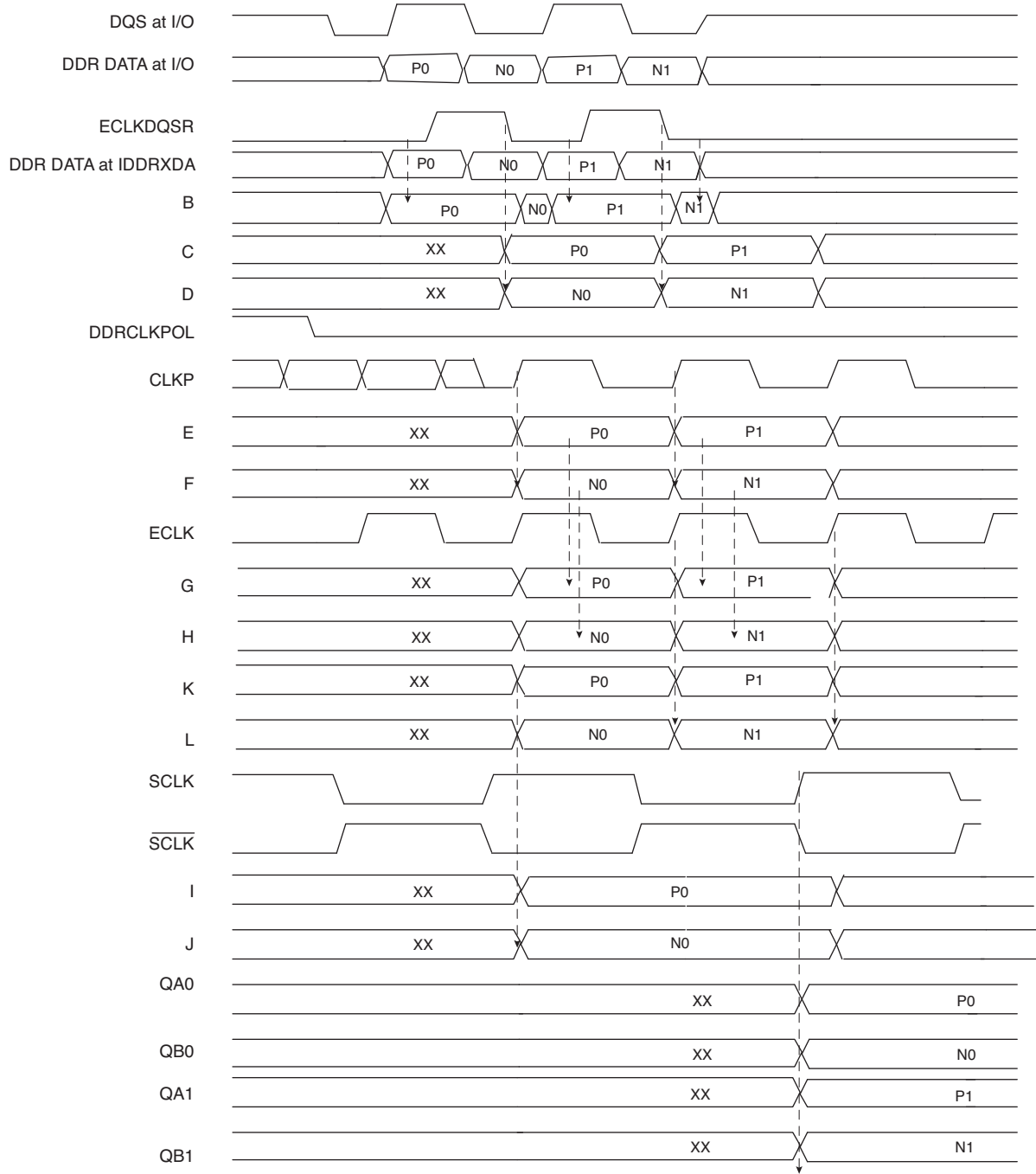
Figure 12-89. Input Register Block in IDDRX2D Mode



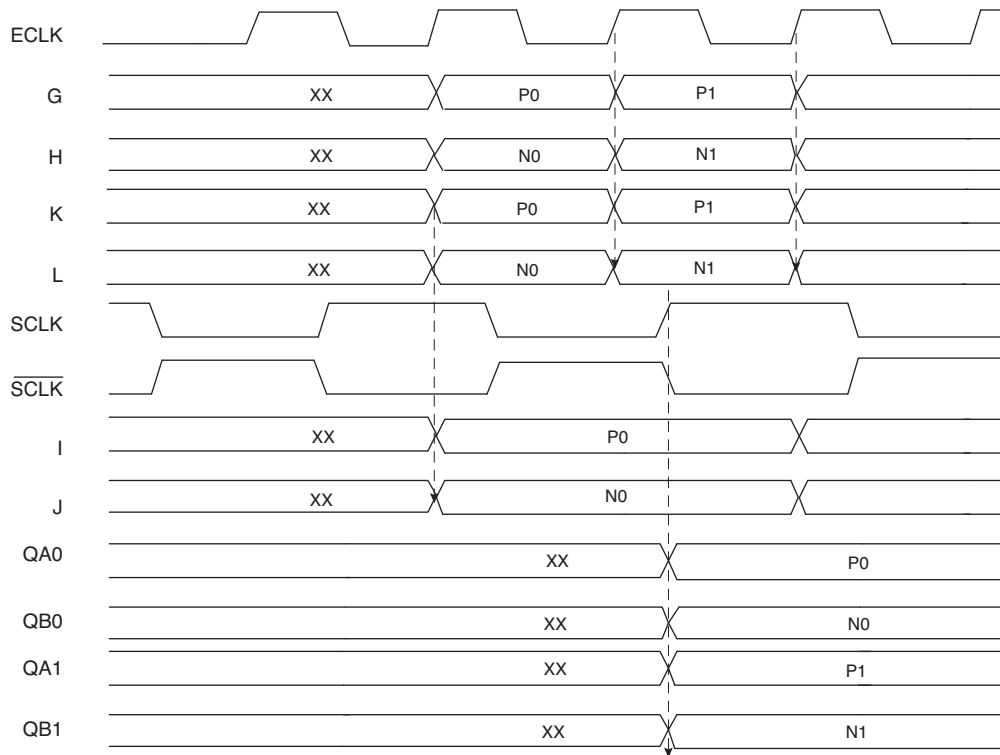
Notes:

1. Simplified version does not show CE/SET/RST details. All latches are transparent when LOW.
2. ECLKDQSR is connected to the DQS signal when in DDR memory mode. In DDR generic mode ECLKDQSR should be connected to the ECLK signal.

**Figure 12-90. IDDRX2D Waveform DDRLAT=0**



**Figure 12-91. IDDRX2D Waveform DDRLAT=1**



**IDDRX2D1**

This primitive is a simplified version of IDDRX2D without the DDRLAT, DDCLKPOL and the ECLKDQSR signals for the LatticeECP3 “EA” devices. It is used for input generic DDRX2 input data in “EA” devices.

In this case, the first stage of registers is clocked by the ECLK signal and the second stage is clocked by the SCLK signal. The “EA” device does not require the control signals from the DQSBUF module in the IDDRX2D1 element. This makes the “EA” device more flexible for placement than the “E” device.

Figure 12-92 shows the IDDRX2D1 primitive symbol and all the I/O ports.

**Figure 12-92. IDDRX2D1 Symbol (“EA” Devices)**

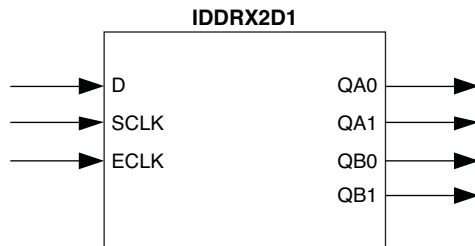


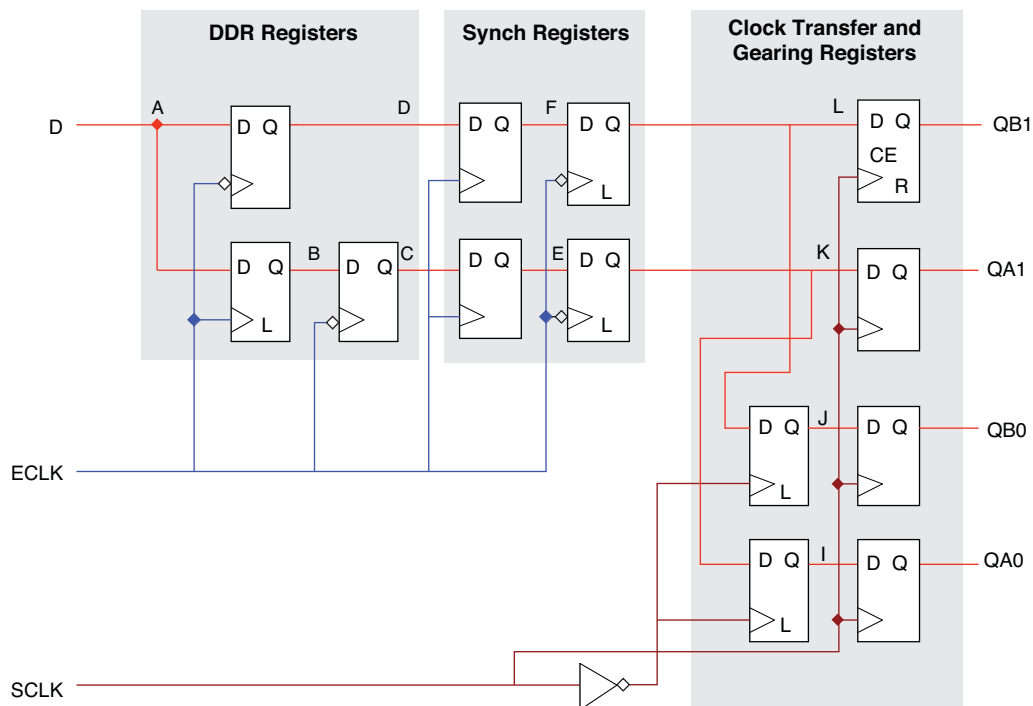
Table 12-26 provides a description of all I/O ports associated with the IDDRX2D1 primitive.

**Table 12-26. IDDRX2D1 Ports**

Port Name	I/O	Definition
D	I	DDR data
ECLK	I	Edge clock. Should be connected to DQS strobe for DDR3 memory interfaces.
SCLK	I	System clock running at one-half ECLK
QA0, QA1	O	Data at the positive edge of the clock.
QB0, QB1	O	Data at the negative edge of the clock.

Figure 12-93 shows the LatticeECP3 Input Register Block configured to function in IDDRX2D1 mode. The ECLK input is connected to the edge clock. The SCLK input should be connected to the system (FPGA) clock. The SCLK should run at half the frequency of ECLK.

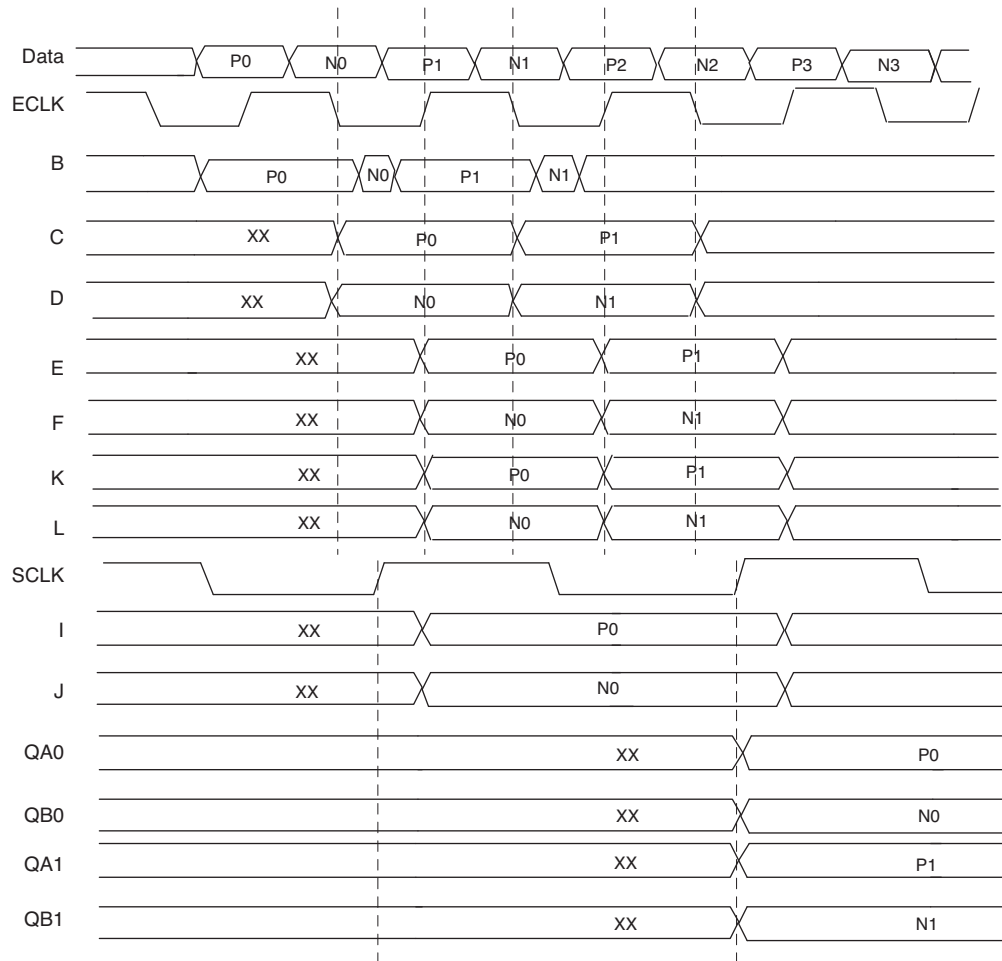
**Figure 12-93. Input Register Block in IDDRX2D1 Mode (“EA” Devices)**



Note: Simplified version does not show CE/SET/RST details. All latches are transparent when LOW.



**Figure 12-94. IDDRX2D1 Waveform**



**ECLKSYNCA**

ECLKSYNCA is used in x2 gearing Tx interfaces to synchronize the signals generated from the ECLK after RESET. These signals include the SCLK, DQCLK0/DQCLK1 and DQSW for DDR memory interfaces.

This module will STOP the ECLK to the CLKDIV and DQSBUF modules until the RESET is released.

Asserting the STOP input of the ECLKSYNCA will stop the ECLK output. When the STOP signal is released, every clock toggling from the second rising edge of the ECLK input will be output from this block. This block resides after the muxes for edge clock sources and before driving onto the actual edge clock.

Figure 12-95 shows the ECLKSYNCA primitive symbol.

**Figure 12-95. ECLKSYNCA Symbol**

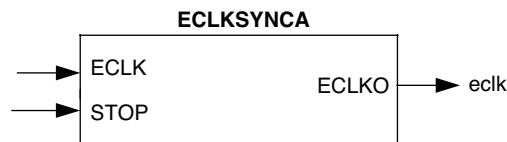


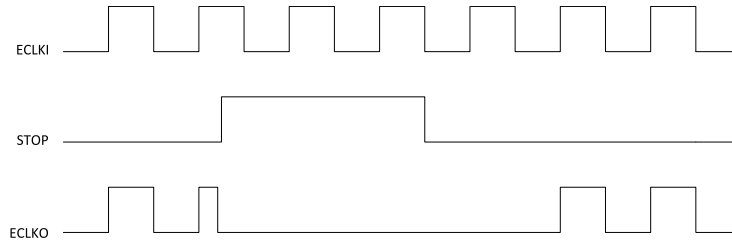
Table 12-27 lists the port descriptions of the ECLKSYNCA primitive.

**Table 12-27. ECLKSYNCA Port Descriptions**

Port Name	I/O	Definition
ECLK	I	Edge clock input
STOP	I	Signal used to stop the edge clock
ECLKO	O	Edge clock output

Figure 12-96 is a waveform that shows this operation.

**Figure 12-96. ECLKSYNC Operation**



This will stop the ECLK to the CLKDIV and DQSBUF until after these blocks are out of reset. By doing this, the user can synchronize the ECLK, SCLK and the DQCLKs used in the ODDR module.

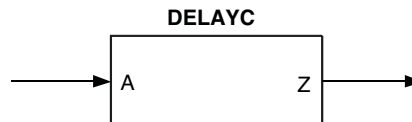
It is required that there is at least two clock cycles between the release of RESET and the release of the STOP input to ECLKSYNC. In the IPexpress-generated module, a soft IP consisting of two flip-flops is used to generate this delayed STOP signal to ECLKSYNC. The reset input to the CLKDIV and DQSBUF is used as an input to these two flip-flops. The clock input to these flip-flops must be slower than the ECLK.

Refer to the [GDDR2\\_TX.Aligned](#), [GDDR2\\_TX.DQSDLL.Centered](#) and [GDDR2\\_TX.PLL.Centered](#) interface descriptions in the [High-Speed DDR Interface Details](#) section.

**DELAYC**

Data going to the DDR registers can be optionally delayed using the delay block. The DELAYC block is used to compensate for clock injection delay times. The amount of the delay is determined by the software based on the type of interface implemented using the Interface ID attribute IDDRAPPS. Refer to [Interface ID Attribute](#) section for details. If an incorrect Interface ID is used for a given interface, then the DELAYC setting will be incorrect. It is important that the correct Interface ID attribute be assigned for each interface to allow the software to set the correct value for DELAYC.

**Figure 12-97. DELAYC Symbol**



**Table 12-28. DELAYC Port Names**

Port Name	I/O	Description
A	I	DDR input from sysIO™ buffer
Z	O	Delayed output

**DELAYB**

Data going to the DDR registers can also be delayed the DELAYB block. Unlike the DELAYC block where the software will control the amount of data delay, DELAYB block will allow user to control the amount of data delay. This block receives 4-bit delay control. The 4-bit delay can be set by using static delay values or can be dynamically

controlled by the user logic. DELAYB can only be used when the interface type is dynamic. See the [Building Generic High-Speed Interfaces](#) section for details.

The delay can be adjusted in 35ps steps. The user can choose from two types of delay values:

1. **Dynamic** – The delay value is controlled by the user logic using the inputs DEL[3:0] of the DELAYB block.
2. **User-Defined** – In this mode, the user chooses a static delay value from one of the 16 delay values. This will tie the inputs DEL[3:0] of the DELAYB block to a fixed value depending on the value chosen.

Figure 12-98 shows the primitive symbol for the DELAYB mode.

**Figure 12-98. DELAYB Symbol**

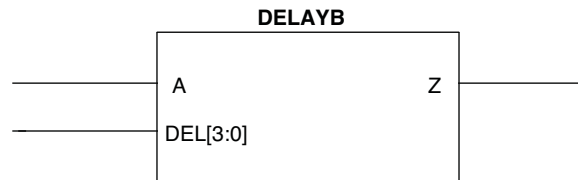


Table 12-29 lists the port names and descriptions for the DELAYB primitive.

**Table 12-29. DELAYB Port Names**

Port Name	I/O	Definition
A	I	DDR input from the sysIO buffer
DEL (0:3)	I	Delay inputs
Z	O	Delay DDR data

## Output DDR Primitives

The output DDR primitives represent the output DDR module used to generate both the generic DDR output data and the DDR memory interface data. There are two available modes for DDR output registers. One is used to implement DDRX1 gearing and the other for DDRX2 gearing.

### ODDRXD

This primitive will implement the output register block in x1 gearing mode. This mode is used to implement DDR/DDR2 memory interfaces on the “E” and “EA” devices. It is also used to generate the generic DDRX1 data on “E” devices.

Figure 12-99 shows the ODDRXD primitive symbol and its I/O ports.

**Figure 12-99. ODDRXD Symbol**

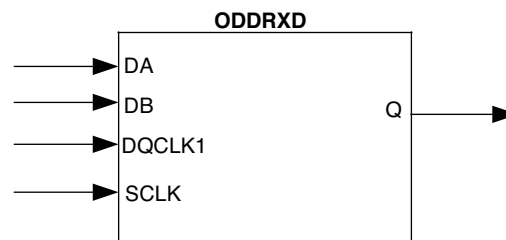


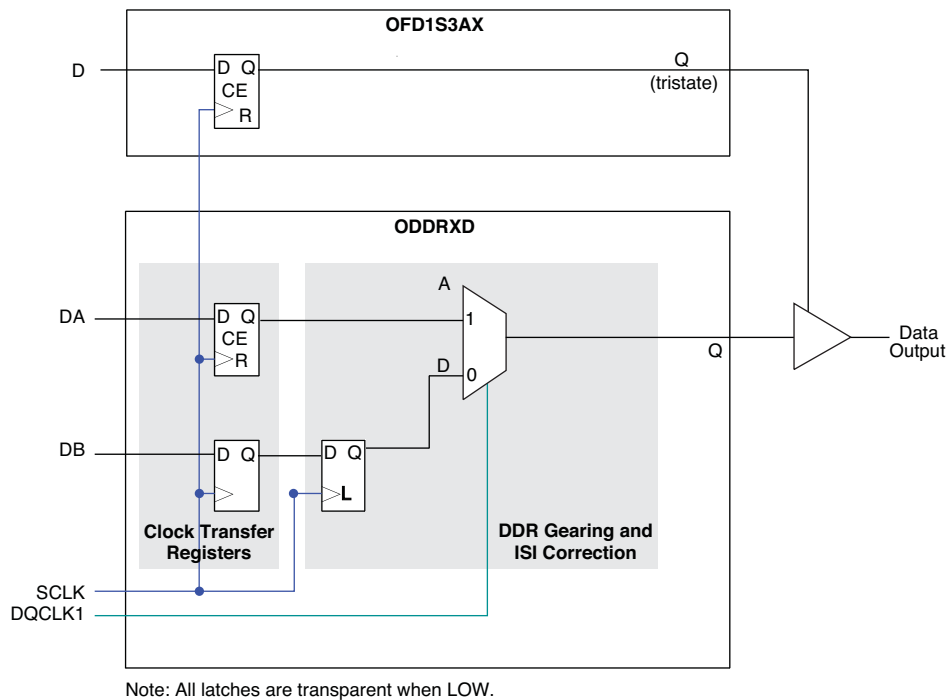
Table 12-30 provides a description of all I/O ports associated with the ODDRXD primitive.

**Table 12-30. ODDRXD Ports**

Port Name	I/O	Definition
SCLK	I	System CLK or ECLK
DA	I	Data at the positive edge of the clock
DB	I	Data at the negative edge of the clock
DQCLK1	I	Clock output at frequency of SCLK used for output gearing
Q	O	DDR data output

Figure 12-100 shows the LatticeECP3 Output Register Block configured in the ODDRXD mode.

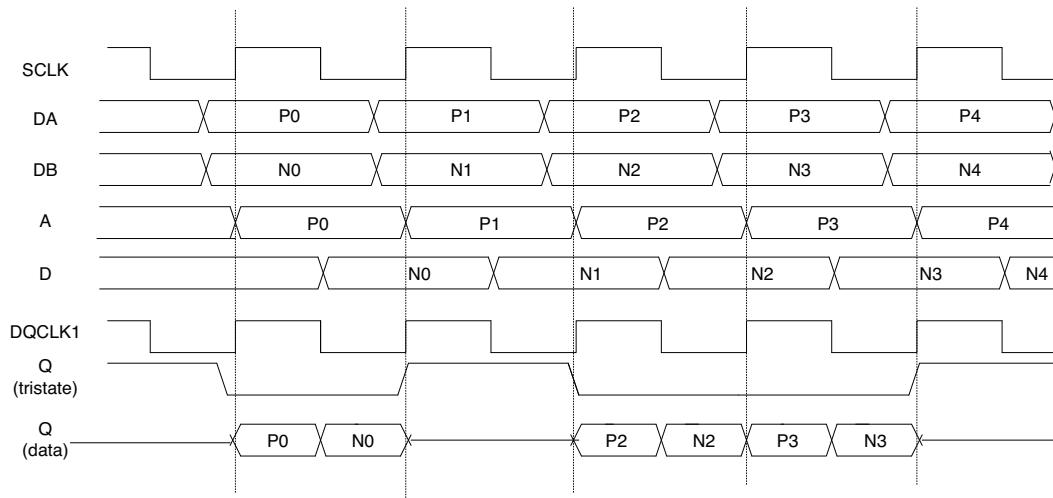
**Figure 12-100. Output Register Block in ODDRXD Mode**



*Note: Tristate control for ODDRXD can only be implemented using the OFD1S3AX module. If this module is not implemented in the user's design then software will infer this module. The clock used in the OFD1S3AX should be the same as the one used in the ODDRXD module. This module will not support tristate inversion.*

Figure 12-101 shows the ODDRXD timing waveform.

**Figure 12-101. ODDRXD Waveform**



**ODDRXD1**

This element is used to generate the generic DDRX1 data on “EA” devices. The ODDRXD1 in the “EA” device does not require the DQCLK1 control signal from the DQSBUF block. This makes the “EA” device more flexible for placement than “E” devices.

Figure 12-102 shows the ODDRXD1 primitive symbol and its I/O ports.

**Figure 12-102. ODDRXD1 Symbol (“EA” Devices)**

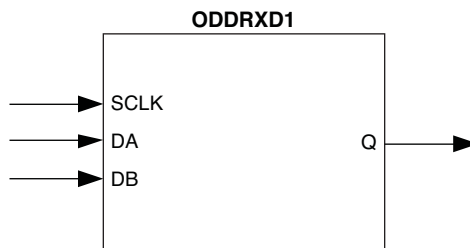


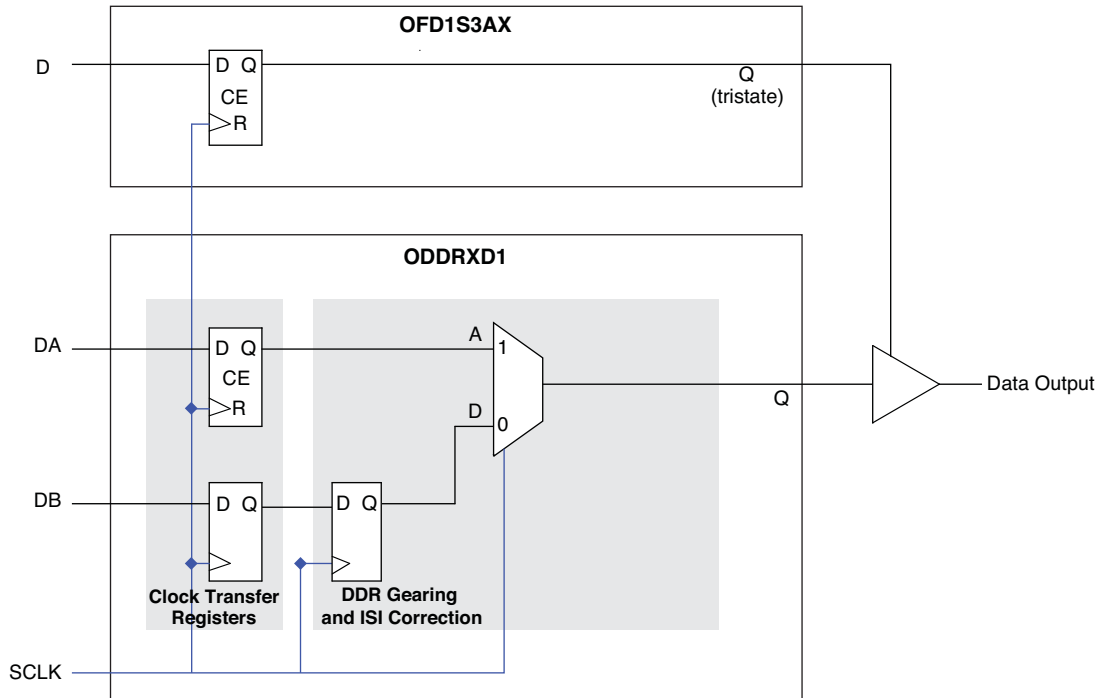
Table 12-31 provides a description of all I/O ports associated with the ODDRXD1 primitive.

**Table 12-31. ODDRXD1 Ports**

Port Name	I/O	Definition
SCLK	I	System CLK or ECLK
DA	I	Data at the positive edge of the clock
DB	I	Data at the negative edge of the clock
Q	O	DDR data output

Figure 12-103 shows the LatticeECP3 Output Register Block configured in the ODDRXD1 mode.

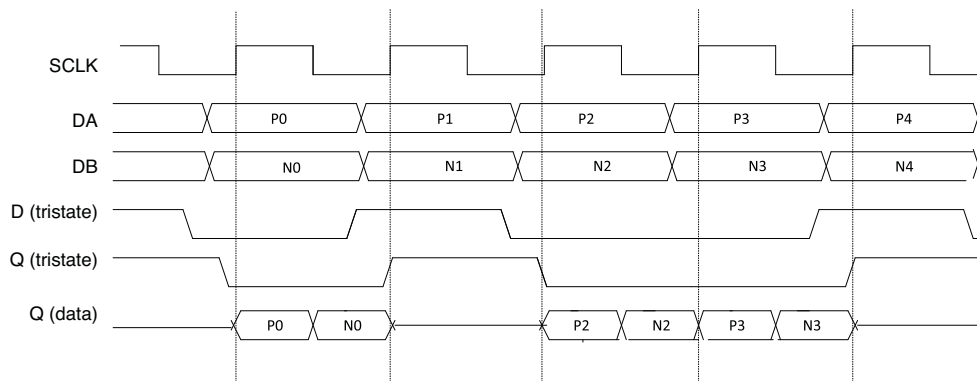
Figure 12-103. Output Register Block in ODDRXD1 Mode (“EA” Devices)



Note: All latches are transparent when LOW.

Note: Tristate control for ODDRXD1 can only be implemented using the OFD1S3AX module. If this module is not implemented in the user’s design then software will infer this module. The clock used in the OFD1S3AX should be the same as the one used in the ODDRXD1 module. This module will not support tristate inversion.

Figure 12-104. ODDRXD1 Waveform



**ODDRX2D**

The ODDRX2D primitive implements the output register for DDR3 memory and generic DDRX2 write functions.

Figure 12-105 shows the ODDRX2D primitive symbol and its I/O ports.

Figure 12-105. ODDRX2D Symbol

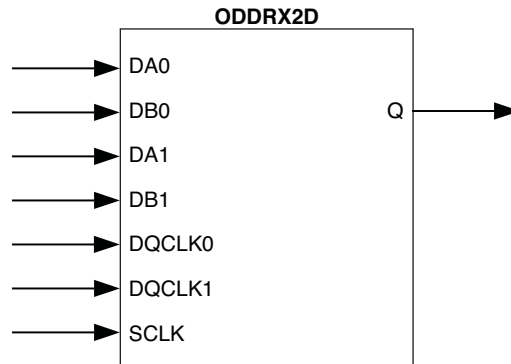


Table 12-32 provides a description of all I/O ports associated with the ODDRX2D primitive.

Table 12-32. ODDRX2D Ports

Port Name	I/O	Definition
SCLK	I	System CLK or ECLK
DA0	I	First data at the positive edge of the clock
DB0	I	First data at the negative edge of the clock
DA1	I	Second data at the positive edge of the clock
DB1	I	Second data at the negative edge of the clock
DQCLK0	I	Clock Output at frequency of SCLK used for output gearing
DQCLK1	I	Clock output at frequency of SCLK used for output gearing (90° shifted from DQCLK0)
Q	O	DDR data output

### ODDRTDQA

The ODDRTDQA primitive implements the tristate register block for DDR3 memory and generic x2 DDR write functions.

Figure 12-106 shows the ODDRTDQA primitive symbol and its I/O ports.

Figure 12-106. ODDRTDQA Symbol

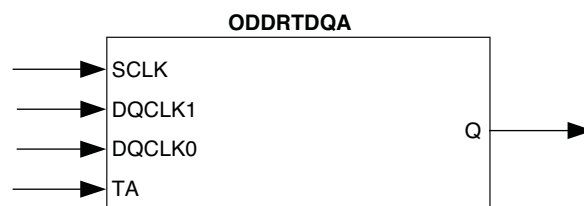


Figure 12-33 provides a description of all I/O ports associated with the ODDRTDQA primitive.

**Table 12-33. ODDRTDQA Ports**

Port Name	I/O	Definition
SCLK	I	System CLK or ECLK
TA	I	Tristate input
DQCLK0	I	Clock output at frequency of SCLK used for output gearing
DQCLK1	I	Clock output at frequency of SCLK used for output gearing (90° shifted from DQCLK0)
Q	O	DDR tristate output

**Figure 12-107. Output Register Block in ODDR2D/ODDRTDQA Mode**

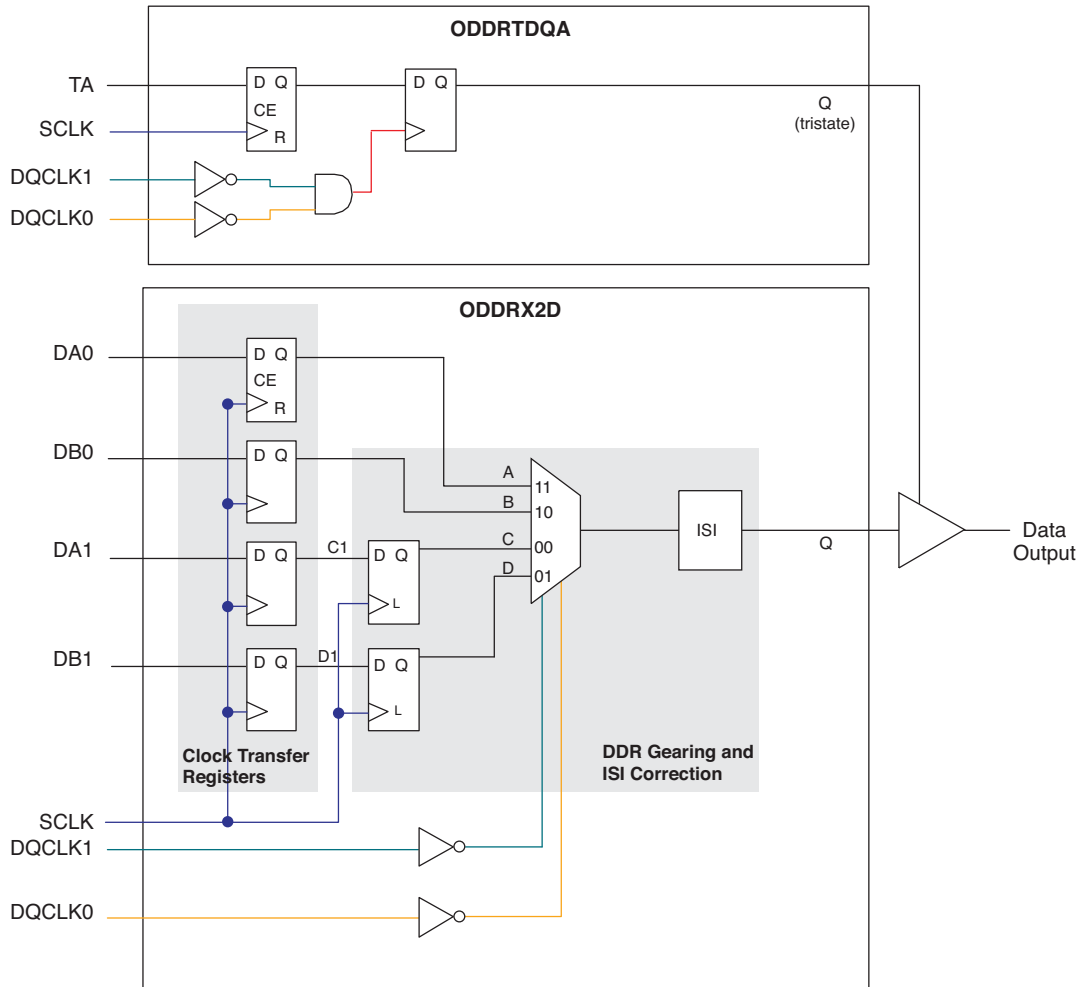


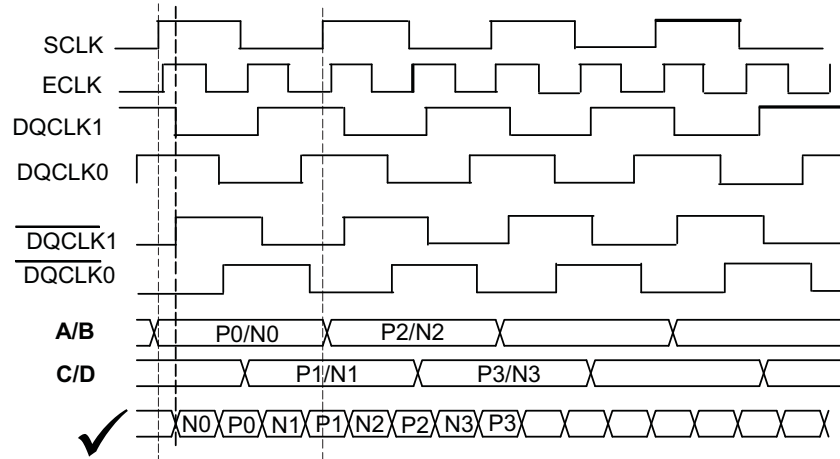
Figure 12-107 shows the LatticeECP3 Output Register Block configured in the ODDR2D and ODDRTDQA tristate modes.

*Note: Tristate control for ODDR2D can only be implemented using the ODDRTDQA module. The clock used in the ODDRTDQA should be the same as the one used in the ODDR2D module. This module will not support tristate inversion.*

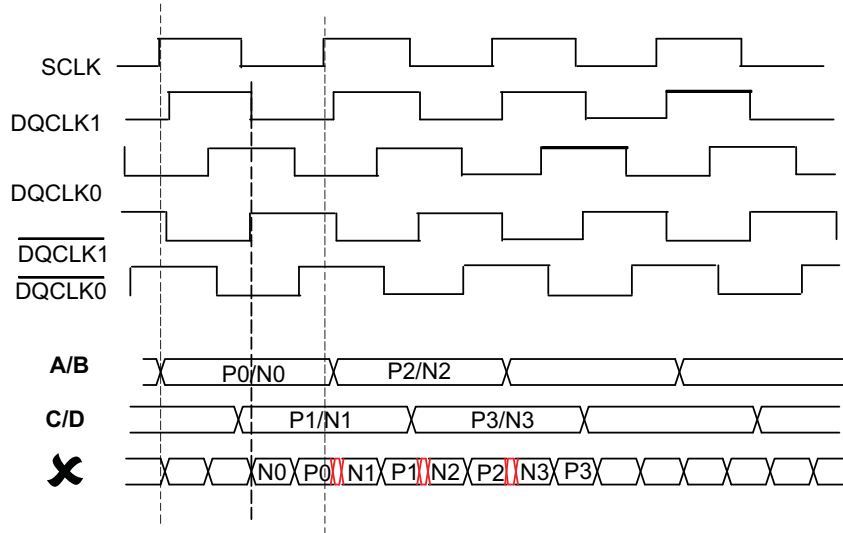


On the ODDR2, it is required that the SCLK be in correct phase with DQCLK1 for the data to be captured correctly inside the ODDR2. The figure below explains the correct relationship between the SCLK and DQCLK1. SCLK edge must be delayed to occur after the DQCLK1 edge for the data to be captured without any glitches.

**Figure 12-108. Correct DQCLK1 Polarity**



**Figure 12-109. Incorrect DQCLK1 Polarity**



The soft IP, ECLKSYNCA, CLKDIVB, DQSBUFE1 and the SCLK routing delay will ensure the correct phase relationship between SCLK and DQCLK inside the ODDR2 module. The user must generate the interface using IPexpress to guarantee this delay is achieved.

**ODDRXDQSA**

The ODDRXDQSA primitive implements the output register for generating the DQS strobe signal for DDR and DDR2 memory

Figure 12-110 shows the ODDRXDQSA primitive symbol and its I/O ports.

Figure 12-110. ODDRXDQSA Symbol



Table 12-34 provides a description of all I/O ports associated with the ODDRXDQSA primitive.

Table 12-34. ODDRXDQSA Ports

Port Name	I/O	Definition
SCLK	I	System CLK or ECLK
DA	I	Data input
DQSW	I	DQS write clock
DQCLK1	I	Clock output at frequency of SCLK used for output gearing
DQSTCLK	O	DQS tristate clock
Q	O	DQS data output

### ODDRTDQSA

The ODDRTDQSA primitive implements the tristate register block for DDR/DDR2 and DDR3 memory DQS output clock generation.

Figure 12-111 shows the ODDRTDQSA primitive symbol and its I/O ports.

Figure 12-111. ODDRTDQSA Symbol

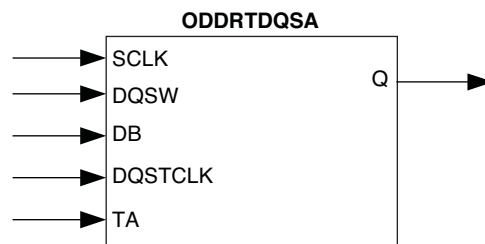


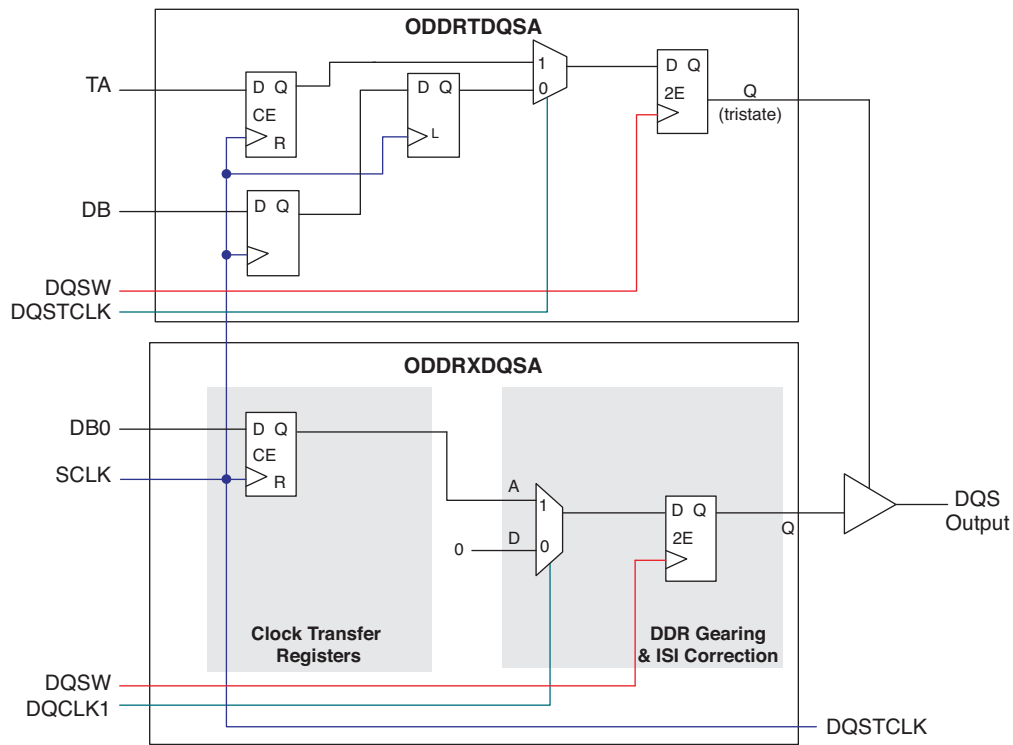
Table 12-35 provides a description of all I/O ports associated with the ODDRTDQSA primitive.

Table 12-35. ODDRTDQSA Ports

Port Name	I/O	Definition
SCLK	I	System CLK or ECLK
DB	I	Data input
DQSW	I	DQS write clock
DQSTCLK	I	DQS tristate Clock
TA	I	Tristate input
Q	O	DQS tristate output

Figure 12-112 shows the LatticeECP3 Output Register Block configured in the ODDRXDQSA and ODDRTDQSA tristate modes.

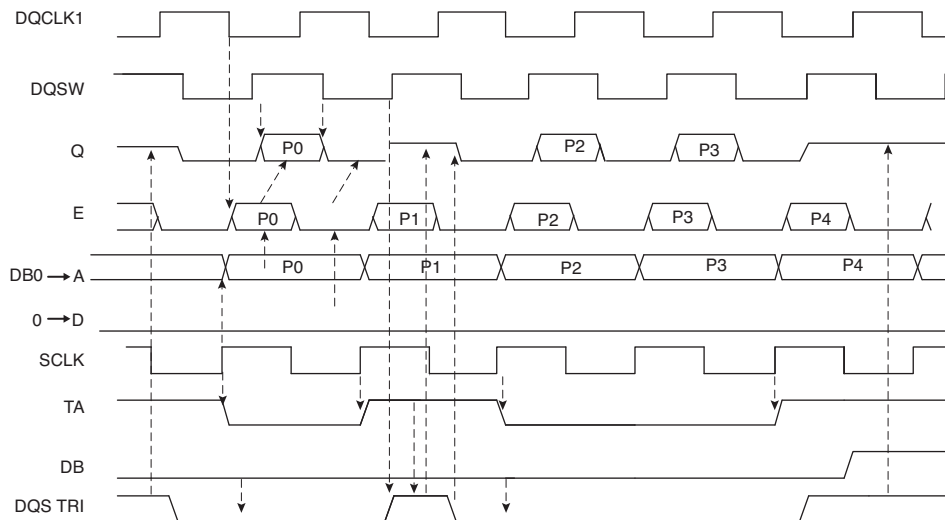
Figure 12-112. Output Register Block in ODDRXDQSA/ODDRTDQSA Mode



Note: Tristate control for ODDRXDQSA can only be implemented using the ODDRTDQSA module. The clock used in the ODDRTDQSA should be the same as the one used in the ODDRXDQSA module. This module will not support tristate inversion.

Figure 12-113 shows the ODDRXDQSA and ODDRTDQSA timing waveform.

Figure 12-113. ODDRXDQSA/ODDRTDQSA Waveform



**ODDRX2DQSA**

The ODDRX2DQSA primitive implements the output register for generating the DQS strobe signal for DDR3 memory interfaces.

Figure 12-114 shows the ODDRX2DQSA primitive symbol and its I/O ports.

**Figure 12-114. ODDRX2DQSA Symbol**

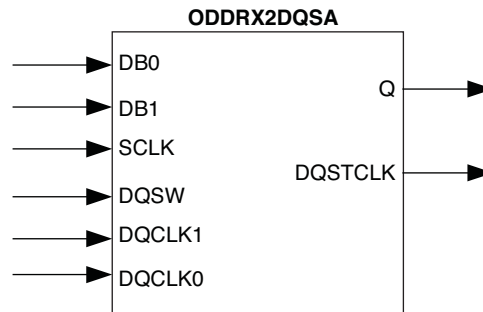


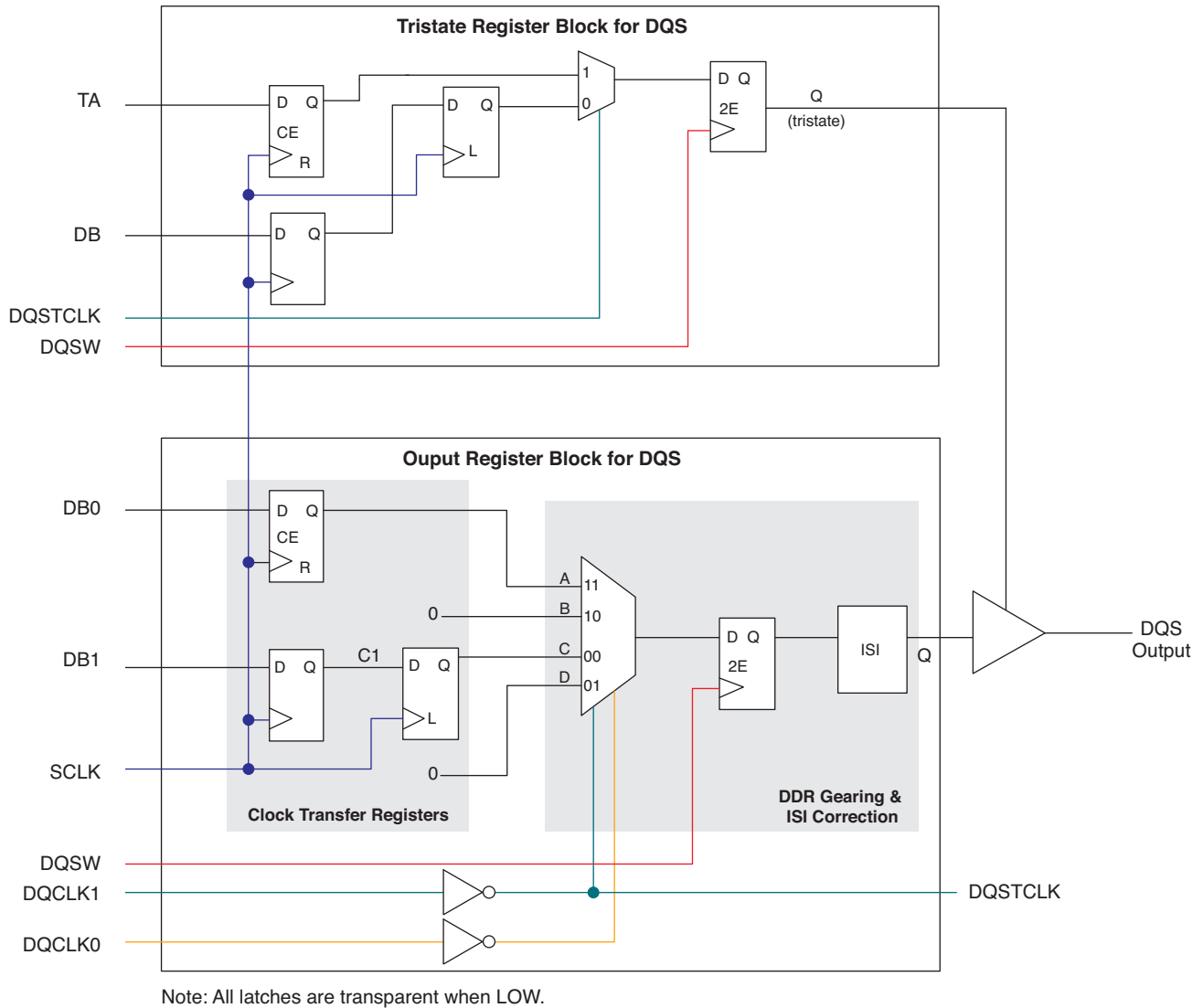
Table 12-36 provides a description of all I/O ports associated with the ODDRX2DQSA primitive.

**Table 12-36. Table 33 ODDRX2DQSA Ports**

Port Name	I/O	Definition
SCLK	I	System CLK or ECLK
DB0	I	Data input
DB1	I	Data input
DQSW	I	DQS write clock
DQCLK0	I	Clock output at frequency of SCLK used for output gearing
DQCLK1	I	Clock output at frequency of SCLK and shifted 90°, used for output gearing
DQSTCLK	O	DQS tristate clock
Q	O	DDR data output

Table 12-115 shows the LatticeECP3 Output Register Block configured in the ODDRX2DQSA and ODDRTDQSA tristate mode.

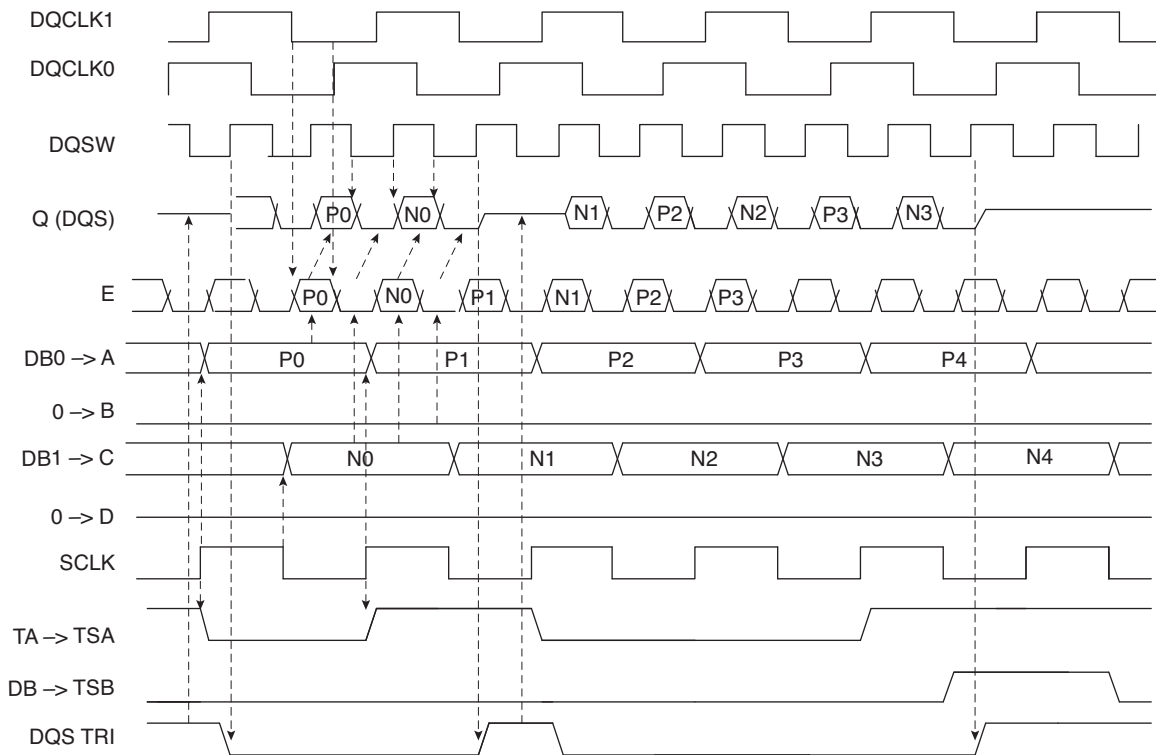
Figure 12-115. Output Register Block in ODDR2DQSA/ODDRTDQSA Mode



Note: Tristate control for ODDR2DQSA can only be implemented using the ODDRTDQSA module. The clock used in the ODDRTDQSA should be the same as the one used in the ODDR2DQSA module. This module will not support tristate inversion.

Figure 12-116 shows the ODDR2DQSA timing waveform.

**Figure 12-116. ODDR2DQSA/ODDRTDQSA Waveform**



**Interface ID Attribute**

This attribute provides an ID setting for each of the high-speed interfaces in the LatticeECP3 “EA” device. IDDRAPPS attribute is used for input interfaces and ODDRAPPS attribute is used for output interfaces. The value for these is pre-determined for each high-speed DDR interfaces. These attributes will be set to the correct values when the interfaces are generated by IPexpress. If an IDDRAPPS or ODDRAPPS attribute is not set for a given interface, the software will error out. If an attribute value is not set correctly for a given interface, then the wrong data delay is used in the DELAYC element for that interface.

The IDDRAPPS and ODDRAPPS attributes are strings. They are only generated for “EA” devices.

**Table 12-37. Interface ID (IDDRAPPS/ODDRAPPS) Attribute Values**

Interfaces Name	Interface ID (Primitive: ATTRIBUTE = Value) <sup>1,2</sup>
<b>Generic Interfaces</b>	
GIREG_RX.SCLK	N/A
GDDR1_RX.SCLK.Aligned	IDDRXD1: IDDRAPPS = SCLK_ALIGNED
GDDR1_RX.SCLK.Centered	IDDRXD1: IDDRAPPS = SCLK_CENTERED
GDDR1_RX.DQS.Aligned	IDDRXD: IDDRAPPS = DQS_ALIGNED
GDDR1_RX.DQS.Centered	IDDRXD: IDDRAPPS = DQS_CENTERED
GDDR2_RX.ECLK.Aligned	IDDRXD2D1: IDDRAPPS = ECLK_ALIGNED
GDDR2_RX.ECLK.Centered	IDDRXD2D1: IDDRAPPS = ECLK_CENTERED
GDDR2_RX.DQS.Aligned	IDDRXD2D: IDDRAPPS = DQS_ALIGNED
GDDR2_RX.DQS.Centered	IDDRXD2D: IDDRAPPS = DQS_CENTERED
GDDR2_RX.ECLK.Dynamic	IDDRXD2D1: IDDRAPPS = ECLK_DYNAMIC
GDDR2_RX.DQS.Dynamic	IDDRXD2D: IDDRAPPS = DQS_DYNAMIC
GDDR2_RX.PLL.Dynamic	IDDRXD2D1: IDDRAPPS = PLL_DYNAMIC

**Table 12-37. Interface ID (IDDRAPPS/ODDRAPPS) Attribute Values (Continued)**

Interfaces Name	Interface ID (Primitive: ATTRIBUTE = Value) <sup>1,2</sup>
GOREG_TX.SCLK	ODDRXD1: ODDRAPPS = SCLK_ALIGNED
GDDR1_TX.SCLK.Centered	ODDRXD1: ODDRAPPS = SCLK_CENTERED (CLOCK) ODDRXD1: ODDRAPPS = SCLK_ALIGNED (DATA)
GDDR1_TX.SCLK.Aligned	ODDRXD1: ODDRAPPS = SCLK_ALIGNED (CLOCK) ODDRXD1: ODDRAPPS = SCLK_ALIGNED (DATA)
GDDR1_TX.DQS.Centered	ODDRXDQSA: ODDRAPPS = DQS_CENTERED (CLOCK) ODDRXD: ODDRAPPS = DQS_ALIGNED (DATA)
GDDR2_TX.Aligned	ODDRXD2: ODDRAPPS = ECLK_ALIGNED (CLOCK) ODDRXD2: ODDRAPPS = ECLK_ALIGNED (DATA)
GDDR2_TX.DQSDLL.Centered	ODDRXDQSA: ODDRAPPS = DQS_CENTERED (CLOCK) ODDRXD2: ODDRAPPS = DQS_ALIGNED (DATA)
GDDR2_TX.PLL.Centered	ODDRXD2: ODDRAPPS = ECLK_CENTERED (CLOCK) ODDRXD2: ODDRAPPS = ECLK_ALIGNED (DATA)
GDDR1_RX.ECLK.Aligned	N/A: not a LatticeECP3 “EA” configuration.
GDDR1_RX.ECLK.Centered	N/A: not a LatticeECP3 “EA” configuration.
<b>DDR Memory Interfaces</b>	
DDR MEM	ODDRXDQSA: ODDRAPPS = DDR_MEM_DQS IDDRXD: IDDRAPPS = DDR_MEM_DQ ODDRXD: ODDRAPPS = DDR_MEM_DQ
DDR2 MEM	ODDRXDQSA: ODDRAPPS = DDR2_MEM_DQS IDDRXD: IDDRAPPS = DDR2_MEM_DQ ODDRXD: ODDRAPPS = DDR2_MEM_DQ

1. Attribute should be assigned on the corresponding IDDRX/ODDRX primitives.

2. Interface IDs are only valid for LatticeECP3 “EA” devices.

## ISI Calibration

ISI correction is only available in the ODDR2D or ODDR2DQSA modes on the left and right sides of the device. ISI calibration settings exist once per output, so each I/O in a DQS-12 group may have a different ISI calibration setting.

The ISI Calibration is set using the ISI\_CAL attribute. Table 12-38 shows the values that can be set for this attribute.

**Table 12-38. ISI Calibration Attribute**

Attribute	Description	Values	Software Default
ISI_CAL	Used to set the ISI Correction values	BYPASS, DEL1, DEL2, DEL3, DEL4, DEL5, DEL6, DEL7	BYPASS

The ISI block extends output signals at certain times, as a function of recent signal history, so it can be read at the output signal’s destination. If the output pattern consists of long strings of 0s to long strings of 1s, there are no delays on output signals. However, if there are quick, successive transitions from 010, the block will stretch out the binary 1. This is because the long trail of 0s will cause these symbols to interfere with the logic 1. Likewise, if there are quick, successive transitions from 101, the block will stretch out the binary 0.

This block is controlled by a 3-bit delay “stretching” control, set in the DQS logic section. There are eight settings in the range from “BYPASS” to “DEL7”.

## Migrating Designs from LatticeECP3 “E” to “EA”

This section lists the changes in design required when moving from LatticeECP3 “E” device to an “EA” device. The LatticeECP3-150EA device was designed as an “E” device in ispLEVER 7.2 SP2. So these changes will also be required if moving a LatticeECP3-150EA design from ispLEVER 7.2 SP2 to ispLEVER 8.0 or later versions. It is required that all LatticeECP3-150EA designs be implemented in ispLEVER 8.0 or newer software. Refer to the section [Migrating Designs from ispLEVER 7.2 SP2 to ispLEVER 8.0](#) to see other changes required when you are moving designs from ispLEVER 7.2 SP2 to ispLEVER 8.0 or later versions of the software. The same changes will apply if moving from ispLEVER 7.2 to the Lattice Diamond design software as well.

A summary of the differences between the “E” and “EA” devices are listed below:

- DDR x1 interfaces on “E” devices use ECLK (edge clock) as the clock input limiting the number of interfaces to 1 per side. DDR x1 interfaces on “EA” devices use the SCLK clock input, so you can have more than one interface per side of the device.
- “E” device requires that DQSBUF be used to implement both x1 and x2 input and output DDR functions. “EA” devices do not require the DQSBUF to implement the input and x1 output DDR functions. X2 output DDR functions will require the use of DQSBUF.
- The “E” devices require the data pins to be grouped into DQS groups so that every 10 data bits are locked to a DQS group. This grouping is not required on the “EA” devices for inputs and 1x output interfaces. 2x output interfaces on “EA” would require DQS grouping.
- The primitives used for Generic DDR input and output functions are different between “E” and “EA”.
- All “EA” designs require that the IDDRAPPS/ODDRAPPS be assigned to each interface which is not required on the “E” device. Refer to the section [Interface ID Attribute](#) for a description of this attribute. IPexpress-generated modules will contain this attribute.
- In DDR and DDR2 memory, the implementation will mostly be same between “E” and “EA” devices. But since DDR and DDR2 use generic DDR to generate the output CLKP/CLKN signal, these must use new primitives on the “EA” device. In addition, all the primitives require the IDDRAPPS/ODDRAPPS attributes to be added for the “EA” device.

Refer to the section [DDR Software Primitives and Attributes](#) for a detailed description of the primitives. Table 12-39 lists the clocking differences between “E” and “EA” devices for each interface. Table 12-40 lists the differences in library elements used for each interfaces. Refer to the [High-Speed DDR Interface Details](#) section for block diagrams of each of the interfaces in the tables below.

**Table 12-39. “E” to “EA” Clocking Differences**

“E” Interface	Clocking Resource	DQS Grouping for Pins	Equivalent “EA” Interface	Clocking Resource	DQS Grouping for Pins
GIREG_RX.SCLK	SCLK	No	GIREG_RX.SCLK	SCLK	No
GDDR1_RX.ECLK.Aligned	ECLK	Yes	GDDR1_RX.SCLK.Aligned	SCLK	No
GDDR1_RX.ECLK.Centered	ECLK	Yes	GDDR1_RX.SCLK.Centered	SCLK	No
GDDR1_RX.DQS.Aligned	DQS Tree	Yes	GDDR1_RX.DQS.Aligned	DQS Tree	Yes
GDDR1_RX.DQS.Centered	DQS Tree	Yes	GDDR1_RX.DQS.Centered	DQS Tree	Yes
GDDR2_RX.ECLK.Aligned	ECLK	Yes	GDDR2_RX.ECLK.Aligned	ECLK	No



**Table 12-40. “E” to “EA” Primitive Changes**

“E” Interface	Primitives Required	Equivalent “EA” Interface	Primitives Used
GIREG_RX.SCLK	IFS1P3DX	GIREG_RX.SCLK	IFS1P3DX
GDDR1_RX.ECLK.Aligned	IDDRXD DQSBUFG TRDLLB DLLDELB CLKDIVB	GDDR1_RX.SCLK.Aligned	IDDRX1D TRDLLB DLLDELB CLKDIVB
GDDR1_RX.ECLK.Centered	IDDRXD DQSBUFG	GDDR1_RX.SCLK.Centered	IDDRX1D
GDDR1_RX.DQS.Aligned	IDDRXD DQSBUFF DQSDLL	GDDR1_RX.DQS.Aligned	IDDRXD DQSBUFF DQSDLL
GDDR1_RX.DQS.Centered	IDDRXD DQSBUFF DQSDLL	GDDR1_RX.DQS.Centered	IDDRXD DQSBUFF DQSDLL
GDDR2_RX.ECLK.Aligned	IDDRX2D DQSBUFE TRDLLB DLLDELB CLKDIV	GDDR2_RX.ECLK.Aligned	IDDRX2D1 TRDLLB DLLDELB CLKDIVB
GDDR2_RX.ECLK.Centered	IDDRX2D DQSBUFE CLKDIVB	GDDR2_RX.ECLK.Centered	IDDRX2D1 CLKDIVB
GDDR2_RX.DQS.Aligned	IDDRX2D DQSBUFD DQSDLL PLL	GDDR2_RX.DQS.Aligned	IDDRX2D DQSBUFD DQSDLL PLL
GDDR2_RX.DQS.Centered	IDDRX2D DQSBUFD DQSDLL CLKDIVB	GDDR2_RX.DQS.Centered	IDDRX2D DQSBUFD DQSDLL CLKDIVB
GOREG_TX.SCLK	OFS1P3DX (data) ODDRXD (clock) DQSBUFG (left/right sides only)	GOREG_TX.SCLK	OFS1P3DX (data) ODDRXD1
GDDR1_TX.SCLK.Centered	ODDRXD DQSBUFG (left/right sides only) PLL	GDDR1_TX.SCLK.Centered	ODDRXD1 PLL
GDDR1_TX.SCLK.Aligned	ODDRXD DQSBUFG	GDDR1_TX.SCLK.Aligned	ODDRX1D
GDDR1_TX.DQS.Centered	ODDRXD (data) ODDRDQSA (clock) DQSBUFF DQSDLL	GDDR1_TX.DQS.Centered	ODDRXD (data) ODDRDQSA (clock) DQSBUFF DQSDLL

Due to the differences described above, some of the “E” interfaces must be regenerated in the software as an “EA” device. Table 12-41 can be used as a guide to determine which interfaces need to be regenerated. When necessary, interfaces should be regenerated using the IPexpress software tool.

**Table 12-41. “E” to “EA” Design Conversion Table**

“E” Interface	Equivalent “EA” Interface	Regeneration Required <sup>1</sup>
GIREG_RX.SCLK	GIREG_RX.SCLK	No
GDDR1_RX.ECLK.Aligned	GDDR1_RX.SCLK.Aligned	Yes
GDDR1_RX.ECLK.Centered	GDDR1_RX.SCLK.Centered	Yes
GDDR1_RX.DQS.Aligned	GDDR1_RX.DQS.Aligned	No
GDDR1_RX.DQS.Centered	GDDR1_RX.DQS.Centered	No
GDDR2_RX.ECLK.Aligned	GDDR2_RX.ECLK.Aligned	Yes
GDDR2_RX.ECLK.Centered	GDDR2_RX.ECLK.Centered	Yes
GDDR2_RX.DQS.Aligned	GDDR2_RX.DQS.Aligned	No
GDDR2_RX.DQS.Centered	GDDR2_RX.DQS.Centered	No
GOREG_TX.SCLK	GOREG_TX.SCLK	Yes
GDDR1_TX.SCLK.Centered	GDDR1_TX.SCLK.Centered	Yes
GDDR1_TX.SCLK.Aligned	GDDR1_TX.SCLK.Aligned	Yes
GDDR1_TX.DQS.Centered	GDDR1_TX.DQS.Centered	No

1. All designs should be regenerated using IPexpress.

## Migrating Designs from ispLEVER 7.2 SP2 to ispLEVER 8.0

This section lists changes that need to be made to existing designs when migrating from ispLEVER 7.2 SP2 to ispLEVER 8.0 and later versions of the software. The same changes will apply if moving from ispLEVER 7.2 to the Lattice Diamond design software as well.

### LatticeECP3-70E and LatticeECP3-90E designs:

- No HDL changes are required to move the LatticeECP3 “E” generic designs from ispLEVER 7.2 SP2 to ispLEVER 8.0
- It is required to re-run the designs through the software Map, Place and Route. ispLEVER 8.0 has an added Design Rule Check that will flag any general non-dedicated routed used on clock paths to DDR interfaces.
- If any of these errors occur, they must be fixed by either changing the clock pin assignment to a dedicated pin or by adding preferences to route the clock on dedicated clock routes.
- All the interface rules and placement guidelines specified for the interface must be followed.

### LatticeECP3-150EA designs:

- All “EA” designs were implemented as “E” designs in ispLEVER 7.2 SP2. All “EA” generic DDR designs will have to be regenerated when moving over to ispLEVER 8.0 except for the designs using DQS interfaces.
- The requirements listed in the section [Migrating Designs from LatticeECP3 “E” to “EA”](#) should be followed when moving “EA” designs from ispLEVER 7.2 SP2 to ispLEVER 8.0.
- All modules should be regenerated using IPexpress.
- “EA” designs require that an Interface ID attribute, IDDRAPPS or ODDRAPPS, be added to all the IDDR and ODDR elements to indicate the interface topology being used. The software will error out if it does not see this attribute on the DDR elements. These attributes are added automatically when IPexpress is used to generate the interface.

### Other important migration rules:

- All DDR designs must only use one of the pre-defined topologies listed in this technical note.
- All DDR interfaces must be generated using the IPexpress software tool.

- 
- This technical note must be strictly followed to understand the interface rules, placement guidelines and timing analysis for all interfaces.
  - ispLEVER 7.2 SP2 allows the use of either the DELAYB or DELAYC element to delay the incoming data. In ispLEVER 8.0, all non-dynamic interfaces must only use the DELAYC element to delay the data input. DELAYB can only be used only for dynamic interfaces.
  - A new Interface ID attribute IDDRAPPS and ODDRAPPS is required for “EA” devices. The software uses this attribute to determine the delay settings to be programmed into the DELAYC element. Refer to the [DDR Software Primitives and Attributes](#) section for the values to be used for this attribute. In ispLEVER 8.0 the software will error out if this attribute is not assigned for “EA” devices. “E” devices do not require this attribute.
  - It is required that clocks connected to the DDR registers only use dedicated clock resources. No general routing should be used on these clocks. A new DRC check was added to ispLEVER 8.0 Place and Route that will generate an error message if the clocks to any of the DDR elements are not using a dedicated clock route. This check will also generate an error message if the clock going to the PLL/DLL that is generating the DDR clocks is not using a dedicated route. As a result, you may see designs passing all DRC checks in ispLEVER 7.2 SP2 fail in ispLEVER 8.0. It is recommended that this problem be fixed by changing either clock location or adding preferences to reroute the clock using a dedicated clock route.

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
June 2009	01.1	Updated for ispLEVER 7.2 SP2. Some of the implementation listed here may not be valid if using ispLEVER 7.2 SP1.
November 2009	01.2	Updated for the LatticeECP3 "EA" device and for ispLEVER 8.0 software support. Includes new methodology to implement DDR interfaces in LatticeECP3 devices.
November 2009	01.3	Updated DDR3 termination and pin assignments.
March 2010	01.4	Updated the termination scheme for DDR3 to external VTT termination.
April 2010	01.5	Updated to reflect all the Generic DDR and DDR memory enhancements made in ispLEVER 8.0 Service Pack 1.
June 2010	01.6	Added Appendix B. Building SDR/DDR Interfaces Using IPexpress in Diamond.
April 2011	01.7	Updated to include additional clarification in the Generic Timing Analysis, DDR Primitive and Attribute and DDR3 Clock Synchronization modules.
September 2011	01.8	Updated the DDR3 Pinout Guidelines.
		Added the DDR3 Termination Guidelines and Layout Considerations.
		Updated the DQSDLL Update Control section.
December 2011	01.9	Updated Generic DDR Design Guidelines with new sections on Design guidelines, Clocking guidelines, Common software errors and valid window calculation.
		Updated DDR3 Clock Synchronization Module with new table for DDR3 clock and PGROUP locations.
		Updated DDR3 Pinout Guidelines with a new sections on DDR3 pin placements for Improved Noise Immunity and table for DQS Group Allocation.
		Updated ECLKSYNCA & ODDR2D in the DDR Software Primitives and Attributes section.
		Updated GDDR2_TX.DQSDLL.Centered Interface with the requirement that CLKOUT must be assigned to DQS pin and this pin cannot support LVDS IO Standard.
		Updated DQSDLLB with domain transfer consideration on the UDDCNTLN input.
February 2012	02.0	Updated document with new corporate logo.
November 2012	02.2	Updated GDDR1_RX.DQS.Centered Interface ("E" and "EA" Devices) diagram.
		Updated DQSDLL Configuration text section and removed DQSDLL Attributes table.
		Updated IDDRXD1 Waveform diagram.
		Updated ODDRXD Waveform diagram.
April 2013	02.3	Updated GDDR2_RX.DQS.Aligned Interface ("E" and "EA" Devices) figure.
		Updated Interface Rules for GDDR1_RX.DQS.Centered, GDDR1_RX.DQS.Aligned, GDDR2_RX.DQS.Aligned, GDDR2_RX.DQS.Centered, GDDR2_TX.DQSDLL.Centered, and GDDR2_TX.PLL.Centered.
		Update the Pin Placement Guidelines for High-Speed Interfaces table.
		Added information to the High-Speed Clock Bridge ("EA" Devices) section.
July 2013	02.4	Updated Technical Support Assistance information.

## Appendix A. Building DDR Interfaces Using IPexpress in ispLEVER 7.2 SP2

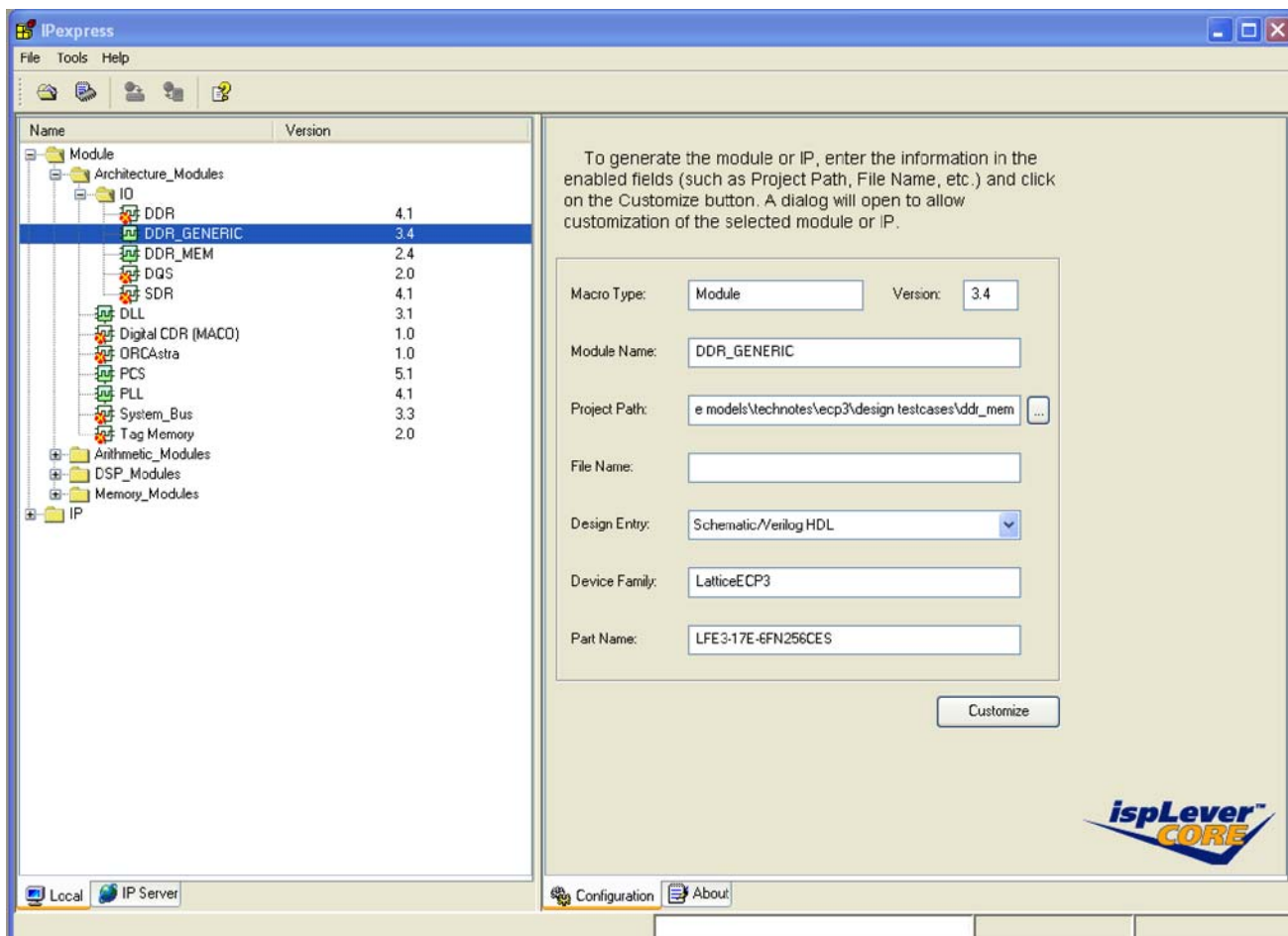
This appendix describes the interface generation for DDR Generic and DDR Memory using IPexpress in ispLEVER 7.2 SP2. **It is highly recommended to update your software to the latest version.** This appendix is only for reference.

IPexpress can be used to configure and generate the DDR Memory Interface and Generic DDR Module. The tool will generate an HDL module that will contain the DDR primitives. This module can be used in the top-level design.

To implement the correct clocking structures to be used for high-speed source synchronous interfaces, see the [High-Speed DDR Interface Details](#) section.

Figure 12-117 shows the main window of IPexpress. The DDR\_Generic and DDR\_MEM options are under Architecture.

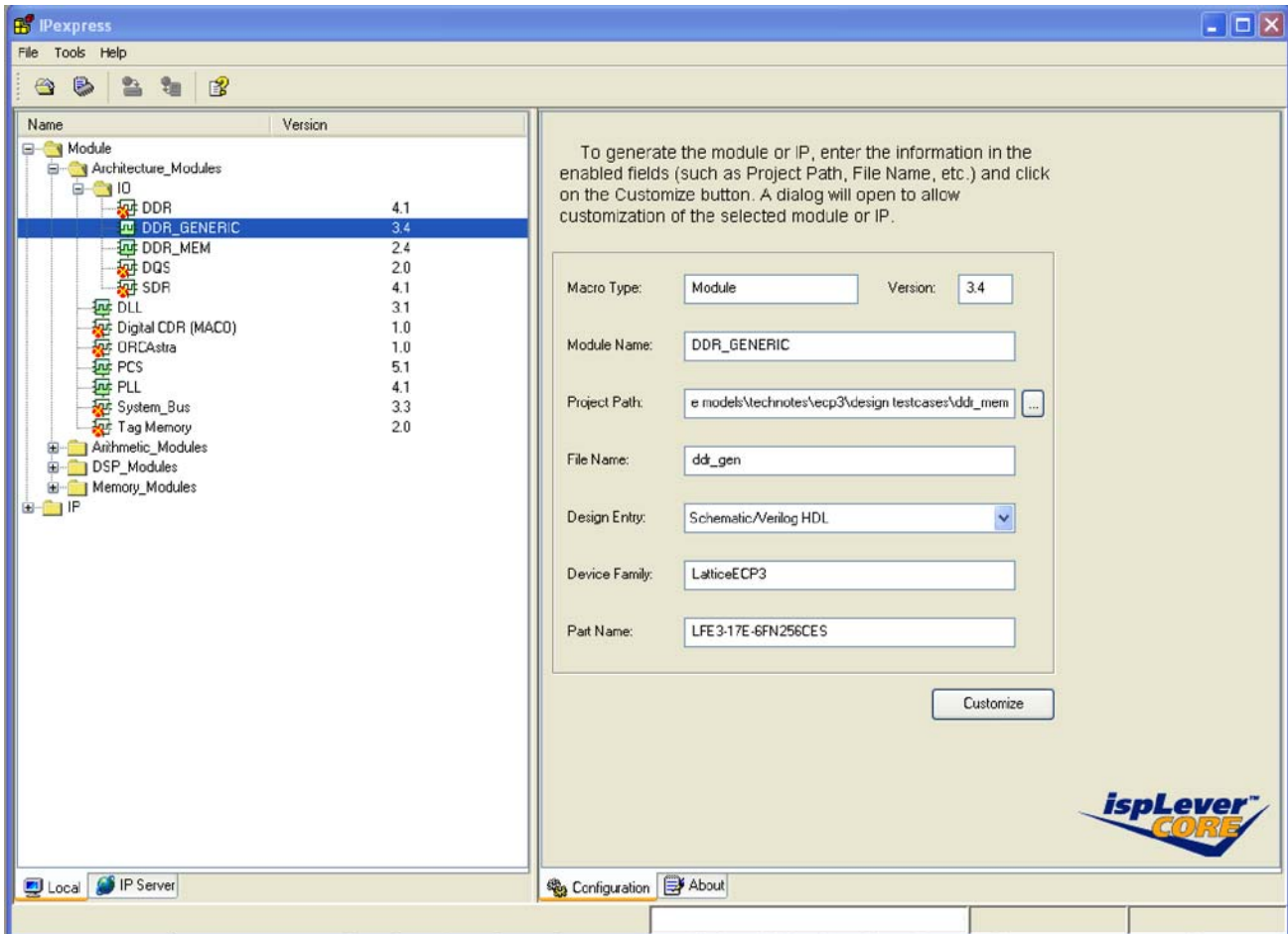
**Figure 12-117. IPexpress Main Window**



## DDR Generic

Figure 12-117 shows the main window when DDR\_GENERIC is selected. The only entry required in this window is the module name. Other entries are set to the project settings. The user may change these entries if desired. After entering the module name, click on **Customize** to open the Configuration Tab window as shown in Figure 12-118.

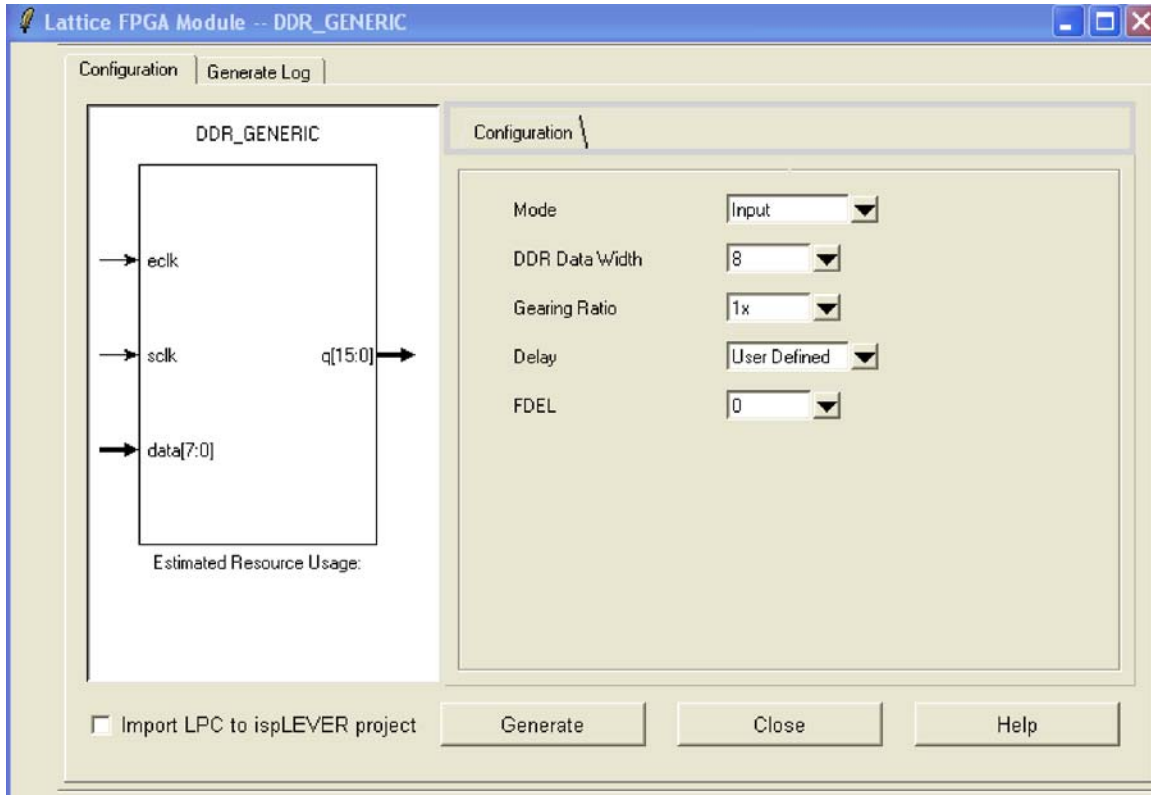
**Figure 12-118. IPexpress Main Window for DDR\_Generic**



## Configuration Tab

The Configuration Tab lists all user-accessible attributes with default values set. Upon completion, click **Generate** to generate source and constraint files. The user may choose to use the .lpc file to load parameters.

**Figure 12-119. Configuration Tab for DDR\_Generic**



The user can change the Mode parameter to choose either Input or Output Tristate DDR module.

The other configuration parameters will change according to the mode selected. The delay and parameter are only available for Input modes.

Table 12-42 describes all user parameters in the IPexpress GUI and their usage.

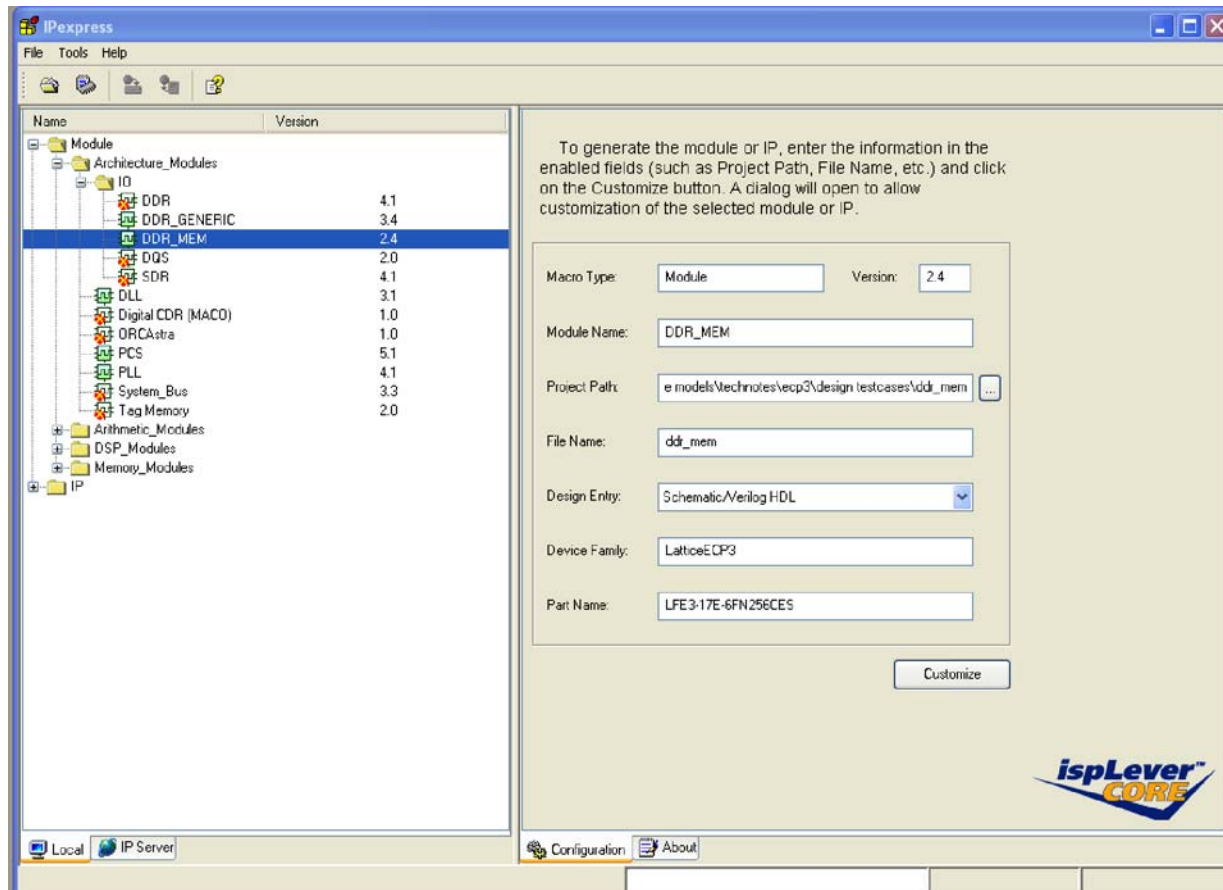
**Table 12-42. User Parameters in the IPexpress GUI**

User Parameter	Description	Values/Range	Default
Mode	Mode selection for the DDR block	Input, Output, Tristate	Input
Data Width	Width of the data bus	1-64	8
Gearing Ratio	Gearing ratio selection	1x, 2x	1x
Delay	Input delay configuration	Dynamic, User Defined, Fixed	User Defined
FDEL	User-defined delay values. Available only when Delay is configured to User Defined.	0-15	0

## DDR\_MEM

Figure 12-120 shows the main window when DDR\_MEM is selected. Similar to the DDR\_Generic, the only entry required here is the module name. Other entries are set to the project settings. The user may change these entries if desired. After entering the module name, click on **Customize** to open the Configuration Tab window as shown in Figure 12-120. Although the user can generate the DDR3\_MEM using IPexpress in ispLEVER 7.2 SP2, it is recommended that the LatticeECP3 DDR3 Memory Controller IP Core be referred to prior to building any DDR3 memory controllers with LatticeECP3 devices.

**Figure 12-120. IPexpress Main Window for DDR\_MEM**

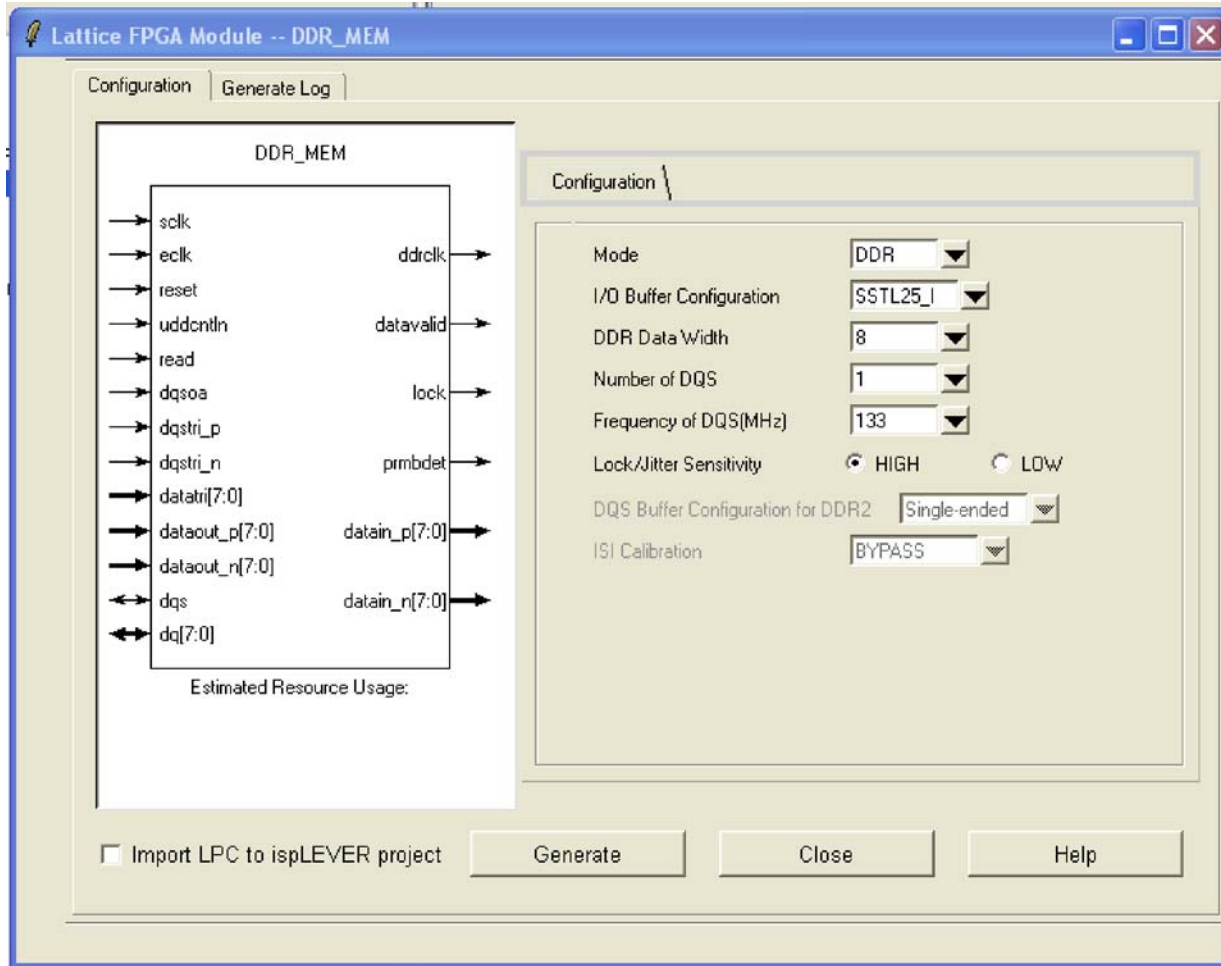




## Configuration Tab

The Configuration Tab lists all user-accessible attributes with default values set. Upon completion, click **Generate** to generate source and constraint files. The user may choose to use the .lpc file to load parameters.

**Figure 12-121. Configuration Tab for DDR\_MEM**



The user can change the Mode parameter to the DDR, DDR2 or DDR3 interface. The other configuration parameters will change according to the mode selected. The Number of DQS parameter determines the number of DDR Interfaces. The software will assume there are eight data bits for every DQS. The user can also choose the frequency of operation; the DDRDLL will be configured to this frequency.

It is recommended that the Lock/Jitter be enabled if the DDR interface is running at 133 MHz or higher. ISI Calibration is only allowed for DDR3 configuration. The parameters available depend on the mode selected. Tables 35, 36 and 37 describe all user parameters in the IPexpress GUI and their usage for modes DDR, DDR2 and DDR3.

**Table 12-43. User Parameters in the IPexpress GUI when in DDR Mode**

User Parameter	Description	Values/Range	Default
I/O Buffer Configuration	I/O standard used for the interface. This also depends on the mode selected.	SSTL25_I, SSTL25_II	SSTL25_I
Data Width	Width of the data bus.	8-64	8
Number of DQS	Number of DQS will determine the number of DQS groups	1, 2, 4, 8	1
Frequency of DQS	DDR Interface Frequency. This is also input to the DDR DLL. The values will depend on the mode selected.	100 MHz, 133MHz, 166MHz, 200MHz	200 MHz
Lock/Jitter Sensitivity	DLL sensitivity to jitter.	High, Low	High

**Table 12-44. User Parameters in the IPexpress GUI when in DDR2 Mode**

User Parameter	Description	Values/Range	Default
I/O Buffer Configuration	I/O standard used for the interface. This also depends on the mode selected.	SSTL18_I, SSTL18_II	SSTL18_I
Data Width	Width of the data bus.	8-64	8
Number of DQS	Number of DQS will determine the number of DQS groups	1, 2, 4, 8	1
Frequency of DQS	DDR Interface Frequency. This is also input to the DDR DLL. The values will depend on the mode selected.	166 MHz, 200 MHz, 266 MHz	200 MHz
Lock/Jitter Sensitivity	DLL sensitivity to jitter.	High, Low	High
DQS Buffer Configuration for DDR2	DQS buffer can be optionally configured as Differential for DDR2.	On/Off	Off

**Table 12-45. User Parameters in the IPexpress GUI when in DDR3 Mode**

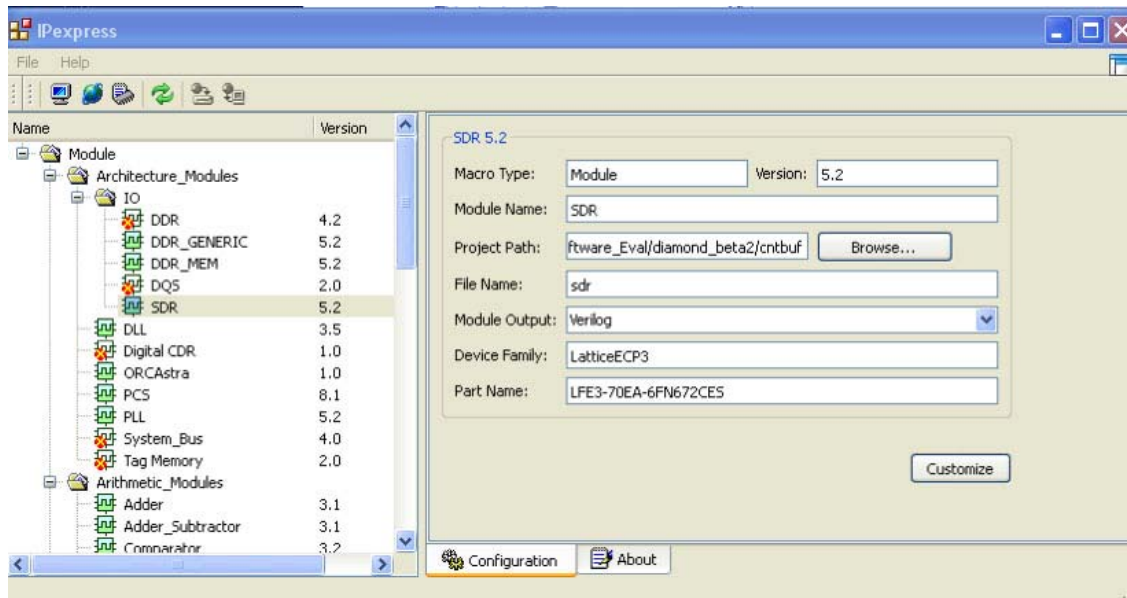
User Parameter	Description	Values/Range	Default
I/O Buffer Configuration	I/O standard used for the interface. This also depends on the mode selected.	SSTL15_I, SSTL15_II	SSTL15_I
Data Width	Width of the data bus.	8-64	8
Number of DQS	Number of DQS will determine the number of DQS groups	1, 2, 4, 8	1
Frequency of DQS	DDR Interface Frequency. This is also input to the DDR DLL. The values will depend on the mode selected.	400 MHz	400 MHz
Lock/Jitter Sensitivity	DLL sensitivity to jitter.	High, Low	High
ISI Calibration	ISI calibration setting for DDR3 output.	BYPASS, DEL1, DEL2, DEL3, DEL4, DEL5, DEL6, DEL7	BYPASS

## Appendix B. Building SDR/DDR Interfaces Using IPexpress in Diamond

This appendix describes the interface generation for DDR Generic and DDR Memory using IPexpress in Lattice Diamond design software. The IPexpress tool is used to configure and generate all the high-speed interfaces described in this document. IPexpress generates a complete HDL module including clocking requirements for each of the interfaces.

For a detailed block diagram of each interface generated by IPexpress, see the section [High-Speed DDR Interface Details](#). IPexpress can be opened from the Tools menu in Project Navigator. All DDR modules are located under **Architecture Modules > IO**. This section will cover SDR and DDR\_GENERIC. DDR\_MEM as discussed in the [Implementing DDR/DDR2/DDR3 Memory Interfaces](#) section.

**Figure 12-122. IPexpress Main Window**

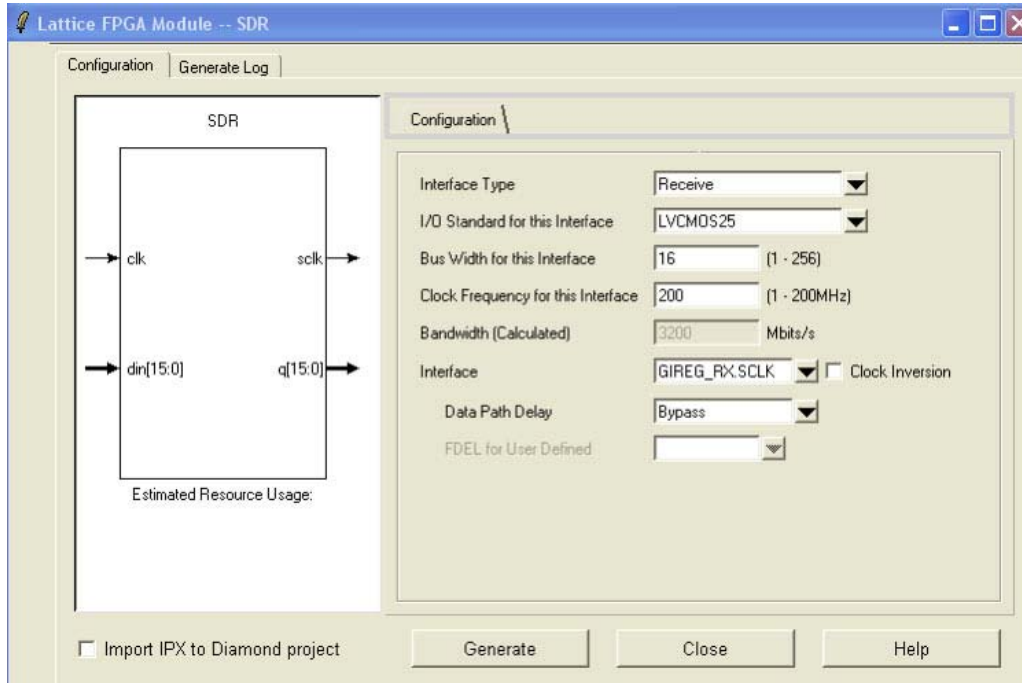


Select the type of interface you would like to build and enter the name of the module. Figure 12-122 shows the type of interface selected as “SDR” and the module name entered. Each module can then be configured by clicking the **Customize** button.

### Building SDR Modules

Choose the interface type SDR, enter the module name and click the **Customize** to open the configuration tab. Figure 12-123 shows the Configuration Tab for the SDR module in IPexpress. Table 12-46 lists the various configurations options available for SDR modules.

**Figure 12-123. SDR Configuration Tab**



**Table 12-46. SDR Configuration Parameters**

GUI Option	Description	Values	Default
Interface Type	Type of interface (transmit or receive)	Transmit, Receive	Receive
I/O Standard for this Interface	I/O standard to be used for the interface.	Transmit and Receive: LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, LVCMOS33, LVCMOS33D, LVDS25, BLVDS25, MLVDS, LVPECL33, HSTL18_I, HSTL18_II, HSTL18D_I, HSTL18D_II, HSTL15_I, HSTL15D_I, SSTL33_I, SSTL33_II, SSTL33D_I, SSTL33D_II, SSTL25_I, SSTL25_II, SSTL25D_I, SSTL25D_II, SSTL18_I, SSTL18_II, SSTL18D_I, SSTL18D_II, SSTL15, SSTL15D, PCI33, LVTTTL33  Transmit only: RSDS, MINILVDS, PPLVDS, LVDS25E, RSDSE	LVCMOS25
Bus Width for this Interface	Bus size for the interface.	1 - 256	16
Clock Frequency for this Interface	Speed at which the interface will run.	1 - 200	200
Bandwidth (Calculated)	Calculated from the clock frequency entered.	(Calculated)	(Calculated)
Interface	Interface selected based on previous entries.	Transmit: GOREG_TX.SCLK Receive: GIREG_RX.SCLK (default)	GIREG_RX.S CLK
Clock Inversion	Option to invert the clock input to the I/O register.	DISABLED, ENABLED	DISABLED

Table 12-46. SDR Configuration Parameters (Continued)

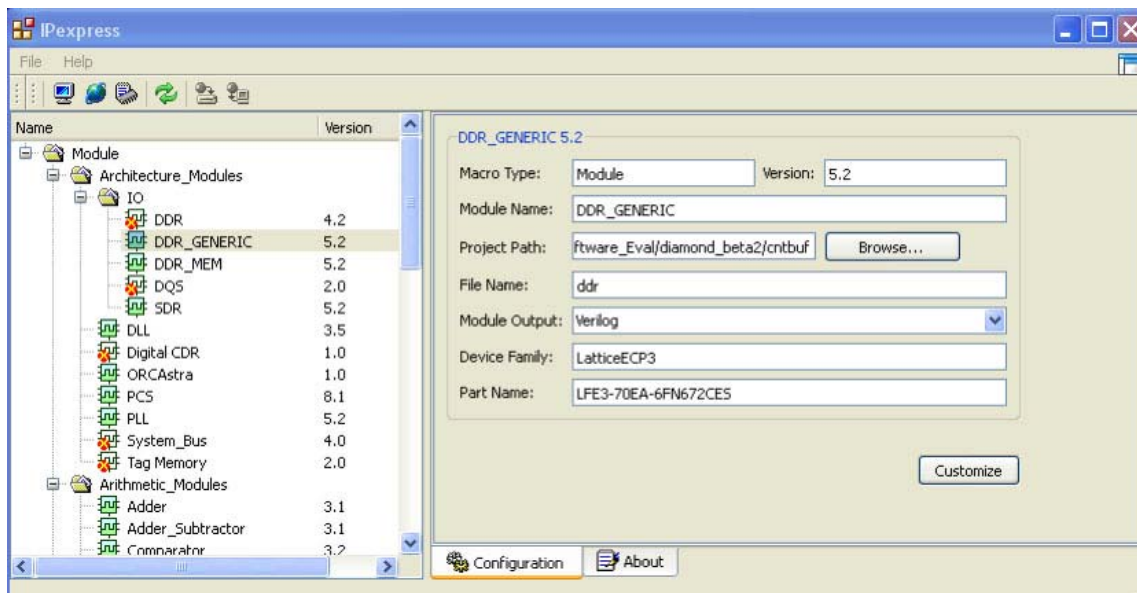
GUI Option	Description	Values	Default
Data Path Delay	Data input can be optionally delayed using the DELAY block.	Bypass, Dynamic <sup>1</sup> , User Defined	Bypass
FDEL for User Defined	If Delay type selected above is user defined, delay values can be entered with this parameter.	0 to 15 <sup>2</sup>	0

1. When Delay type Dynamic is selected, the 16-step delay values must be controlled from the user's design.
2. A FDEL is a fine-delay value that is additive. The delay value for a FDEL can be found in the [LatticeECP3 Family Data Sheet](#).

## Building DDR Generic Modules

Choose interface type **DDR\_GENERIC**, enter the module name and click **Customize** to open the configuration tab.

Figure 12-124. "DDR\_Generic" Selected in Main IPexpress Window



When you click **Customize**, DDR modules have a Pre-Configuration tab and a Configuration tab. The Pre-Configuration tab allows users to enter information about the type of interface to be built. Based on the entries in the Pre-configuration tab, the Configuration tab will be populated with the best interface selection. The user can also, if necessary, override the selection made for the interface in the Configuration tab and customize the interface based on design requirements.

Figure 12-22 shows the Pre-Configuration tab for DDR generic interfaces. Table 12-47 lists the various parameters in the tab.

Figure 12-125. DDR Generic Pre-Configuration Tab

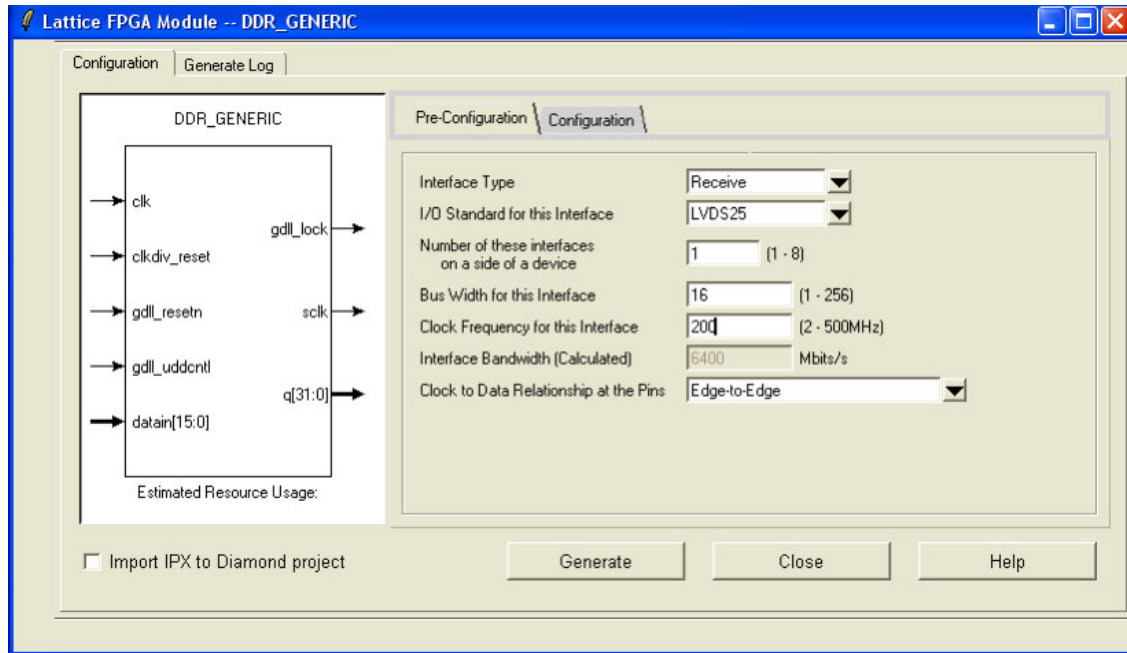


Table 12-47. Pre-Configuration Tab Settings

GUI Option	Description	Values
Interface Type (Transmit or Receive)	Type of interface (Receive or Transmit)	Transmit, Receive
I/O Standard for this Interface	I/O Standard used for the interface	Transmit and Receive: LVCMOS25, LVCMOS18, LVCMOS15, LVCMOS12, LVCMOS33, LVCMOS33D, LVDS25, BLVDS25, MLVDS, LVPECL33, HSTL18_I, HSTL18_II, HSTL18D_I, HSTL18D_II, HSTL15_I, HSTL15D_I, SSTL33_I, SSTL33_II, SSTL33D_I, SSTL33D_II, SSTL25_I, SSTL25_II, SSTL25D_I, SSTL25D_II, SSTL18_I, SSTL18_II, SSTL18D_I, SSTL18D_II, SSTL15, SSTL15D, PCI33, LVTTTL33  Transmit only: RSDS, MINILVDS, PPLVDS, LVDS25E, RSDSE
Number of interfaces on a side of a device	Number of interfaces to be implemented per side. This is used primarily for narrow bus width interfaces (<10). Otherwise it is recommended to leave this at 1.	1 to 8
Bus Width for this Interface	Bus width for each interface. If the number of interfaces per side is >1 then the bus width per interface is limited to 10. If number of interfaces per side is >1 and if using differential I/O standards then bus width is limited to 5.	1-256
Clock Frequency for this Interface	Interface speed	2 - 500 MHz
Interface Bandwidth (Calculated)	Bandwidth is calculated from the clock frequency.	Calculated

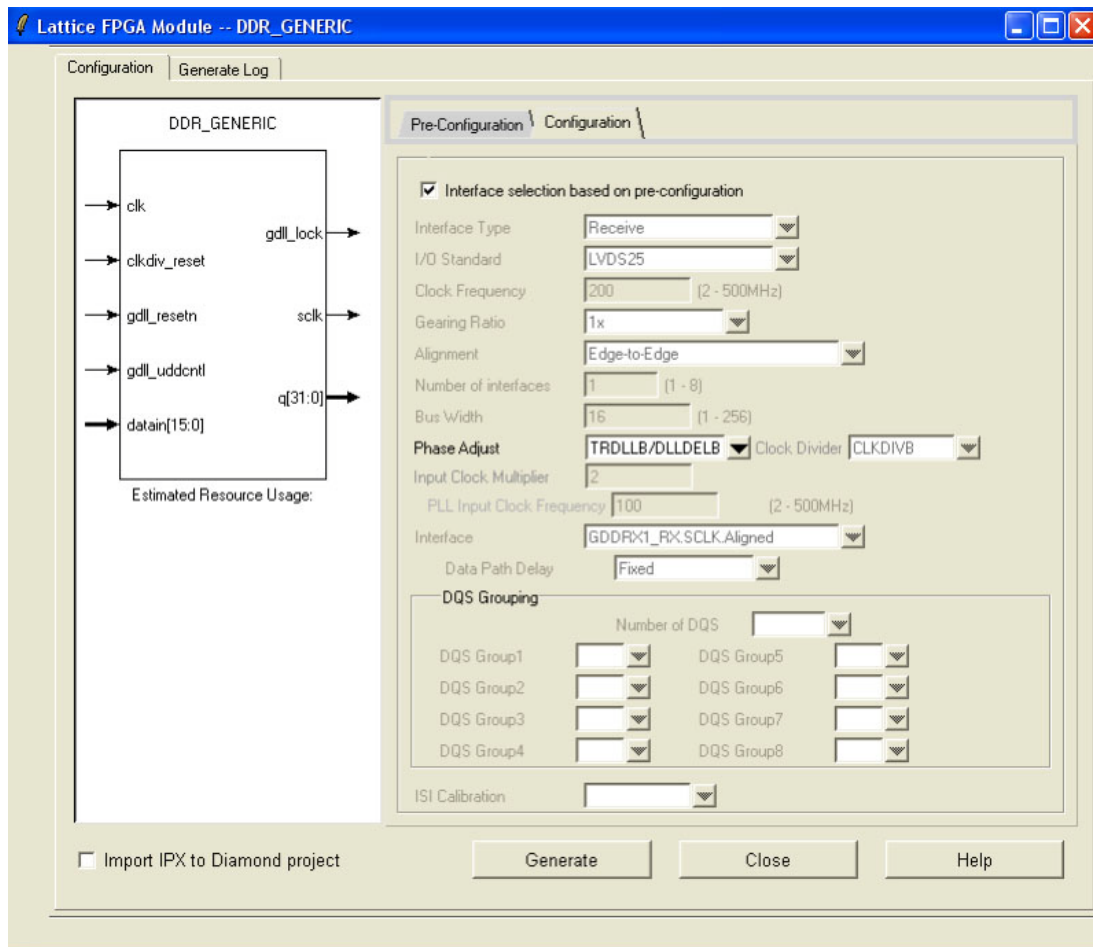
**Table 12-47. Pre-Configuration Tab Settings (Continued)**

GUI Option	Description	Values
Clock to Data Relationship at the Pins	Relationship between clock and data.	Edge-to-Edge, Centered, Dynamic Data Phase Alignment Required <sup>1</sup> , Dynamic Clock Phase Alignment Required

1. Dynamic Phase Alignment is only available for x2 interfaces (i.e, when the clock frequency is higher than 200 MHz).

Based on the selections made in the Pre-Configuration Tab, the Configuration Tab is populated with the selections. Figure 12-126 shows the Configuration Tab for the selections made in the Pre-Configuration Tab.

**Figure 12-126. DDR Generic Configuration Tab**



The checkbox at the top of this tab indicates that the interface is selected based on entries in the Pre-Configuration tab. The user can choose to change these values by disabling this entry. Note that IPexpress chooses the most suitable interface based on selections made in the Pre-Configuration tab.

Table 12-48 lists the various parameters in the Configuration tab.



**Table 12-48. Configuration Tab Settings**

GUI Option	Description	Values	Default Value
Interface Selection Based on Pre-configuration	Indicates interface is selected based on selection made in the Pre-configuration tab. Disabling this checkbox allows users to make changes if needed.	ENABLED, DISABLED	ENABLED
Interface Type	Type of interface (receive or transmit)	Transmit, Receive	Receive
I/O Standard	I/O standard used for the interface	All the ones listed in the Pre-configuration tab	LVC MOS25
Clock Frequency	Speed of the interface	2 to 500 MHz	200 MHz
Gearing Ratio	DDR register gearing ratio (1x or 2x)	1x, 2x	1x
Alignment	Clock to data alignment	Edge-to-Edge, Centered, Dynamic Data Phase Alignment Required, Dynamic Clock Phase Alignment Required	Edge-to-Edge
Number of Interfaces	Number of interfaces to be implemented per side. This is primarily used for narrow bus width interfaces (<10), otherwise it is recommended to leave this at 1.	1 to 8	1
Bus Width	Bus width for each interface. If the number of interfaces per side is >1 then the bus width per interface is limited to 10. If the number of interfaces per side is >1 and if using differential I/O standards then the bus width is limited to 5.	1 to 256	10
Phase Adjust	Module used for phase shifting input clock.	TRDLLB/DLLDELB, PLL <sup>1</sup>	TRDLLB/DLLDELB
Clock Divider	Module used for generation of SCLK from ECLK.	CLKDIVB, TRDLLB <sup>2</sup>	CLKDIVB
Interface	Shows list of all valid high-speed interfaces for a given configuration.	See Table 12-5 for interfaces available for a given configuration.	GDDR1_RX.SCLK. Aligned (EA devices); GDDR1_RX.ECLK. Aligned (E devices)
Data Path Delay	Data input can be optionally delayed using the DELAY block. Value is selected based on Interface Type.	Bypass, Fixed, Dynamic <sup>3</sup>	Fixed
Number of DQS Groups	Enabled when a DQS interface is selected in the Interface selection.	1 to 8	
Number of DQ: DQS Group1 to DQS Group8	This option can be used to change the number of DQ assigned to each DQS lane. Each DQS lane can support up to 10 DQ.	1 to 10	

1. Only available when using GDDR2\_RX.ECLK. Aligned interface.

2. Only available when using GDDR2\_RX.SCLK Aligned interface.

3. When Dynamic Delay is selected, the 16-step delay values must be controlled from the user's design.



Table 12-49 shows how the interfaces are selected by IPexpress based on the selections made in the Pre-Configuration tab.

**Table 12-49. IPexpress Interface Selection**

Device Selected	Interface Type	Gearing Ratio <sup>1</sup>	Alignment	Number of Interfaces	Interface
EA	Receive	1x	Edge-to-Edge	1	GDDR1_RX.SCLK.Aligned
EA	Receive	1x	Centered	1	GDDR1_RX.SCLK.Centered
E	Receive	1x	Edge-to-Edge	1	GDDR1_RX.ECLK.Aligned
E	Receive	1x	Centered	1	GDDR1_RX.ECLK.Centered
E, EA	Receive	1x	Edge-to-Edge	>1	GDDR1_RX.DQS.Aligned
E, EA	Receive	1x	Centered	>1	GDDR1_RX.DQS.Centered
E, EA	Receive	2x	Edge-to-Edge	1	GDDR2_RX.ECLK.Aligned
E, EA	Receive	2x	Centered	1	GDDR2_RX.ECLK.Centered
E, EA	Receive	2x	Edge-to-Edge	>1	GDDR2_RX.DQS.Aligned
E, EA	Receive	2x	Centered	>1	GDDR2_RX.DQS.Centered
EA	Receive	2x	Dynamic	1	GDDR2_RX.ECLK.Dynamic (Default)
EA					GDDR2_RX.DQS.Dynamic <sup>2</sup>
EA					GDDR2_RX.PLL.Dynamic <sup>2</sup>
E, EA	Transmit	1x	Centered	1	GDDR1_TX.SCLK.Centered
E, EA	Transmit	1x	Edge-to-Edge	1	GDDR1_TX.SCLK.Aligned
E, EA	Transmit	1x	Centered	>1	GDDR1_TX.DQS.Centered
EA	Transmit	2x	Edge-to-Edge	1	GDDR2_TX.Aligned
EA	Transmit	2x	Centered	>1	GDDR2_TX.DQSDLL.Centered
EA	Transmit	2x	Centered	<1	GDDR2_TX.PLL.Centered

1. Gearing Ratio of 1x is selected for clock frequencies less than 200MHz. Gearing ratio of 2x is selected for frequencies above 200 MHz.

2. These interfaces can only be selected in the Configuration Tab.

The implementation for several of the interfaces described above differs between the “E” and “EA” devices. Refer to the [High-Speed DDR Interface Details](#) section to see implementation details for “E” and “EA” devices.

The Data Delay setting for each interface is predetermined and cannot be changed by the user. User can only control Data Delay values when using a dynamic interface.

*Note: Some modules generated by IPexpress have a SCLK and ECLK output port. If present, this port must be used to drive logic outside the interface driven by the same signal. In these modules, the input buffer for the clock is inside the IPexpress module and therefore cannot be used to drive other logic in the top level.*

## Building DDR Memory Interfaces

The IPexpress tool is used to configure and generate the DDR, DDR2 and DDR3 memory interfaces.

To see the detailed block diagram for each interface generated by IPexpress see the [Memory Read Implementation](#) and [Memory Write Implementation](#) sections. IPexpress can be opened from the **Tools** menu in Project Navigator. All the DDR modules are located under **Architecture Modules > IO**. DDR\_MEM is used to generate DDR memory interfaces.

Figure 12-127. “DDR\_MEM” Selected in Main IPexpress Window

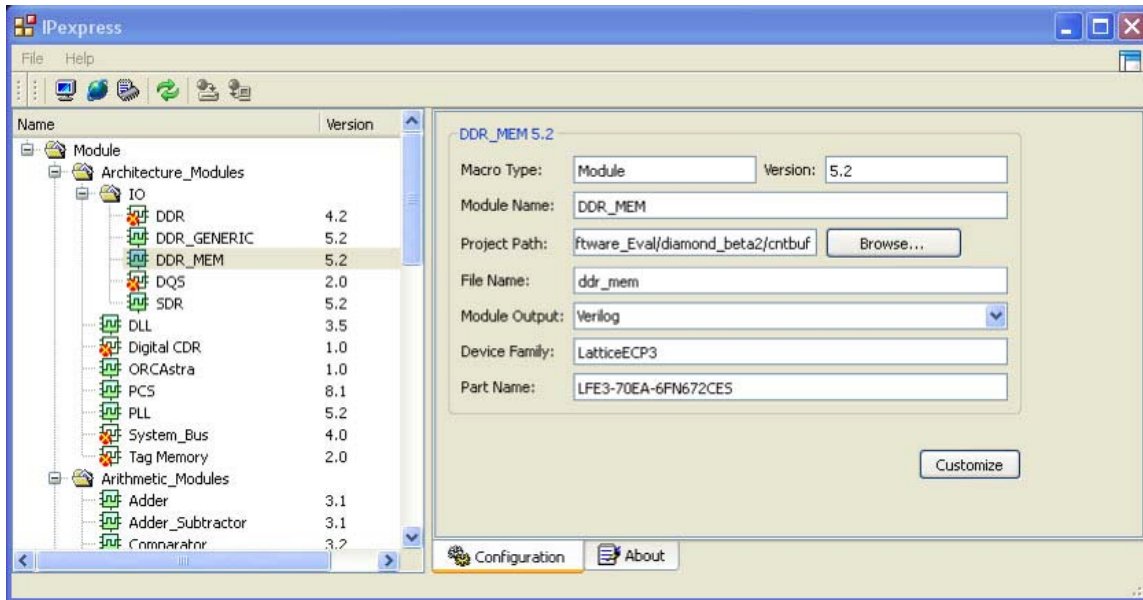


Figure 12-127 shows the IPexpress Main Window. To generate a DDR memory interface, select **DDR\_MEM**, assign a module name and click on **Customize** to see the Configuration tab. Figure 12-128 shows the Configuration tab for the DDR\_MEM interface. You can choose to implement the DDR1\_MEM, DDR2\_MEM or DDR3\_MEM interface.

Figure 12-128. Configuration Tab for DDR\_MEM

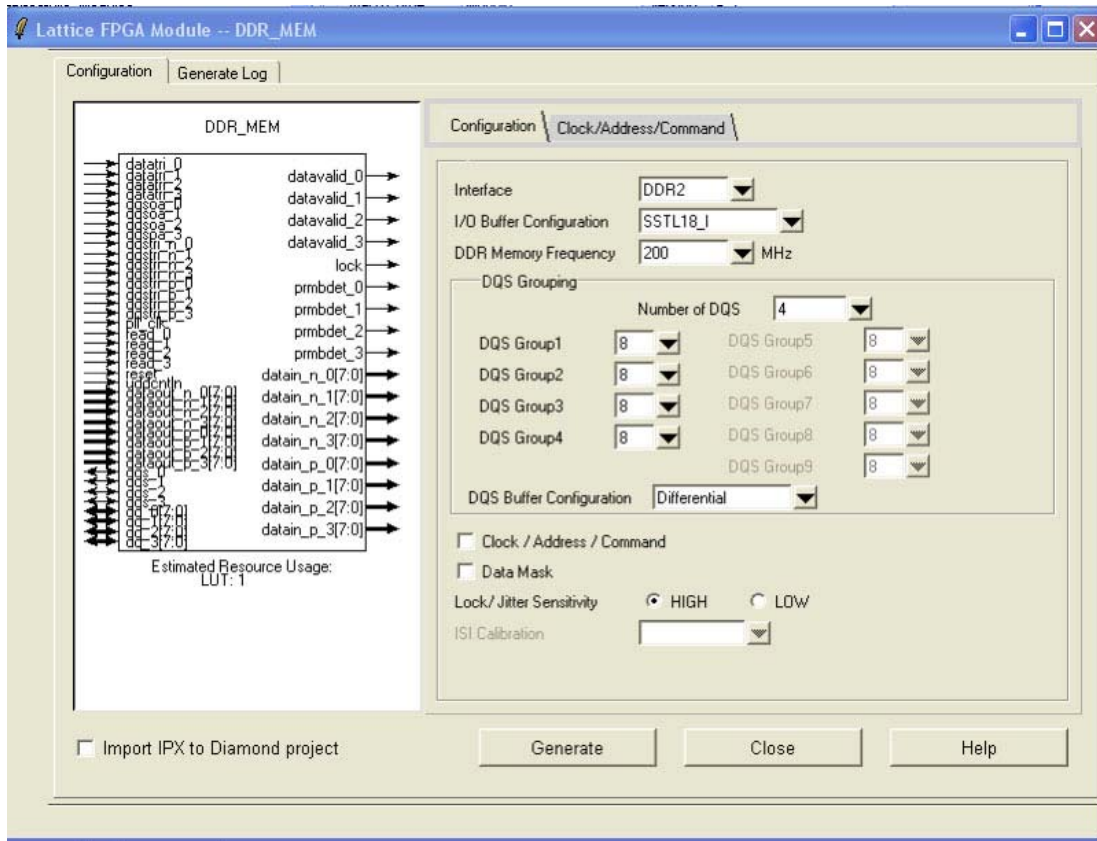


Table 12-50 describes the various settings shown in the Configuration tab above.

Table 12-50. Configuration Tab Settings for DDR\_MEM

GUI Option	Description	Range	Default Value
Interface	DDR memory interface type	DDR, DDR2, DDR3	DDR2
I/O Buffer Configuration	I/O type configuration for DDR pins	SSTL25_I, SSTL25_II SSTL18_I, SSTL18_II, SSTL15	DDR – SSTL25_I DDR2 – SSTL18_I DDR3 – SSTL15
Number of DQS	Interface width (1 DQS per 8 bits of data)	1 to 9	4
DQS Group1 to DQS Group8	Number of DQ per DQS pin	1 to 8	8
DQS Buffer Configuration for DDR2	DQS buffer type	Single-ended, Differential	DDR – Single-ended DDR2 – Single-ended DDR3 – Differential
Clock/Address/Command	Clock/address/command interface will be generated when this option is checked	ENABLED, DISABLED	DISABLED
Data Mask	Data mask signal will be generated when this option is checked	ENABLED, DISABLED	DISABLED
Lock/Jitter Sensitivity	Lock Sensitivity attribute for DQSDLL <sup>1</sup>	HIGH, LOW	HIGH

GUI Option	Description	Range	Default Value
DDR Memory Frequency	DDR Memory Interface Frequency	DDR – 87.5 MHz, 100 MHz, 133.33 MHz, 166.67 MHz, 200 MHz DDR2 – 125 MHz, 200 MHz, 266.67 MHz DDR3 – 150 MHz, 200 MHz, 300 MHz, 400 MHz	DDR – 200 MHz DDR2 – 200 MHz DDR3 – 400 MHz
ISI Calibration	ISI calibration is available for the DDR3 interface to adjust for inter-symbol interference adjustment per DQS group	BYPASS, DEL1, DEL2, DEL3, DEL4, DEL5, DEL6, DEL7	BYPASS

1. It is recommended to set Lock Sensitivity to HIGH for DDR Memory Frequency higher than 133 MHz.

If the user chooses to generate the Clock/Address/Command signals using IPexpress, then the settings in the Clock/Address/Command Tab are active and can be set up as required. Figure 12-129 shows the Clock/Address/Command Tab in the IPexpress for DDR2 Memory.

**Figure 12-129. Clock/Address/Command Tab in the IPexpress for DDR\_MEM**

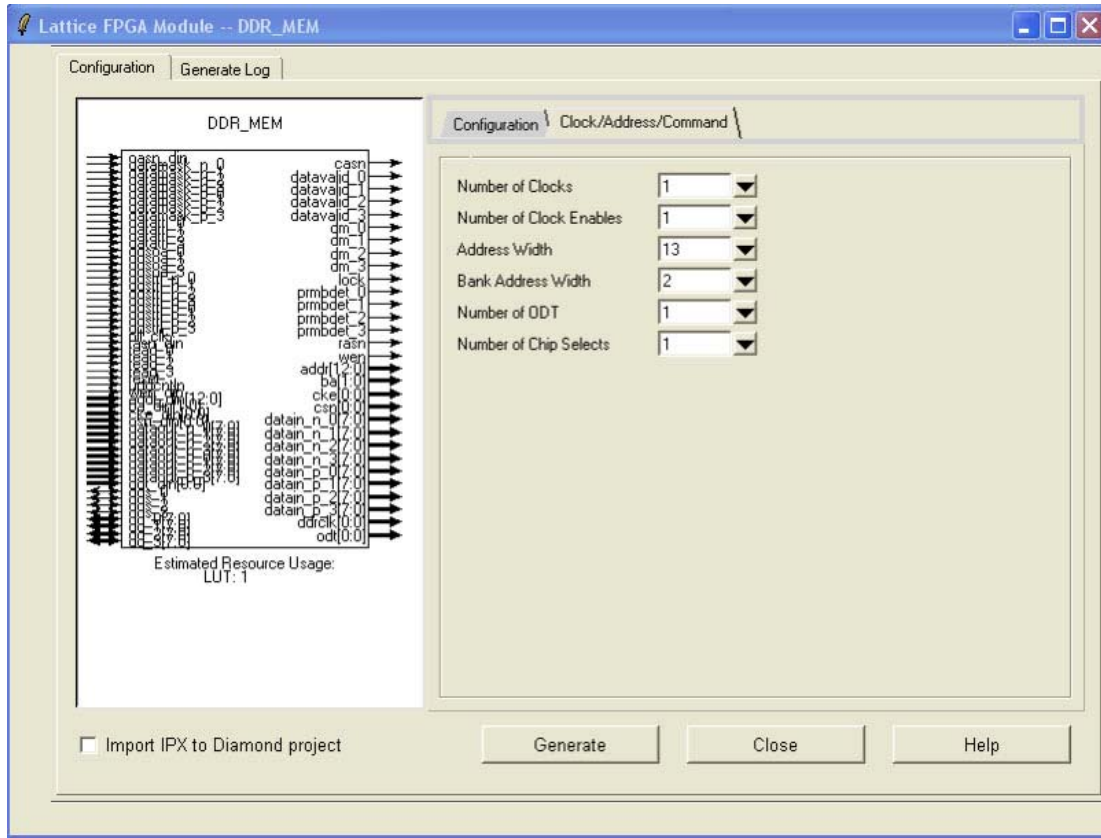


Table 12-51 lists the values that can be used for the Clock/Address/Command settings.

**Table 12-51. Clock/Address/Command Settings for DDR\_MEM**

GUI Option	Range	Default Value
Number of Clocks	1, 2, 4	1
Number of Clock Enables	1, 2, 4	1
Address Width	DDR: 12-14 DDR2: 13-16 DDR3: 13-16	DDR: 13 DDR2: 13 DDR3: 14
Bank Address Width	DDR: 2 DDR2: 2, 3 DDR3: 3	DDR: 2 DDR2: 2 DDR3: 3
Number of ODT	DDR: N/A DDR2: 1, 2, 4 DDR3: 1, 2, 4	DDR: N/A DDR2: 1 DDR3: 1
Number of Chip Selects	DDR: 1, 2, 4, 8 DDR2: 1, 2, 4 DDR3: 1, 2, 4	DDR: 1 DDR2: 1 DDR3: 1

## Introduction

A key requirement for designers using FPGA devices is the ability to calculate the power dissipation of a particular device used on a board. LatticeECP3™ devices bring together the lowest-power FPGA with SERDES and the state-of-the-art ispLEVER® Power Calculator tool. This technical note provides information on power supply considerations and the power calculations that the Power Calculator tool provides. Also included are some guidelines to reduce power consumption.

## Power Supply Sequencing and Hot Socketing

LatticeECP3 devices have been designed to ensure predictable behavior during power-up and power-down. During power-up and power-down sequences, the I/Os remain in tri-state until the power supply voltage is high enough (VCCMIN) to ensure reliable operation. In addition, leakage into I/O pins is controlled to within the limits specified in the [LatticeECP3 Family Data Sheet](#), allowing for easy integration with the rest of the system.

These capabilities, along with lowest-power FPGA with SERDES, makes the LatticeECP3 the ideal choice for many low-power, high-speed SERDES, multiple power supply and hot-swap applications.

## Recommended Power-up Sequence

Refer to the DC and Switching Characteristics section of the [LatticeECP3 Family Data Sheet](#) for more information on any power-up sequence for LatticeECP3 family.

## Power Calculator Hardware Assumptions

Power consumption for a device can be coarsely broken down into the static (or DC) element and the dynamic (or AC) element. These elements have the following dependencies with respect to the junction temperature ( $T_J$ ) of the die.

- Static power is a result of the leakage associated with the transistors. There are two types of static leakage.
  - Static leakage which has a strong temperature dependency
  - DC bias which is fairly constant across temperature
- Dynamic power is caused by the toggling of signals in the transistor.
  - Dynamic power is fairly constant across temperature

Each component in an FPGA (e.g., LUT, register, EBR block, I/O etc.) has its own coefficients for static and dynamic positions. Certain selections in the Power Calculator tool affect some of these coefficients which are discussed in the Power Calculator section.

## Power Calculator

Power Calculator is the fastest power simulation tool available in the industry. The tool offers Estimation Mode for “what-if” analysis, and also allows designers to import NCD design files to accurately estimate power for their designs. The background engine performs each calculation quickly and accurately.

When running the Power Calculator tool in Estimation mode, designers provide estimates of the utilization of various components and the tool provides an estimate of the power consumption. This is a good start, especially for what-if analyses and device selection.

Calculation mode is a more accurate approach, where the designer imports the actual device utilization by importing the post place and route netlist design file (or NCD) file.

Users can also import a Trace Report (or TWR) file where the frequencies for various clocks are also imported. Note that the Trace Report only includes frequencies of the clocks nets that are constrained in the Preference file.

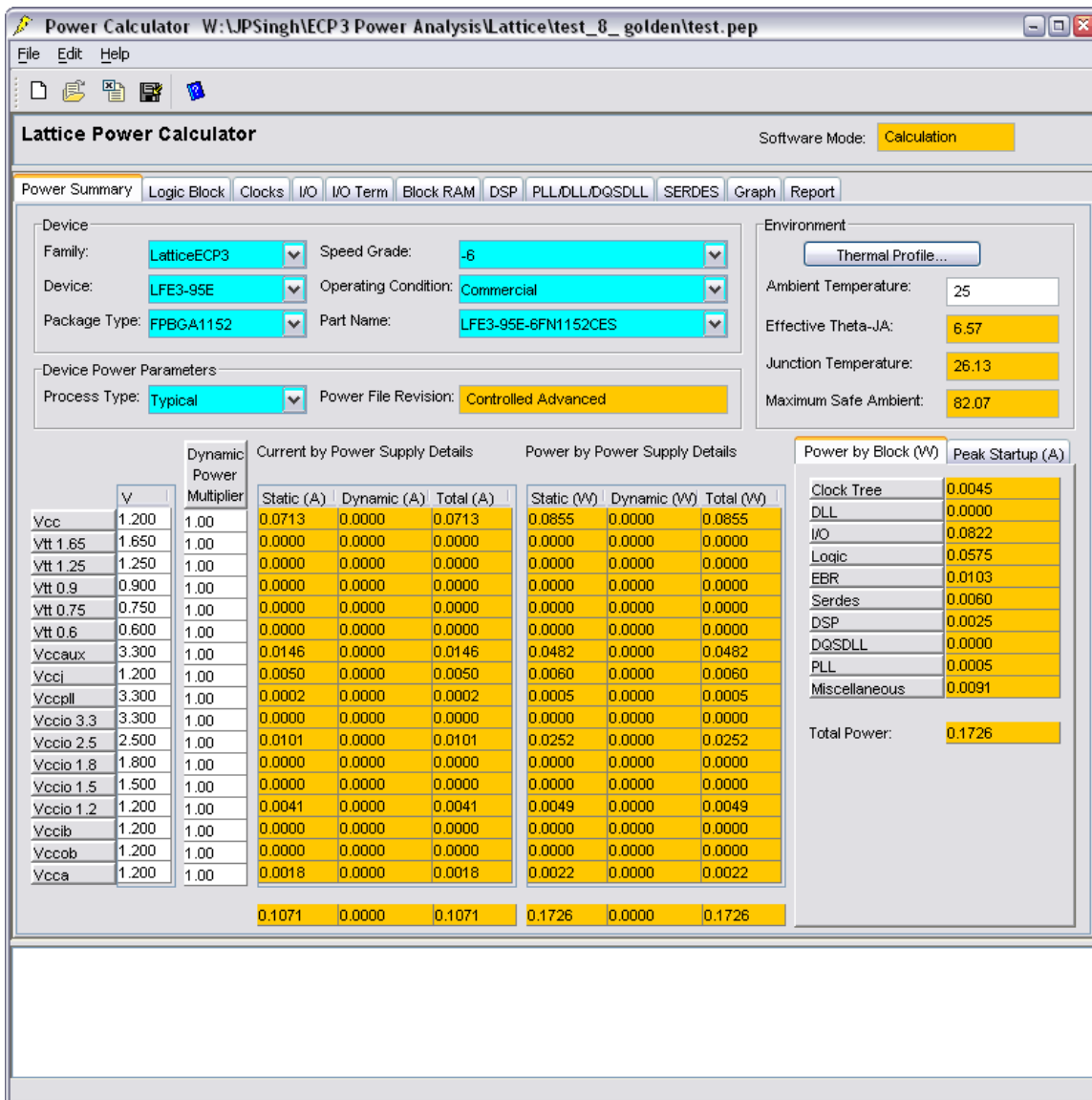
The default Activity Factor (AF%) for dynamic power calculation is set to 10% in the Power Calculator. Users can change the default AF for the entire project or for each clock net individually. Activity Factor is discussed in more detail later in this document.

## Power Calculator and Power Equations

Please refer to the ispLEVER Tutorial for launching and using the Power Calculator tool under **Help > ispLEVER Help**.

Once you step through the procedure, you will see a window that looks like Figure 1.

**Figure 13-1. Power Calculator Main Window**



It is important to understand how the options available with Power Calculator affects the power. For example, if the ambient temperature is changed, it affects the junction temperature, according to the following equation:



$$T_J = T_A + \Theta_{JA\_EFFECTIVE} * P \quad (1)$$

Where  $T_J$  and  $T_A$  are the junction and ambient temperatures, respectively, and  $P$  is the power.

$\Theta_{JA\_EFFECTIVE}$  is the effective thermal impedance between the die and its environment.

The junction temperature is directly proportional to the ambient temperature. An increase in  $T_A$  will increase  $T_J$  and result in an increase of the static leakage component.

Selecting the Process Type again affects the static leakage; in particular the static leakage coefficient changes.

The DC Bias component is constant across the range.

For dynamic power, increasing the frequency of toggling will increase the dynamic component of power.

## Typical and Worst Case Process Power/ICC

Another factor that affects DC power is process variation. This variation, in turn, causes variation in quiescent power.

Power Calculator takes these factors into account and allows designers to specify either a typical process or a worst case process.

## Junction Temperature

Junction temperature is the temperature of the die during operation. It is one of the most important factors that affects the device power. For a fixed junction temperature, voltage and device package combination, quiescent power is fixed.

Ambient temperature affects the junction temperature as shown in Equation 1. Devices operating in a high-temperature environment have higher leakage since their junction temperature will be higher. Power Calculator models this ambient to junction temperature dependency. When the user provides an ambient temperature, it is rolled into an algorithm that calculates the junction temperature and power through an iterative process to find the thermal equilibrium of the system (device running with the design) with respect to its environment ( $T_A$ , airflow etc.).

## Maximum Safe Ambient Temperature

Max. Safe Ambient Temperature is one of the most important numbers displayed in the Summary tab of the Power Calculator. This is the maximum ambient temperature at which the design can run without violating the junction temperature limits for commercial or industrial devices.

Power Calculator uses an algorithm to accurately predict this temperature. The algorithm adjusts itself as the user changes options such as voltage, process, frequency, AF% etc. (or any factor that may affect the power dissipation of the device).

## Operating Temperature Range

When designing a system, engineers must make sure a device operates at specified temperatures within the system environment. This is particularly important to consider before a system is designed. With Power Calculator, users can predict device thermodynamics and estimate the dynamic power budget. The ability to estimate a device's operating temperature prior to board design also allows the designer to better plan for power budgeting and airflow.

Although total power, ambient temperature, thermal resistance and airflow all contribute to device thermodynamics, the junction temperature (as specified in the [LatticeECP3 Family Data Sheet](#)) is the key to device operation. The allowed junction temperature range is 0°C to 85°C for commercial devices and -40°C to 105°C for industrial devices. Anytime the junction temperature of the die falls out of these ranges, the performance and reliability of the device's operation must be evaluated. The reliability limit of junction temperature, on the other hand, for this generation of device technology is 125°C.



---

## Dynamic Power Multiplier (DPM)

It is difficult to estimate the temperature dependence of dynamic power due to various ways in which a design can be placed and routed. The user-defined frequency of operation makes this problem even more complex. To help resolve this issue, the Dynamic Power Multiplier provides some guard bands for system and board designers.

The Dynamic Power Multiplier is defaulted to “1” which means the dynamic power is what it is. If the user wishes to add 20% additional dynamic power, the DPM can be set to 1.2 (1 + 20%) and it can be placed against the appropriate power supply. This increases the dynamic power for that supply by 20% and provides users with some guard band (if needed).

## Power Budgeting

Power Calculator provides the power dissipation of a design under a given set of conditions. It also predicts the junction temperature ( $T_j$ ) for the design. Any time this junction temperature is outside the limits specified in the [LatticeECP3 Family Data Sheet](#), the viability of operating the device at this junction temperature must be re-evaluated.

A commercial device is likely to show speed degradation with a junction temperature above 85°C and an industrial device at a junction temperature will degrade above 100°C. It is required that the die temperature be kept below these limits to achieve the guaranteed speed operation.

Operating a device at a higher temperature also means a higher SICC. The difference between the SICC and the total ICC (both Static ICC and Dynamic ICC) at a given temperature provides the dynamic budget available. If the device runs at a dynamic ICC higher than this budget, the total ICC is also higher. This causes the die temperature to rise above the specified operating conditions.

There are a number of ways to handle this situation. Some of these are discussed in the Power Management section of this document.

The four factors of power, ambient temperature, thermal resistance and airflow, can also be varied and controlled to reduce the junction temperature of the device. Power Calculator is a powerful tool to help system designers to properly budget the FPGA power that, in turn, helps improve overall system reliability.

## Activity Factor Calculation

The Activity Factor % (or AF%) is defined as the percentage of frequency (or time) that a signal is active or toggling the output. Most resources associated with a clock domain are running or toggling at some percentage of the frequency at which the clock is running. Users must provide this value as a percentage under the AF% column in the Power Calculator tool.

Another term for I/Os is the I/O Toggle Rate. The AF% is applicable to the PFU, Routing, and Memory Read Write Ports, etc. The activity of I/Os is determined by the signals provided by the user (in the case of inputs) or as an output of the design (in the case of outputs). The rates at which the I/Os toggle define their activity. The I/O Toggle Rate or the I/O Toggle Frequency is a better measure of their activity.

The Toggle Rate (or TR) in MHz of the output is defined in the following equation:

$$\text{Toggle Rate (MHz)} = 1/2 * f * \text{AF\%} \quad (5)$$

Users are required to provide the TR (MHz) value for the I/O instead of providing the frequency and AF% for other resources. AF can be calculated for each routing resource, output or PFU. However, this involves long calculations. The general recommendation for a design occupying roughly 30% to 70% of the device is an AF% between 15% and 25%. This is an average value. The accurate value of an AF depends upon clock frequency, stimulus to the design and the final output.

## Thermal Impedance and Airflow

A common method for characterizing a packaged device's thermal performance is with "Thermal Resistance",  $\Theta$ . For a semiconductor device, thermal resistance indicates the steady state temperature rise of the die junction above a given reference for each watt of power (heat) dissipated at the die surface. Its units are  $^{\circ}\text{C}/\text{W}$ .

The most common examples are  $\Theta_{JA}$ , Thermal Resistance Junction-to-Ambient (in  $^{\circ}\text{C}/\text{W}$ ) and  $\Theta_{JC}$ , Thermal Resistance Junction-to-Case (also in  $^{\circ}\text{C}/\text{W}$ ). Another factor is  $\Theta_{JB}$ , Thermal Resistance Junction-to-Board (in  $^{\circ}\text{C}/\text{W}$ ).

Knowing the reference (i.e. ambient, case, or board) temperature, the power, and the relevant  $\Theta$  value, the junction temperature can be calculated per following equations.

$$T_J = T_A + \Theta_{JA} * P \quad (6)$$

$$T_J = T_C + \Theta_{JC} * P \quad (7)$$

$$T_J = T_B + \Theta_{JB} * P \quad (8)$$

Where  $T_J$ ,  $T_A$ ,  $T_C$  and  $T_B$  are the junction, ambient, case (or package) and board temperatures (in  $^{\circ}\text{C}$ ), respectively.  $P$  is the total power dissipation of the device.

$\Theta_{JA}$  is commonly used with natural and forced convection air-cooled systems.  $\Theta_{JC}$  is useful when the package has a high conductivity case mounted directly to a PCB or heatsink. And  $\Theta_{JB}$  applies when the board temperature adjacent to the package is known.

Power Calculator utilizes the ambient temperature ( $^{\circ}\text{C}$ ) to calculate the junction temperature ( $^{\circ}\text{C}$ ) based on the  $\Theta_{JA}$  for the targeted device. Users can also provide the airflow values (in LFM) to obtain a more accurate junction temperature value.

To improve airflow effectiveness, it is important to maximize the amount of air that flows over the device or the surface area of the heat sink. The airflow around the device can be increased by providing an additional fan or increasing the output of the existing fan. If this is not possible, baffling the airflow to direct it across the device may help. This means the addition of sheet metal or objects to provide the mechanical airflow guides to guide air to the target device. Often the addition of simple baffles can eliminate the need for an extra fan. In addition, the order in which air passes over devices can impact the amount of heat dissipated.

## Reducing Power Consumption

One of the most critical challenges for designers today is reducing the system power consumption. A low-order reduction in power consumption goes a long way, especially in modern hand-held devices and electronics. There are several design techniques that can be used to significantly reduce overall system power consumption. Some of these include:

1. Reducing operating voltage.
2. Operating within the specified package temperature limitations.
3. Using optimum clock frequency reduces power consumption, as the dynamic power is directly proportional to the frequency of operation. Designers must determine if some portions of the design can be clocked at a lower rate that will reduce power.
4. Reducing the span of the design across the device. A more closely-placed design uses fewer routing resources and therefore less power.
5. Reducing the voltage swing of the I/Os where possible.
6. Using optimum encoding where possible. For example, a 16 bit binary counter has, on average, only 12% activity factor and a 7-bit binary counter has an average of 28% activity factor. On the other hand, a 7-bit LFSR counter will toggle at an activity factor of 50%, which causes higher power consumption. A gray code

counter, where only one bit changes at each clock edge will use the least amount of power, as the activity factor is less than 10%.

7. Minimizing the operating temperature by the following methods:
  - Use packages that can better dissipate heat, such as ceramic packages.
  - Placing heat sinks and thermal planes around the device on the PCB.
  - Use better airflow techniques, such as mechanical airflow guides and fans (both system fans and device mounted fans).

## Power Calculator Assumptions

The following are the assumptions made by the Power Calculator.

1. The Power Calculator tool uses equations with constants based on a room temperature of 25°C.
2. Users can define the ambient temperature ( $T_A$ ) for device junction temperature ( $T_J$ ) calculation based on the power estimation.  $T_J$  is calculated from the user-entered  $T_A$  and the power calculation of typical room temperature.
3. I/O power consumption is based on an output loading of 5pF. Users have the ability to change this capacitive loading.
4. Users can estimate power dissipation and current for each type of power supply ( $V_{CC}$ ,  $V_{CCIO}$ ,  $V_{CCJ}$  and  $V_{CCAUX}$ ). For  $V_{CCAUX}$ , only static  $I_{CCAUX}$  values are provided in the Power Calculator.
5. Additional  $V_{CCAUX}$  contributions due to differential output buffers, differential input buffers and reference input buffers must be added per pair for differential buffers or per pin for reference input buffers, according to the user's design. See the equation given in this technical note for Total DC Power ( $I_{CCAUX}$ ).
6. The nominal  $V_{CC}$  is used by default to calculate power consumption. A lower or higher  $V_{CC}$  can be chosen from a list of available values.
7. Users can enter Airflow in Linear Feet per Minute (LFM) along with a Heat Sink option to calculate the junction temperature.
8. The default value of the I/O types for LatticeECP3 devices is LVCMOS25, 12mA.
9. The activity factor (AF) is defined as the toggle rate of the registered output. For example, assuming that the input of a flip-flop is changing at every clock cycle, 100% AF of a flip-flop running at 100 MHz is 50 MHz.

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
February 2011	01.1	Updated document with new corporate logo.
July 2013	01.2	Updated Technical Support Assistance information.

## Introduction

This technical note discusses how to access the features of the LatticeECP3™ sysDSP™ (Digital Signal Processing) slice described in the [LatticeECP3 Family Data Sheet](#). Designs targeting the sysDSP slice can offer significant improvement over traditional LUT-based implementations. Table 14-1 provides an example of the performance and area benefits of this approach.

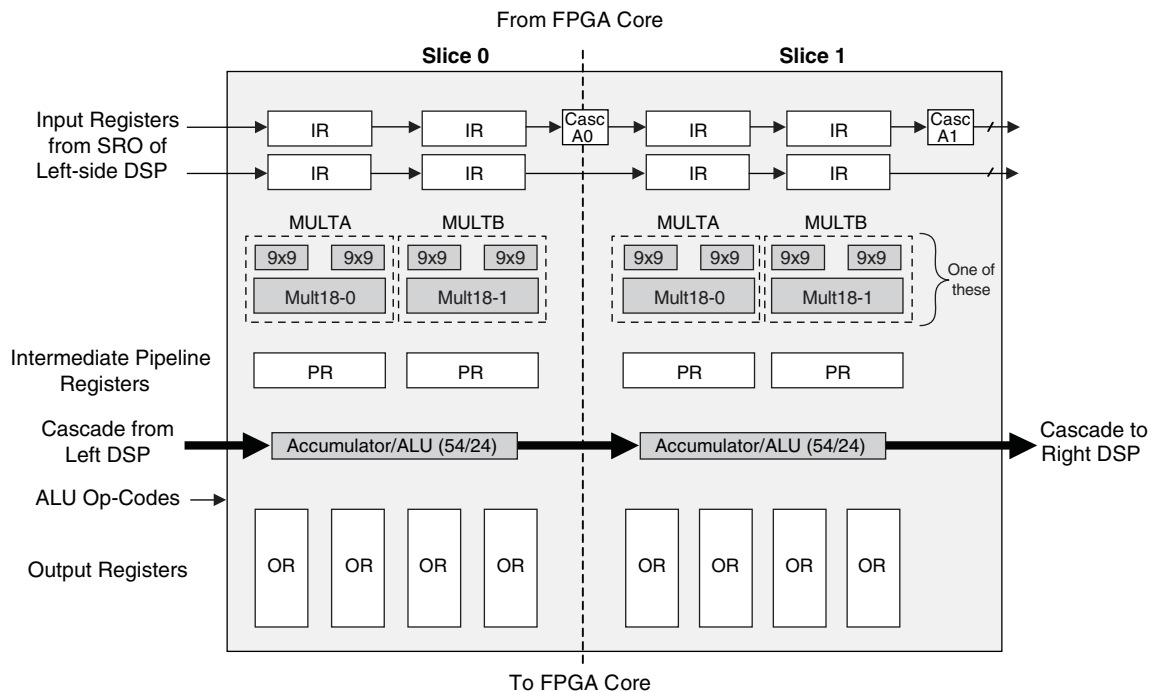
**Table 14-1. sysDSP Slice vs. LUT-based Multipliers**

Multiplier Width	Register Pipelining	LatticeECP3-95-8 Using sysDSP Slice(s)		LatticeECP3-95-8 Using LUTs	
		f <sub>MAX</sub> (MHz)	LUTs	f <sub>MAX</sub> (MHz)	LUTs
9x9	Input, Multiplier, Output	402	0	268	114
18x18	Input, Multiplier, Output	402	0	163	411
36x18	Input, Multiplier, Output	394	0	123	737
36x36	Input, Multiplier, Output	225	0	105	1502

## sysDSP Slice Hardware

The LatticeECP3 sysDSP slices are located in rows throughout the device. Figure 14-1 is a simplified block diagram of two of the sysDSP slices. The programmable resources in a slice include: multipliers, ALU, muxes, pipeline registers, shift register chain and cascade chain. If the shift out register A is selected, the cascade match register (Casc A0) is available. The multipliers can be configured as 18X18 or 9X9 and the ALU can be configured as 54-bit or 24-bit. Advanced features of the sysDSP slice are described later in this document.

**Figure 14-1. LatticeECP3 sysDSP Slice**



© 2013 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at [www.latticesemi.com/legal](http://www.latticesemi.com/legal). All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

The sysDSP slice can be configured in a number of ways. The ALU54/24 can be configured as one of the following: adder, subtracter, or accumulator. Using two sysDSP slices, the most common configurations include:

- **One 36x36 Multiplier**
  - Basic multiplier, no add/sub/accum/sum blocks
- **Four 18x18 Multipliers**
  - Two add/sub/accum blocks
- **Eight 9x9 Multipliers**
  - Two add/sub blocks

## sysDSP Slice Software

### Overview

The sysDSP slice can be targeted in a number of ways.

- Use IPexpress™ to specify and configure the sysDSP module for instantiating in the user HDL design.
- Create HDL code for direct sysDSP slice inference by the synthesis tools.
- Implement the design in The MathWorks® Simulink® tool using a Lattice library of DSP blocks. The ispLeverDSP™ tool in the ispLEVER® and Lattice Diamond™ design software converts these blocks into HDL.
- Instantiate sysDSP primitives directly in the source code.

### Targeting sysDSP Slices Using IPexpress

IPexpress allows you to graphically specify sysDSP elements. Once the element is specified, an HDL file is generated, which can be instantiated in a design. IPexpress allows users to configure all ports and set all available parameters. The following modules target the sysDSP slice. The resource usage estimation will be shown in the GUI. See Appendix B for information about using IPexpress for Diamond.

- ADDER\_TREE
- BARREL\_SHIFTER
- MAC (Multiplier Accumulate)
- MMAC (Multiplier Multiplier Accumulate)
- MULT (Multiplier)
- MULTADDSUB (Multiplier Add/Subtract)
- MULTADDSUBSUM (Multiply Add/Subtract and SUM)
- SLICE (Fully-configurable sysDSP slice used for advanced functions)
- WIDE\_MUX

### MULT Module

The IPexpress MULT Module configures elements to be packed into the sysDSP slice. The function  $A \times B = P$  is implemented. The screen shot shown in Figure 14-2 shows the following:

• **Data Options**

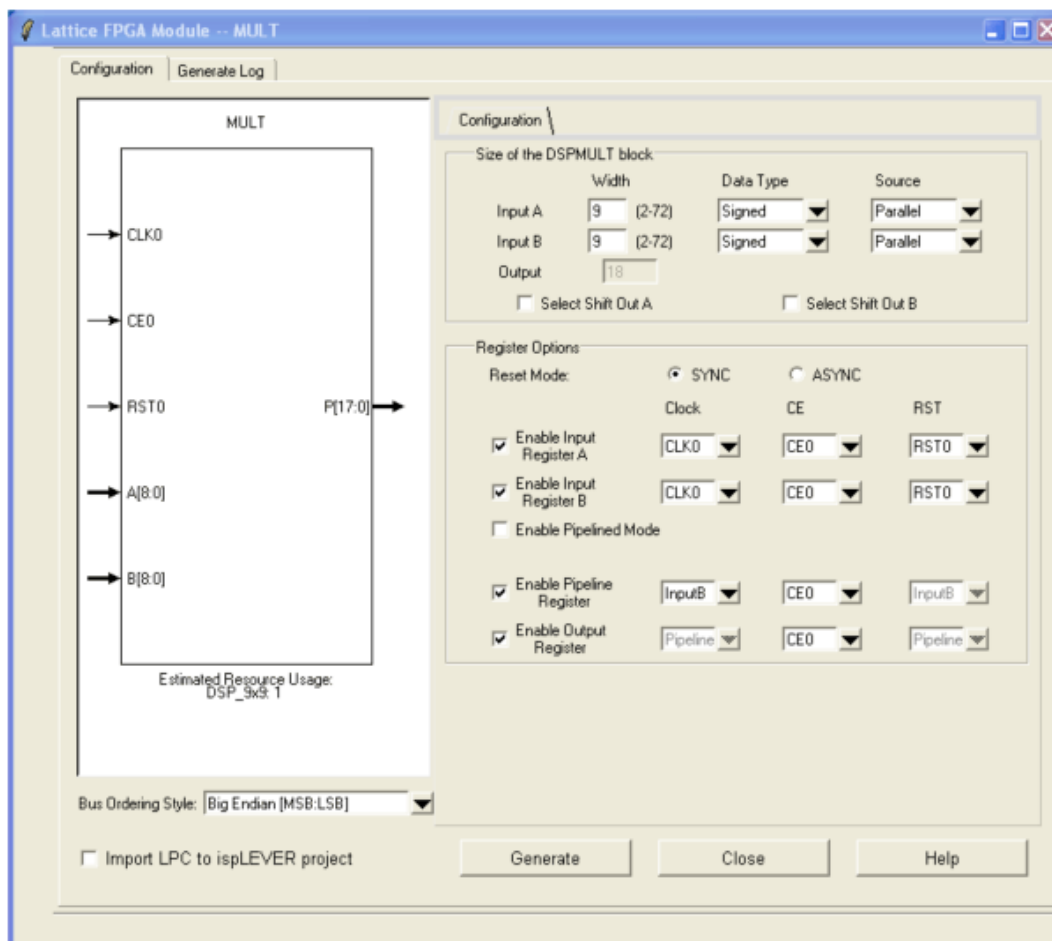
- Input bus widths from 2 to 72 bits
- Data Type specifies whether the data is Signed, Unsigned or Dynamic
- Source specifies whether data is from Parallel, Shift or Dynamic
- Shift Out enables the shift out port on the sysDSP slice

• **Register Options**

- Reset Mode selects whether an Async or Sync Reset is used.
- Enable Input Register selects whether data input registers are used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipeline Register selects whether data pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected. If the option InputA or InputB is selected, the pipeline register uses the same clock as the input register. Output registers will automatically be enabled and will use the same clock.
- Enable Output Register selects whether the data output pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipelined Mode turns on all the input, pipeline and output registers. If this mode is enabled, latency will be provided.

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register, which is useful in applications such as the FIR Filter.

**Figure 14-2. MULT Mode**



### MAC Module

The IPexpress MAC Module configures elements to be packed into the sysDSP slice. The function  $A_n \times B_n \pm P_{n-1} = P_n$  is implemented. The screen shot shown in Figure 14-3 shows the following:

- **Data Options**

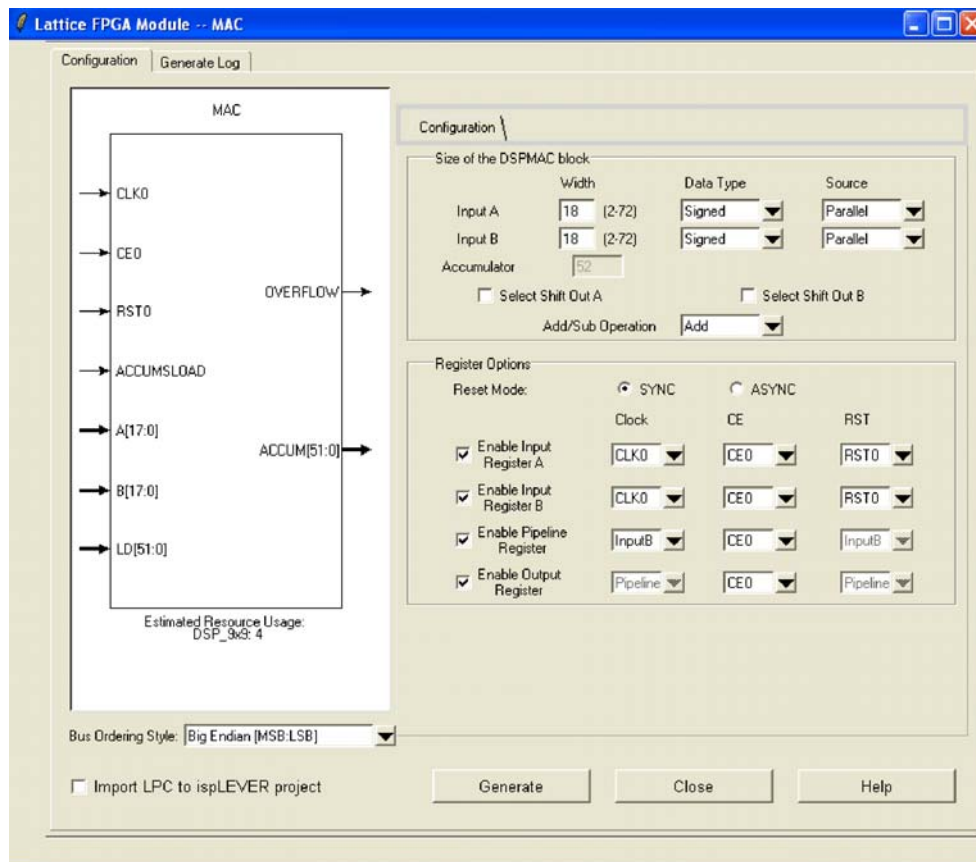
- Input bus widths from 2 to 72 bits
- Data Type specifies whether the data is Signed, Unsigned or Dynamic
- Source specifies whether data is from Parallel, Shift or Dynamic
- Shift Out enables the shift out port on the sysDSP slice
- Add/Sub Operation selects whether the arithmetic operation is Addition, Subtraction or Dynamic

- **Register Options**

- Reset Mode selects whether an Async or Sync Reset is used
- Enable Input Register selects whether data input registers are used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipeline Register selects whether data pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected. If the option InputA or InputB is selected, the pipeline register uses the same clock as the input register. Output registers will automatically be enabled and will use the same clock.
- Enable Output Register is selected by default and selects the data output pipeline registers to be used. Clock, reset and clock enable resources can be selected.

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register. The accumulator is loaded with an initial value from data on the LD port when the signal ACCUMSLOAD is toggled.

**Figure 14-3. MAC Mode**





### MMAC Module

The IPexpress MMAC Module configures elements to be packed into the sysDSP slice. The function  $A0_n \times B0_n \pm A1_n \times B1_n + P_{n-1} = P_n$  is implemented. The screen shot shown in Figure 14-4 shows the following:

- **Data Options**

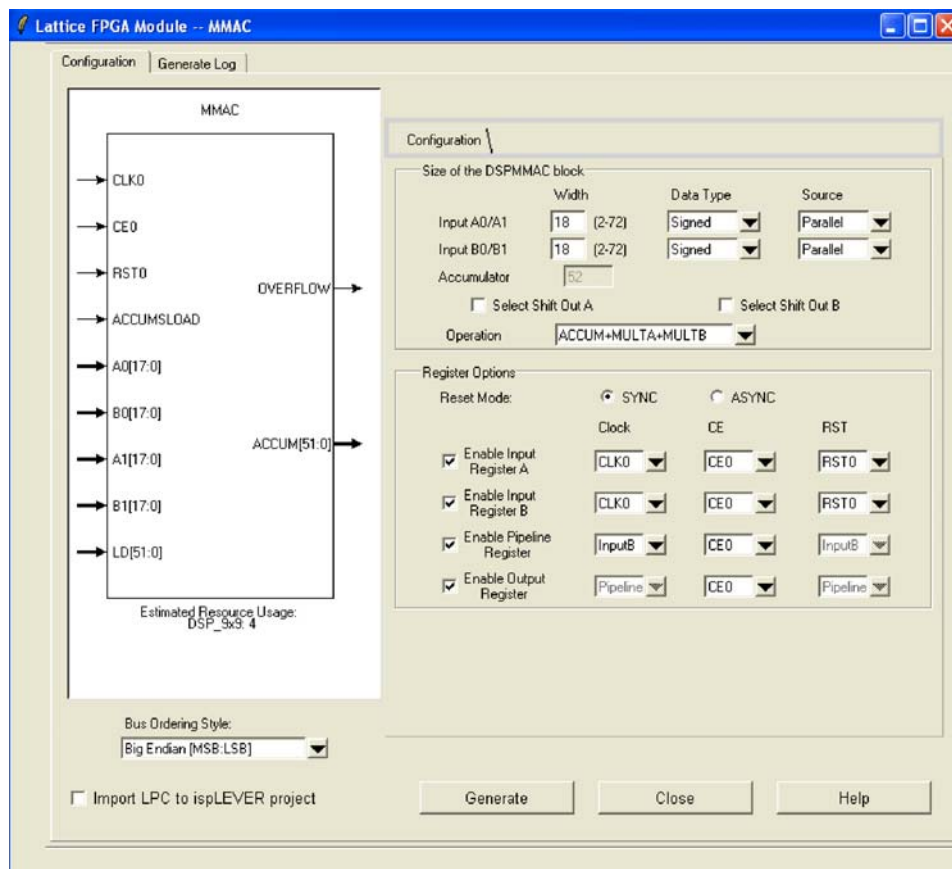
- Input bus widths from 2 to 72 bits
- Data Type specifies whether the data is Signed, Unsigned or Dynamic
- Source specifies whether data is from Parallel, Shift (DSP slice shift out port) or Dynamic
- Shift Out enables the shift out port on the DSP slice
- Operation selects whether the arithmetic operation is Addition, Subtraction or Dynamic

- **Register Options**

- Reset Mode selects whether an Async or Sync Reset is used.
- Enable Input Register selects whether data input registers are used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipeline Register selects whether data pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected. If the option InputA or InputB is selected, the pipeline register uses the same clock as the input register. Output registers will automatically be enabled and will use the same clock.
- Enable Output Register is selected by default and selects the data output pipeline registers to be used. Clock, reset and clock enable resources can be selected.

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register. The accumulator is loaded with an initial value from data on the LD port when the signal ACCUMSLOAD is toggled.

**Figure 14-4. MMAC Mode**





---

**MULTADDSUB Module**

The IPexpress MULTADDSUB Module configures elements to be packed into the sysDSP slice. The function  $A0 \times B0 \pm A1 \times B1 = P$  is implemented. The screen shot shown in Figure 14-5 shows the following:

**• Data Options**

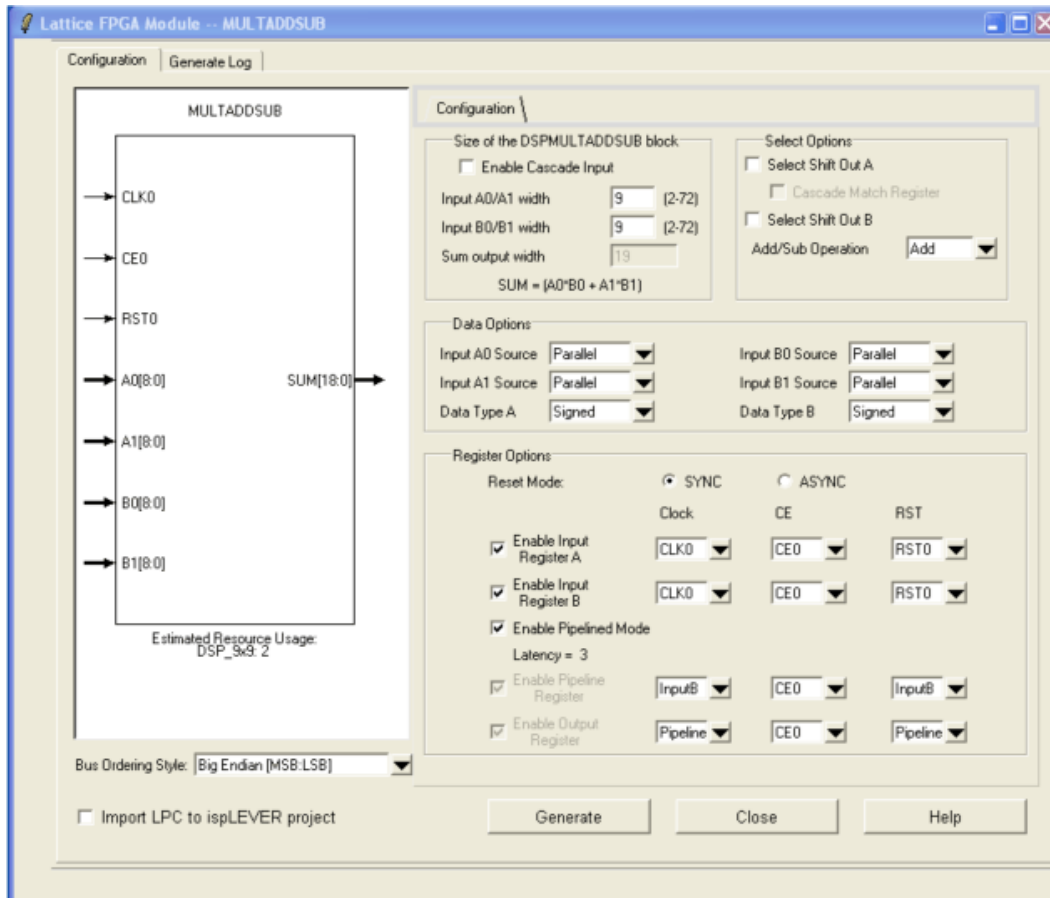
- Input bus widths from 2 to 72 bits
- Cascade input can be enabled. This can be used to cascade the Multaddsub modules. If this is selected, the CIN and SignCIN will appear as additional input ports and SignSUM as an additional output port.
- Data Type specifies whether the data is Signed, Unsigned or Dynamic
- Source specifies whether data is from Parallel, Shift or Dynamic
- Shift Out enables the shift out port on the sysDSP slice. The cascade match register is available if Shift Out A is selected. This is useful when a cascaded chain of Multaddsub modules are implemented.
- Add/Sub Operation selects whether the arithmetic operation is Addition, Subtraction or Dynamic

**• Register Options**

- Reset Mode selects if an Async or Sync Reset is used.
- Enable Input Register selects whether data input registers are used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipeline Register selects whether data pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected. If the option InputA or InputB is selected, the pipeline register uses the same clock as the input register. Output registers will automatically be enabled and will use the same clock.
- Enable Output Register selects if the data output pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipelined Mode turns on all the input, pipeline and output registers. If this mode is enabled, latency will be provided.

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register. Other than this shift register chain, the cascade chain can be enabled by selecting Enable cascade input, as described above. For appropriate operations, SUM and SignSUM (other than the last module in the cascade chain) need to be connected to CIN and SignCIN of an adjacent module. For the first module in the cascade chain, the CIN and SignCIN ports need to be assigned a value of 0.

Figure 14-5. MULTADDSUB Mode



### MULTADDSUBSUM Module

The IPexpress MULTADDSUBSUM Module configures elements to be packed into the sysDSP slice. The function  $A0 \times B0 \pm A1 \times B1 \pm A2 \times B2 \pm A3 \times B3 = P$  is implemented. The screen shot shown in Figure 14-6 shows the following:

- **Data Options**

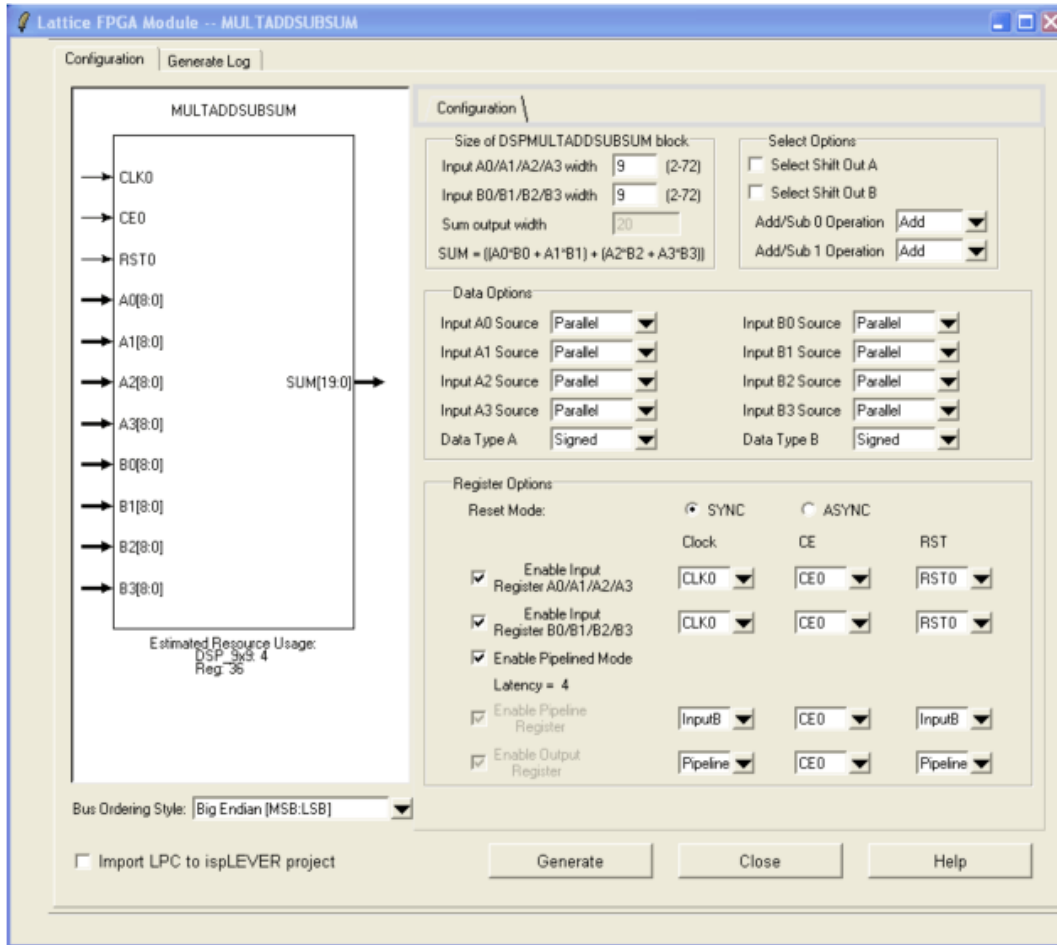
- Input bus widths from 2 to 72 bits
- Data Type specifies whether the data is Signed, Unsigned or Dynamic
- Source specifies whether data is from Parallel, Shift or Dynamic
- Shift Out enables the shift out port on the sysDSP slice
- Add/Sub Operation selects whether the arithmetic operation is Addition, Subtraction or Dynamic

- **Register Options**

- Reset Mode selects whether an Async or Sync Reset is used
- Enable Input Register selects the data input registers to be used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipeline Register selects if data pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected. If the option InputA or InputB is selected, the pipeline register uses the same clock as the input register. Output registers will automatically be enabled and will use the same clock.
- Enable Output Register selects whether the data output pipeline registers are used. If enabled, clock, reset and clock enable resources can be selected.
- Enable Pipelined Mode turns on all the input, pipeline and output registers. If this mode is enabled, latency will be provided.

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format can only be enabled if inputs are less than 18 bits. The Shift format enables a sample/shift register.

**Figure 14-6. MULTADDSUBSUM Mode**



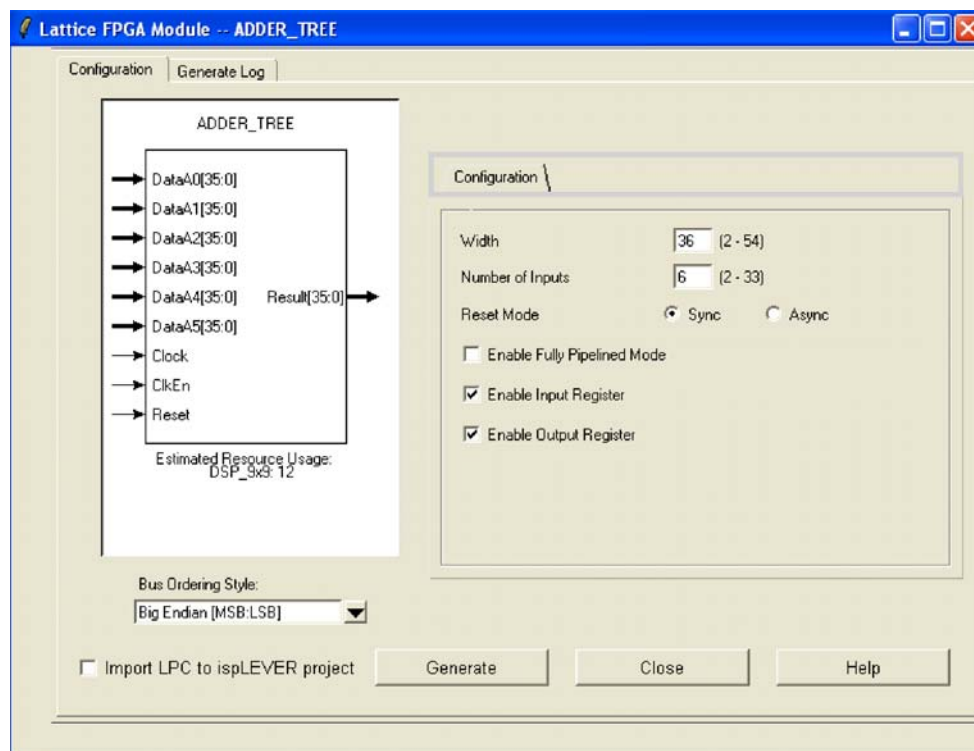
### Adder Tree Module

The IPexpress Adder Tree Module is used to generate an adder tree in the sysDSP slice. The function  $DataA0 + DataA1 + \dots + DataAn = Resultn$  is implemented. The screen shot shown in Figure 14-7 shows the following:

- Data Options
  - Input bus widths from 2 to 54 bits
  - Number of inputs can be from 2 to 33
- Register Options
  - Reset Mode selects whether an Async or Sync Reset is used
  - Enable Fully Pipelined Mode selects if data pipeline and output registers are used
  - Enable Input Register selects if data input registers are used
  - Enable Output Register selects if data output registers are used

If the number of inputs is larger than 3, multiple DSP slices will be needed. Furthermore, if Fully Pipelined Mode is enabled, registers from the generic logic will be needed.

**Figure 14-7. Adder Tree Mode**

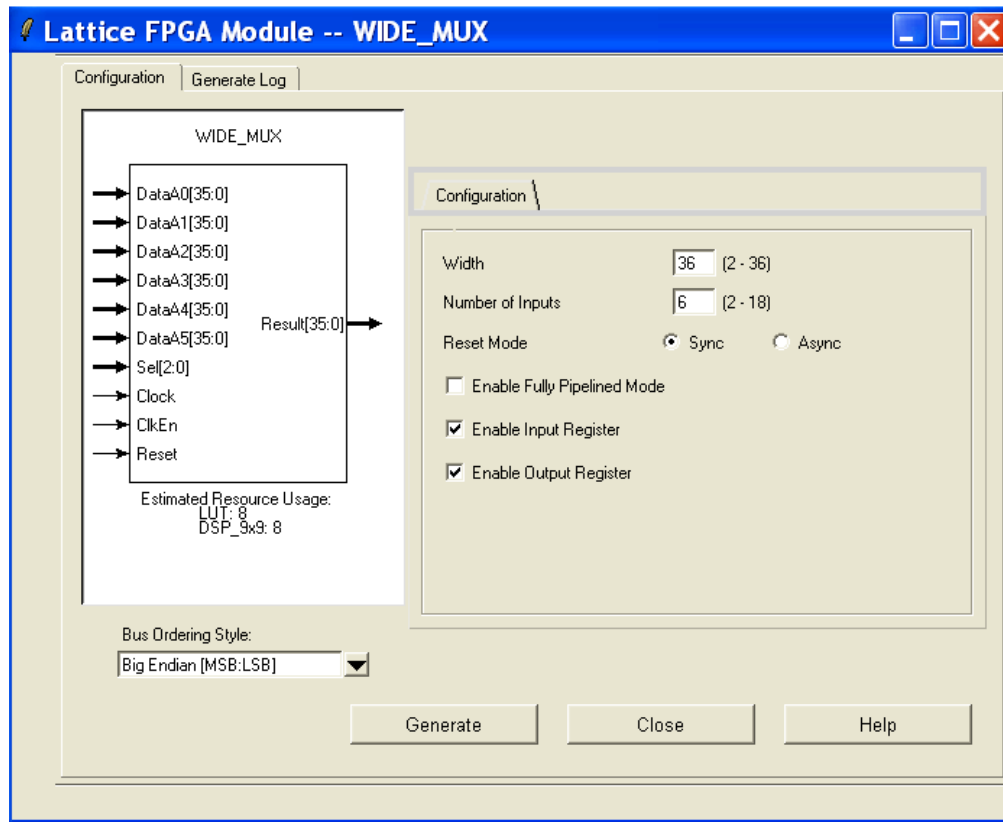


### Wide Mux Module

The IPexpress Wide Mux Module is used to generate a MUX in the sysDSP slice. The screen shot shown in Figure 14-8 shows the following:

- Data Options
  - Input bus widths from 2 to 36 bits
  - Number of inputs can be from 2 to 28
- Register Options
  - Reset Mode selects whether an Async or Sync Reset is used
  - Enable Fully Pipelined Mode selects if data input, pipeline and output registers are used
  - Enable Input Register selects if data input registers are used
  - Enable Output Register selects if data output registers are used.

Figure 14-8. Wide Mux Mode



### Barrel Shifter Module

The IPexpress Barrel Shifter Module is used to generate a Barrel Shifter in the sysDSP slice. Barrel Shifters are useful for applications such as floating point addition, compression/decompression algorithms, and pattern matching. The screen shot shown in Figure 14-9 shows the following:

- Data Options
  - Shift direction options of Left or Right
  - Type options of Zero Insert, Sign Extension (Shift Right only) or Rotate
  - Data Width can be from 1 to 40 bits or 2 to 32 for Type Rotate
- Register Options
  - Reset Mode selects whether an Async or Sync Reset is used
  - Enable Input Register selects if data input registers are used
  - Enable Pipeline Register selects if data pipeline registers are used
  - Enable Output Register selects if data output registers are used

Below are some examples of a Barrel Shifter:

Example 1:

Shift Direction = Left  
 Type = Zero Insert  
 A[7:0] = 0000 0100  
 Shift[1:0] = 10  
 P[7:0] = 0000 1000

Example 2:

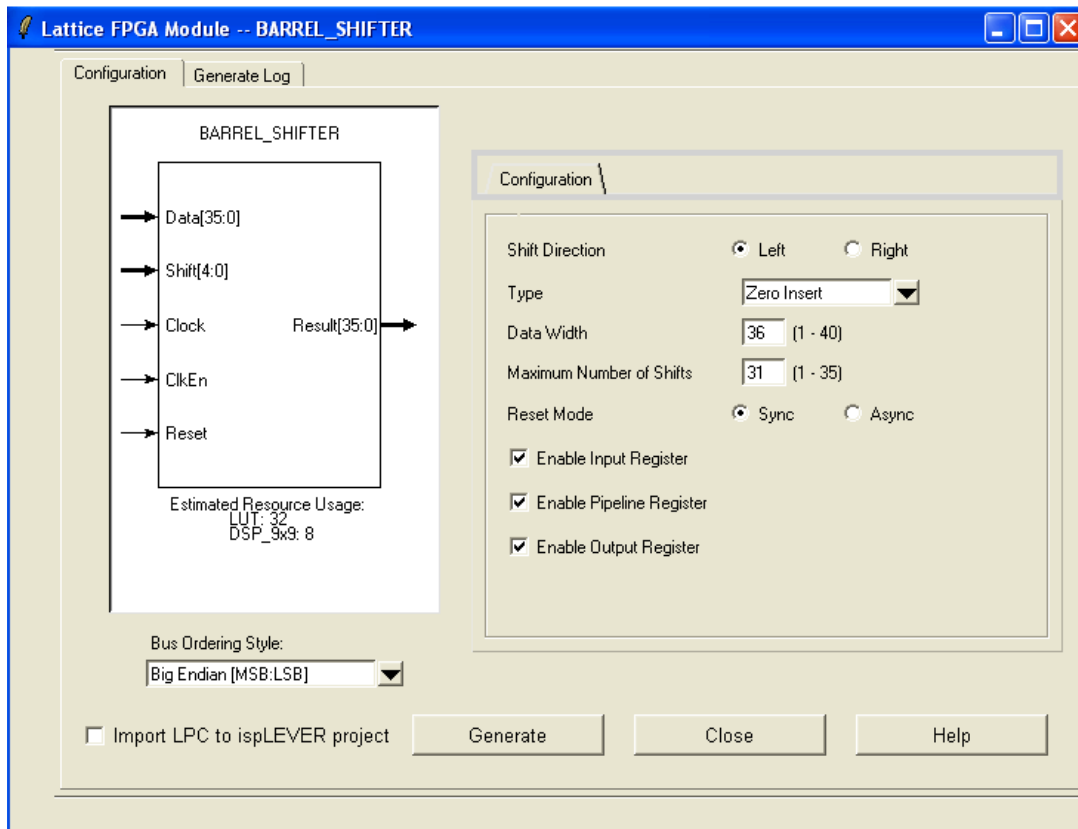
Shift Direction = Right  
 Type = Sign Extension  
 A[7:0] = 1111 1100  
 Shift[1:0] = 10  
 P[7:0] = 1111 1110

Example 3:

Shift Direction = Left  
 Type = Rotate  
 A[7:0] = 1111 1111 1100  
 Shift[1:0] = 10  
 P[7:0] = 1111 1111 1001

Multiple sysDSP slices can be spanned to accommodate large functions. Additional LUTs may be required if multiple sysDSP slices are needed.

**Figure 14-9. Barrel Shifter Mode**



---

## Targeting the sysDSP Slice by Inference

The Inferencing flow enables the design tools to infer sysDSP slices from an HDL design. It is important to note that when using the Inferencing flow, unless the code style matches the sysDSP slice, results will not be optimal. The following are VHDL and Verilog examples.

### VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
--use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity mult is
port (reset, clk      : in std_logic;
      dataax, dataay  : in std_logic_vector(8 downto 0);
      dataout        : out std_logic_vector (17 downto 0));
end;

architecture arch of mult is
signal dataax_reg, dataay_reg : std_logic_vector (8 downto 0);
signal dataout_node : std_logic_vector (17 downto 0);
signal dataout_pipeline : std_logic_vector (17 downto 0);

begin

process (clk, reset)
begin
if (reset='1') then
  dataax_reg <= (others => '0');
  dataay_reg <= (others => '0');
elsif (clk'event and clk='1') then
  dataax_reg <= dataax;
  dataay_reg <= dataay;
end if;
end process;

dataout_node <= dataax_reg * dataay_reg;

process (clk, reset)
begin
if (reset='1') then
  dataout_pipeline <= (others => '0');
elsif (clk'event and clk='1') then
  dataout_pipeline <= dataout_node;
end if;
end process;

process (clk, reset)
begin
if (reset='1') then
  dataout <= (others => '0');
elsif (clk'event and clk='1') then
  dataout <= dataout_pipeline;
end if;
end if;
```

```
end process;
```

```
end arch;
```

### Verilog Example

```
module mult (dataout, dataax, dataay, clk, reset);
    output [35:0] dataout;
    input [17:0] dataax, dataay;
    input clk,reset;
    reg [35:0] dataout;
    reg [17:0] dataax_reg, dataay_reg;

    wire [35:0] dataout_node;
    reg [35:0] dataout_reg;

    always @(posedge clk or posedge reset)
    begin
        if (reset)
            begin
                dataax_reg <= 0;
                dataay_reg <= 0;
            end
        else
            begin
                dataax_reg <= dataax;
                dataay_reg <= dataay;
            end
        end
    end

    assign dataout_node = dataax_reg * dataay_reg;

    always @(posedge clk or posedge reset)
    begin
        if (reset)
            dataout_reg <= 0;
        else
            dataout_reg <= dataout_node;
        end
    end

    always @(posedge clk or posedge reset)
    begin
        if (reset)
            dataout <= 0;
        else
            dataout <= dataout_reg;
        end
    end
endmodule
```

### Targeting the sysDSP Slice Using ispLeverDSP with Simulink

Simulink is a graphical add-on (similar to schematic entry) for MATLAB, which is produced by The MathWorks. For more information, refer to the Simulink web page at [www.mathworks.com/products/simulink/](http://www.mathworks.com/products/simulink/).



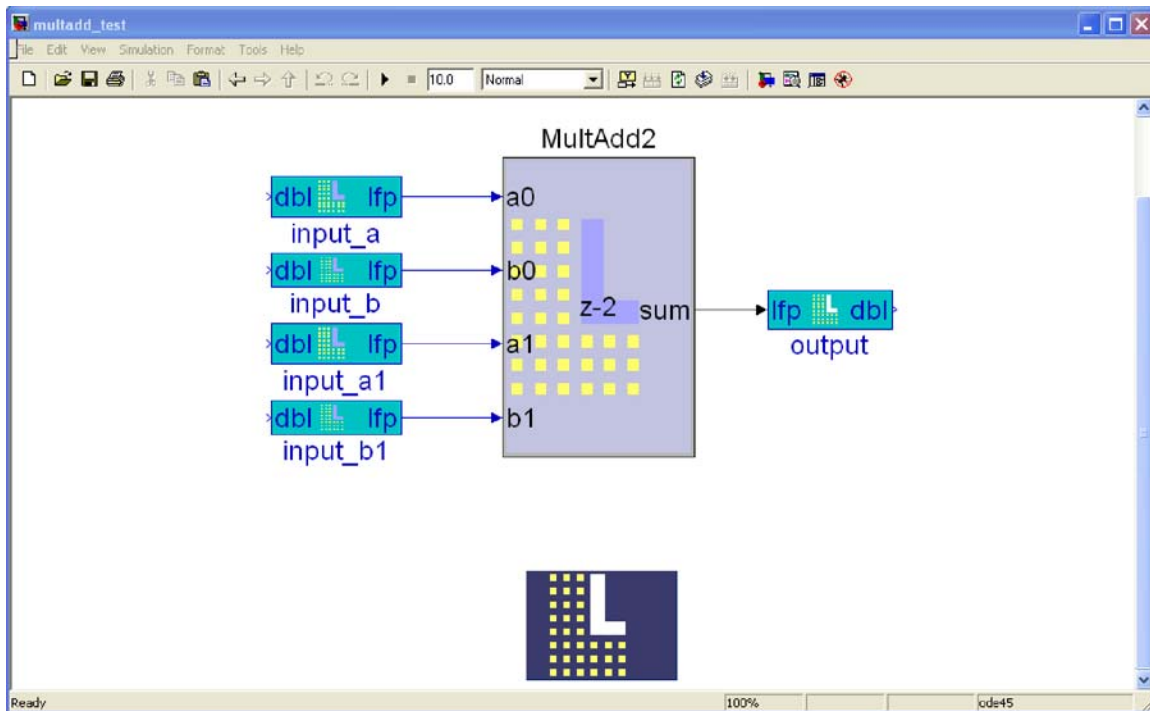
The ispLeverDSP is a block set in Simulink library provided by Lattice. It allows users to create algorithms and models in Simulink and converts the algorithms and models into RTL code.

After successful installation of ispLEVER or Diamond, set a path in MATLAB to make the Lattice block set available in the Simulink library. For more information, refer to the [ispLEVER 8.1 Installation Notice](#) or the [Lattice Diamond Installation Notice](#).

The Lattice block set includes multipliers, adders, registers, and many other building blocks. Besides the basic building blocks there are a couple of unique Lattice blocks:

- **Gateways In and Out** – The design is made of the blocks between Gateway In and Gateway Out and these blocks can be converted into HDL code. When the design is converted into RTL code, the signal fed into the Gateway In is translated into the test vector for the testbench and the signal coming out of the Gateway Out is the reference copy for self-check in the testbench. In Figure 14-10, the input\_x blocks on the left are Gateway In and the output block on the right is the Gateway Out.
- **Generator** – The Generator block is used to convert the design into HDL files that can be instantiated in a HDL design. The Generate Block is identified by the Lattice logo as shown in Figure 14-10.

**Figure 14-10. Simulink Design**



### Targeting the sysDSP Slice by Instantiating Primitives

The sysDSP slice can be targeted by instantiating the sysDSP slice primitives into a design. The advantage of instantiating primitives is that it provides access to all ports and sets all available parameters. The disadvantage of this flow is that all this customization requires extra coding by the user. Appendix A details the syntax for the sysDSP slice primitives.

## sysDSP in the Report Files

### Map Report

The Map Report includes information on how many sysDSP components are used and how many are available. A sysDSP slice is made of Multipliers and ALUs. The Map Report also shows how the sysDSP components are configured. Below is the DSP section from the Map Report Summary and the Component Details for the ALU.

#### Number Of Mapped DSP Components:

```
-----
MULT18X18C      2
MULT9X9C        0
ALU54A          1
ALU24A          0
-----
```

```
Number of Used DSP MULT Sites:  4 out of 256 (1 %)
Number of Used DSP ALU Sites:   2 out of 128 (1 %)
Number of clocks:               1
  Net CLK0_c: 3 loads, 3 rising, 0 falling (Driver: PIO CLK0 )
```

#### DSP Component Details

```
-----
. ALU54A dsp_alu_0:
```

#### 54-Bit ALU

```
Opcode 0      1
Opcode 1      0
Opcode 2      1
Opcode 3      0
Opcode 4      0
Opcode 5      0
Opcode 6      0
Opcode 7      0
Opcode 8      0
Opcode 9      1
Opcode 10     0
```

```
OpcodeOP0 Registers      CLK          CE          RST
```

```
-----
                          Input      CLK0      CE0      RST0
                          Pipeline    CLK0      CE0      RST0
```

```
OpcodeOP1 Registers      CLK          CE          RST
```

```
-----
                          Input      --      --
                          Pipeline    --      --
```

```
OpcodeIN Registers      CLK          CE          RST
```

```
-----
                          Input
                          Pipeline
```

#### Data

```
Input Registers      CLK          CE          RST
```

```
-----
                          C0
                          C1
```

Output Register	CLK	CE	RST
-----			
Output0	CLK0	CE0	RST0
Output1	CLK0	CE0	RST0
Flag Register	CLK	CE	RST
Flag	CLK0	CE0	RST0

Other

MCPAT_SOURCE	STATIC
MASKPAT_SOURCE	STATIC
MASK01	0x0000000000000000
MCPAT	0x0000000000000000
MASKPAT	0x0000000000000000
RNDPAT	0x0000000000000000
PSE17	0b1111111111111111
PSE44	0b11111111111111111111111111111111
PSE53	0b11111111
GSR	DISABLED
RESETMODE	SYNC
MULT9_MODE	DISABLED

**PAR Report**

The PAR Report shows how the sysDSP components are packed into the sysDSP slices. The PAR Report shown below has two MULT18x18s and one ALU that are packed into one sysDSP slice:

```

----- DSP Report -----
DSP Slice #: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
# of MULT9X9C
# of MULT18X18C
# of ALU24A
# of ALU54A

DSP Slice #: 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
# of MULT9X9C
# of MULT18X18C      2
# of ALU24A
# of ALU54A          1

DSP Slice 33      Component_Type      Physical_Type      Instance_Name
MULT18_R52C2      MULT18X18C        MULT18             dsp_mult_1
MULT18_R52C3      MULT18X18C        MULT18             dsp_mult_0
ALU54_R52C5       ALU54A            ALU54              dsp_alu_0
----- End of DSP Report -----

```

## Trace Report

The Trace Report includes the timing of the design. The timing paths are analyzed to, from or through the sysDSP slice. Below is an example from the Post PAR Trace Report.

```
=====
Preference: FREQUENCY NET "CLK0_c" 350.000000 MHz ;
           72 items scored, 0 timing errors detected.
-----
```

Passed: The following path meets requirements by 0.320ns

```
Logical Details: Cell type Pin type Cell/ASIC name (clock net +/-)

Source:          MULT18X18C Port dsp_mult_0(ASIC) (from CLK0_c +)
Destination:    ALU54A Port dsp_alu_0(ASIC) (to CLK0_c +)

Delay:           0.340ns (100.0% logic, 0.0% route), 1 logic levels.
```

### Constraint Details:

```
0.340ns physical path delay dsp_mult_0 to dsp_alu_0 meets
2.857ns delay constraint less
0.000ns skew and
2.197ns MU_SET requirement (totaling 0.660ns) by 0.320ns
```

### Physical Path Details:

Name	Fanout	Delay (ns)	Site	Resource
C2OUT_DEL	---	0.340	*18_R52C3.CLK0 to *T18_R52C3.P35	dsp_mult_0 (from CLK0_c)
ROUTE	1	0.000	*T18_R52C3.P35 to *54_R52C5.MB35	test_ecp3_mult_out_p_1_35 (to CLK0_c)

```
-----
0.340 (100.0% logic, 0.0% route), 1 logic levels.
```

### Clock Skew Details:

#### Source Clock:

Delay	Connection
0.778ns	N4.PADDI to MULT18_R52C3.CLK0

#### Destination Clock :

Delay	Connection
0.778ns	N4.PADDI to ALU54_R52C5.CLK0

Report: 394.166MHz is the maximum frequency for this preference.

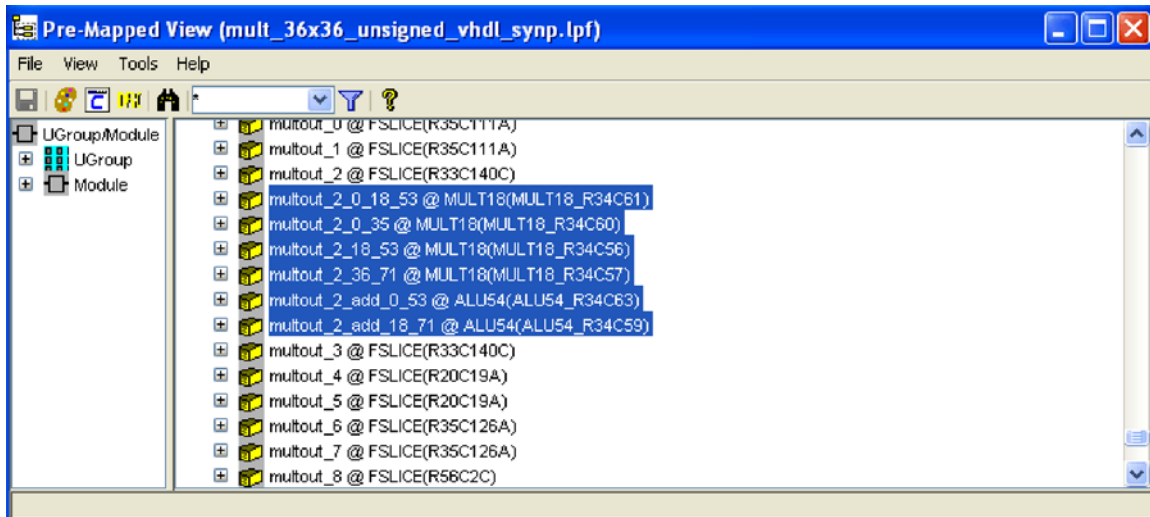
## sysDSP in the ispLEVER Design Planner

Note: See Appendix B for usage information for Lattice Diamond design software.

### Pre-Mapped View

In the Design Planner Pre-Mapped View, sysDSP instances can be viewed or grouped together. Figure 14-11 is a screen shot showing MULT and ALU instances.

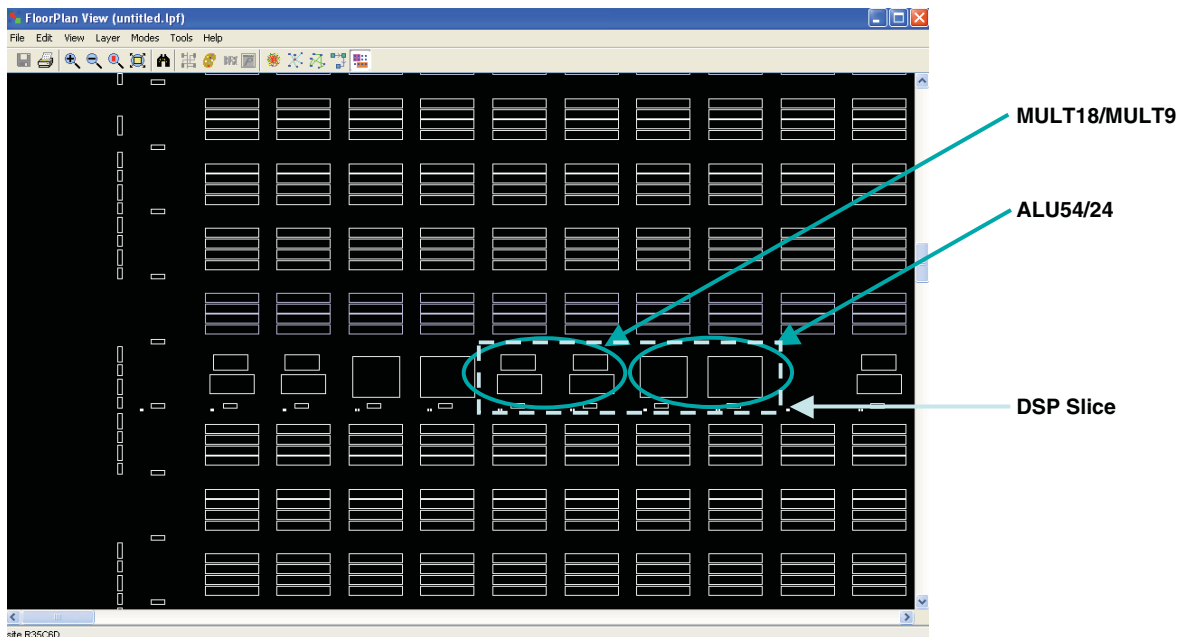
**Figure 14-11. Design Planner Pre-Mapped View**



### Floorplan View

The DSP slices are organized in rows, as shown in Figure 14-12. It can be seen that each slice extends to four columns with the multipliers on the left and ALUs on the right.

**Figure 14-12. Floorplan View**

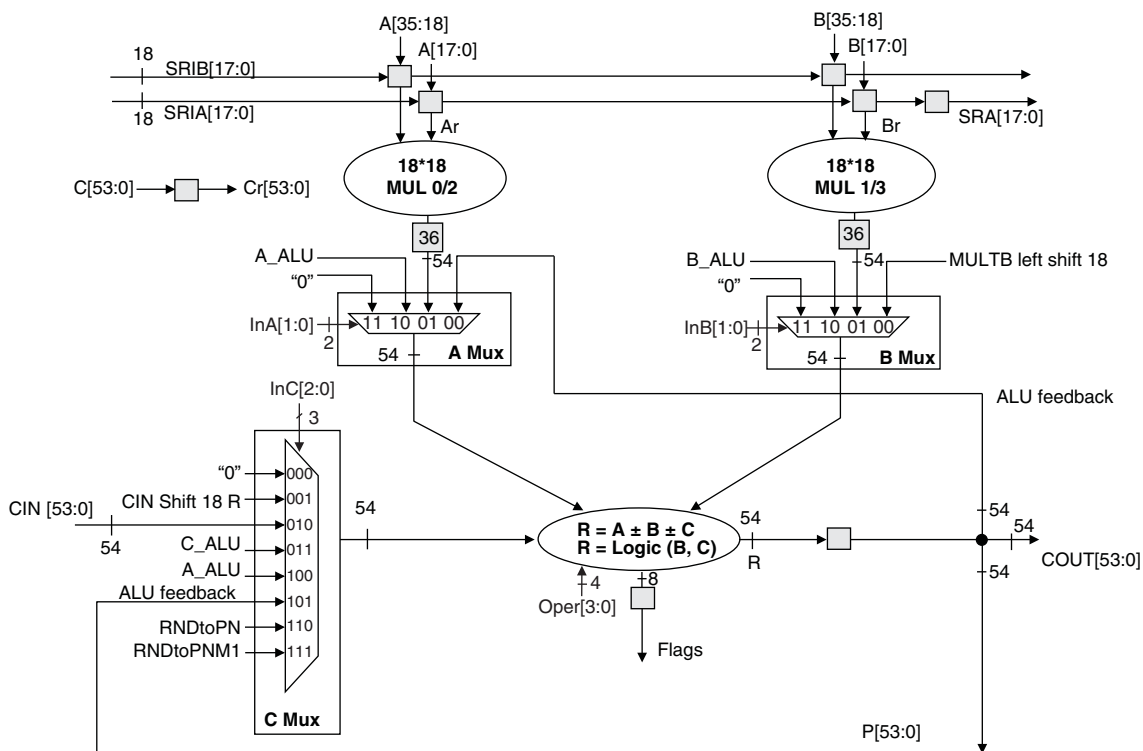


## Advanced Features of the sysDSP Slice

Examining the LatticeECP3 sysDSP slice in more detail (Figure 14-13) reveals the Arithmetic Logic Unit (ALU) and three programmable muxes, AMUX, BMUX and CMUX. These components are used in combination to enable the advanced functions of the sysDSP slice, such as:

- Cascading of slices for implementing Adder Trees fully in sysDSP slices.
- Ternary addition functions are implemented through bypassing of multipliers.
- Various rounding techniques modify the data using the ALU.
- ALU flags.
- Dynamic muxes input selection allows for Time Division Multiplexing (TDM) of the sysDSP slice resources.

**Figure 14-13. Detailed sysDSP Slice Diagram**



**Notes:**

1. Two slices are shown: Slice0 (Mult0 and Mult1) and Slice1 (Mult2 and Mult3)
2. Each 18x18 can also implement two 9x9s.
3. A Mux[53:36] and B Mux[53:36] are sign extended if SIGNEDA or SIGNEDB = 1; otherwise, they are zero extended.
4. P[17:0] has an independent multiplexer to bypass or not bypass the COUT[17:0] register. This is required to support 36\*36 multiplication mode.
5. A 54-bit ALU does not always feed 54 bits from A, B and C. Sometimes it will select only 36b, as seen in "Arithmetic Modes".

□ = Bypassable registers

### AMUX

AMUX selects between multiple 54-bit inputs to the ALU statically or dynamically. The inputs to AMUX are listed in Table 14-2.

**Table 14-2. AMUX Inputs**

InA[1:0]	AMUX Signal Selection
00	ALU feedback
01	Output of MULTA, sign-extended to 54 bits
10	A_ALU={Cra[17:0], Ara}, concatenated result of Cra and Ara
11	GND or 54 bits of 0

Signal Descriptions:

- Cra is generated from Cr by sign extending Ar35 (on Cr(17:0)), and Br35 (on Cr(44:27))
- Ara is generated from the registered input of A (AR) sign extended to 36 bits

### BMUX

BMUX selects between multiple 54-bit inputs to the ALU statically or dynamically. The inputs to BMUX are listed in Table 14-3.

**Table 14-3. BMUX Inputs**

InB[1:0]	BMUX Signal Selection
00	Output of MULTB, left shift 18 bits
01	Output of MULTB, sign-extended to 54 bits
10	B_ALU={Cra[44:27], Bra}, concatenated result of Cra and Bra
11	GND or 54 bits of 0

Signal Descriptions:

- Bra is generated from the registered input of B (BR) sign extended to 36 bits

### CMUX

CMUX selects between multiple 54-bit inputs to the ALU statically or dynamically. The inputs to CMUX are listed in Table 14-4.

**Table 14-4. CMUX Inputs**

InC[2:0]	CMUX Signal Selection
000	GND or 54 bits of 0
001	CIN right shift 18 bits
010	CIN
011	C_ALU: sign extended Cr
100	A_ALU
101	ALU feedback
110	Constant for rounding (RNDtoPN)
111	Constant for rounding (RNDtoPNM1)

Signal Descriptions:

- CIN (Cascade Input) is connected to the COUT (Cascade Output) of an adjacent sysDSP slice

## ALU

**Operation Codes:** The ALU does the processing of the sysDSP slice. Its operation can be static or dynamic and is set using a 4-bit operation code. The ALU can operate with either two inputs (BMUX and CMUX) or with three inputs mode (AMUX, BMUX and CMUX). Table 14-5 lists the operation codes and the ALU functions that are performed.

**Table 14-5. Operation Codes and ALU Functions**

#	OPR(3)	OPR(2)	OPR(1)	OPR(0)	Function	
1	0	1	0	0	$R = A + B + C$	Arithmetic
2	0	1	0	1	$R = A - B + C$	
3	0	1	1	0	$R = A + B - C$	
4	0	1	1	1	$R = A - B - C$	
5	1	1	0	0	$R = B \text{ XNOR } C$	Logical
6	1	1	1	0	$R = B \text{ XOR } C$	
7	0	0	0	0	<b><math>R = B \text{ NAND } C</math></b>	
8	1	0	0	0	$R = B \text{ AND } C$	
9	0	0	1	1	$R = B \text{ OR } C$	
10	1	0	1	1	$R = B \text{ NOR } C$	

Signal Descriptions:

- OPR(x) is the opcode bit number
- A is the output from the AMUX
- B is the output from the BMUX
- C is the output from the CMUX

For arithmetic operation, the ALU is a ternary adder with selectable addition or subtraction for each input. The data width for the ALU operation is 54 bits with the following exception: MUX A and B opcodes are 10 and MUX C opcode is 011. For this operation mode, the data width is limited to 36 bits. If the 0 input port for one of the MUXes is selected by the port selection port, the ALU function effectively becomes a two-input operation.

**Flags:** ALU flags are indicators generated by comparing the ALU result and constants or fixed patterns. Figures 14-14 and 14-15 show the circuitry for the flags. Tables 14-6 and 14-7 provide definitions of the inputs and outputs.



Figure 14-14. Flag Circuitry

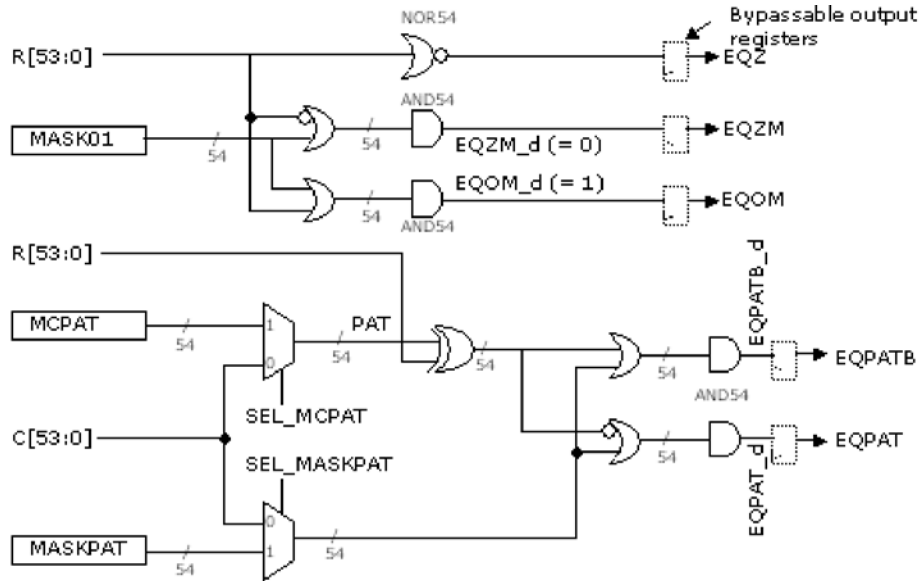


Figure 14-15. Flag Pattern Circuitry

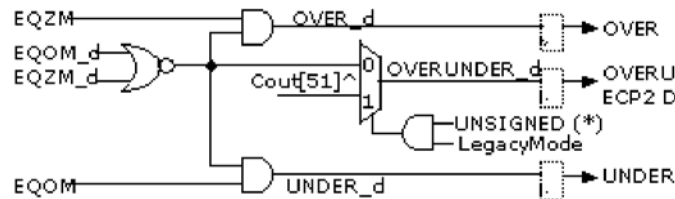


Table 14-6. Flag Input Definitions

Flag Input	Definition
R	54-bit result from the ALU
MASK01	Memory cell constant which is a mask for EQZM/EQOM
MASKPAT	Memory cell constant which is a mask constant for EQPAT/EQPATB
MCPAT	Memory cell constant which is a MEM Cell Pattern constant
SELPAT	Mux select chooses the PAT source
SELMASK	Mux select chooses the mask source for EQPAT/EQPATB
LegacyMode	Sets OVERUNDER which is used to support LatticeECP2 legacy overflow

Table 14-7. Flag Output Definitions

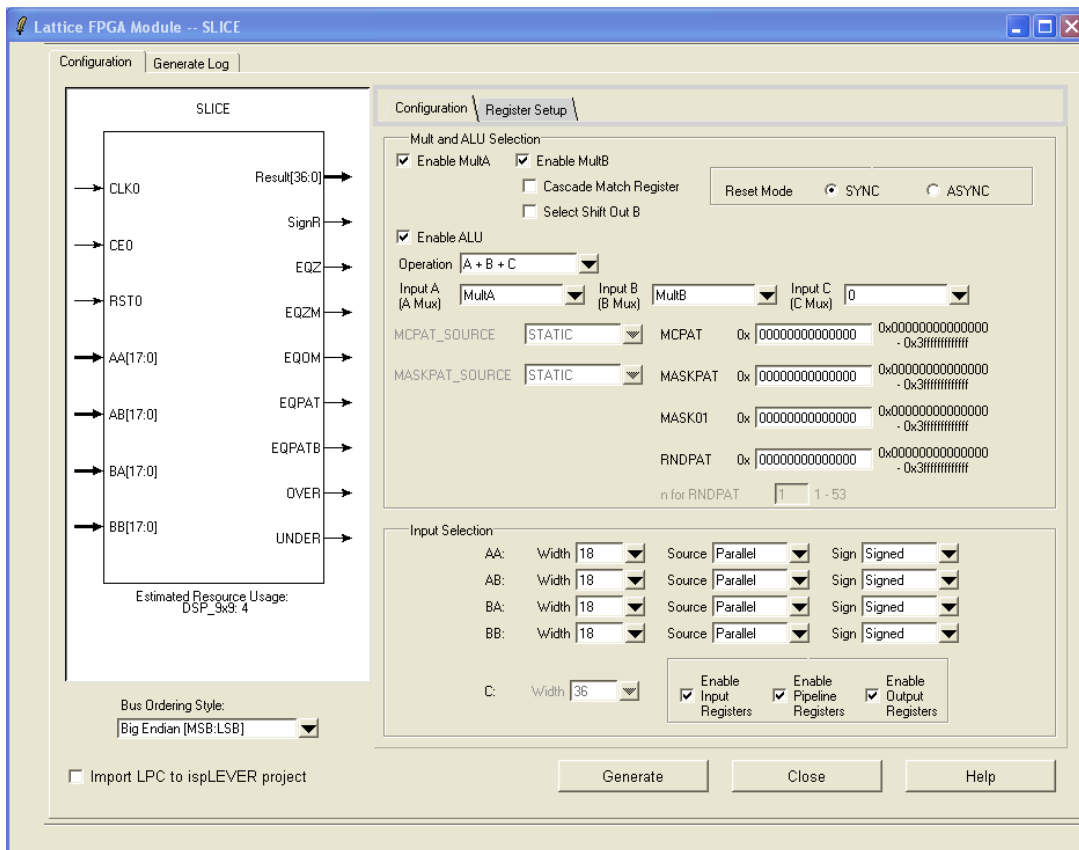
Flag Output	Definition	Equation
EQZ	Equal to zero	$EQZ = \text{not}(\text{bitwise\_or}(R))$
EQZM	Equal to zero with mask	$EQZM = \text{bitwise\_and}(\text{not}(R) \text{ or } MASK01)$
EQOM	Equal to one with mask	$EQOM = \text{bitwise\_and}(R \text{ or } MASK01)$
EQPAT <sup>1</sup>	Equal to PAT with mask	$EQPAT = \text{bitwise\_and}(\text{not}(R \text{ xor } MCPAT) \text{ or } MASKPAT)$
EQPATB <sup>1</sup>	Equal to bit inverted PAT with mask	$EQPATB = \text{bitwise\_and}((R \text{ xor } MCPAT) \text{ or } MASKPAT)$
OVER	Accumulator overflow	$OVER = EQZM \text{ and } (\text{not}(EQOM\_d \text{ or } EQZM\_d))$
UNDER	Accumulator underflow	$UNDER = EQOM \text{ and } (\text{not}(EQOM\_d \text{ or } EQZM\_d))$
OVERUNDER	Either over or underflow	
_d	The suffix “_d” denotes non-registered signals	

1. For EQPAT and EQPATB, the pattern to be compared with can be specified through MCPAT and MASKPAT or through C input of sysDSP slice dynamically.

## IPexpress Slice Module

See Appendix B for information on using the IPexpress Slice Module in Diamond. The IPexpress Slice Module allows users to configure a sysDSP slice and enable all of its advanced features. Figure 14-16 shows the GUI for the Slice Module.

Figure 14-16. Slice Module GUI



There are two sections for configuration, Mult/ALU and Input Selections. The output ports, such as data and flags, are enabled automatically depending on sysDSP operation and user options. The two sections are described below.

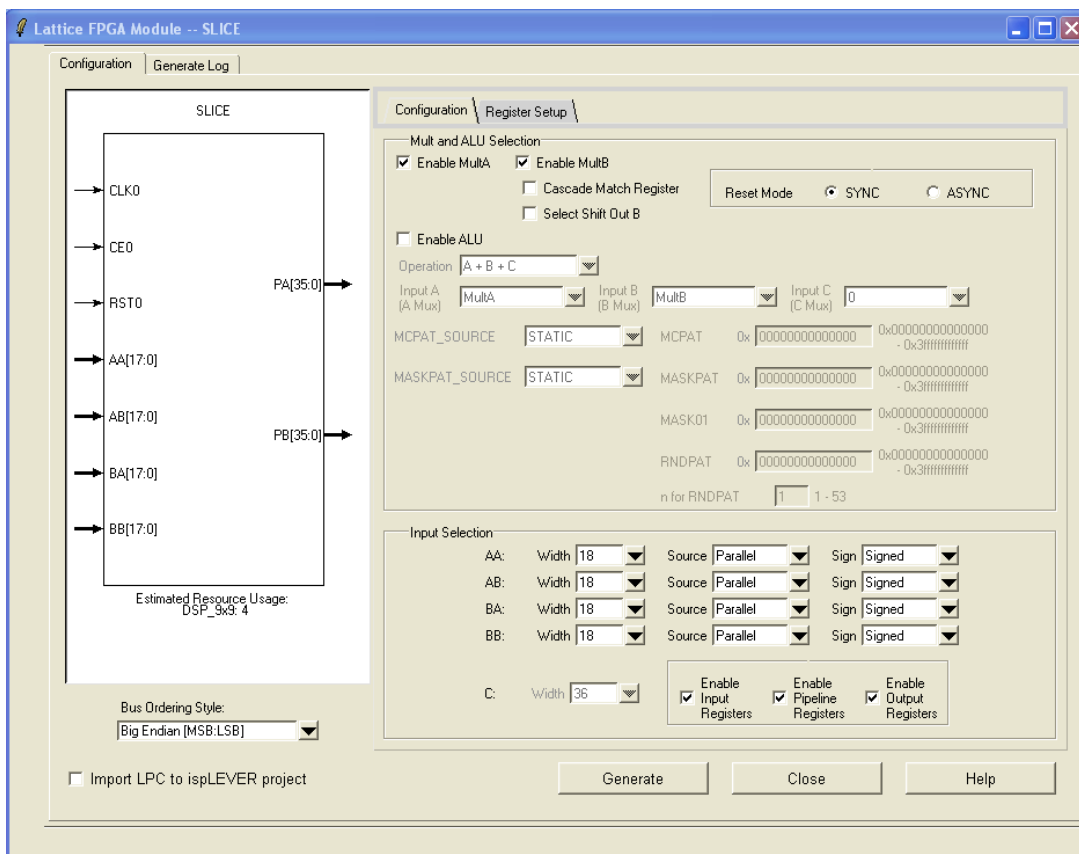
### Input Selection

- Input bus widths which can be from 1-18 bit for AA, AB, BA, BB. AA and AB are the inputs for MultA, BA and BB are the inputs for MultB.
- Input bus widths which can be from 1-54 bit for C (only when "Enable B" is unchecked). This C input port is different from the CMUX output (input of ALU).
- Data Type specifies if the data is Signed, Unsigned or Dynamic
- Source specifies if data is from Parallel, Shift or Dynamic
- Shift Out enables the shift out port on the sysDSP slice
- Input, Pipeline and Output Registers are user-selectable

### Mult and ALU Selection

To configure a sysDSP slice, the user must first consider whether the ALU is needed. The result of the ALU is a function of A, B and C, where A, B and C are outputs of AMUX, BMUX and CMUX, respectively. If the check box "Enable ALU" is unchecked, all the ALU related options will be grayed out and thus not available, as shown in Figure 14-17. These include Operation, Input A (A Mux), Input B (B Mux), Input C (C Mux), MCPAT\_SOURCE, MASKPAT\_SOURCE, MCPAT, MASKPAT, MASK01, and RNDPAT. This will set up the slice as two independent multipliers.

**Figure 14-17. Slice Module GUI with "Enable ALU" Unchecked**



The check box "Enable ALU" should be checked if the ALU is needed. The ALU can operate in either three-input or two-input mode or it can be configured dynamically.

- **Three-input operation options:**  $A+B+C$ ,  $A+B-C$ ,  $A-B+C$ ,  $A-B-C$ .
- **Two-input operation options:**  $B \text{ XNOR } C$ ,  $B \text{ XOR } C$ ,  $B \text{ NAND } C$ ,  $B \text{ AND } C$ ,  $B \text{ OR } C$ ,  $B \text{ NOR } C$ .
- **Dynamic option:** Any of the possible Opcodes listed in Table 14-5.

Next, determine whether MultA, MultB, or both, will be used.

For CMUX, the output can be one of the eight inputs or it can be configured dynamically. The available configurations for the outputs of AMUX and BMUX are dependent on whether MultA and MultB are enabled.

- If “Enable MultA” is checked, the output of AMUX can be either MultA output or configured dynamically.
- If “Enable MultA” is unchecked, the options are ALU feedback, A\_ALU, 0 and DYNAMIC.
- If “Enable MultB” is checked, the output of BMUX can be MultB output, MultB shift 18L or configured dynamically.
- If “Enable MultB” is unchecked, the options are B\_ALU, 0 and DYNAMIC.

Configuration of “Enable MultA” will affect the availability of the operation options:

- If “Enable MultA” is checked, the available options are  $A + B + C$ ,  $A - B + C$ ,  $A + B - C$ ,  $A - B - C$  and DYNAMIC;
- If “Enable MultA” is unchecked, the available options are  $A + B + C$ ,  $A - B + C$ ,  $A + B - C$ ,  $A - B - C$ ,  $B \text{ XNOR } C$ ,  $B \text{ XOR } C$ ,  $B \text{ NAND } C$ ,  $B \text{ AND } C$ ,  $B \text{ OR } C$ ,  $B \text{ NOR } C$  and DYNAMIC.

Configuration of “Enable MultB” will affect the availability of the cascade match register and select shift out B.

- If “Enable MultB” is checked, “cascade match register” and “select shift out B” will be available. CascA0 and CascA1 in Figure 14-1 are controlled by the check box of cascade match register.
- If “cascade match register” is checked, registers CascA0 and CascA1 will be added to the line of shift register A.
- If “select shift out B” is checked, both SROA and SROB will be added to output ports of the Slice module.

Table 14-8 lists the possible configuration scenarios.

**Table 14-8. Possible Configuration Scenarios**

Scenario	MULTA	MULTB	ALU	Operation Mode
1	Yes	No	No	a. Single multiplier instance: MULTA only. b. Static mode only. <sup>1</sup>
2	No	Yes	No	a. Single multiplier instance: MULTB only. b. Static mode only.
3	Yes	Yes	No	a. Two multiplier instances: MULTA and MULTB b. Static mode only.
4	Yes	Yes	Yes	a. Two multiplier instances and ALU instance: MULTA, MULTB and ALU. b. Static mode: A + B + C or A - B + C or A + B - C or A - B - C. c. Dynamic mode <sup>2</sup> : A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C
5	No	Yes	Yes	a. One multiplier instance and ALU instance: MULTB and ALU. b. Static mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C c. Dynamic mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C.
6	Yes	No	Yes	a. One multiplier instance and ALU instance: MULTA and ALU. b. Static mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C c. Dynamic mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C
7	No	No	Yes	a. ALU instance: ALU only. b. Static mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C c. Dynamic mode: A + B + C, A - B + C, A + B - C, A - B - C, B XNOR C, B XOR C, B NAND C, B AND C, B OR C, B NOR C

1. Static mode means available operation options other than DYNAMIC.

2. Dynamic mode means the operation can be configured dynamically.

If the ALU is used, some flag signals are generated by comparing the ALU results with pre-specified constants or C input. In the GUI, these constants are MASK01, MCPAT, and MASKPAT.

- If “Enable MultB” is unchecked, options for MCPAT\_SOURCE and MASKPAT\_SOURCE will be available. These options are STATIC and DYNAMIC.
- If “STATIC” is checked, the values for MCPAT and MASKPAT need to be specified by the user.
- If “DYNAMIC” is checked, the values for MCPAT and MASKPAT are specified though the input port C dynamically.
- If "Enable ALU" is checked, there will be an output port SignR available, which indicates the sign of the result. This port should be left unconnected if not being used.

## Rounding

LatticeECP3 sysDSP slices provide the capability of rounding through the two rounding patterns. Rounding on a number is achieved by adding a constant (e.g. 0.5 for decimal) followed by a truncation, or in the following equation:

$$\text{Output} = \text{Floor}(\text{Input} + 0.5), \text{ with input and output in decimal}$$

In sysDSP slices, there are two rounding constant inputs for CMUX. One is 0...0001000...0, (called RNDtoPN) with the binary or rounding point between the “1” and the “0” in front of the “1”, and the value of RNDtoPN is 0.5 in decimal. The other one is 0...00001111...1, (called RNDtoPNM1) with the binary or rounding point between the last “0” and the first “1”. Where N is the position of binary point or the number of 1s. By choosing one of these constants, the output of the sysDSP slice is (result of ALU)+ RNDtoPN or (result of ALU)+ RNDtoPNM1. The ALU result at the previous clock cycle is the input for the rounding operation and the rounding result is:

$$\text{Output\_truncated} = \text{Floor}(\text{Input} + \text{RNDtoPN})$$

or

$$\text{Output\_truncated} = \text{Floor}(\text{Input} + \text{RNDtoPNM1})$$

Other manipulations are also possible, depending on the rounding scheme to be implemented. Thus, these two constants are chosen by the user and it is up to the user to determine which rounding scheme is implemented. The information of the ALU output value (positive, negative etc.) can be extracted and used for port selection of CMUX, thus the selection of rounding constants.

The following are examples of rounding. Example 1 is described in detail and the others are similar.

### Example 1: Rounding Toward Zero

To implement rounding to zero, for a positive input, RNDtoPNM1 is used; for a negative input, RNDtoPN is used. In the GUI, the value of RNDPAT needs to be specified by the user, which is used to specify the rounding point position (e.g. 000000 00000000 00000000 00000000 00000000 00000000 000 10000). The user is responsible for the switching of the RND\_CONSTANTS depending on the sign of the ALU result and rounding scheme.

Steps for rounding toward zero:

1. Extract the sign of the ALU result.
2. Select the input port of the CMUX based on the result of step 1. If the sign is positive, RNDtoPNM1 will be selected to pass through CMUX. If the sign is negative, RNDtoPN will be selected to pass through CMUX.
3. Add the RND\_CONSTANT and the ALU result.
4. Truncate the result of step 3. This is the rounding result.

Rounding to zero of a 54-bit value to 49-bit

Sign	Data Type	Data Value	Rounding point
+	Input	000000 00000000 00000000 00000000 00000000 01010101 010 10000	▼
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111	
	Output Data	000000 00000000 00000000 00000000 00000000 01010101 010 11111	
	Output truncated	000000 00000000 00000000 00000000 00000000 01010101 010 01010101 010	
-	Input	111111 11111111 11111111 11111111 11111111 01010101 010 10000	
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000	
	Output Data	111111 11111111 11111111 11111111 11111111 01010101 011 00000	
	Output truncated	111111 11111111 11111111 11111111 11111111 01010101 011 01010101 011	

### Example 2: Rounding to Positive Infinity

Rounding to positive infinity of a 54-bit value to 49-bit

Sign	Data Type	Data Value	Rounding point
+	Input	000000 00000000 00000000 00000000 00000000 01010101 010 10000	▼
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000	
	Output Data	000000 00000000 00000000 00000000 00000000 01010101 011 00000	
	Output truncated	000000 00000000 00000000 00000000 00000000 01010101 011 01010101 011	
-	Input	111111 11111111 11111111 11111111 11111111 01010101 010 10000	
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000	
	Output Data	111111 11111111 11111111 11111111 11111111 01010101 011 00000	
	Output truncated	111111 11111111 11111111 11111111 11111111 01010101 011 01010101 011	

### Example 3: Rounding to Negative Infinity

Rounding to negative infinity of a 54-bit value to 49-bit

Sign	Data Type	Data Value	Rounding point
+	Input	000000 00000000 00000000 00000000 00000000 01010101 010 10000	
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111	
	Output Data	000000 00000000 00000000 00000000 00000000 01010101 010 11111	
	Output truncated	000000 00000000 00000000 00000000 00000000 01010101 010	
-	Input	111111 11111111 11111111 11111111 11111111 01010101 010 10000	
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111	
	Output Data	111111 11111111 11111111 11111111 11111111 01010101 010 11111	
	Output truncated	111111 11111111 11111111 11111111 11111111 01010101 010	

### Example 4: Rounding Away from Zero

Rounding away from zero of a 54-bit value to 49-bit

Sign	Data Type	Data Value	Rounding point
+	Input	000000 00000000 00000000 00000000 00000000 01010101 010 10000	
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000	
	Output Data	000000 00000000 00000000 00000000 00000000 01010101 011 00000	
	Output truncated	000000 00000000 00000000 00000000 00000000 01010101 011	
-	Input	111111 11111111 11111111 11111111 11111111 01010101 010 10000	
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111	
	Output Data	111111 11111111 11111111 11111111 11111111 01010101 010 11111	
	Output truncated	111111 11111111 11111111 11111111 11111111 01010101 010	

For convergent rounding, there are two options: one is rounding toward even and the other is rounding toward odd. For rounding toward even, the number of exact x.5 is rounding toward the closest even and the others are rounding toward the nearest.

In addition to the steps listed above for rounding toward zero or infinity, some extra steps are needed. If the value of (result of ALU) + RND\_CONSTANT matches a certain pattern, or if flag EQPAT = 1 is generated, the least significant bit after truncation will be assigned 0 (rounding toward even) or 1 (rounding toward odd). For rounding toward even, MAPAT = 11...1100...00, MASKPAT = 11...1100...00, where the rounding point is between the last "1" and the first "0". The flag EQPAT generated with this pattern indicates whether the (result of ALU) + RND\_CONSTANT matches a certain pattern of XX...X100...00.

### Example 5: Rounding Toward Even

Rounding toward even steps (54-bit value to 49-bit MSBs, rounding point = 5)

1. Add RNDtoPN to the result of ALU
2. If EQPAT=1(matches pattern XX...X100...00), then integer LSB is replaced with 0; otherwise, keep the LSB.

Rounding toward even of a 54-bit value to 49-bit

Sign	Data Type	Data Value	Rounding point
+	Input	000000 00000000 00000000 00000000 00000000 00000000 010 10000	
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000	
	Output Data	000000 00000000 00000000 00000000 00000000 00000000 011 00000	
	Match 100000	Yes	
	Output truncated	000000 00000000 00000000 00000000 00000000 00000000 00000000 010	
-	Input	111111 11111111 11111111 11111111 11111111 11111111 101 10000	
	RNDtoPN	000000 00000000 00000000 00000000 00000000 00000000 000 10000	
	Output Data	111111 11111111 11111111 11111111 11111111 11111111 110 00000	
	Match 100000	No	
	Output truncated	111111 11111111 11111111 11111111 11111111 11111111 11111111 110	

### Example 6: Rounding Toward Odd

For rounding toward odd, MCPAT = 11...11011...11, MASKPAT = 11...1100...00, where the rounding point is between the last "1" and the first "0". The flag EQPAT generated with this pattern indicates whether the (result of ALU) + RND\_CONSTANT matches a certain pattern of XX...X011...11.

Rounding toward odd steps (rounding point = 5):

1. Add RNDtoPNM1 to the result of ALU
2. If EQPAT=1(matches pattern XX...X01...11), then integer LSB is replaced with 1; otherwise, keep the LSB.

Rounding toward odd of a 54-bit value to 49-bit

Sign	Data Type	Data Value	Rounding point
+	Input	000000 00000000 00000000 00000000 00000000 00000000 010 10000	
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111	
	Output Data	000000 00000000 00000000 00000000 00000000 00000000 010 11111	
	Match 011111	Yes	
	Output truncated	000000 00000000 00000000 00000000 00000000 00000000 00000000 011	
-	Input	111111 11111111 11111111 11111111 11111111 11111111 101 10000	
	RNDtoPNM1	000000 00000000 00000000 00000000 00000000 00000000 000 01111	
	Output Data	111111 11111111 11111111 11111111 11111111 11111111 101 11111	
	Match 011111	No	
	Output truncated	111111 11111111 11111111 11111111 11111111 11111111 11111111 101	

For dynamic and random rounding, the selection port for CMUX is opened to the user. When and how the two rounding patterns, RNDtoPN and RNDtoPNM1, are switched is up to the user.



## sysDSP Slice Control Signal and Data Signal Descriptions

RST	Asynchronous reset of selected registers
SIGNEDA	Dynamic signal: 0 = unsigned, 1 = signed
SIGNEDB	Dynamic signal: 0 = unsigned, 1 = signed
ACCUMSLOAD	Dynamic signal: 0 = accumulate, 1 = load
ADDNSUB	Dynamic signal: 0 = subtract, 1 = add
SOURCEA	Dynamic signal: 0 = parallel input, 1 = shift input
SOURCEB	Dynamic signal: 0 = parallel input, 1 = shift input

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
June 2009	01.1	Updated the performance table.
		Updated the IPexpress modules GUI and description.
		Updated the MAP, PAR and Trace reports.
		Updated the DSP slice block diagram and description on arithmetic operation.
June 2010	01.2	Updated for Lattice Diamond software support.
February 2012	01.3	Updated document with new corporate logo.
July 2013		Updated Technical Support Assistance information.

## Appendix A. DSP Primitives

```

module MULT18X18C (
A17,A16,A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0,
B17,B16,B15,B14,B13,B12,B11,B10,B9,B8,B7,B6,B5,B4,B3,B2,B1,B0,
SIGNEDA,SIGNEDB,SOURCEA,SOURCEB,
CE3,CE2,CE1,CE0,CLK3,CLK2,CLK1,CLK0,RST3,RST2,RST1,RST0,
SRIA17,SRIA16,SRIA15,SRIA14,SRIA13,SRIA12,SRIA11,SRIA10,SRIA9,SRIA8,SRIA7,SRIA6,SRIA
5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0,
SRIB17,SRIB16,SRIB15,SRIB14,SRIB13,SRIB12,SRIB11,SRIB10,SRIB9,SRIB8,SRIB7,SRIB6,SRIB
5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0,
SROA17,SROA16,SROA15,SROA14,SROA13,SROA12,SROA11,SROA10,SROA9,SROA8,SROA7,SROA6,SROA
5,SROA4,SROA3,SROA2,SROA1,SROA0,
SROB17,SROB16,SROB15,SROB14,SROB13,SROB12,SROB11,SROB10,SROB9,SROB8,SROB7,SROB6,SROB
5,SROB4,SROB3,SROB2,SROB1,SROB0,
ROA17,ROA16,ROA15,ROA14,ROA13,ROA12,ROA11,ROA10,ROA9,ROA8,ROA7,ROA6,ROA5,ROA4,ROA3,R
OA2,ROA1,ROA0,
ROB17,ROB16,ROB15,ROB14,ROB13,ROB12,ROB11,ROB10,ROB9,ROB8,ROB7,ROB6,ROB5,ROB4,ROB3,R
OB2,ROB1,ROB0,
P35,P34,P33,P32,P31,P30,P29,P28,P27,P26,P25,P24,P23,P22,P21,P20,P19,
P18,P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0,SIGNEDP) ;
input  A17,A16,A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0 ;
input  B17,B16,B15,B14,B13,B12,B11,B10,B9,B8,B7,B6,B5,B4,B3,B2,B1,B0 ;
input  SIGNEDA,SIGNEDB,SOURCEA,SOURCEB ;
input  CE3,CE2,CE1,CE0,CLK3,CLK2,CLK1,CLK0,RST3,RST2,RST1,RST0 ;
input  SRIA17,SRIA16,SRIA15,SRIA14,SRIA13,SRIA12,SRIA11,SRIA10,SRIA9 ;
input  SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0 ;
input  SRIB17,SRIB16,SRIB15,SRIB14,SRIB13,SRIB12,SRIB11,SRIB10,SRIB9 ;
input  SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0 ;
output SROA17,SROA16,SROA15,SROA14,SROA13,SROA12,SROA11,SROA10,SROA9 ;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0 ;
output SROB17,SROB16,SROB15,SROB14,SROB13,SROB12,SROB11,SROB10,SROB9 ;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0 ;
output
ROA17,ROA16,ROA15,ROA14,ROA13,ROA12,ROA11,ROA10,ROA9,ROA8,ROA7,ROA6,ROA5,ROA4,ROA3,R
OA2,ROA1,ROA0 ;
output
ROB17,ROB16,ROB15,ROB14,ROB13,ROB12,ROB11,ROB10,ROB9,ROB8,ROB7,ROB6,ROB5,ROB4,ROB3,R
OB2,ROB1,ROB0 ;
output P35,P34,P33,P32,P31,P30,P29,P28,P27,P26,P25,P24,P23,P22,P21,P20,P19,P18 ;
output P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0 ;
output SIGNEDP ;

parameter REG_INPUTA_CLK = "NONE" ;
parameter REG_INPUTA_CE = "CE0" ;
parameter REG_INPUTA_RST = "RST0" ;
parameter REG_INPUTB_CLK = "NONE" ;
parameter REG_INPUTB_CE = "CE0" ;
parameter REG_INPUTB_RST = "RST0" ;
parameter REG_PIPELINE_CLK = "NONE" ;
parameter REG_PIPELINE_CE = "CE0" ;
parameter REG_PIPELINE_RST = "RST0" ;
parameter REG_OUTPUT_CLK = "NONE" ;
parameter REG_OUTPUT_CE = "CE0" ;

```

```

parameter REG_OUTPUT_RST = "RST0";
parameter CAS_MATCH_REG = "FALSE";
parameter MULT_BYPASS = "DISABLED";
parameter GSR = "ENABLED";
parameter RESETMODE = "SYNC";
endmodule

module MULT9X9C (
A8,A7,A6,A5,A4,A3,A2,A1,A0,B8,B7,B6,B5,B4,B3,B2,B1,B0,
SIGNEDA,SIGNEDB,SOURCEA,SOURCEB,
CE3,CE2,CE1,CE0,CLK3,CLK2,CLK1,CLK0,RST3,RST2,RST1,RST0,
SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0,
SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0,
SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0,
SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0,
ROA8,ROA7,ROA6,ROA5,ROA4,ROA3,ROA2,ROA1,ROA0,
ROB8,ROB7,ROB6,ROB5,ROB4,ROB3,ROB2,ROB1,ROB0,
P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0,
SIGNEDP,
);
input A8,A7,A6,A5,A4,A3,A2,A1,A0;
input B8,B7,B6,B5,B4,B3,B2,B1,B0;
input SIGNEDA,SIGNEDB,SOURCEA,SOURCEB;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
input SRIA8,SRIA7,SRIA6,SRIA5,SRIA4,SRIA3,SRIA2,SRIA1,SRIA0;
input SRIB8,SRIB7,SRIB6,SRIB5,SRIB4,SRIB3,SRIB2,SRIB1,SRIB0;
output SROA8,SROA7,SROA6,SROA5,SROA4,SROA3,SROA2,SROA1,SROA0;
output SROB8,SROB7,SROB6,SROB5,SROB4,SROB3,SROB2,SROB1,SROB0;
output ROA8,ROA7,ROA6,ROA5,ROA4,ROA3,ROA2,ROA1,ROA0;
output ROB8,ROB7,ROB6,ROB5,ROB4,ROB3,ROB2,ROB1,ROB0;
output P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0;
output SIGNEDP;

parameter REG_INPUTA_CLK = "NONE";
parameter REG_INPUTA_CE = "CE0";
parameter REG_INPUTA_RST = "RST0";
parameter REG_INPUTB_CLK = "NONE";
parameter REG_INPUTB_CE = "CE0";
parameter REG_INPUTB_RST = "RST0";
parameter REG_PIPELINE_CLK = "NONE";
parameter REG_PIPELINE_CE = "CE0";
parameter REG_PIPELINE_RST = "RST0";
parameter REG_OUTPUT_CLK = "NONE";
parameter REG_OUTPUT_CE = "CE0";
parameter REG_OUTPUT_RST = "RST0";
parameter CAS_MATCH_REG = "NONE";
parameter MULT_BYPASS = "DISABLED";
parameter GSR = "ENABLED";
parameter RESETMODE = "SYNC";
endmodule

module ALU54A(
CE3,CE2,CE1,CE0,CLK3,CLK2,CLK1,CLK0,RST3,RST2,RST1,RST0,SIGNEDIA,SIGNEDIB,
A35,A34,A33,A32,A31,A30,A29,A28,A27,A26,A25,A24,A23,A22,A21,A20,A19,A18,

```

```

A17,A16,A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0,
B35,B34,B33,B32,B31,B30,B29,B28,B27,B26,B25,B24,B23,B22,B21,B20,B19,B18,
B17,B16,B15,B14,B13,B12,B11,B10,B9,B8,B7,B6,B5,B4,B3,B2,B1,B0,
C53,C52,C51,C50,C49,C48,C47,C46,C45,C44,C43,C42,C41,C40,C39,C38,C37,C36,
C35,C34,C33,C32,C31,C30,C29,C28,C27,C26,C25,C24,C23,C22,C21,C20,C19,C18,
C17,C16,C15,C14,C13,C12,C11,C10,C9,C8,C7,C6,C5,C4,C3,C2,C1,C0,
MA35,MA34,MA33,MA32,MA31,MA30,MA29,MA28,MA27,MA26,MA25,MA24,MA23,MA22,MA21,
MA20,MA19,MA18,MA17,MA16,MA15,MA14,MA13,MA12,MA11,MA10,MA9,MA8,MA7,MA6,MA5,
MA4,MA3,MA2,MA1,MA0,
MB35,MB34,MB33,MB32,MB31,MB30,MB29,MB28,MB27,MB26,MB25,MB24,MB23,MB22,MB21,
MB20,MB19,MB18,MB17,MB16,MB15,MB14,MB13,MB12,MB11,MB10,MB9,MB8,MB7,MB6,MB5,
MB4,MB3,MB2,MB1,MB0,
CIN53,CIN52,CIN51,CIN50,CIN49,CIN48,CIN47,CIN46,CIN45,CIN44,CIN43,CIN42,
CIN41,CIN40,CIN39,CIN38,CIN37,CIN36,CIN35,CIN34,CIN33,CIN32,CIN31,CIN30,CIN29,
CIN28,CIN27,CIN26,CIN25,CIN24,CIN23,CIN22,CIN21,CIN20,CIN19,CIN18,CIN17,CIN16,
CIN15,CIN14,CIN13,CIN12,CIN11,CIN10,CIN9,CIN8,CIN7,CIN6,CIN5,CIN4,CIN3,CIN2,CIN1,CIN
0,
OP10,OP9,OP8,OP7,OP6,OP5,OP4,OP3,OP2,OP1,OP0,SIGNEDCIN,
R53,R52,R51,R50,R49,R48,R47,R46,R45,R44,R43,R42,R41,R40,R39,R38,R37,R36,
R35,R34,R33,R32,R31,R30,R29,R28,R27,R26,R25,R24,R23,R22,R21,R20,R19,R18,
R17,R16,R15,R14,R13,R12,R11,R10,R9,R8,R7,R6,R5,R4,R3,R2,R1,R0,
EQZ,EQZM,EQOM,EQPAT,EQPATB,OVER,UNDER,OVERUNDER,SIGNEDR);
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDIA,SIGNEDIB;
input A35,A34,A33,A32,A31,A30,A29,A28,A27,A26,A25,A24,A23,A22,A21,A20,A19,A18;
input A17,A16,A15,A14,A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0;
input B35,B34,B33,B32,B31,B30,B29,B28,B27,B26,B25,B24,B23,B22,B21,B20,B19,B18;
input B17,B16,B15,B14,B13,B12,B11,B10,B9,B8,B7,B6,B5,B4,B3,B2,B1,B0;
input C53,C52,C51,C50,C49,C48,C47,C46,C45,C44,C43,C42,C41,C40,C39,C38,C37,C36;
input C35,C34,C33,C32,C31,C30,C29,C28,C27,C26,C25,C24,C23,C22,C21,C20,C19,C18;
input C17,C16,C15,C14,C13,C12,C11,C10,C9,C8,C7,C6,C5,C4,C3,C2,C1,C0;
input MA35,MA34,MA33,MA32,MA31,MA30,MA29,MA28,MA27,MA26,MA25,MA24,MA23,MA22,MA21;
input MA20,MA19,MA18,MA17,MA16,MA15,MA14,MA13,MA12,MA11,MA10,MA9,MA8,MA7,MA6,MA5;
input MA4,MA3,MA2,MA1,MA0;
input MB35,MB34,MB33,MB32,MB31,MB30,MB29,MB28,MB27,MB26,MB25,MB24,MB23,MB22,MB21;
input MB20,MB19,MB18,MB17,MB16,MB15,MB14,MB13,MB12,MB11,MB10,MB9,MB8,MB7,MB6,MB5;
input MB4,MB3,MB2,MB1,MB0;
input CIN53,CIN52,CIN51,CIN50,CIN49,CIN48,CIN47,CIN46,CIN45,CIN44,CIN43,CIN42;
input CIN41,CIN40,CIN39,CIN38,CIN37,CIN36,CIN35,CIN34,CIN33,CIN32,CIN31,CIN30,CIN29;
input CIN28,CIN27,CIN26,CIN25,CIN24,CIN23,CIN22,CIN21,CIN20,CIN19,CIN18,CIN17,CIN16;
input
CIN15,CIN14,CIN13,CIN12,CIN11,CIN10,CIN9,CIN8,CIN7,CIN6,CIN5,CIN4,CIN3,CIN2,CIN1,CIN
0;
input OP10,OP9,OP8,OP7,OP6,OP5,OP4,OP3,OP2,OP1,OP0,SIGNEDCIN;
output R53,R52,R51,R50,R49,R48,R47,R46,R45,R44,R43,R42,R41,R40,R39,R38,R37,R36;
output R35,R34,R33,R32,R31,R30,R29,R28,R27,R26,R25,R24,R23,R22,R21,R20,R19,R18;
output R17,R16,R15,R14,R13,R12,R11,R10,R9,R8,R7,R6,R5,R4,R3,R2,R1,R0;
output EQZ,EQZM,EQOM,EQPAT,EQPATB,OVER,UNDER,OVERUNDER,SIGNEDR;

parameter REG_INPUTC0_CLK = "NONE";
parameter REG_INPUTC0_CE = "CE0";
parameter REG_INPUTC0_RST = "RST0";
parameter REG_INPUTC1_CLK = "NONE";
parameter REG_INPUTC1_CE = "CE0";
parameter REG_INPUTC1_RST = "RST0";

```

```

parameter REG_OPCODEOP0_0_CLK = "NONE";
parameter REG_OPCODEOP0_0_CE = "CE0";
parameter REG_OPCODEOP0_0_RST = "RST0";
parameter REG_OPCODEOP1_0_CLK = "NONE";
parameter REG_OPCODEOP0_1_CLK = "NONE";
parameter REG_OPCODEOP0_1_CE = "CE0";
parameter REG_OPCODEOP0_1_RST = "RST0";
parameter REG_OPCODEOP1_1_CLK = "NONE";
parameter REG_OPCODEIN_0_CLK = "NONE";
parameter REG_OPCODEIN_0_CE = "CE0";
parameter REG_OPCODEIN_0_RST = "RST0";
parameter REG_OPCODEIN_1_CLK = "NONE";
parameter REG_OPCODEIN_1_CE = "CE0";
parameter REG_OPCODEIN_1_RST = "RST0";
parameter REG_OUTPUT0_CLK = "NONE";
parameter REG_OUTPUT0_CE = "CE0";
parameter REG_OUTPUT0_RST = "RST0";
parameter REG_OUTPUT1_CLK = "NONE";
parameter REG_OUTPUT1_CE = "CE0";
parameter REG_OUTPUT1_RST = "RST0";
parameter REG_FLAG_CLK = "NONE";
parameter REG_FLAG_CE = "CE0";
parameter REG_FLAG_RST = "RST0";
parameter MCPAT_SOURCE = "STATIC";
parameter MASKPAT_SOURCE = "STATIC";
parameter MASK01 = "0x0000000000000000";
parameter MCPAT = "0x0000000000000000";
parameter MASKPAT = "0x0000000000000000";
parameter RNDPAT = "0x0000000000000000";
parameter GSR = "ENABLED";
parameter RESETMODE = "SYNC";
parameter MULT9_MODE = "DISABLED";
parameter FORCE_ZERO_BARREL_SHIFT = "DISABLED";
parameter LEGACY = "DISABLED";
endmodule

```

```

module ALU24A(
MA17,MA16,MA15,MA14,MA13,MA12,MA11,MA10,MA9,MA8,MA7,MA6,MA5,
MA4,MA3,MA2,MA1,MA0,
MB17,MB16,MB15,MB14,MB13,MB12,MB11,MB10,MB9,MB8,MB7,MB6,MB5,
MB4,MB3,MB2,MB1,MB0,
CIN23,CIN22,CIN21,CIN20,CIN19,CIN18,CIN17,CIN16,
CIN15,CIN14,CIN13,CIN12,CIN11,CIN10,CIN9,CIN8,CIN7,CIN6,CIN5,CIN4,CIN3,CIN2,CIN1,CIN
0,
CE3,CE2,CE1,CE0,CLK3,CLK2,CLK1,CLK0,RST3,RST2,RST1,RST0,SIGNEDIA,SIGNEDIB,OPADDN-
SUB,OPCINSEL,
R23,R22,R21,R20,R19,R18,
R17,R16,R15,R14,R13,R12,R11,R10,R9,R8,R7,R6,R5,R4,R3,R2,R1,R0);

input MA17,MA16,MA15,MA14,MA13,MA12,MA11,MA10,MA9,MA8,MA7,MA6,MA5;
input MA4,MA3,MA2,MA1,MA0;
input MB17,MB16,MB15,MB14,MB13,MB12,MB11,MB10,MB9,MB8,MB7,MB6,MB5;
input MB4,MB3,MB2,MB1,MB0;

```

---

```
input CIN23,CIN22,CIN21,CIN20,CIN19,CIN18,CIN17,CIN16;
input
CIN15,CIN14,CIN13,CIN12,CIN11,CIN10,CIN9,CIN8,CIN7,CIN6,CIN5,CIN4,CIN3,CIN2,CIN1,CIN
0;
input          CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNE-
DIA,SIGNEDIB,OPADDNSUB,OPCINSEL;
output R23,R22,R21,R20,R19,R18;
output R17,R16,R15,R14,R13,R12,R11,R10,R9,R8,R7,R6,R5,R4,R3,R2,R1,R0;

parameter REG_OUTPUT_CLK = "NONE";
parameter REG_OUTPUT_CE = "CE0";
parameter REG_OUTPUT_RST = "RST0";
parameter REG_OPCODE_0_CLK = "NONE";
parameter REG_OPCODE_0_CE = "CE0";
parameter REG_OPCODE_0_RST = "RST0";
parameter REG_OPCODE_1_CLK = "NONE";
parameter REG_OPCODE_1_CE = "CE0";
parameter REG_OPCODE_1_RST = "RST0";
parameter GSR = "ENABLED";
parameter RESETMODE = "SYNC";
endmodule
```

## Appendix B. Using IPexpress for Diamond

### Invoking IPexpress for Diamond

There are several ways IPexpress can be invoked. To invoke IPexpress from the Start menu, select:

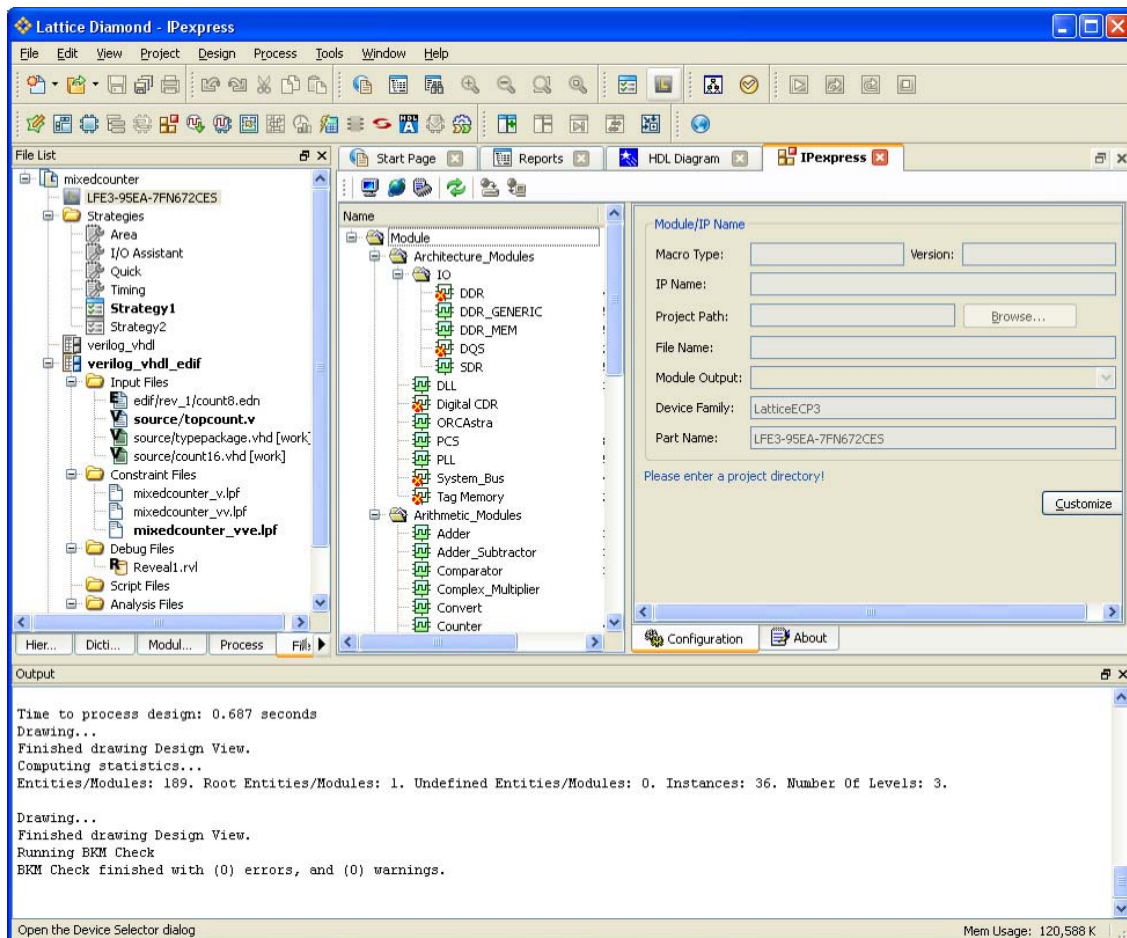
**Start > Programs > Lattice Diamond 1.0 > Accessories > IPexpress**

To invoke IPexpress from within Diamond, a project must be opened, then invoke the IPexpress icon or select:

**Tools > IPexpress**

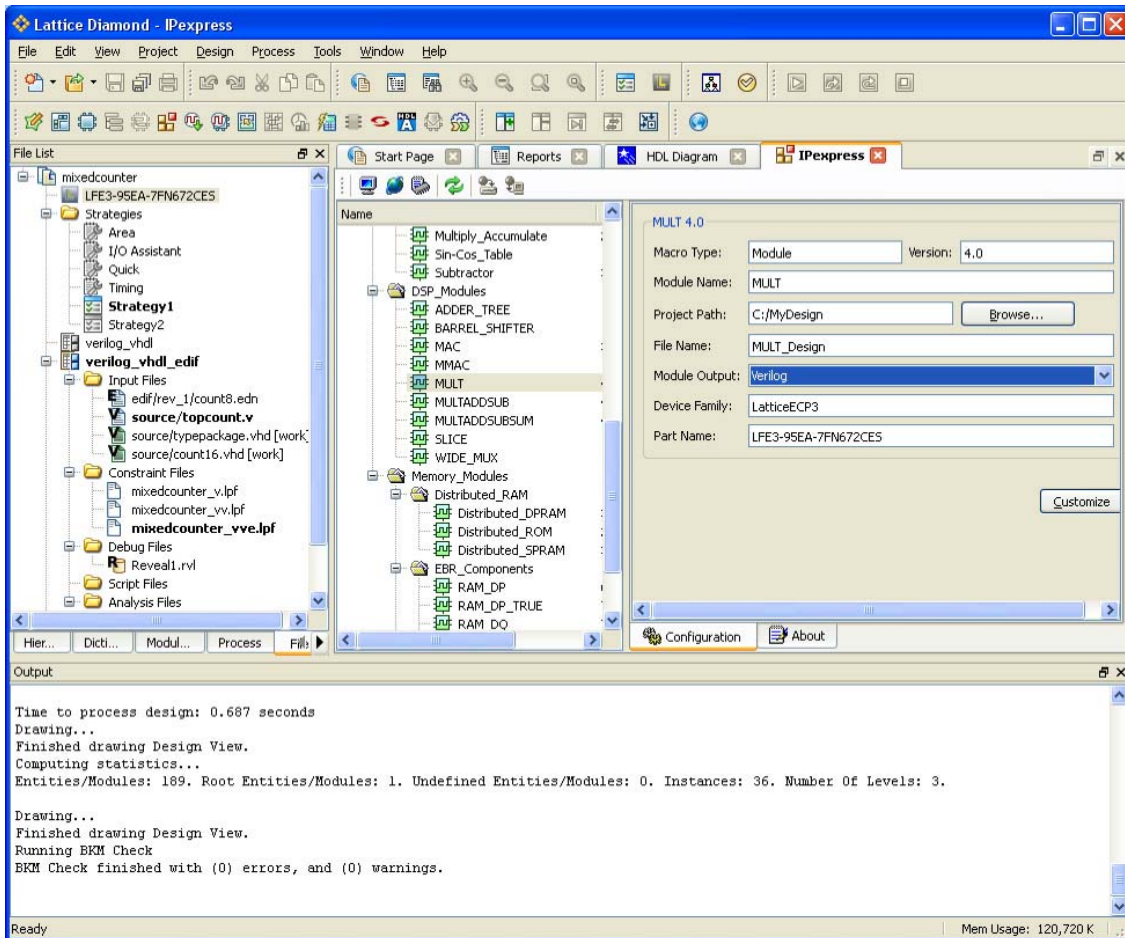
The IPexpress interface appears as shown below:

**Figure 14-18. IPexpress Interface**



Scroll to the modules and left-click on the module you wish to use. Enter the information into the IPexpress interface as shown in the example Figure 14-19.

Figure 14-19. Creating the Module Instance

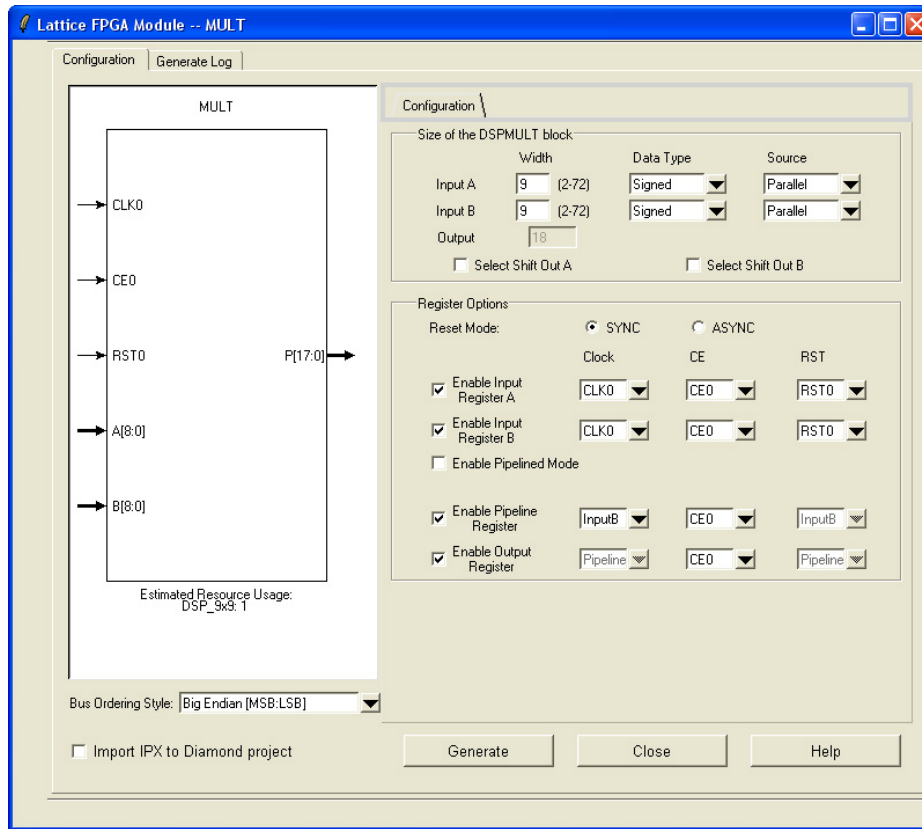


Select **Customize**.

An example of a MULT Module Dialog window appears as shown in Figure 14-20. Select **Help** for information about the fields in this window.

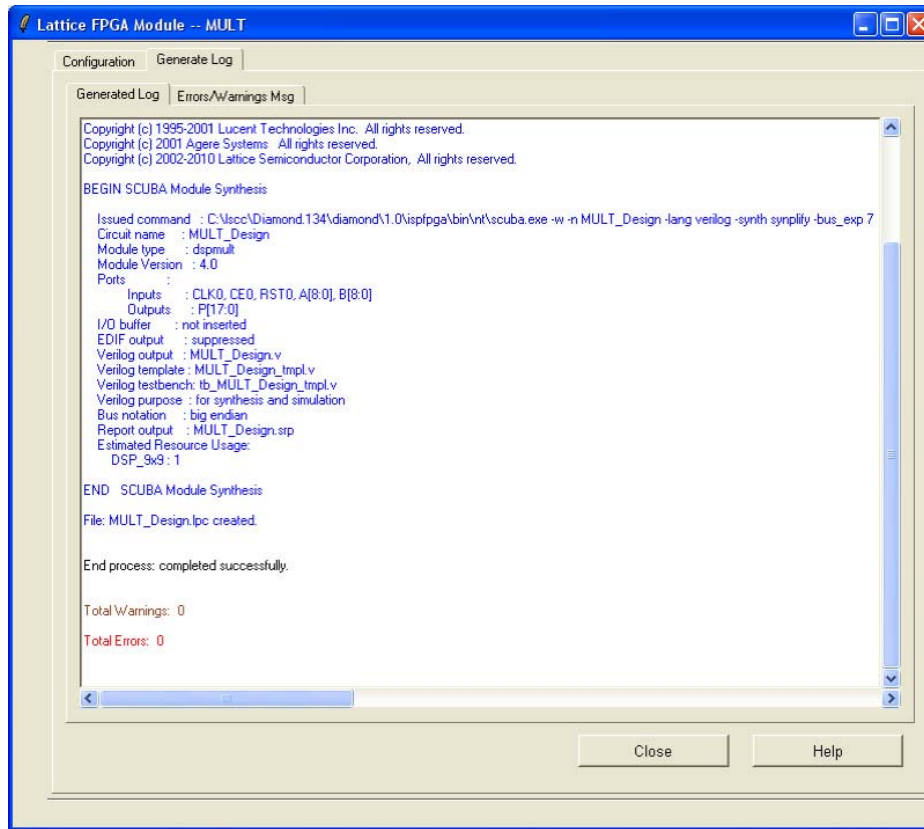


Figure 14-20. MULT Module



When you are finished selecting your options, Select **Generate**. A log window will appear similar to what is shown in Figure 14-21.

Figure 14-21. IP Generation Log Window



All other DSP Module interfaces for the LatticeECP3 are shown in the figures below.

Figure 14-22. MAC Module

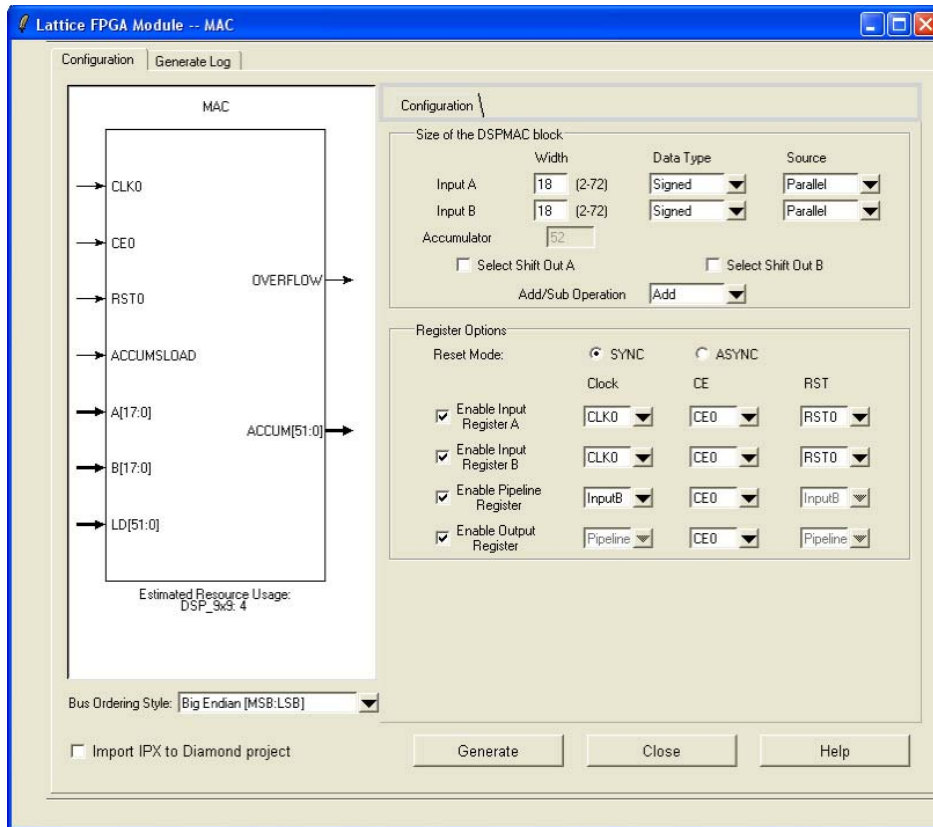


Figure 14-23. MMAC Module

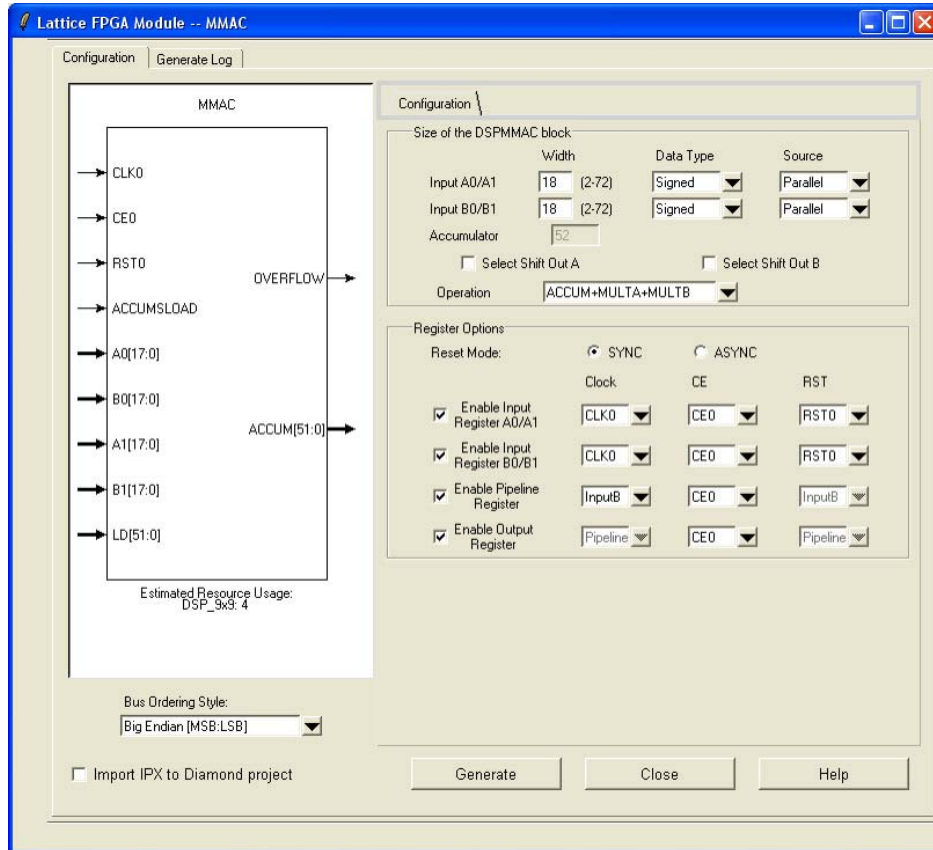


Figure 14-24. MULTADDSUB Module

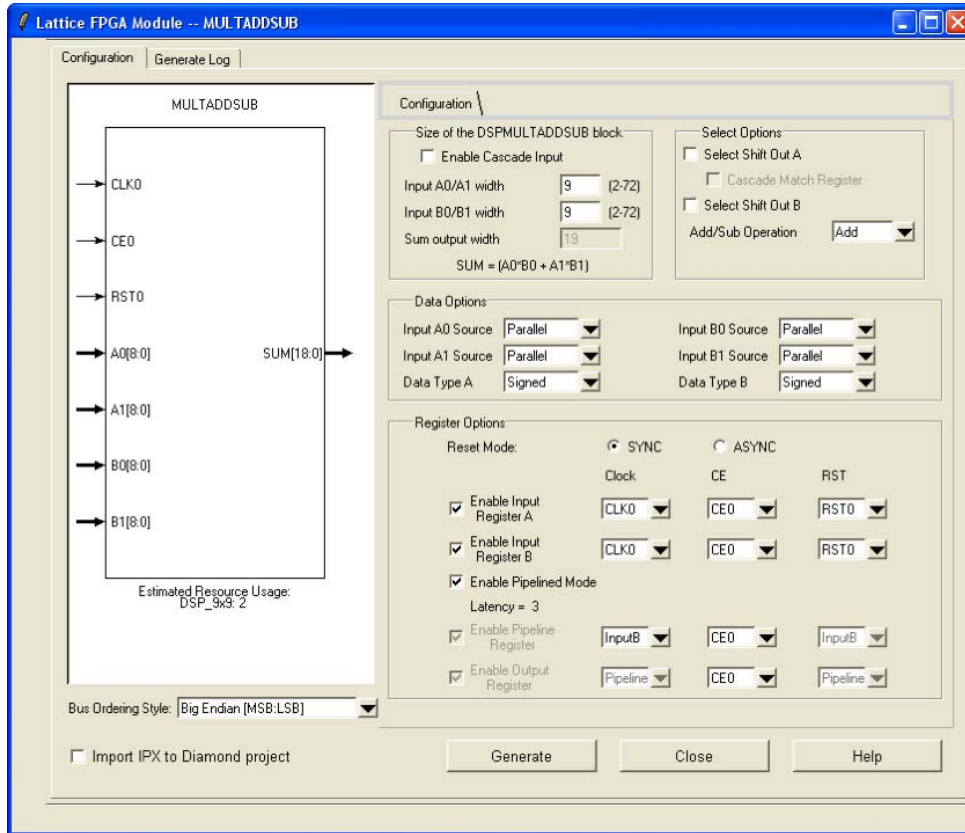


Figure 14-25. MULTADDSUBSUM Module

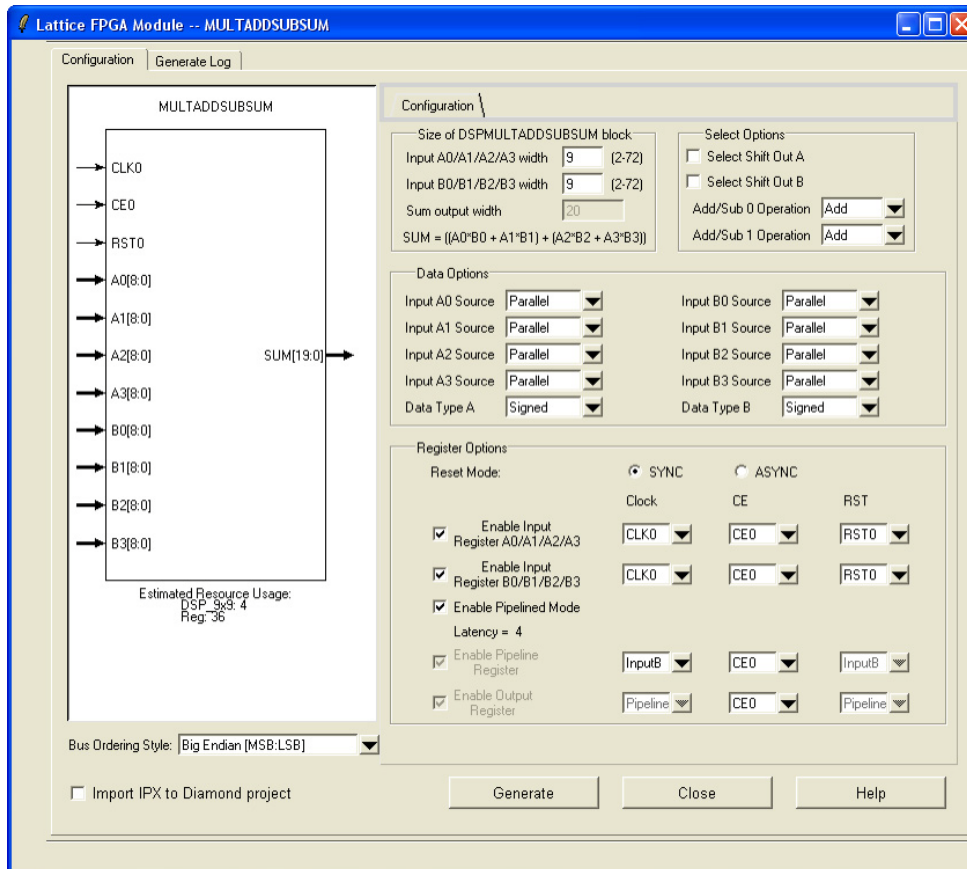


Figure 14-26. ADDER\_TREE Module

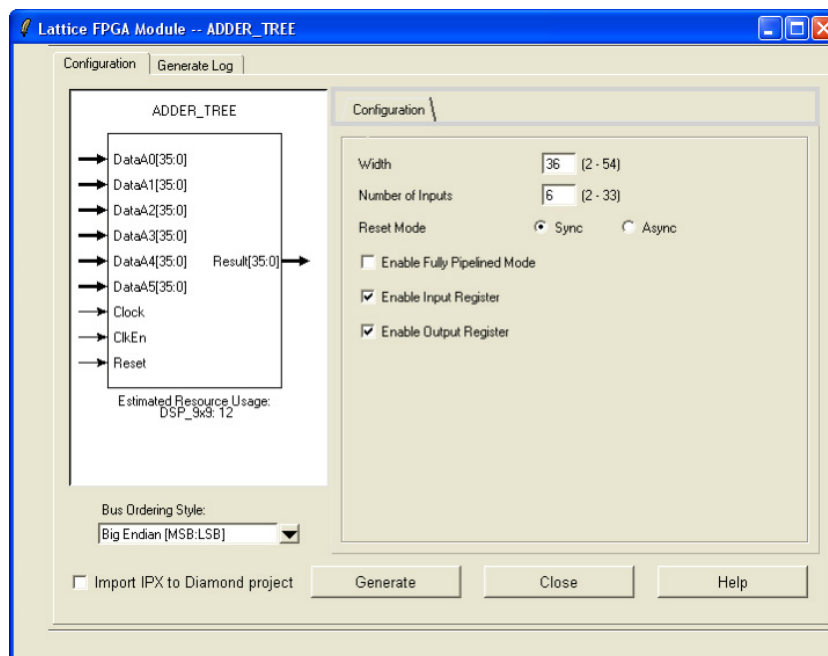


Figure 14-27. BARREL\_SHIFTER Module

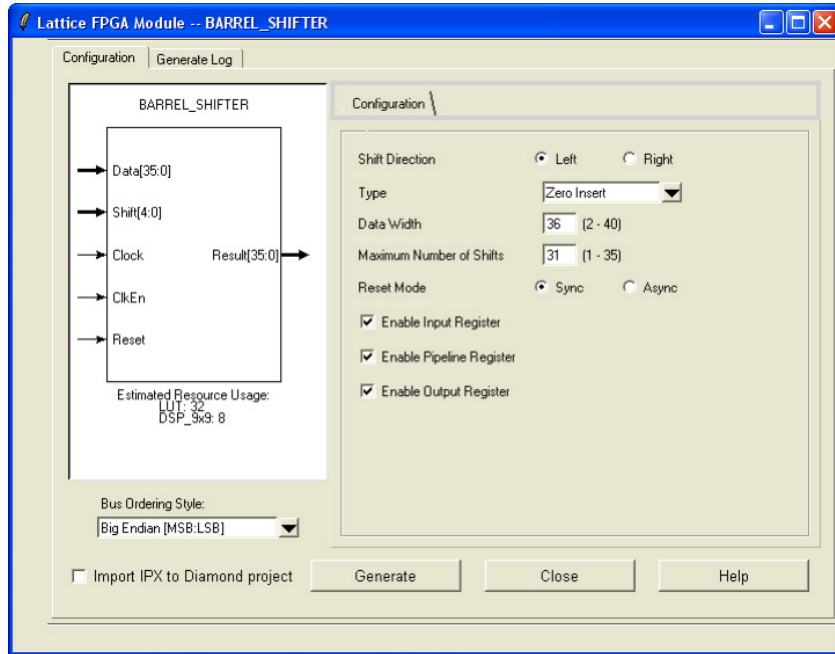


Figure 14-28. WIDE\_MUX Module

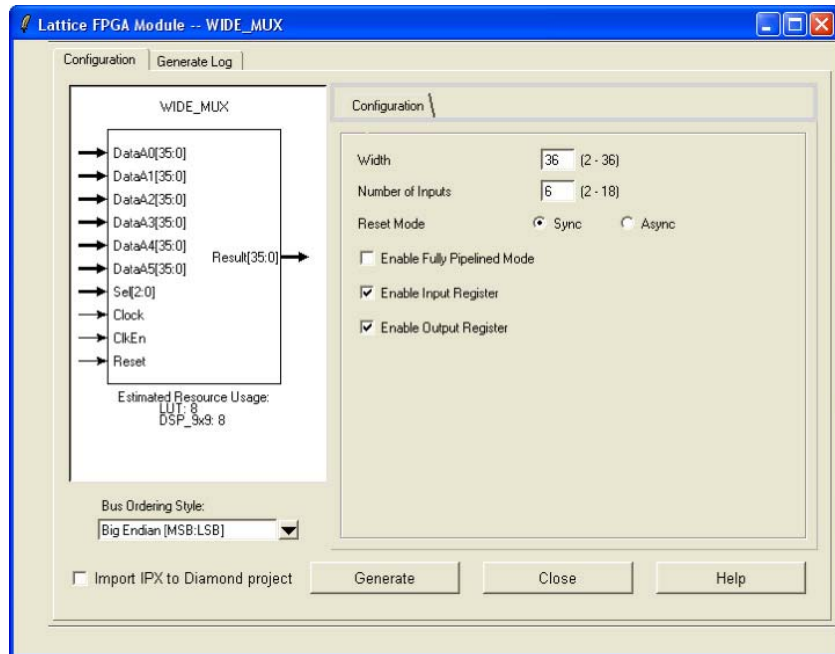


Figure 14-29. SLICE Module – Configuration

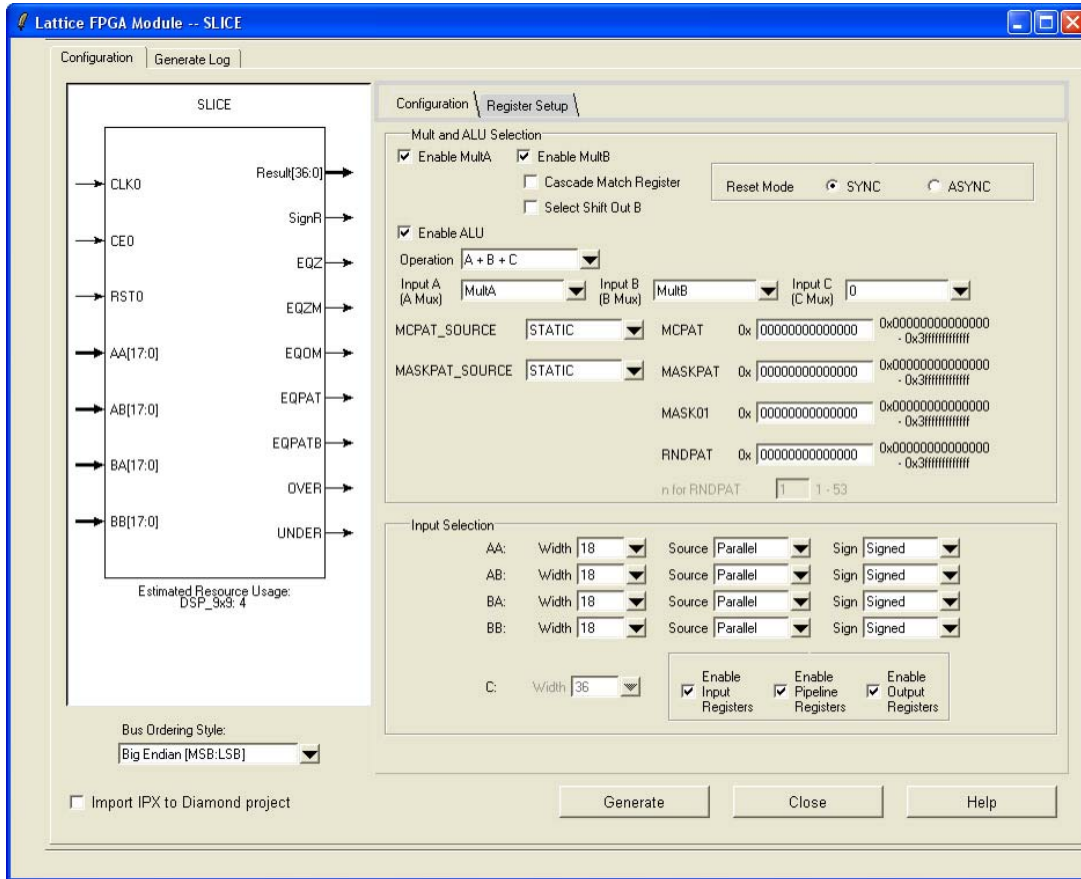
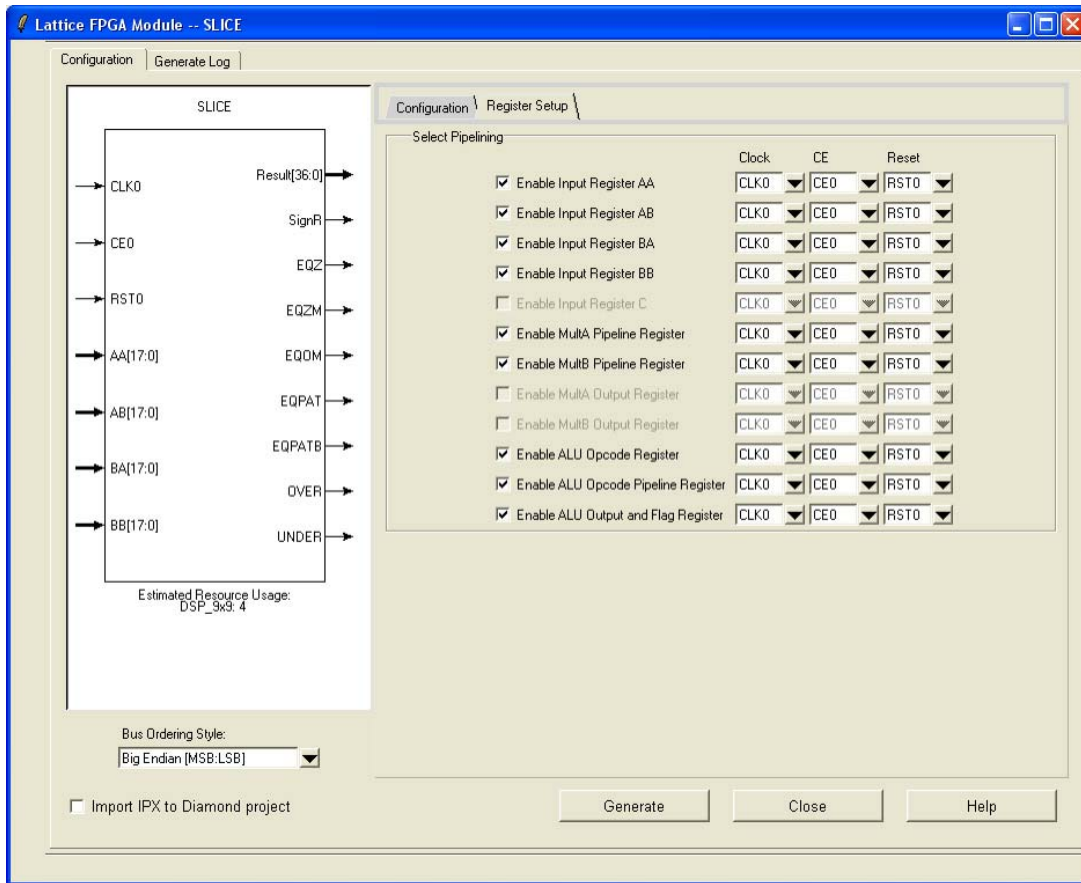




Figure 14-30. SLICE Module – Register Setup



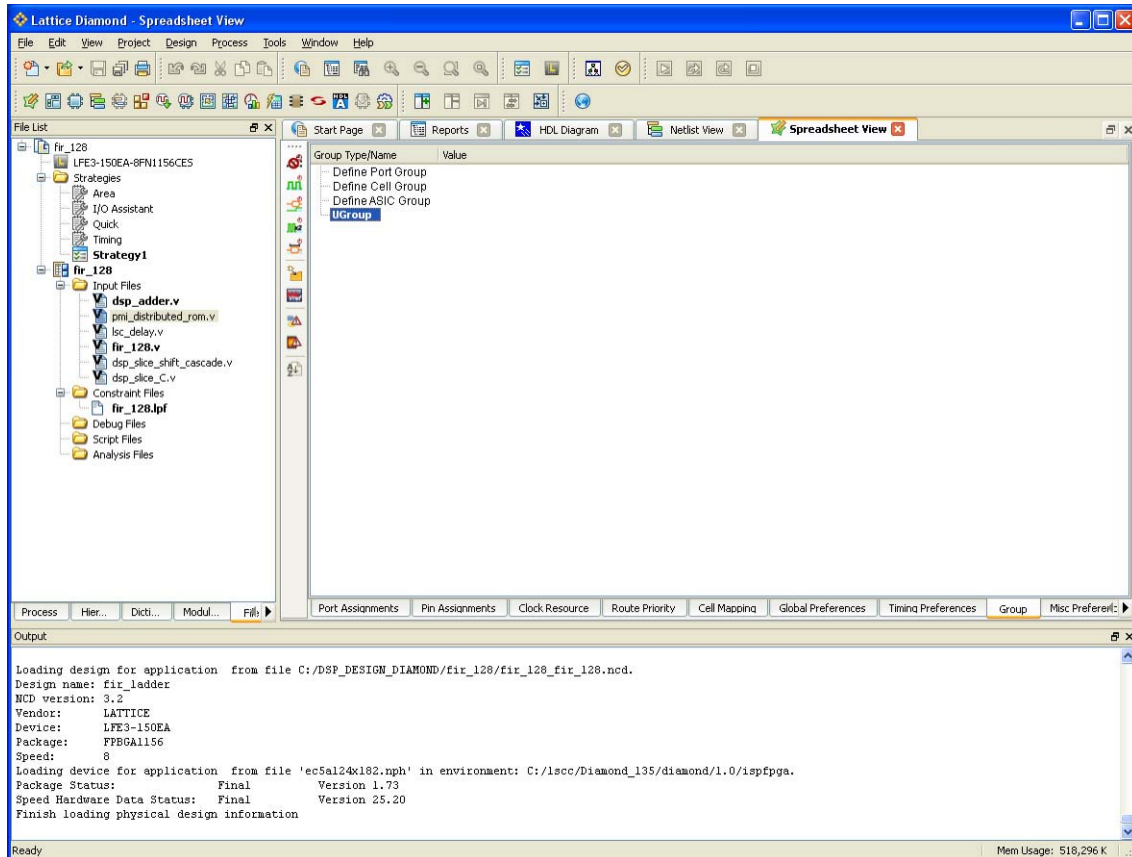
## sysDSP in Diamond

### Pre-Mapped View

#### Grouping Instances

sysDSP instances can be viewed or grouped together. In Diamond, select **Tools > Spreadsheet View**. Select the **Groups** tab. Right-click on **UGROUP** and select **New UGroup...**

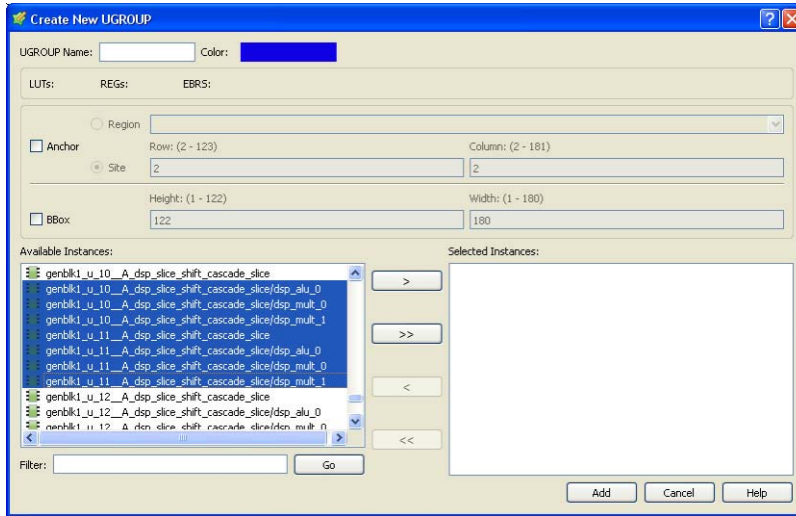
Figure 14-31. Setting UGroups



A Create New UGroup window appears as shown in Figure 14-32.

Left-click on an instance and select the right arrow (>) to add to the UGroup. To select multiple instances, hold the **Ctrl** or **Shift** key while selecting the instances with the left mouse button. Select the **Add** button to complete the UGroup.

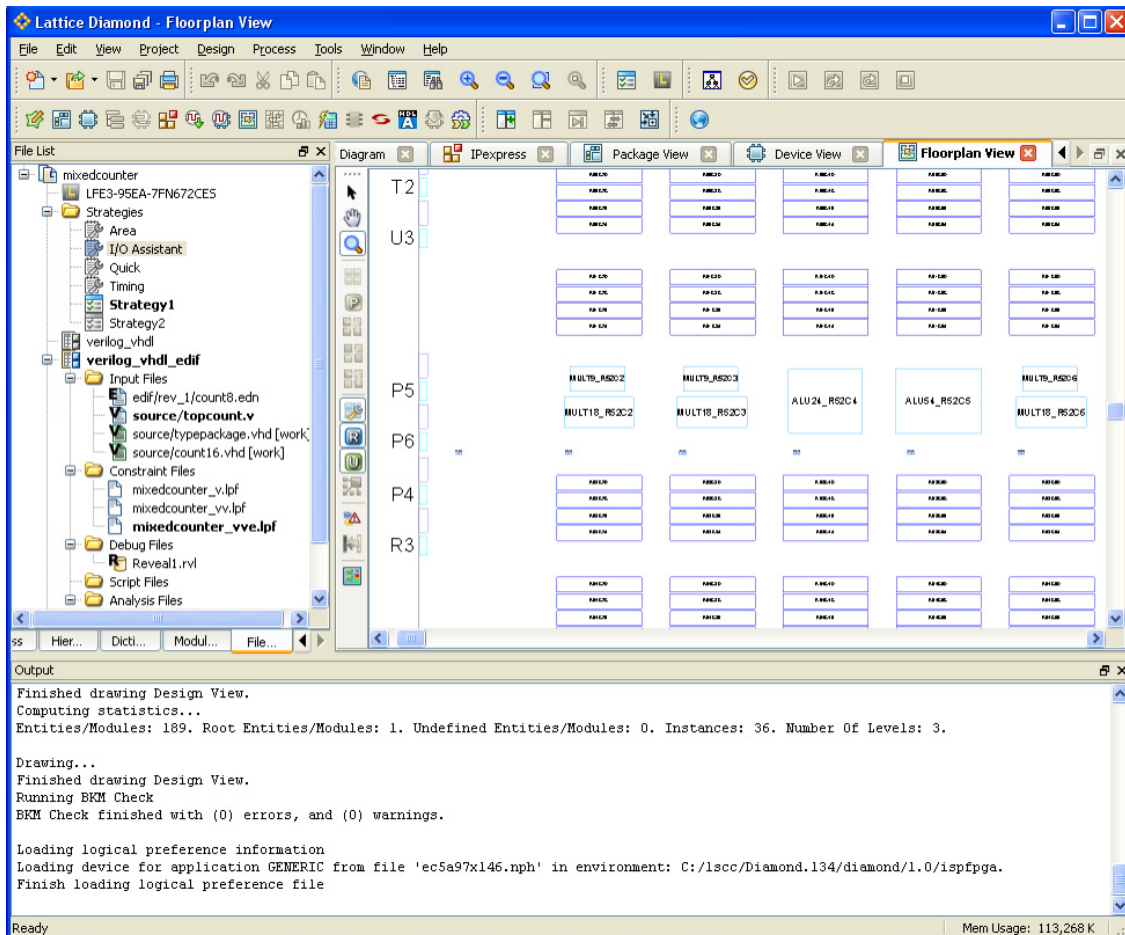
Figure 14-32. Selecting and Creating UGroups



### Floorplan View in Diamond

The DSP slices are organized in rows, as shown in Figure 14-33. It can be seen that each slice extends to four columns with the multipliers on the left and ALUs on the right.

Figure 14-33. Floorplan View in Diamond



## Introduction

Configuration is the process of loading or programming a design into volatile memory of an SRAM-based FPGA. This is accomplished via a bitstream file, representing the logical states, that is loaded into the FPGA internal configuration SRAM memory. The functional operation of the device after programming is determined by these internal configuration RAM settings. The SRAM cells must be loaded with configuration data each time the device powers up.

The configuration memory in LatticeECP3™ FPGAs is built using volatile SRAM; therefore, an external non-volatile configuration memory is required to maintain the configuration data when the power is removed. This non-volatile memory supplies the configuration data to the LatticeECP3 when it powers up or anytime the device needs to be updated.

To support multiple configuration options the LatticeECP3 supports the Lattice sysCONFIG™ interface as well as the dedicated JTAG port. The available configuration options, or modes, are listed in Table 15-1.

**Table 15-1. Supported Configuration Modes**

Interface	Port	Description
sysCONFIG	SPI	Serial Peripheral Interface to single or multiple FPGA devices.
	SPIm	Serial Peripheral Interface to single Flash memory devices with partitioned memory.
	SSPI	Configure and readback by standard SPI Master driver or devices.
	SCM	Slave Serial Mode for daisy chain configuration.
	SPCM	Slave 8-bit parallel CPU-like programming interface.
JTAG	JTAG (IEEE 1149.1 and IEEE 1532 compliant)	Standard 4-pin JTAG interface.

This technical note covers all of the configuration options available for LatticeECP3.

The LatticeECP3 configuration RAM can be loaded in a number of different modes. In these configuration modes, the FPGA can act as a master, a peripheral to a CPU, or a slave of other system devices. It also supports in-system configuration via the JTAG port.

The decision about which configuration mode to use is a system design concern. There are many methods for configuring the FPGA utilizing four basic schemes.

- **Master:** As a master, the FPGA is the source of the clock, and accesses an external PROM or EPROM storage device through a serial connection, with no additional timing or control signals used. This scheme includes Serial Programming Interface (SPI) that supports a seamless connection for programming using industry-standard external Flash-based memory devices.
- **Slave:** In slave mode the FPGA receives bit-serial or byte-wide data and a clock from an external data and timing source, either from a microprocessor, or from the lead device in an FPGA-daisy chain. As a slave device, the clock used to configure the FPGA is generated externally and provided to the CCLK input.
- **JTAG:** The device can be configured through the JTAG port. The JTAG port is always on and available regardless of the configuration mode selected.

The system designer should determine the requirements for configuration very early in the design. Many factors must be considered when deciding which configuration mode is best suited for the design. The flexible features for configuration can provide a seamless design to the system.

## General Configuration Flow

The LatticeECP3 enters Configuration mode when one of three things happens: power is applied to the device, the PROGRAMN pin is driven low, or a JTAG or SSPI Refresh instruction is issued. Upon entering Configuration mode the INITN pin and the DONE pin are driven low to indicate that the device is initializing (i.e. getting ready to receive configuration data).

Once initialization is complete, the INITN pin will be driven high. The low-to-high transition of the INITN pin causes the CFG pins to be sampled, telling the LatticeECP3 which port it will configure from. The LatticeECP3 then begins reading data from the selected port and starts looking for the preamble header (BDB3 hex for unencrypted bitstreams, and BFB3 for encrypted bitstreams). All data after the preamble is valid configuration data.

When the LatticeECP3 has finished reading all of the configuration data, and assuming there have been no errors, the DONE pin goes high and the LatticeECP3 enters user mode. In other words, the device begins to function according to your design.

Note that the LatticeECP3 may also be programmed via JTAG. When programming via JTAG, the INITN and DONE signals have no meaning, because JTAG, per the IEEE standard, takes complete control of all device I/Os. It is recommended that the PROGRAMN input be held high when using the JTAG port to configure the FPGA. This prevents the FPGA SRAM memory from being cleared when the JTAG programming cycle is complete.

The following sections define each configuration pin, each configuration mode, and all of the configuration options for the LatticeECP3.

## Configuration Pins

The LatticeECP3 supports two types of configuration pins, dedicated and dual-purpose. The dedicated pins are used exclusively for configuration; the dual-purpose pins, when configuration is complete, are available as extra I/O pins. If a dual-purpose pin is to be used both for configuration and as a general purpose I/O (GPIO) the following conditions must be met:

- The I/O type must remain the same. For example, if the pin is a 3.3V CMOS pin (LVCMOS33) during configuration, it must remain a 3.3V CMOS pin as a GPIO.
- You must select the correct CONFIG\_MODE setting and set the PERSISTENT attribute to OFF. Doing so permits the dual-purpose sysCONFIG pins to be used as GPIO when configuration completes. These settings can be found in the ispLEVER® Design Planner or Spreadsheet View in Lattice Diamond™ design software. See Table 15-3 for more information.
- You are responsible for insuring that no internal or external logic will interfere with the control signals required by configuration mode you have selected.

The dual-purpose configuration pins are controlled using HDL source file attributes, or with the ispLEVER or Diamond Preference Editor. You can read about how to apply HDL preferences in TN1177, [LatticeECP3 sysIO Usage Guide](#).

The LatticeECP3 also supports JTAG for configuration, transparent read back, and JTAG testing. The following sections describe the function of the various sysCONFIG and JTAG pins. Table 15-2 is provided for reference.

**Table 15-2. LatticeECP3 Configuration Pins**

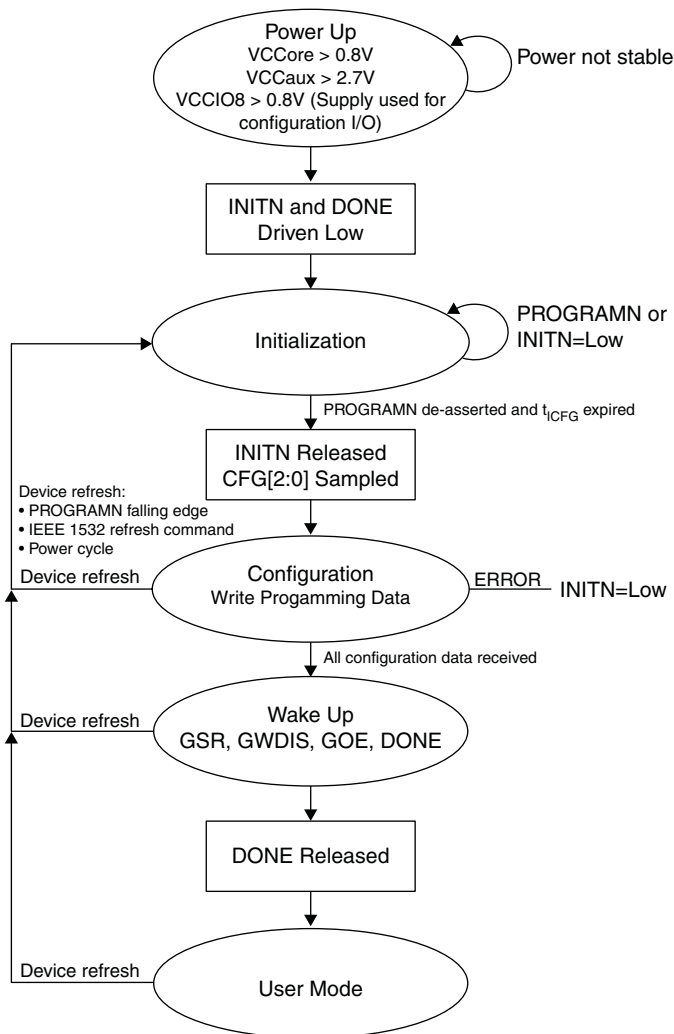
Pin Name	I/O Type	Pin Type	Configuration Mode					
			SPI	SPI <sub>m</sub>	SSPI <sup>1</sup>	SCM <sup>1</sup>	SPCM <sup>1</sup>	JTAG
CFG[2:0]	Input, weak pull-up	Dedicated	=000	=010	=001	=101	=111	
PROGRAMN	Input, weak pull-up	Dedicated	ALL					
INITN	Bi-directional open drain <sup>5</sup>	Dedicated	ALL					
DONE	Bi-directional open drain <sup>5</sup>	Dedicated	ALL					
CCLK	Input	Dedicated	Slave mode, determined by the CFG0 setting =1					
MCLK	Bi-directional, weak pull-up	Dual-Purpose	Master mode, determined by the CFG0 setting =0					
D0/SPIFASTN <sup>2</sup>	Bi-directional <sup>2</sup>	Dual-Purpose	SPIFASTN	SPIFASTN			D0	
D1 <sup>2,3</sup>	Bi-directional <sup>2</sup>	Dual-Purpose					D1	
D2 <sup>2,3</sup>	Bi-Directional <sup>2</sup>	Dual-Purpose					D2	
D3/SI <sup>2,3</sup>	Bi-directional <sup>2</sup>	Dual-Purpose			SI		D3	
D4/SO <sup>2,3</sup>	Bi-directional <sup>2</sup>	Dual-Purpose			SO		D4	
D5 <sup>2</sup>	Bi-directional <sup>2</sup>	Dual-Purpose					D5	
D6 <sup>2,3</sup>	Bi-directional <sup>2</sup>	Dual-Purpose					D6	
D7/SPID0 <sup>2,3</sup>	Bi-directional <sup>2</sup>	Dual-Purpose	SPID0	SPID0	Note 4		D7	
CSN/SN	Bi-directional, weak pull-up	Dual-Purpose			SN		CSN	
CS1N/HOLDN	Bi-directional, weak pull-up	Dual-Purpose			HOLDN <sup>3</sup>		CS1N	
WRITEN	Active low control input pin	Dual-Purpose					WRITEN	
BUSY/SISPI	Bi-directional, weak pull-up	Dual-Purpose	SISPI	SSIPI	Note 4		BUSY	
DI/CSSPI0N	Bi-directional, weak pull-up <sup>6</sup>	Dual-Purpose	CSSPI0N	CSSPI0N	Note 4	DI		
DOUT/CSON	Bi-directional, weak pull-up	Dual-Purpose	DOUT		DOUT	DOUT	DOUT/ CSON	

1. SSPI = Slave SPI, SCM = Serial Configuration Mode, SPCM = Slave Parallel Configuration Mode.
2. D[0:7] pins have no pull-up during power-up and configuration in all programming modes. This allows you to use large pull-up or pull-down resistors to pre-set those pins to a certain state while powering up your systems.
3. Weak pull-ups consist of a current source of 30µA to 150µA. The pull-ups for sysCONFIG dedicated and dual-purpose pins track VCCIO8. The pull-ups for TDI, TDO, and TMS track VCCJ.
4. This pin is used for programming the SPI Flash boot PROM.
5. Optional weak pull-up resistor available.
6. Requires external pull-up to VCCIO8.

## Configuration Process and Flow

Prior to becoming operational, the FPGA goes through a sequence of states, including initialization, configuration and wake-up.

**Figure 15-1. Configuration Flow**



PROGRAMN must not be asserted low until after all power rails have reached stable operation.

PROGRAMN must not make a falling edge transition during the time the FPGA is in the Initialization state. PROGRAMN must be asserted for a minimum low period of  $t_{PRGM}$  in order for it to be recognized by the FPGA. Failure to meet this requirement can cause the device to become non-operational, requiring power to be cycled.

## Power-up Sequence

In order for the LatticeECP3 to operate, power must be applied to the device. During a short period of time, as the voltages applied to the system rise, the FPGA will have an indeterminate state. Other devices in the system will also be in an indeterminate state.

As power continues to ramp, a Power On Reset (POR) circuit inside the FPGA becomes active. The POR circuit, once active, makes sure the external I/O pins are in a high-impedance state. It also monitors the  $V_{CCcore}$ ,  $V_{CCaux}$ , and the  $V_{CCI08}$  input rails. The POR circuit waits for the following conditions:

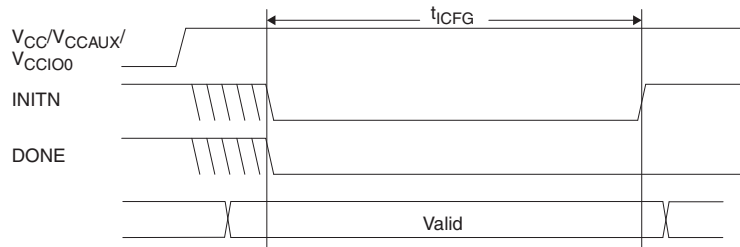
- $V_{CCcore} > 0.8V$
- $V_{CCaux} > 2.7V$
- $V_{CCI08} > 0.8V$  (Supply used for configuration I/O)



When these conditions are met the POR circuit releases an internal reset strobe, allowing the device to begin its initialization process. The FPGA samples the CFG[2:0] input pins to determine if a master or a slave mode configuration is selected. The FPGA uses this information to determine the  $t_{ICFG}$  initialization period. The next step is to assert INITN active low, and to drive DONE low. When INITN and DONE are asserted low the device moves to the initialization state, as shown in Figure 15-1.

The PROGRAMN input must not be asserted low as power is applied to the FPGA. Nor should it be allowed to transition from high to low at any time that INITN is in the initialization state.

**Figure 15-2. Configuration from Power-On-Reset Timing**



## Initialization

The LatticeECP3 enters the memory initialization phase immediately after the Power On Reset circuit drives the INITN and DONE status pins low. The purpose of the initialization state is to clear all of the SRAM memory inside the FPGA.

The FPGA remains in the initialization state until all of the following conditions are met:

- The  $t_{ICFG}$  time period has elapsed
- The PROGRAMN pin is deasserted
- The INITN pin is no longer asserted low by an external master

The dedicated INITN pin provides two functions during the initialization phase. The first is to indicate the FPGA is currently clearing its configuration SRAM. The second is to act as an input preventing the transition from the initialization state to the configuration state.

During the  $t_{ICFG}$  time period the FPGA is clearing the configuration SRAM. When the LatticeECP3 is part of a chain of devices each device will have different  $t_{ICFG}$  initialization times. The FPGA with the slowest  $t_{ICFG}$  parameter can prevent other devices in the chain from starting to configure. Premature release of the INITN in a multi-device chain may cause configuration of one or more chained devices to fail to configure intermittently.

The active-low, open-drain initialization signal INITN must be pulled high by an external resistor when initialization is complete. To synchronize the configuration of multiple FPGAs, one or more INITN pins should be wire-ANDed. If one or more FPGAs or an external device holds INITN low, the FPGA remains in the initialization state.

## Loading the Configuration Memory

The rising edge of the INITN pin causes the FPGA to enter the configuration state. The FPGA is able to accept the configuration bitstream created by the ispLEVER and Diamond development tools.

If the FPGA CFG[2:0] input pins have placed it in a master configuration mode it will begin fetching data from an external non-volatile memory.

If the FPGA CFG[2:0] input pins have placed it in a slave configuration mode, the FPGA waits for configuration data to be presented to it on each CCLK rising edge.



During the time the FPGA receives its configuration data the INITN control pin takes on its final function. INITN is used to indicate an error exists in the configuration data. When INITN is active high configuration is proceeding without issue. If INITN is asserted low, an error has occurred and the FPGA will not operate.

## Wake-up

Wake-up is the transition from configuration mode to functional mode. Wake-up starts when the device has correctly received all of its configuration data. When all configuration data is received, the FPGA asserts an internal DONE strobe. The assertion of the internal DONE causes a Wake Up state machine to run that sequences four controls. The four control strobes are:

- External DONE
- Global Write Disable (GWDISn)
- Global Output Enable (GOE)
- Global Set/Reset (GSR)

External DONE is a bi-directional, open-drain I/O. The FPGA releases the open-drain DONE pin at the programmed wake-up phase. If an external agent is holding the external DONE pin low, the wake-up process of the LatticeECP3 does not proceed. Only after the external DONE is active high do the final wake-up phases complete. Once the final wake-up phases are complete, the FPGA enters user mode.

The Global Output Enable, when it is asserted, permits the FPGA's I/O to exit a high-impedance state and take on their programmed output function. The FPGA inputs are always active. However, they are typically prevented from performing any action on the FPGA logic by the assertion of the Global Set/Reset (GSR).

The Global Set/Reset is an internal strobe that, when asserted, causes all I/O/LUT flip-flops, distributed RAM output flip-flops, and Embedded Block RAM output flip-flops that have the GSR enabled attribute to be set/cleared per their HDL definition.

The Global Write Disable is a control that overrides the write enable strobe for all RAM logic inside the FPGA. The inputs on the FPGA are always active, as mentioned in the Global Output Enable section. Keeping GWDIS asserted prevents accidental corruption of the instantiated RAM resources inside the FPGA.

## Clearing the Configuration Memory and Re-initialization

Several methods are available to clear the internal configuration memory of the LatticeECP3 device. The first is mentioned earlier when the device powers up (see the [“Power-up Sequence”](#) section of this document). A second method is to toggle the PROGRAMN pin. Also, JTAG can reinitialize the memory through an ISC Refresh command. SSPI can also initiate a reconfiguration with the Refresh command.

The other methods available are:

- Assertion of the PROGRAMn dedicated input
- Sending the ISC Refresh command using a configuration port (JTAG, or Slave SPI)

Invoking one of these methods causes the LatticeECP3 to drive INITN and DONE low. The FPGA enters the initialization state described above.

## Reconfiguration Priority

There are many sources that can initiate a reconfiguration while a configuration is already in process. There is a priority as to which of these sources can interrupt the configuration process depending on which of the sources initiated the original configuration. Note that if an interruption occurs, the reconfiguration will occur without informing the original configuration source that the configuration did not complete. JTAG always has the highest priority and any JTAG initiated configuration event will cause a reconfiguration to occur. Toggling the PROGRAMN pin has the next highest priority. It will interrupt any current configuration other than a JTAG configuration.

## FPGA Configuration Control Pin Definitions

The LatticeECP3 FPGA provides a set of I/O pins that can be used to load a configuration bitstream into the device. Some of these I/O are single purpose and are always available to perform configuration operations. Those configuration pins that are not dedicated can be configured for your use after the FPGA has entered user mode. This section describes what each I/O is, how it functions, and how to reclaim some for your own use.

### Configuration Pin Management

The dual-purpose sysCONFIG pins described in the Table 15-2 are dedicated configuration pins during the device configuration process. The PERSISTENT attribute is used to determine whether the dual-purpose sysCONFIG pins remain sysCONFIG pins during normal operation. The LatticeECP3 provides three settings for the PERSISTENT feature. The available options are shown in Table 15-3.

**Table 15-3. PERSISTENT Setting and Affected Pins**

Persistent Setting	Pins
OFF	All dual-purpose configuration pins are available as GPIO
SLAVE_PARALLEL	D [0:7], CSN, CS1N, WRITEN, BUSY, CSON, MCLK <sup>1</sup>
SSPI	SI, SO, SN, HOLDN, SISPI, SPID0, SPID1, CSSPIN and CSSPI1N

1. These pins are not used by the Slave Parallel logic, but they are reserved by the Slave Parallel logic. They are not available for use as GPIO.

You can use the SLAVE\_PARALLEL or the Slave SPI configuration port to access some of the resources connected to the FPGA. Accessing the FPGA resources requires special command sequences, which are described in other documents.

### Dedicated Control Pins

The LatticeECP3 makes a set of dedicated control pins available to allow you to select the way you want to configure the FPGA. The following sub-sections describe the LatticeECP3 dedicated sysCONFIG pins. These pins are powered by  $V_{CCIO8}$ .

While the device is under IEEE 1149.1 or 1532 JTAG control the dedicated programming pins have no meaning. This is because a boundary scan cell will control each pin, per JTAG 1149.1, rather than normal internal logic.

### JTAG Pins

The JTAG pins are standard IEEE 1149.1 TAP (Test Access Port) pins. The JTAG pins are dedicated pins and are always accessible when the LatticeECP3 device is powered up. While the device is under 1149.1 or 1532 JTAG control the dedicated programming pins INITN, DONE, and CCLK have no meaning. This is because a boundary scan cell will control each pin, per the IEEE standard, rather than normal internal logic. If the device is being accessed by JTAG then PROGRAMN will still be an active input even in JTAG mode.

These pins are powered by  $V_{CCJ}$ .

#### TDO

The Test Data Output pin is used to shift out serial test instructions and data. When TDO is not being driven by the internal circuitry, the pin will be in a high impedance state. This pin should be wired to TDO of the JTAG connector, or to TDI of a downstream device in a JTAG chain. An internal pull-up resistor on the TDO pin is provided. The internal resistor is pulled up to  $V_{CCJ}$ .

#### TDI

The Test Data Input pin is used to shift in serial test instructions and data. This pin should be wired to TDI of the JTAG connector, or to TDO of an upstream device in a JTAG chain. An internal pull-up resistor on the TDI pin is provided. The internal resistor is pulled up to  $V_{CCJ}$ .

**TMS**

The Test Mode Select pin controls test operations on the TAP controller. On the falling edge of TCK, depending on the state of TMS, a transition will be made in the TAP controller state machine. An internal pull-up resistor on the TMS pin is provided. The internal resistor is pulled up to  $V_{CCJ}$ .

**TCK**

The test clock pin, TCK, provides the clock to run the TAP controller state machine, which loads and unloads the JTAG data and instruction registers. TCK can be stopped in either the high or low state and can be clocked at frequencies up to that indicated in the [LatticeECP3 Family Data Sheet](#). The TCK pin supports hysteresis; the typical hysteresis is approximately 100mV when  $V_{CCJ} = 3.3V$ . The TCK pin does not have a pull-up. A pull-down resistor between TCK and ground on the PCB of 4.7 K is recommended to avoid inadvertent clocking of the TAP controller as  $V_{CC}$  ramps up.

**Optional TRST**

Test Reset, TRST, is not supported on the LatticeECP3.

 **$V_{CCJ}$** 

Having a separate JTAG  $V_{CC}$  ( $V_{CCJ}$ ) pin lets you apply a voltage level to the JTAG port that is independent from the rest of the device. Valid voltage levels are 3.3V, 2.5V, 1.8V, 1.5V, and 1.2V, but the voltage used must match the other voltages in the JTAG chain.  $V_{CCJ}$  must be connected even if JTAG is not used.

Please see [In-System Programming Design Guidelines for ispJTAG Devices](#) for further JTAG chain information.

**Configuration and JTAG Pin Physical Description**

All of the sysCONFIG dedicated and dual-purpose pins are part of Bank 8. Bank 8  $V_{CCIO}$  determines the output voltage level of these pins, input thresholds are determined by the I/O Type selected in the ispLEVER Design Planner (default is 3.3V LVCMOS) and Diamond Spreadsheet View.

JTAG voltage levels and thresholds are determined by the  $V_{CCJ}$  pin, allowing the LatticeECP3 to accommodate JTAG chain voltages from 1.2V to 3.3V.

**CFG[2:0]**

The Configuration Mode pins, CFG[2:0], are used to inform the FPGA how you want to configure the device. The actions performed by the remaining configuration pins depend on the state of the CFG[2:0] inputs. The CFG[2:0] input pins have weak internal pull-up resistors, that guarantee a valid configuration mode is selected should they be left unconnected. Lattice recommends the CFG pins be connected with independent pull-up/pull-down resistors. It is also recommended that these pins not be directly connected to the power or ground planes.

The CFG[2:0] pins are sampled at two different points in the configuration process. The first sample point is when the Power-On Reset state machine starts up. The POR sample point determines if the FPGA to be configured in master or slave mode. The  $t_{CFG}$  time period changes based on this information.

The second time the CFG pins are sampled is at the rising edge of the INITN pin. This sample is used to make the final configuration selection. Table 15-4 describes the configuration mode that is active based on the CFG input pins. The state on the CFG pins at any other time is not important. The state pins can be changed freely, which may be useful for selecting a new configuration mode.

**Table 15-4. LatticeECP3 Configuration Pins<sup>1</sup>**

Configuration Mode	Clock	CFG [2]	CFG [1]	CFG [0]
SPI Master (Single)	MCLK	0	0	0
SPI Master (Multiple)		0	1	0
Slave SPI	CCLK	0	0	1
		X	X	X
SCM	CCLK	1	0	1
SPCM	CCLK	1	1	1

1. JTAG is always available for IEEE 1149.1 and 1532 support.

**PROGRAMN**

The PROGRAMn is a dedicated input that is used to configure the FPGA. The PROGRAMn pin is a falling edge sensitive, and has an internal weak pull-up. When a falling edge occurs, the FPGA exits user mode and starts a device configuration sequence at the Initialization phase, as described earlier.

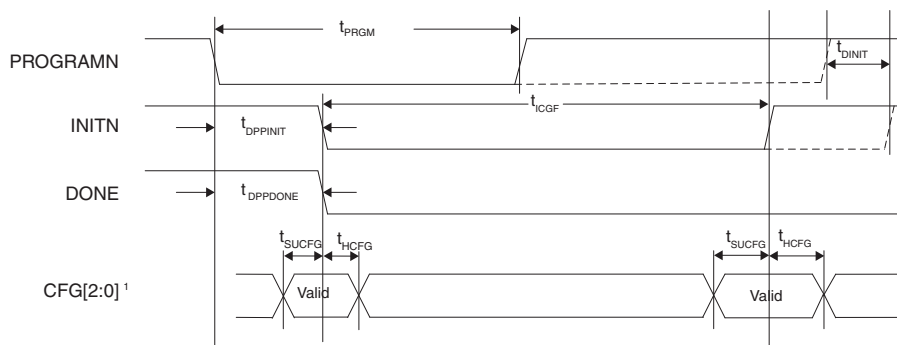
Proper operation of the LatticeECP3 FPGA depends on the PROGRAMn pin. The following conditions must be met:

- The PROGRAMn pin must not be asserted until after all of the supply rails are stable. This can be achieved by either placing an external pullup resistor and tying it to the VCCIO8 voltage, or permitting the FPGA's internal pull-up resistor to pull the input high.
- The PROGRAMn pin must make a high to low transition in order to cause the FPGA to enter configuration mode. This is not necessary when first powering the FPGA, as the FPGA will enter configuration mode after the internal Power On Reset circuit releases the internal reset.
- The PROGRAMn pin must not be allowed to transition from high to low at any time INITn is active (i.e. low) as a result of being in the Initialization state.
- PROGRAMn must meet the minimum active pulse width  $t_{PRGM}$ .
- PROGRAMn remains an active input even when the JTAG bus is being used to program the FPGA. The PROGRAMn pin should not be asserted during JTAG programming sequences.

Failing to follow these guidelines may prevent the FPGA from operating.

PROGRAMn must be de-asserted in order for the FPGA to exit the Initialization state.

**Figure 15-3. Configuration from PROGRAMN Timing**



1. The CFG pins are normally static (hard wired).

## INITN

The INITN pin is a bidirectional open-drain control pin. It has the following functions:

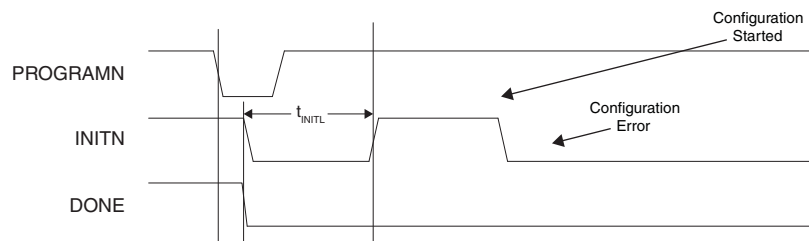
- After power is applied, or after a PROGRAMN assertion it goes low to indicate the FPGA configuration cells are being erased. The low time assertion is specified with the  $t_{ICFG}$  parameter.
- After the  $t_{ICFG}$  time period has elapsed the INITN pin is deasserted (i.e. is active high) to indicate the FPGA is ready for its configuration bits. In master mode the FPGA starts loading bits from an attached non-volatile memory. In slave mode the FPGA waits for the bits to arrive over the interface selected by the CFG[2:0] input pins. The rising edge of the INITN samples the CFG[2:0] inputs, allowing the FPGA to determine how it is to be configured.
- INITN can be asserted low by an external agent before the  $t_{ICFG}$  time period has elapsed in order to prevent the FPGA from reading configuration bits. This is useful when there are multiple FPGA's chained together. The FPGA with the longest  $t_{ICFG}$  time can hold all other FPGA's in the chain from starting to get data until it is ready itself.
- The last function provided by INITN is to signal an error during the time configuration data is being read. Once  $t_{ICFG}$  has elapsed, and the INITN pin has gone high, any INITN assertion signals the FPGA has detected an error during configuration.

The following conditions will cause INITN to become active, indicating the Initialization state is active:

- Power has just been applied
- PROGRAMN falling edge occurred
- The IEEE 1532 Refresh command has been sent using a slave configuration port (JTAG, SSPI, etc.).

If the INITN pin is asserted due to an error condition, the error can be cleared by correcting the configuration bit-stream and forcing the FPGA into the Initialization state.

**Figure 15-4. Configuration Error Notification**



## DONE

The DONE pin is a dedicated bi-directional open drain with a weak pull-up that signals the FPGA is in User mode. DONE is first able to indicate entry into User mode only after the internal DONE pin is asserted. The internal DONE signal defines the beginning of the FPGA Wake-Up state.

The DONE output pin is controlled by the DONE\_EX configuration parameter. The default state of this pin is OFF. The default mode causes the LatticeECP3 to start immediately after the internal DONE bit is asserted. The FPGA does not stall waiting for the DONE pin to be asserted high.

The FPGA can be held from entering User mode indefinitely by having an external agent keep the DONE pin asserted low. In order to use DONE to stall entering user mode the DONE\_EX configuration preference must be set ON. A common reason for keeping DONE driven low is to allow multiple FPGAs to be completely configured. As each FPGA reaches the DONE state, it is ready to begin operation. The last FPGA to configure can cause all FPGAs to start in unison.

It is critical that DONE be asserted low when the LatticeECP3 is in a chain of FPGAs. The LatticeECP3 continues to pass configuration clock pulses to FPGAs attached downstream as long as DONE is de-asserted. Any FPGA

permitted to assert DONE and enter User mode will only pass a few hundred more configuration clock cycles. Downstream FPGAs will never complete their configuration process if this occurs.

The DONE pin drives low in tandem with the INITN pin when the FPGA enters Initialization mode. As described earlier, this condition happens when power is applied, PROGRAMN is asserted, or an IEEE 1532 Refresh command is received via a slave configuration port.

Sampling the DONE pin is a good way for an external device to tell if the FPGA has finished configuration. However, when using IEEE 1532 JTAG to configure SRAM the DONE pin is driven by a boundary scan cell, so the state of the DONE pin has no meaning during IEEE 1532 JTAG configuration (once configuration is complete, DONE takes on the behavior defined by the DONE\_EX configuration parameter).

### **Configuration Clock (CCLK)**

The CCLK is a dedicated input-only whose purpose is to provide a reference clock for the FPGA when one of the slave configuration modes is selected by the CFG[2:0] inputs.

Please refer to the LatticeECP3 AC Timing information in the [LatticeECP3 Family Data Sheet](#) for information about maximum clock rates, and data setup and hold parameters.

When the LatticeECP3 is in a chain of FPGAs it is necessary to continue to drive CCLK until all of the FPGAs have received their configuration data. It is recommended the CCLK continue to be toggled until the DONE signal is active.

### **Dual-Purpose sysCONFIG Pins**

The Dual-Purpose sysCONFIG pins, depending on the configuration mode selected by the CFG[2:0] input pins, provide special configuration functions during the Configuration phase of the device wake-up process. The dual-purpose pins can be recovered for your use once the FPGA enters User mode. Successful recovery of the dual-purpose pins relies on following the guidelines described in the “[Configuration Pins](#)” section of this document.

The dual-purpose configuration pins are located in the same I/O bank as the dedicated configuration pins. The configuration pins in the LatticeECP3 are powered by the VCCIO8 voltage rail.

### **Master Clock (MCLK)**

The MCLK provides a reference clock for synchronous non-volatile memories attached to the FPGA. MCLK is only active when the FPGA CFG[2:0] inputs select a master configuration mode. See Table 15-4 for a full description of the available configuration modes selectable by the CFG[2:0] input pins. MCLK acts as a general purpose I/O if the FPGA is in a slave configuration mode.

The LatticeECP3 generates MCLK from an internal oscillator. The initial output frequency of the MCLK is approximately 2.5MHz. The MCLK frequency can be altered using the MCCLK\_FREQ parameter. You can select the MCCLK\_FREQ using the Diamond Spreadsheet View. For a complete list of the supported MCLK frequencies, see Table 15-8.

During the initial stages of device configuration the frequency value specified using MCCLK\_FREQ is loaded into the FPGA. Once the FPGA accepts the new MCLK\_FREQ value the MCLK output begins driving the selected frequency. Make certain that when selecting the MCLK\_FREQ that you do not exceed the frequency specification of your configuration memory, or of your PCB. Lattice recommends reviewing the LatticeECP3 AC specifications in the [LatticeECP3 Family Data Sheet](#) when making MCLK\_FREQ decisions.

The LatticeECP3 provides the ability to be configured from a bitstream that is encrypted. There are additional requirements on the MCCLK\_FREQ selection that you must adhere to when configuring the LatticeECP3 with an encrypted bitstream. These conditions are discussed in the “[Bitstream Encryption/Decryption Flow](#)” section of this document.

**DI/CSSPI0N**

The DI/CSSPI0N configuration pin provides one of two functions depending on the FPGA's configuration mode. When the FPGA is in Serial Configuration Mode the pin is set to DI (Data Input) mode. When the FPGA is in SPI Master or SPI Master Multiboot mode, the pin is set to CSSPI0N (Chip Select SPI 0).

When the FPGA is in Serial Configuration Mode the DI pin receives incoming configuration data. See the Serial Configuration Mode section of this document for more information.

When the FPGA is in SPI Master or SPI Master Multiboot mode the CSSPI0N is the chip select strobe to the attached SPI memory that contains the FPGA's configuration bits. The FPGA asserts this pin active low during the Configuration phase of the wake-up process.

An external pull-up resistor is required on CSSPI0N in SPI and SPIm modes of operation. Some SPI memory devices require the CSn input to rise in tandem with their input voltage. The internal pull-up on CSSPI0N does not become active until the FPGA determines all input voltage rails are stable. This does not meet the requirements of some SPI memory vendors.

**DOUT/CSON**

The DOUT/CSON configuration pin is used only when placing the LatticeECP3 into a chain of FPGAs requiring configuration.

The DOUT/CSON pin is an output from the LatticeECP3 and is only active when the FPGA is connected to another FPGA in a daisy chain.

When in a daisy chain, the pin may act as either a data output (DOUT) or a chip select (CSON). The behavior is described in detail in the Configuring Multiple FPGA Devices section of this document.

For serial and parallel configuration modes, when Bypass mode is selected, this pin becomes DOUT (see Figure 15-10). When the device is fully configured a Bypass instruction in the bitstream is executed and the data on DI, or D[0:7] in the case of a parallel configuration mode, will then be routed to the DOUT pin. This allows data to be passed, serially, to the next device. In a parallel configuration mode D0 will be shifted out first followed by D1, D2, and so on.

Daisy chaining the Parallel devices can be implemented with the Flowthrough attribute. This attribute allows the CSON pin to be driven when the done bit is set and configuration of the first part is complete. The CSON of the first part will drive the CSN of the second part.

You will find this attribute in the ispLEVER Generate Bitstream Data property under Chain Mode or in the Diamond Bitstream options window. See Appendix B for more information on setting these options in Diamond.

The DOUT/CSON drives out a high on power-up and will continue to do so until the execution of the Bypass instruction within the bitstream, or until the I/O Type is changed by your code.

**CSN/SN**

The CSN/SN is a bidirectional pin that is active in Slave Parallel Configuration mode, or in Slave SPI mode. The pin is a chip select that gates the incoming configuration data.

Detailed information about using the chip select pin can be found in the [“Slave Parallel Mode \(SPCM\)”](#) and [“Slave SPI \(SSPI\)”](#) configuration sections of this document.

**CS1N/HOLDN**

The CS1N/HOLDN configuration pin is active only in Slave Parallel Configuration mode or in Slave SPI mode.

When the LatticeECP3 is in a Slave Parallel Configuration mode the pin acts as a chip select that works in conjunction with CSN. Information detailing the interaction of these two pins is described in the Parallel Configuration mode section of this document.



The configuration pin takes on the HOLDN function when the LatticeECP3 is in Slave SPI Configuration mode. Assertion of the HOLDN input causes the FPGA to ignore the SPI SCLK. Details for using HOLDN are provided in the Slave SPI Configuration section.

When CSN or CS1N is high, the D[0:7], and BUSY pins are tri-stated. CSN and CS1N are interchangeable when controlling the D[0:7], and BUSY pins.

#### **WRITEN**

The WRITEN configuration pin is an input pin that is active in Slave Parallel Configuration mode. It is a write enable strobe that controls the direction data flows on the D[0:7] data bus pins. When WRITEN is asserted active low the FPGA D[0:7] pins are tri-stated to allow an external bus master to write data into the FPGA.

#### **BUSY/SISPI**

The BUSY/SISPI configuration pin is active in Slave Parallel Configuration mode and in SPI Master modes.

When the LatticeECP3 is in a Slave Parallel Configuration mode, the pin is a tri-state output pin. When the FPGA parallel bus is active due to the assertion of CSN/CS1N the BUSY pin indicates the FPGA's ability to accept a byte of configuration data. The FPGA is able to accept another configuration byte when this output is driven low.

When the LatticeECP3 is in SPI Master mode the pin is connected to the data input of the SPI PROM that contains the configuration data. SISPI is an output used by the LatticeECP3 to transmit commands to the SPI PROM.

#### **D[0]/SPIFASTN**

The D[0]/SPIFASTN configuration pin is available in Slave Parallel Configuration and SPI Master configuration modes.

In Slave Parallel Configuration mode the D[0] pin is the most-significant data bit in the combined D[0:7] parallel data bus.

In SPI Master configuration modes it becomes the SPIFASTN input. The input is sampled at the rising edge of the INITN output.

The SPIFASTN selects the Read Opcode transmitted to the SPI PROM. When SPIFASTN is deasserted (i.e. driven to  $V_{ih}$ ) the FPGA requests a Read Operation using the 03 hex Read Opcode. When SPIFASTN is asserted (i.e. driven to  $V_{il}$ ) the FPGA requests a Read Operation using the 0B hex Read Opcode. A SPI PROM that accepts the 0B Read Opcode is able to operate at higher serial clock rates. Consult the SPI memory vendor's data sheet for the exact capabilities of the SPI memory device.

Do not leave this input floating when a SPI Master mode is selected.

In parallel mode this pin is D[0] and operates in the same way as D[1:6] below. Taken together D[0:7] form the parallel data bus, D[0] is the most significant bit in the byte. The D[0:7] data bus are open-drain I/O without a pullup resistor during the time that power is applied to the FPGA. They also remain in this state until the FPGA is fully configured. When the FPGA is configured the D[0:7] I/O take on the attributes defined in your HDL source code, or using the Spreadsheet View preference editor.

As with D[1:6], if SRAM (configuration memory) needs to be accessed using the parallel pins while the part is in user mode (the DONE pin is high) then the PERSISTENT control cell must be set to preserve this pin as D[0]. Note that SRAM may only be read using JTAG or Slave Parallel mode.

#### **D[1], D[2], D[5] and D[6]**

These configuration pins are only available in Slave Parallel Configuration mode and are intermediary data bits for the parallel data bus made up of bits D[0:7].

Remember that D[0] is the most-significant data bit and D[7] is the least-significant.



**D[3]/SI**

The D[3]/SI configuration pin is only available in Slave Parallel Configuration or in Slave SPI Configuration mode.

When the LatticeECP3 is in Slave Parallel Configuration mode the pin acts as D[3].

In Slave SPI Configuration mode the pin acts as the Serial Input pin for data supplied by a SPI Master Controller.

**D[4]/SO**

The D[4]/SO configuration pin is only available in Slave Parallel Configuration or in Slave SPI Configuration mode.

When the LatticeECP3 is in Slave Parallel Configuration mode the pin acts as D[4].

In Slave SPI Configuration mode the pin acts as the Serial Output pin for data transmitted from the FPGA back to the SPI Master Controller.

**D[7]/SPID0**

The D[7]/SPID0 configuration pin is only available in Slave Parallel Configuration or in Master SPI Configuration mode.

When the LatticeECP3 is in Slave Parallel Configuration mode the pin acts as D[7]. This is the least-significant-bit of the D[0:7] data bus.

In Master SPI Configuration mode the pin acts as the SPI Data Input pin receiving data from an attached SPI PROM.

## Configuration Modes

LatticeECP3 devices support many different configuration modes, utilizing either serial or parallel data paths. The configuration method used by the LatticeECP3 is selected by driving the CFG[2:0] input pins. The CFG[2:0] input pins are sampled at the falling edge of INITN to determine if the part is in a master or a slave configuration mode. The pins are sampled a second time at the rising edge of INITN to determine the specific configuration mode.

The LatticeECP3 starts the configuration process in one of three ways:

- Initial application of power
- Assertion of the PROGRAMN input pin
- Reception of a REFRESH command from a configuration port (JTAG, Slave SPI, Slave Parallel)

## SPI Interface

The Serial Peripheral Interface (SPI) is a four-wire de facto bus standard used to transmit and receive serial data. The LatticeECP3 can use a SPI data bus to retrieve its configuration data from most SPI ROMs.

The amount of ROM storage required depends on the number of Look Up Tables (LUTs) in the LatticeECP3 device you have selected. Figure 15-5 shows how many bits of configuration data are required for each member of the LatticeECP3 family.

**Table 15-5. Maximum Configuration Bits – SPI Flash Mode Bitstream File<sup>1</sup>**

Device	Bitstream Size <sup>1</sup>	SPI Flash	Dual Boot SPI Flash	Units
ECP3-17	4.5	8	16	Mb
ECP3-35	8.2	16	32	Mb
ECP3-70	22.5	32	64	Mb
ECP3-95	22.5	32	64	Mb
ECP3-150	35.7	64	128	Mb

1. The Bitstream Size column is the maximum number of bits the FPGA may require. This number takes into account the pre-initialization of all Embedded Block RAMs.

The estimated configuration time can be calculated by dividing the bitstream size (in bits) from Table 15-5 by the configuration clock (MCLK or CCLK) frequency. The MCLK frequency is modified using the Global Preferences tab within the Diamond Spreadsheet View or in the ispLEVER Design Planner.

The LatticeECP3 provides the following three SPI configuration modes:

- SPI Master (SPI)
- SPI Master Multiboot (SPIm)
- Slave SPI (SSPI)

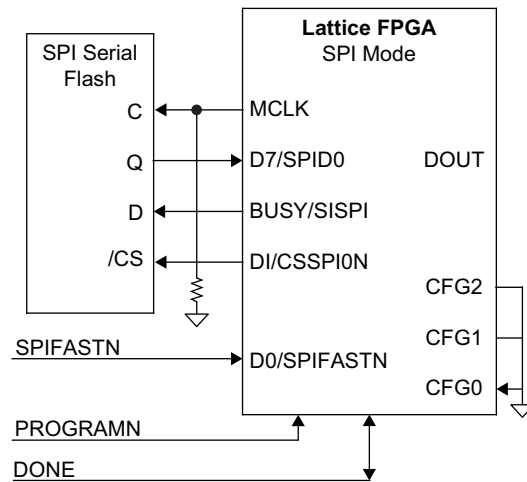
#### **SPI Master Mode**

The simplest SPI configuration consists of one SPI Serial Flash connected to one LatticeECP3, as shown in Figure 15-5. In this configuration the LatticeECP3 is the master of the SPI bus. The FPGA controls the chip select and the MCLK, and receives the configuration data on the SPID0 input.

During FPGA configuration the SISPI output sends a command sequence to reset the SPI PROM's internal address pointer. The SPIFASTN informs the FPGA which SPI Read Command to send to the SPI PROM. When the SPIFASTN input is driven high, the FPGA sends a 03 Hex Read Opcode. When it is asserted active low, the FPGA sends a 0B Hex Read Opcode to the SPI PROM.

As mentioned in the section describing the MCLK behavior, the configuration clock frequency can be altered. The MCLK frequency must not exceed the clock input frequency of the SPI PROM.

Figure 15-5. One FPGA, One SPI Serial Flash



Note: The board-level pull-down on MCLK should have a 1-3 Kohms resistance. This counteracts the weak internal pull-up on MCLK and prevents an unintentional rising edge at power-up.

In order to configure properly, the LatticeECP3 must transmit at least 128 clock pulses before it receives the preamble code (BDB3 hex for unencrypted bitstreams, and BFB3 for encrypted bitstreams). It is required that the data in the SPI PROM be padded to account for these extra clocks. The bitstream generation tool automatically adds the necessary padding information.

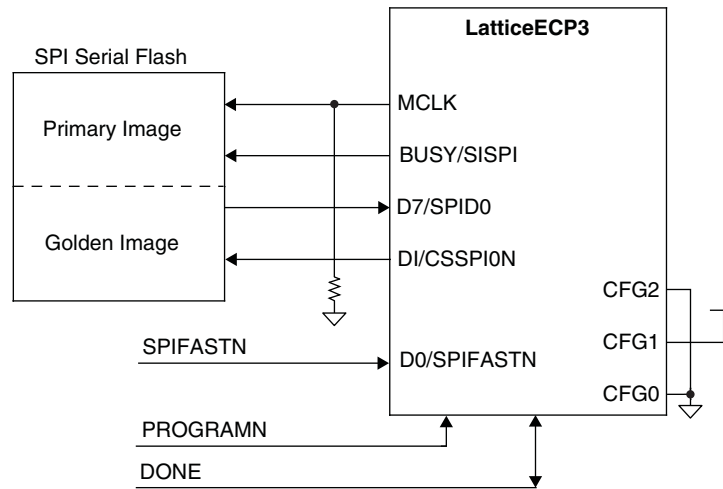
There are some general rules that user has to follow in order to configure properly:

- The POR of the SPI flash device should be lower than the POR of LatticeECP3 or the SPI flash must be powered first.
- SPI flash maximum frequency must be higher than LatticeECP3 MCLK frequency.
- Board routing requirements need to ensure proper timing. Please refer to ECP3 data sheet for detailed timing requirement.

### SPI Master Multiboot (SPIm) Mode

SPI Master Multiboot mode is an enhancement to the SPI Master boot mode. The SPI memory is attached to the FPGA in exactly the same way as SPI Master mode. SPIm enables you to partition the SPI PROM to store two configuration bitstreams. The FPGA will attempt to configure from the Primary image, and if the FPGA fails to configure from the primary image it tries to load a fail-safe Golden bitstream. Figure 15-6 shows the concept from a high level.

Figure 15-6. SPI Mode for Dual-Boot Capability



Internally, SPI mode adds logic to detect a configuration failure and the ability to reattempt configuration from a different address within the SPI Flash device. While SPI mode treats the SPI Flash device as a single block of storage starting at address zero, SPI mode allows segmentation of the Flash device for the golden bitstream.

In SPI mode, the primary bitstream is stored at address offset 0x010000. When configuring, the LatticeECP3 device automatically reads the data beginning at this address first. If after  $2^{14}$  clocks the device still does not receive the pre-amble code or a bitstream error is encountered after receiving the pre-amble code, the configuration logic resets and loads the data located at address offset 0xFFFF00.

The LatticeECP3 uses a 24-bit addressing scheme to access the SPI memory array. The amount of storage remaining in the SPI starting at address 0xFFFF00 is only 256 bytes. This is not enough to store a complete LatticeECP3 configuration bitstream. The LatticeECP3 configuration bitstream is stored elsewhere in the SPI PROM. The data retrieved by the FPGA at address 0xFFFF00 is an instruction pointing to the start of the failsafe configuration data.

An example of the SPI Flash memory organization for SPI mode is shown in Table 15-6.

**Table 15-6. SPI Mode Data Map<sup>1, 2, 3</sup>**

Block (512Kb)	SPI Flash Address	Contents
0	0x000000	Unused
1	0x010000	Primary Bitstream
2	0x020000	
3	0x030000	
.	.	
.	.	
18	0x120000	
32	0x200000	Golden Bitstream
33	0x210000	
34	0x220000	
.	.	
.	.	
49	0x310000	
N	0xFF0000	Jump instruction to 0x200000
	0xFFFF00	

1. The bitstream sizes shown are examples. Actual sizes and address boundaries vary with device density.
2. After the golden bitstream is written into the SPI Flash device, the top half of the SPI Flash can be write-protected to secure the golden bitstream from alterations.
3. When the SPI Flash device reaches the address 0xFFFFF, it rolls over to address 0x000000. If the last page is un-programmed, the device can read the jump instruction programmed on address 0x000000 effectively implementing a multiple patterns support for board development or debugging need.

## Slave SPI (SSPI)

The LatticeECP3 can be configured by a SPI Master controller. Using the CFG[2:0] inputs to select SSPI configuration mode the FPGA becomes a SPI Slave device, receiving data from a SPI Master controller. The FPGA can be accessed using Mode 0 and Mode 3 bus transactions.

The slave SPI interface allows for a the following functions to be performed:

- Configuration of the FPGA
- Readback of the FPGA configuration bitstream
- Forcing the device to REFRESH as if PROGRAMN were asserted
- Read/Write access to a SPI PROM attached to the SPI Master configuration pins
- Clearing the FPGA configuration
- Reading the FPGA ID code
- Reading the FPGA User ID code

**Table 15-7. Slave SPI Pins**

Signal Name	Type	Description	Function
SN	Active Low Input	Chip Select	A falling edge on SN causes the FPGA to enter Command State. A rising edge on SN causes the FPGA to enter Reset State.
SI	Input	Serial Input Data	Serial input for command and data.
SO	Tri-state Output	Serial Output Data	Serial output data to the SPI Master.
SCK	Clock Input	Serial Data Clock	Serial input clock for command and data.
HOLDN	Asynchronous Active Low Input	Put the Device on Hold	Tri-state SO and set the device into the suspension state by ignoring the CCLK. Do not assert when loading encrypted bitstreams.

The chip select pin (SN) is a chip select input to the FPGA. The LatticeECP3 responds on the falling and rising edges of the SN input. SN is not a level sensitive input. When the SN falling edge occurs the FPGA is ready to accept commands from a SPI Master Controller. A rising edge on the SN input resets the internal state machine and tri-states the SO output pin. The only exception to this is when the FPGA has received a SPI\_PROGRAM command. This command can only be interrupted by the assertion of the PROGRAMN input.

The HOLDN pin is provided to allow a SPI Master Controller to pause transactions on the Slave SPI port. When HOLDN is asserted low the FPGA tri-states the SO output and ignores the SCLK input. This allows the SPI Master Controller to interact with another SPI device and then resume transactions to the LatticeECP3. Encrypted bitstreams must be sent without interruption. You are not permitted to assert HOLDN or deassert SN once an encrypted bitstream transmission has begun.

**Figure 15-7. Slave SPI Example**

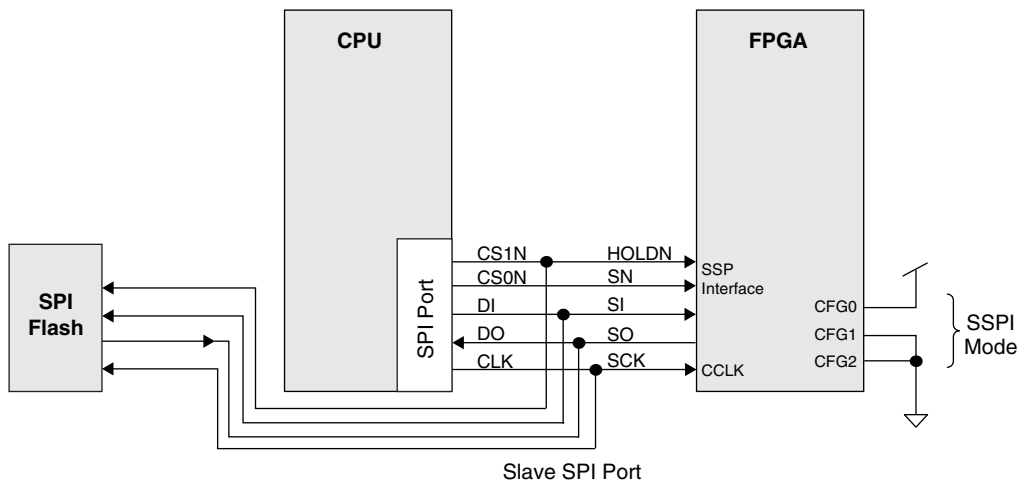


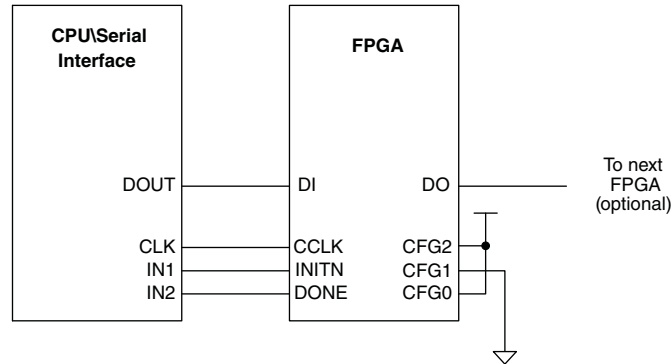
Figure 15-7 illustrates how an on-board CPU can be connected to the LatticeECP3 using the Slave SPI programming interface. The CPU can fetch configuration data from the attached SPI PROM. The CPU is not required to deassert the SN input to the FPGA. When the CPU asserts the CS1N to access the SPI PROM the FPGA HOLDN is asserted causing it to ignore SCLK transitions. The HOLDN input can not be asserted when transferring an encrypted bitstream.

Full details on using Slave SPI mode on the LatticeECP3 are provided in TN1222, [LatticeECP3 Slave SPI Port User's Guide](#).

### Slave Serial Configuration Mode (SCM)

Slave Serial Configuration mode provides a simple, low pin count method for configuring one or more FPGAs. Data is presented to the FPGA on the Data Input pin DI at every CCLK rising edge.

Figure 15-8. Slave Serial Block Diagram



The bitstream data generated by Lattice Diamond is formatted so that it is ready to shift into the FPGA. Left shift the data out of the file in order for it to be correctly received by the FPGA.

The FPGA synchronizes itself on either a 0xBDB3 or 0xBFB3 code word. It is critical that any data presented on DIN not be recognized as one of these two synchronization words early. To guarantee proper recognition of the synchronization word it is recommended that the synchronization word always be preceded by a minimum of 128 '1' bits. Presenting any other bitstream data, Programmer generated header information for example, risks the being misinterpreted due to bit slippage.

Slave Serial Configuration Mode can be used to configure a chain of FPGAs. Details about configuring a chain of devices is discussed in [“Combining Configuration Modes” on page 38](#) of this document.

### Slave Parallel Mode (SPCM)

The LatticeECP3 can be configured using Slave Parallel Configuration Mode. Slave Parallel permits an external master to configure the FPGA using an 8-bit synchronous SRAM bus interface. Slave Parallel Configuration Mode is a flexible method for configuring one or more FPGAs. It is also the fastest mode available for configuring the LatticeECP3.

The slave parallel interface allows for a multitude of different functions to be performed:

- Configuration of the FPGA
- Readback of the FPGA configuration bitstream
- Reading the device CRC
- Reading the programming status register
- Reading the FPGA ID code
- Reading the FPGA User ID code

Figure 15-9. Slave Parallel Block Diagram

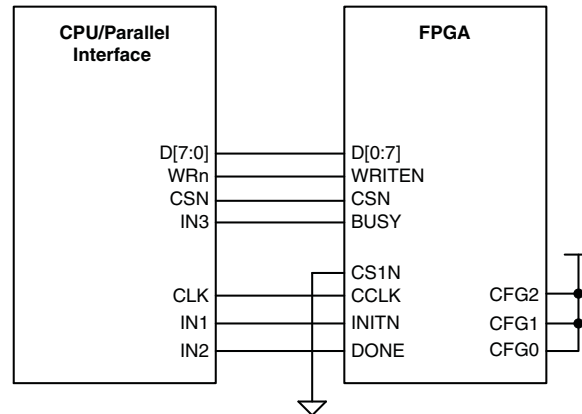
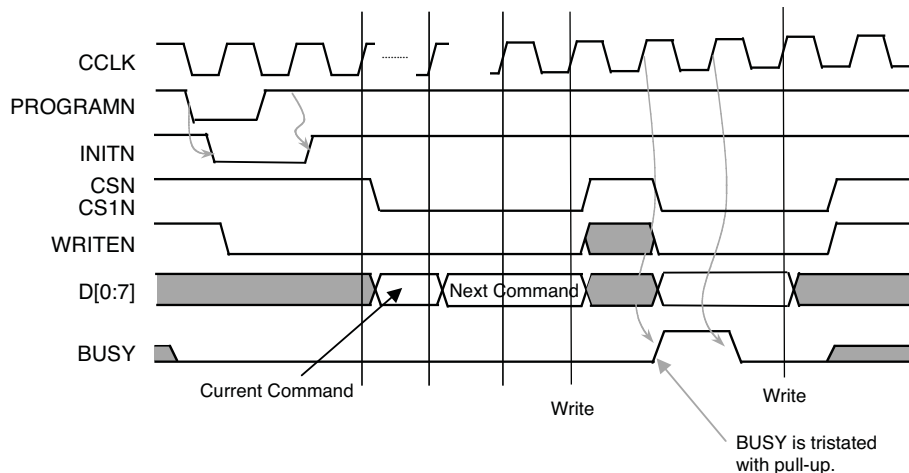


Figure 15-9 shows the typical Slave Parallel Configuration Mode usage. Configuration data can be written to the LatticeECP3 immediately following the INITN rising edge. The LatticeECP3 data bus bit ordering is denoted using a big-endian nomenclature. This means that D0 receives the MSBit, and D7 receives the LSBit. One byte of data can be sent to the FPGA on each rising CCLK edge as long as CSN, CS1N, and WRITEN are asserted. When the LatticeECP3 is the only device being configured the FPGA can receive configuration data at the full CCLK rate. The master device is not required to monitor the BUSY pin in this situation, because the configuration bitstreams are padded to avoid BUSY assertion.

Sending an encrypted bitstream must be done atomically, i.e. without interruption. The bus master is not permitted to pause the transfer of an encrypted bitstream by deasserting the CSN or CS1N inputs. The CCLK pin can be stretched or stopped if desired, but the CSN and CS1N pins must remain asserted.

Slave Parallel mode can also be used to read status registers and the configuration bitstream. In order for the Slave Parallel port to be used to perform read operations the FPGA must have the PERSISTENT preference set to SLAVE\_PARALLEL mode. See the Configuration Pin Management section of this document for more information.

Figure 15-10. Parallel Port Write Timing Diagram



Note: The BUSY pin cannot go high while both CS1N and CSN are low. The second BUSY high shown is OK since CS1N or CSN was low previously.



## JTAG Mode (IEEE 1149.1 and IEEE 1532)

The LatticeECP3 provides an IEEE 1532 compliant interface. The IEEE 1532 specification, a superset of the IEEE 1149.1 JTAG specification, describes a standard methodology for configuring programmable logic devices. The LatticeECP3 only requires the four IEEE 1149.1 control signals (TCK, TMS, TDI, and TDO) in order to initiate and complete programming operations. The LatticeECP3 JTAG port is always available for use, regardless of the configuration mode selected.

Programming the LatticeECP3 using the JTAG port is typically accomplished in one of several ways:

- You can use Lattice Diamond Programmer software in combination with a Lattice download cable
- You can use Automatic Test Equipment that can read Serial Vector Format (SVF), In-System Configuration (ISC), STAPL/JAM, or ATE vector files
- You can use an embedded microprocessor to run Lattice's ispVM Embedded configuration software

### Lattice Diamond Programmer's Fast Program

The Lattice Diamond development tools translate your design into a bitstream containing an optional header, mandatory preamble, and the device configuration data. The configuration data includes its own preamble, fuse data, and finally a trailing CRC. This basic structure is used for all of the configuration modes supported by the LatticeECP3. The IEEE 1532 mode adds some additional operations to the device configuration process.

Prior to sending the configuration data the FPGA's Boundary Scan I/O Cells are placed in a high-impedance state, and the FPGA's configuration memory is cleared. Because the I/O are tri-stated the DONE and INITn output signals do not provide status information while the configuration data is being written to the FPGA. The JTAG configuration mode uses an internal status register to confirm the FPGA DONE and INITn status signals indicate the device configured correctly. After the internal DONE and INITn controls are confirmed, the Boundary Scan I/O Cells are re-enabled, and all I/O take on the function assigned to them.

The JTAG interface, because it can control the Boundary Scan I/O Cells, can also be used to configure the LatticeECP3 without putting the I/O into a high-impedance state. During device configuration the I/O cells can be locked in their last known active state. This mode of operation is called TransFR Programming. A full description of how to use TransFR is provided in TN1087, [Minimizing System Interruption During Configuration Using TransFR Technology](#).

### JTAG Configuration Data Read and Save

The JTAG interface can be used to read the configuration data stored in the FPGA's SRAM array. There are two modes available to retrieve the data, foreground mode or background mode.

Foreground readback is accessed using IEEE 1532 mode. When using this method the JTAG Boundary Scan Cells are placed in a high-impedance state, and the configuration data read. Once the configuration data is retrieved the Boundary Scan Cells are restored, and the FPGA returns to normal operation.

The Background Read and Save operation allows you to read the content of the device while the device remains in operation. All I/O, as well as the non-JTAG configuration pins, continue normal operation during the Background Read and Save operation. You must not violate the following conditions when using the Background Read and Save function:

- The Soft Error Detection system must not be running. De-assert the SEDENABLE pin to prevent the SED circuit from interfering with the Background Read and Save operation. It is recommended that you wait at least one full SEDSTART to SEDDONE time period after the deassertion of the SEDENABLE to make sure the SED circuit has discontinued operation. Alternately monitor the SEDINPROG output, and wait for it to de-assert.
- Write operations to distributed RAM blocks must be suspended. Write operations that occur at the same time the SRAM cell is being read are non-deterministic. It is possible for the SRAM to receive, or retain, incorrect RAM data.

Regardless of which read and save mode is used the configuration data will not include the EBR or the distributed RAM contents. Distributed RAM contents will always be return zeroes.

### **Boundary Scan and Boundary Scan Description Language (BSDL) Files**

The LatticeECP3, as mentioned previously, provides an IEEE 1149.1 compliant JTAG interface. The JTAG interface can be controlled by Automatic Test Equipment (ATE) that uses Boundary Scan Description Language (BSDL) files. Lattice makes BSDL files available for the LatticeECP3 on the Lattice Semiconductor website.

The boundary scan ring covers all of the I/O pins, as well as the dedicated and dual-purpose sysCONFIG pins. Note that PROGRAMN, CCLK, and the CFG pins are observe only (BC4, JTAG read-only) boundary scan cells.

When performing JTAG 1149.1 EXTEST instructions, the SERDES CML termination for both Tx and Rx is set to 50 ohm pull-ups. This allows the high-speed channels to operate properly if DC data is sent or received. During JTAG EXTEST, the termination will be set to 50 ohm. This overrides the termination resistance programmed into the SERDES logic.

## **Bitstream Generation Software Usage**

This section describes the settings for bitstream generation performed by the Diamond software program that generates a bitstream. These options are controlled through the Global Preferences of the Diamond Spreadsheet View and the property settings of the Bit Generation Software tool. To set the Global Preferences and properties in Diamond, see Appendix B. By setting the proper parameter in the Lattice design software the selected configuration options are set in the generated bitstream. As the bitstream is loaded into the device the selected configuration options take effect. These options are described in the following sections.

Bit Generation takes a fully routed Physical Design (.ncd file) as input and produces a configuration bitstream (bit images). The bitstream file contains all of the configuration information from the Physical Design defining the internal logic and interconnections of the FPGA, as well as device-specific information from other files associated with the target device. The data in the bitstream can then be downloaded directly into the FPGA memory cells or used to generate files for PROM programming (using a separate program, ispVM). Please refer to the ispVM documentation for details on creating PROM image files.

### **Configuration Options**

Several configuration options are available for each configuration mode. These options are controlled from the Spreadsheet View for each Strategy. They include the following items.

- When daisy chaining multiple FPGA devices an overflow option is provided for serial and parallel configuration modes
- When using SPI or SPIm mode, the master clock frequency can be set
- A security bit can be set to prevent SRAM readback
- The Persistent option can be set
- Configuration pins can be protected
- DONE pin options can be selected

By setting the proper parameter in the Lattice design software the selected configuration options are set in the generated bitstream. As the bitstream is loaded into the device the selected configuration options take effect. These options are described in the following sections.

#### **Master Clock**

If the CFG pins indicate an SPI or SPIm mode, the MCLK pin will become an output with a default frequency, or one selected when you added preferences to your design. The default Master Clock Frequency is 2.5 MHz. For a complete list of the supported Master Clock frequencies, please see the [LatticeECP3 Family Data Sheet](#). When using the LatticeECP3 devices, the available frequencies are restricted, as shown in the data sheet.

You can change the Master Clock frequency by setting the MCCLK\_FREQ global preference in the Diamond Spreadsheet view tool. During configuration one of the first pieces of information loaded is the MCCLK\_FREQ parameter. When this parameter is loaded the master clock frequency changes to the selected value without glitching. Care should be exercised not to exceed the frequency specification of the slave devices or the signal integrity capabilities of the PCB layout.

Configuration time is computed by dividing the maximum number of configuration bits, as given in Table 15-5 above, by the Master Clock frequency.

**Table 15-8. Selectable Master Clock (MCCLK) Frequencies During Configuration (Nominal)**

MCCLK (MHz)	MCCLK (MHz)
2.5 <sup>1</sup>	10
	13
4.3	15 <sup>2</sup>
5.4	20
6.9	26
8.1	
9.2	33 <sup>3</sup>

1. Software default MCLK frequency. Hardware default is 3.1MHz.
2. Maximum MCCLK with encryption enabled.
3. Maximum MCCLK without encryption

### Security Bit

Setting the CONFIG\_SECURE option to ON prevents readback of the SRAM from JTAG or the sysCONFIG pins. When CONFIG\_SECURE is set to ON the only operations available are erase and write. The security control bit is updated as the last operation of SRAM configuration. If a secured device is read it will output all ones.

For LatticeECP3 devices the CONFIG\_SECURE option is accessed via the Design Planner in ispLEVER. To set this option in Diamond, see Appendix B. The default is OFF.

### Persistent Option

The PERSISTENT option is used to direct the place and route tools about how it can use the sysCONFIG pins. By default the PERSISTENT option is turned OFF, which allows the place and route tools to reclaim most of the configuration pins as general purpose input/output. Changing the PERSISTENT configuration option from its default state prevents the place and route tools from either the Slave SPI or the Slave Parallel configuration ports from becoming general purpose I/O.

Enabling the dedicated sysCONFIG ports is useful for performing additional capabilities while the FPGA is in user mode.

You use the SLAVE\_PARALLEL setting when:

- You want to read back the FPGAs SRAM contents. The LatticeECP3 provides a command set and access protocol that allows the configuration SRAM to be read from the FPGA. The Slave Parallel port can read all of the configuration data, except the EBR and the distributed RAM contents.
- You have a LatticeECP3, configured as a SPI Master, in series of FPGAs in a device chain. The SPI Master FPGA must keep the MCLK pin active in order to provide a configuration clock for all of the chained FPGAs. Table 15-3 describes the configuration pins that are preserved. The MCLK output is only preserved Slave Parallel configuration mode. If PERSISTENT is set to OFF, or SSPI the MCLK output tri-states after the lead FPGA is configured, which prevents chained FPGAs from configuring.

Use the SSPI PERSISTENT setting when:

- You want to access a SPI PROM attached to the SPI Master configuration pins. You can attach a SPI memory

controller and using a custom command you can perform erase, program, and verify sequences on the SPI PROM while the FPGA is in operation. Table 15-3 describes the configuration pins that are preserved. Information about the Slave SPI transactions are published in TN1222 [LatticeECP3 Slave SPI Port User's Guide](#). You can also use the SSPI Embedded device programming software provided by Lattice.

## Configuration Mode

The CONFIG\_MODE option tells the software which configuration port the hardware is using to program the FPGA. Setting this parameter permits the design software to check to make sure configuration port pins are not oversubscribed. The oversubscription is only flagged as a warning. In some cases it is acceptable to oversubscribe the configuration port. For example it is acceptable to have the FPGA in SPI Master configuration mode and use the SISPI, SPID0, and SPICS pins as general purpose I/O.

The CONFIG\_MODE is also used to make sure encrypted bitstreams are generated correctly. To guarantee correct operation of encrypted bitstreams you need to set the CONFIG\_MODE parameter.

## DONE EX

During configuration the DONE output pin is low. Once configuration is complete, indicated by assertion of an internal DONE bit, the device wake-up sequence takes place. The external DONE pin is able to operate in one of two modes during the wake up sequence. The default behavior, set when DONE\_EX = OFF, is for it to actively drive to VIL. When DONE\_EX is set ON, the external DONE pin becomes an open-drain output. The LatticeECP3 wake up sequence will stall until the external DONE pin is pulled high. Set DONE\_EX to ON when you want to synchronize the when a chain of FPGAs wakes up. Make sure you place a pullup resistor that is able to drive all of the DONE pins.

## Device Wake-Up

When configuration is complete the device will wake up in a predictable fashion. Wake-Up occurs after successful configuration, without errors, and provides the transition from Configuration Mode to User Mode. The Wake-Up process begins when the internal DONE bit is set.

Table 15-9 provides a list of the Wake-Up sequences supported by the LatticeECP3; Figure 15-11 shows the Wake-Up timing. The default Wake-Up sequence works fine for most single device applications.

**Table 15-9. Wake-Up Options**

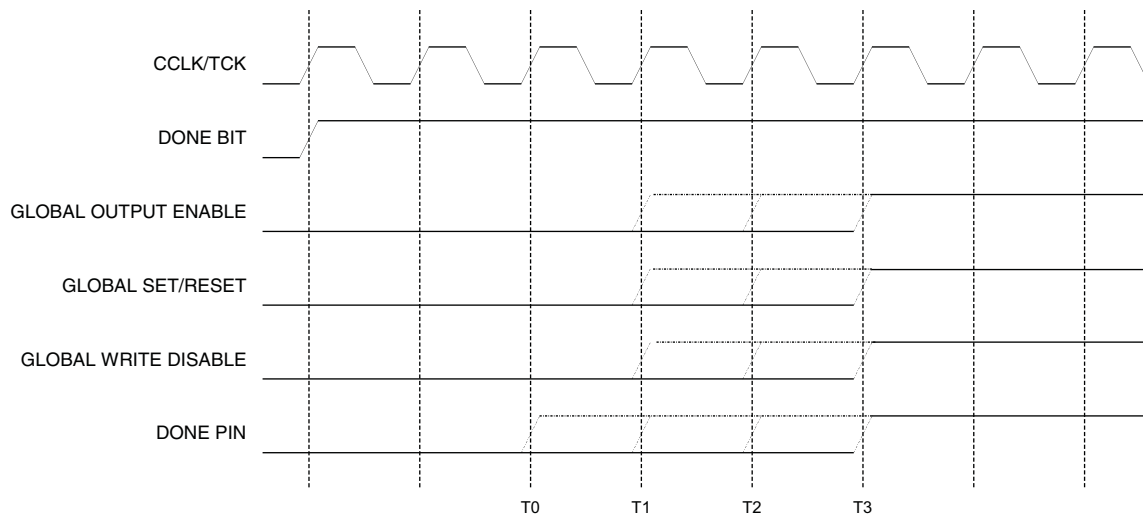
Sequence	Phase T0	Phase T1	Phase T2	Phase T3
1	DONE	GOE, GWDIS, GSR		
2	DONE		GOE, GWDIS, GSR	
3	DONE			GOE, GWDIS, GSR
4 <sup>1</sup>	DONE	GOE	GWDIS, GSR	
5	DONE	GOE		GWDIS, GSR
6	DONE	GOE	GWDIS	GSR
7	DONE	GOE	GSR	GWDIS
8		DONE	GOE, GWDIS, GSR	
9		DONE		GOE, GWDIS, GSR
10		DONE	GWDIS, GSR	GOE
11		DONE	GOE	GWDIS, GSR
12			DONE	GOE, GWDIS, GSR
13		GOE, GWDIS, GSR	DONE	
14		GOE	DONE	GWDIS, GSR
15		GOE, GWDIS	DONE	GSR
16		GWDIS	DONE	GOE, GSR
17		GWDIS, GSR	DONE	GOE

**Table 15-9. Wake-Up Options (Continued)**

Sequence	Phase T0	Phase T1	Phase T2	Phase T3
18		GOE, GSR	DONE	GWDIS
19			GOE, GWDIS, GSR	DONE
20		GOE, GWDIS, GSR		DONE
21 <sup>2</sup>		GOE	GWDIS, GSR	DONE
22		GOE, GWDIS	GSR	DONE
23		GWDIS	GOE, GSR	DONE
24		GWDIS, GSR	GOE	DONE
25		GOE, GSR	GWDIS	DONE

1. Default when DONE\_EX=ON.
2. Default when DONE\_EX=OFF.

**Figure 15-11. Wake-Up Timing Diagram**



### Synchronizing Wake-Up

The LatticeECP3 is, in most cases, configured using one of the master configuration modes. The FPGA, when in master configuration mode, is driving the configuration clock. The configuration clock is used for stepping through the final four Wake-Up states described in the previous section.

The LatticeECP3 has the ability to use an external clock source to control the final state transitions in the Wake-Up process. There are three possible sources for the clock. The JTAG TCK, the Slave Configuration CCLK, and a general-purpose input.

### Start-Up Clock Selection

Once the FPGA is configured, it enters the start-up state, which is the transition between the configuration and operational states. This sequence is synchronized to a clock source, which defaults to CCLK when a slave configuration mode is used, or TCK when JTAG is used.

If desired, a user-defined clock source can be used instead of CCLK/TCK. You need to specify this clock signal, and instantiate the STRTUP library element in your design. The example shown below shows the proper syntax of instantiating the STRTUP library element.

**Verilog**

```
STRTUP u1 (.UCLK(<clock_name>)) /* synthesis syn_noprune=1 */;
```

**VHDL**

```
component STRTUP
  port(STRTUP: in STD_ULOGIC );
end component;

attribute syn_noprune: boolean ;
attribute syn_noprune of STRTUP: component is true;
begin
  u1: STRTUP port map (UCLK =><clock name>);
```

**Synchronous to Internal DONE Bit**

If the LatticeECP3 is the only device in the configuration chain, or the last device in the chain, DONE\_EX should be set to the default value (OFF). The Wake-Up process will be initiated by setting of the internal DONE bit on successful completion of configuration.

**Synchronous to External DONE Pin**

The DONE pin can be used to synchronize Wake-Up to other devices in a configuration chain. If DONE\_EX (see the DONE EX section above) is ON then the DONE pin is an open-drain bi-directional pin. If an external device drives the DONE pin low then the Wake-Up sequence stalls until DONE is active high. Once the DONE pin goes high the device will follow the selected WAKE\_UP sequence.

In a configuration chain, a chain of devices configuring from one source (such as Figure 15-17), it is usually desirable, or even necessary, to delay wake-up of all of the devices until the last device finishes configuration. This is accomplished by setting DONE\_EX to OFF on the last device while setting DONE\_EX to ON for the other devices.

**Wake-up Sequence Options**

The wake-up sequence options determine the order of application for three internal signals, GSR, GWDIS, and GOE, and one external signal, DONE.

- GSR is used to set and reset the core of the device. GSR is asserted (low) during configuration and de-asserted (high) in the Wake-Up sequence.
- When the GWDIS signal is low it safeguards the integrity of the RAM Blocks and LUTs in the device. This signal is low before the device wakes up. The GWDIS signal is internal to the FPGA, and does not appear on any FPGA I/O. During the time it is driven low all EBR and LUT RAM elements are safe from being modified.
- During initialization and configuration the FPGA I/O are placed in a high impedance state. The GOE control controls when the FPGA I/O leave the high impedance state. The I/O are Hi-Z when GOE is asserted low.
- The DONE pin, when high, indicates that FPGA has completed configuration and is in user mode. DONE will only be high if DONE\_EX=ON, the output driver is released, and the external pin is pulled up.

If DONE\_EX (see DONE EX above) is OFF then sequence 21 is the default, but you can select any sequence from 8 to 25; if DONE\_EX is ON the default sequence is 4, but you can select any sequence from 1 to 7.

**WAKE\_ON\_LOCK**

The wake-up sequence can be delayed until the selected PLLs have a chance to lock. The WAKE\_ON\_LOCK attribute selects which PLLs will delay the wake-up sequence until the PLL locks. If you choose an external signal for PLL feedback rather than an internal clock signal, wake-up must occur without waiting for PLL lock because all I/Os are tri-stated until the device wakes up, preventing the PLL from locking.

Using the default mode of operation, the device PLLs do not have to be locked for wake-up to commence. You can choose to make the wake-up sequence dependent on any of the PLLs. If multiple PLLs are included in the design, all PLLs in the design have to be locked to satisfy the wake-up sequence.

## Bitstream Generation Property Options

### Run DRC (T/F)

When the Run DRC option is set to TRUE, a physical design rule check will be run prior to generating a bitstream. The output will be saved to the Bit Generation report (.bgn file). Running DRC before a bitstream is produced will detect any errors that could cause the FPGA to function improperly. If no fatal errors are detected, it will produce a bitstream file. The default is True and will run DRC. When this option is set to False, a design rule check (DRC) will not be run prior to generating a bitstream.

### Create Bitfile (T/F)

This option allows you to decide whether or not to generate an output bitstream. The default setting is to create a bitstream.

### Bitstream File Formats

- Bit file (binary)
- Raw bit file
- Mask & Readback file (ASCII)
- Mask & Readback file (binary)

These options allow you to choose the format of a bitstream file. The Raw Bit option causes the Bitstream Generator to create a Raw Bit (.rbit) file instead of a binary file (.bit). A binary .bit file can be viewed with a binary editor.

The Raw Bit File is a text file containing ASCII ones and zeros representing the bits in the bitstream file. If you are using a microprocessor to configure a single FPGA, you can include the Raw Bit file in the source code as a text file to represent the configuration data. The sequence of characters in the Raw Bit file is the same as the bit sequence that will be written into the FPGA. This file is a large file.

A Mask file (.msk) can be generated in either an ASCII formatted file or binary file. The Mask file is used to compare relevant bit locations for executing a readback of configuration data contained in an operating FPGA. You can compare readback data from the device to the mask file after downloading the bitstream. The ASCII mask file will contain 1's and 0's, and X's. The file contains all FPGA data frames. It contains no header, ID frames, address frames and no preloaded frames.

### No Header (T/F)

The generated bitstream contains no header. The default will be false and will always produce a bitstream file including all the header information.

## Bitstream Encryption/Decryption Flow

The LatticeECP3 supports both encrypted and non-encrypted bitstreams. The encrypted flow adds only two steps to the normal FPGA design flow, encryption of the configuration bitstream and programming the encryption key into the LatticeECP3 devices.

### Encrypting the Bitstream

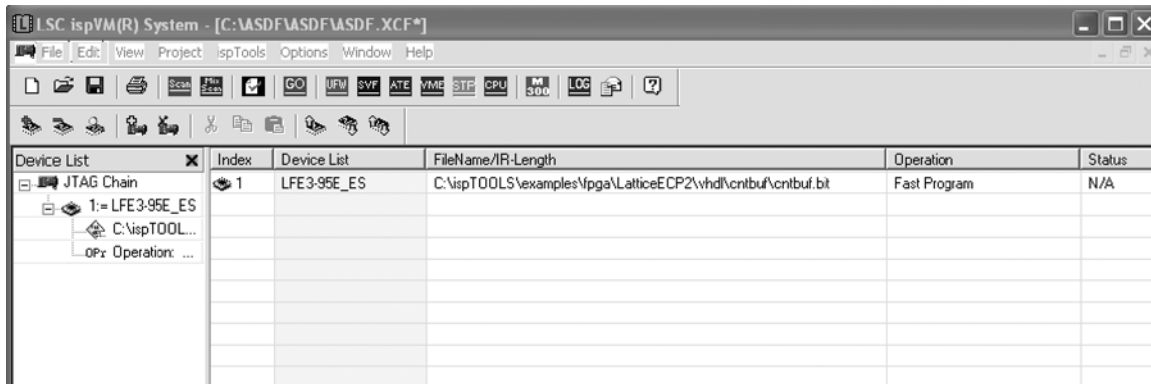
As with any other Lattice FPGA design flow, the engineer must first create the design using a device and version of ispLEVER or Diamond which supports the encryption feature. You must obtain the Encryption Installer from Lattice prior to using Encryption capabilities. The design is synthesized, mapped, placed and routed, and verified. Once the engineer is satisfied with the design a bitstream is created and loaded into the FPGA for final debug. After the design has been debugged it is time to secure the design.

The bitstream can be encrypted using an appropriate version of ispLEVER by going to the Tools pull-down menu and selecting Security features or by using the Universal File Writer (UFW), which is part of the Lattice ispVM™



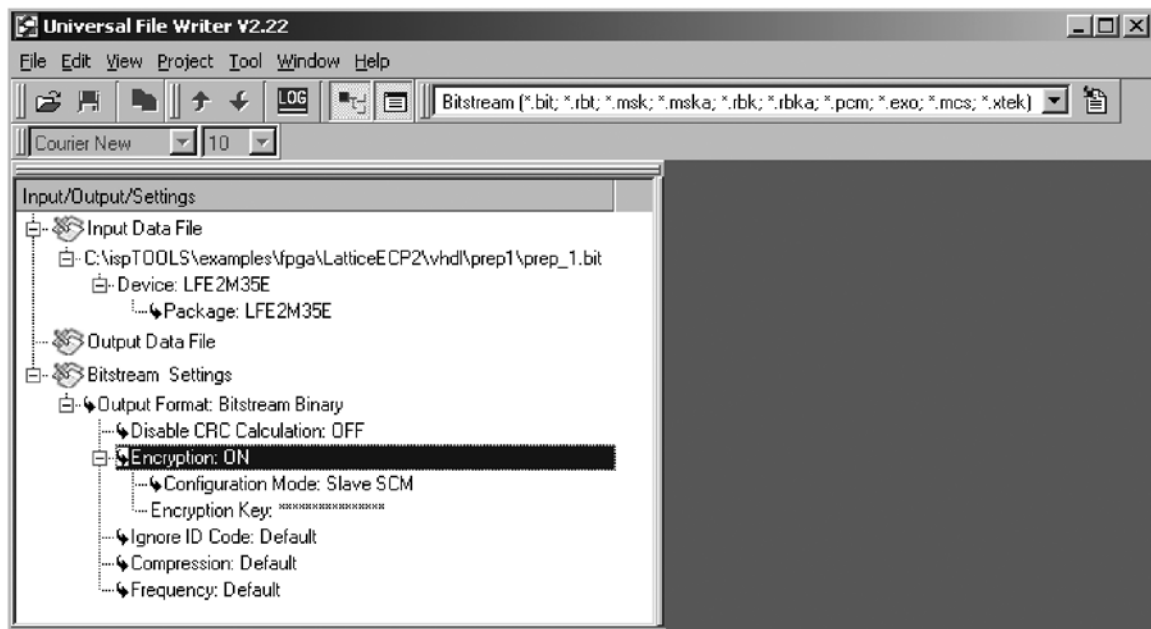
System tool suite. The file is encrypted using ispVM as follows. To encrypt the bitstream in Diamond, see Appendix B.

**Figure 15-12. ispVM Main Window**



1. Start ispVM. You can start ispVM from the Tools menu in ispLEVER or from the **Start -> Programs** menu in Windows. ispVM is not accessible from the Tools menu in Diamond. You should see a window that looks similar to Figure 15-12. Click on the **UFW** button on the toolbar. You will see a window similar to Figure 15-13.

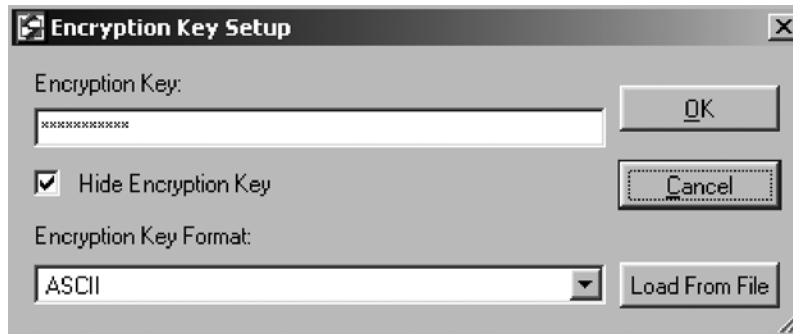
**Figure 15-13. Universal File Writer (Encryption Option)**



2. Double click on **Input Data File** and browse to the non-encrypted bitstream created using ispLEVER or Diamond. Double-click on **Output Data File** and select an output file name. Right-click on **Encryption** and select **ON**. Right-click on **Configuration Mode** and select the type of device the FPGA will be configuring from, such as SPI Serial Flash. Right-click on **Encryption Key** and select **Edit Encryption Key**. You will see a window that looks similar to Figure 15-14.



Figure 15-14. Encryption Key Dialog Window



3. Enter the desired 128-bit key. The key can be entered in Hexadecimal or ASCII. Hex supports 0 through f and is not case sensitive. ASCII supports all alphanumeric characters, as well as spaces, and is case sensitive. Note: be sure to remember this key. Lattice cannot recover an encrypted file if the key is lost. Click on **OK** to go back to the main UFW window.
4. From the menu bar, click on **Project -> Generate** to create the encrypted bitstream file.
5. The bitstream can now be loaded directly into non-volatile configuration storage (such as SPI Serial Flash) using a Lattice ispDOWNLOAD<sup>®</sup> Cable, a third-party programmer, or any other method normally used to program a non-encrypted bitstream. However, before the LatticeECP3 can configure from the encrypted file the 128-bit key used to encrypt the file must be programmed into the one-time programmable cells on the FPGA.

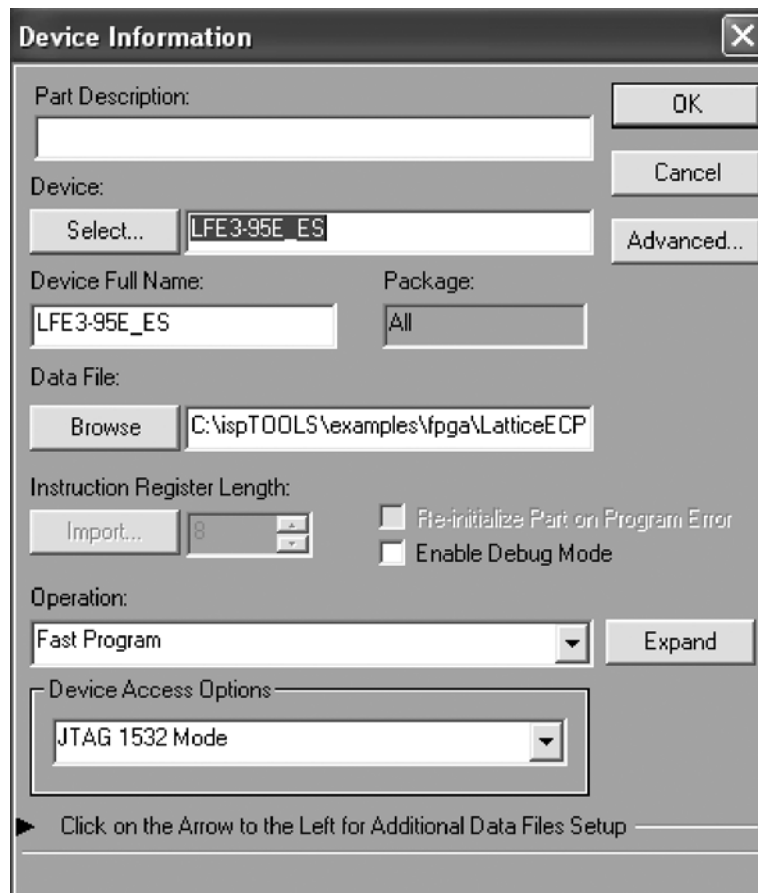
### Programming the 128-bit Key

The next step is to program the 128-bit encryption key into the one-time programmable cells on the LatticeECP3. This is done through the device JTAG interface. Note that this step is separated from file encryption to allow flexibility in the manufacturing flow. For instance, the board manufacturer might program the encrypted file into the SPI Serial Flash, but the key might be programmed at your facility. This flow adds to design security and it allows you to control over-building of a design. Over-building occurs when a third party builds more boards than are authorized and sells them to grey market customers. If the key is programmed at the factory, then the factory controls the number of working boards that enter the market. The LatticeECP3 will only configure from a file that has been encrypted with the same 128-bit key that is programmed into the FPGA.

To program the key into the LatticeECP3, proceed as follows.

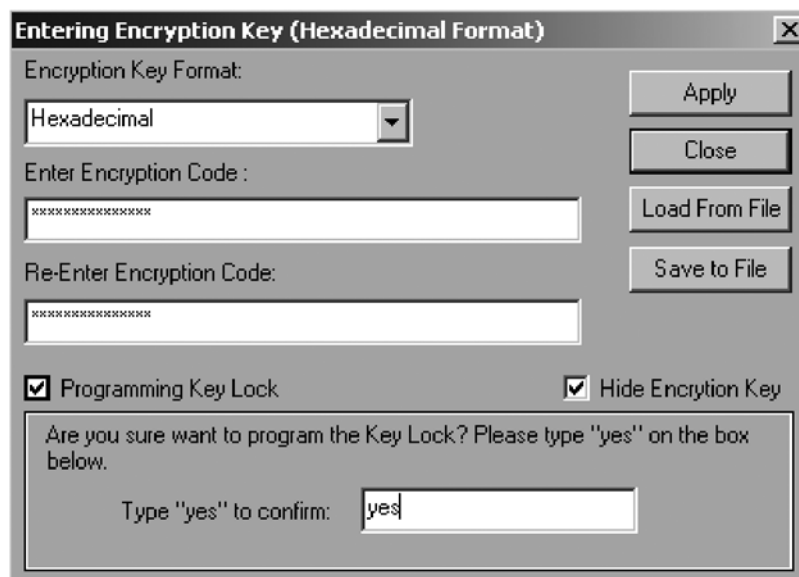
1. Attach a Lattice ispDOWNLOAD cable from a PC to the JTAG connector wired to the LatticeECP3 (note that the 128-bit key can only be programmed into the LatticeECP3 using the JTAG port). Apply power to the board.
2. Start the ispVM System software. ispVM can be started from within the ispLEVER Tools menu or from the **Start -> Programs** menu in Windows. ispVM cannot be invoked from the Tools menu in Diamond. You should see a window that looks similar to Figure 15-12. If the window does not show the board's JTAG chain then proceed as follows. Otherwise, proceed to step 3.
  - a. Click the **SCAN** button in the toolbar to find all Lattice devices in the JTAG chain. The chain shown in Figure 15-12 has only one device, the LatticeECP3.

Figure 15-15. Device Information Window (Encryption Option)



3. Double-click on the line in the chain containing the LatticeECP3. This will open the Device Information window (see Figure 15-17). From the Device Access Options drop-down box select **Security Mode**, then click on the **Security Key** button to the right. The window will look similar to Figure 15-16.

Figure 15-16. Enter the Encryption Key



4. Enter the desired 128-bit key. The key can be entered in Hexadecimal or ASCII. Hex supports 0 through f and is not case sensitive. ASCII supports all alphanumeric characters, as well as spaces, and is case sensitive. This key must be the same as the key used to encrypt the bitstream. The LatticeECP3 will only configure from an encrypted file whose encryption key matches the one loaded into the FPGA's one-time programmable cells. *Note: be sure to remember this key. Once the Key Lock is programmed, Lattice Semiconductor cannot read back the one-time programmable key.*
  - a. The key can be saved to a file using the **Save to File** button. The key will be encrypted using an 8-character password that you select. The name of the file will be <project\_name>.bek. In the future, instead of entering the 128-bit key, simply click on **Load from File** and provide the password.
5. Programming the Key Lock secures the 128-bit encryption key. Once the Key Lock is programmed and the device is power cycled, the 128-bit encryption key cannot be read out of the device. When satisfied, type **Yes** to confirm, then click **Apply**.
6. From the main ispVM window (Figure 15-12) click on the green **GO** button on the toolbar to program the key into the LatticeECP3 one-time programmable cells. When complete, the LatticeECP3 will only configure from a bitstream encrypted with a key that exactly matches the one just programmed.

### Verifying a Configuration

As an additional security step when an encrypted bitstream is used, the readback path from the SRAM fabric is automatically blocked. In this case, for all ports, a read operation will produce all 1's. However, even when the configuration bitstream has been encrypted and readback disabled, there are still ways to verify that the bitstream was successfully downloaded into the FPGA.

If the SRAM fabric is programmed directly, the data is first decrypted and then the FPGA performs a cyclic redundancy code (CRC) on the data. (CRC) circuitry is used to validate each configuration data frame (sequence of data bits) as it is loaded into the target device. If all CRCs pass, configuration was successful. If a CRC does not pass, the DONE pin will stay low and INITN will go from high to low.

If the encrypted data is stored in non-volatile configuration memory, such as SPI Serial Flash, the data is stored encrypted. A bit-for-bit verify can be performed between the encrypted configuration file and the stored data.

## File Formats

The base binary file format is the same for all non-encrypted, non-1532 configuration modes. Different file types (hex, binary, ASCII, etc.) may ultimately be used to configure the device, but the data in the file is the same. Table 15-10 shows the format of a non-encrypted bitstream. The bitstream consists of a comment field, a header, the preamble, and the configuration setup and data.

**Table 15-10. Non-Encrypted Configuration Data**

Frame	Contents	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator
Header	1111...1111	16 Dummy bits
	1011110110110011	16-bit Standard Bitstream Preamble (0xBDB3)
Verify ID		64 bits of command and data
Control Register 0		64 bits of command and data
Reset Address		32 bits of command and data
Write Increment		32 bits of command and data
Data 0		Data, 16-bit CRC, and Stop bits
Data 1		Data, 16-bit CRC, and Stop bits
.	.	.
.	.	.
.	.	.
Data n-1		Data, 16-bit CRC, and Stop bits
End	1111...1111	Terminator bits and 16-bit CRC
Usercode		64 bits of command and data
SED CRC		64 bits of command and data
Program Security		32 bits of command and data
Program Done		32 bits of command and data, 16-bit CRC
NOOP	1111...1111	64 bits of NOOP data
End	1111...1111	32-bit Terminator (all ones)

Note: The data in this table is intended for reference only.

Table 15-11 shows a bitstream that is built for encryption but has not yet been encrypted. The highlighted areas will be encrypted. The changes between Table 15-10 and Table 15-11 include the following:

- The Program Security frame (readback disable) has been moved to the beginning of the file so that readback is turned off at the very beginning of configuration. This is an important security feature that prevents someone from interrupting the configuration before completion and reading back unsecured data.
- A copy of the usercode is placed in the non-encrypted comment string. This has been done to allow you a method to identify an encrypted file. For example, the usercode could be used as a file index. Note that the usercode itself, while encrypted in the configuration data file, is not encrypted on the device. At configuration the usercode is decrypted and placed in the JTAG Usercode register. This allows you a method to identify the data in the device. The JTAG Usercode register can be read back at any time, even when all SRAM readback paths have been turned off. The usercode can be set to any 32-bit value. For information on how to set usercode, see the ispLEVER or Diamond help facility.
- A copy of CONFIG\_MODE, one of the global preferences, is placed in the non-encrypted comment string. CONFIG\_MODE can be SPI/SPIm, SSPI, Slave SCM, Slave PCM, Master PCM, or JTAG.

**Table 15-11. Configuration File Just Before Encryption**

Frame	Contents	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator
Header	1111...1111	16 Dummy Bits
		16-bit Standard Bitstream Preamble
Verify ID		64 bits of Command and Data
Control Register 0		64 bits of Command and Data
Program Security		32 bits of Command and Data
Reset Address		32 bits of Command and Data
Write Increment		32 bits of Command and Data
Data 0		Data, 16-bit CRC, and Stop Bits
Data 1		Data, 16-bit CRC, and Stop Bits
.	.	.
.	.	.
.	.	.
Data n-1		Data, 16-bit CRC and Stop Bits
End	1111...1111	Terminator Bits and 16-bit CRC
Usercode		64 Bits of Command and Data
SED CRC		64 Bits of Command and Data
Program Done		32 Bits of Command and Data, 16-bit CRC
NOOP	1111...1111	64 bits of NOOP data
End	1111...1111	32-bit Terminator (All Ones).

Note: The data in this table is intended for reference only. The shaded areas will be encrypted.

Once encrypted, besides the obvious encryption of the data itself, the file will have additional differences from a non-encrypted file (refer to Tables 15-12, 15-13, and 15-14).

- There are three preambles, the encryption preamble, alignment preamble, and the bitstream preamble. The alignment preamble marks the beginning of the encrypted data. The entire original bitstream, including the bitstream preamble are all encrypted, per Table 15-11. The comment string, the encryption preamble, dummy data, and alignment preamble are not encrypted.
- The decryption engine within the FPGA takes some time to perform its task; extra time is provided in one of two ways. For master configuration modes (SPI and SPIm) the FPGA drives the configuration clock, so when extra time is needed the FPGA stops sending configuration clocks. For slave configuration modes (Bitstream-Burst, Slave Serial, and Slave Parallel) the data must be padded to create the extra time. Because of this there are several different file formats for encrypted data (see Tables 15-12, 15-13, and 15-14). Note that because of the time needed to decrypt the bitstream it takes longer to configure from an encrypted data file than it does from a non-encrypted file. The bitstream sizes may vary depending on the configuration mode.

**Table 15-12. LatticeECP3 Master Serial Encrypted Bitstream File Format**

Frame	Contents (D0..D7)	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator <sup>1</sup> .
Header	(LSB) 1111...1111	2 Dummy Bytes.
	1011111110110011	16-bit Encryption Preamble (0xBF3).
30,000 Filler Bits	11...1111	30,000 bits of filler data (all 1's). Ignored by device.
Key Expansion Preamble	1011101010110011	16-bit Key Expansion Preamble (0xBAB3).
240 Filler Bits	11...1111	240 bits of data (all 1's). Ignored by device.
Alignment Preamble	1011110010110011	16-bit Alignment Preamble (0xBCB3).
	1	1-bit Dummy Data.
Encrypted Configuration Data	Encrypted Data <sup>2</sup>	There are no dummy filler bits when the bitstream is generated for master programming mode. The CCLK of the master device stops the clock when it needs time to decrypt the data. It resumes the clock when ready for new data. <sup>2</sup>
Encrypted Frame Program Done	32 Bits Encrypted	32-bit Program Done Command – Encrypted.
Frame (End)	32-bit Encrypted Terminator	32-bit Terminator (all ones) – Encrypted.
Filler Bits	11...1111	Insert filler to meet the 128-bit bound requirement (all 1's).
Dummy Data	111...1111	207-bit Dummy Data (all ones), to provide delay to turn off the decryption engine.

1. Two new fields, Usercode and Config Mode, have been added to the comment string for the ECP2 family encrypted bitstreams.

2. If bitstream compression is selected, the bitstream must be compressed prior to encryption.

**Table 15-13. LatticeECP3 Slave Serial, JTAG Burst, and Slave SPI Encrypted Bitstream File Format**

Frame	Contents (D0..D7)	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator <sup>1</sup> .
Header	(LSB) 1111...1111	2 Dummy Bytes.
	1011111110110011	16-bit Encryption Preamble (0xBFB3).
30,000 Filler Bits	11...1111	30,000 bits of filler data (all 1's). Ignored by device.
Key Expansion Preamble	1011101010110011	16-bit Key Expansion Preamble (0xBAB3).
240 Filler Bits	11...1111	240 bits of data (all 1's). Ignored by device.
Alignment Preamble	1011110010110011	16-bit Alignment Preamble (0xBCB3).
	1	1-bit Dummy Data.
Configuration Data Frames	128 Bits of Encrypted Data	128-bit Configuration Data. <sup>2</sup>
	64 Bits of Filler Data	64 bits of filler data (all 1's). Provide delay clocks for the decryption engine to decrypt the 128 bits of data just received. If the peripheral device can provide 64 extra clocks, then the 64 bits of dummy data is not required.
	.....	
	128 Bits of Encrypted Data	Last 128 bits of the last Frame of the Configuration data.
	64 Bits of Filler Data	64 bits of filler data (all 1's). Delay to decrypt the last 128 bits of Configuration data.
Encrypted Frame Program Done	32 Bits Encrypted	32-bit Program Done Command – Encrypted.
Frame (End)	32-bit Encrypted Terminator	32-bit Terminator (all ones) – Encrypted.
Filler Bits	11...1111	Insert filler to meet the 128-bit bound requirement (all 1's).
	64 Bits of Filler Data	64 bits of filler data (all 1's). Delay to decrypt the Program Done command and the filler.
Dummy Data	111...1111	207-bit Dummy Data (all ones), to provide delay to turn off the decryption engine.

1. Two new fields, Usercode and Config Mode, have been added to the comment string for the ECP2 family encrypted bitstreams.
2. If bitstream compression is selected, the bitstream must be compressed prior to encryption.

**Table 15-14. LatticeECP3 Master Parallel Encrypted Bitstream File Format**

Frame	Contents (D0..D7)	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator <sup>1</sup> .
Header	(LSB) 1111...1111	2 Dummy Bytes.
	1011111110110011	16-bit Encryption Preamble (0xBFB3).
240,000 Filler Bits	11...1111	240,000 bits of filler data (all 1's). Ignored by device.
Key Expansion Preamble	1011101010110011	16-bit Key Expansion Preamble (0xBAB3).
2032 Filler Bits	11...1111	240 bits of data (all 1's). Ignored by device.
Alignment Preamble	1011101010110011	16-bit Alignment Preamble (0xBCB3).
	11111111	8-bit Dummy Data.
Encrypted Configuration Data	Encrypted Data <sup>2</sup>	There are no dummy filler bits when the bitstream is generated for master programming mode. The CCLK of the master device stops the clock when it needs time to decrypt the data. It resumes the clock when ready for new data. <sup>2</sup>
Encrypted Frame Program Done	32 Bits Encrypted	32-bit Program Done Command – Encrypted.
Frame (End)	32-bit Encrypted Terminator	32-bit Terminator (all ones) – Encrypted.
Filler Bits	11...1111	Insert filler to meet the 128-bit bound requirement (all 1's).
Dummy Data	111...1111	1600-bit Dummy Data (all ones), to provide delay to turn off the decryption engine.

1. Two new fields, Usercode and Config Mode, have been added to the comment string for the ECP2 family encrypted bitstreams.
2. If bitstream compression is selected, the bitstream must be compressed prior to encryption.

## Decryption Flow

Compared to the encryption flow just discussed, the decryption flow is much simpler.

When data comes into the FPGA the decoder starts looking for the preamble and all information before the preamble is ignored. The preamble determines the path of the configuration data.

If the decoder detects a standard bitstream preamble in the bitstream it knows that this is a non-encrypted data file. The decoder then selects the Raw data path.

If the decoder detects an encryption preamble in the bitstream it knows that this is an encrypted data file. If an encryption key has not been programmed, the encrypted data is blocked and configuration fails (the DONE pin stays low), if the proper key has been programmed then configuration can continue. The next block read contains 30,000 clocks of filler data. This delay allows time for the FPGA to read the key cells and prepare the decryption engine. The decoder keeps reading the filler data looking for the alignment preamble. Once found, it knows that the following data needs to go through the decryption engine. It first looks for the standard preamble. Once found, then the SRAM cells' programming begins.

But what happens if the key in the FPGA does not match the key used to encrypt the file? Once the data is decrypted, the FPGA expects to find a valid standard bitstream preamble (BDB3), along with proper commands and data that pass CRC checks. If the keys do not match then the decryption engine will not produce a proper configuration bitstream; either configuration will not start because the preamble was not found (the INITN pin stays high and the DONE pin stays low) or CRC errors will occur, causing the INITN pin to go low to indicate the error.

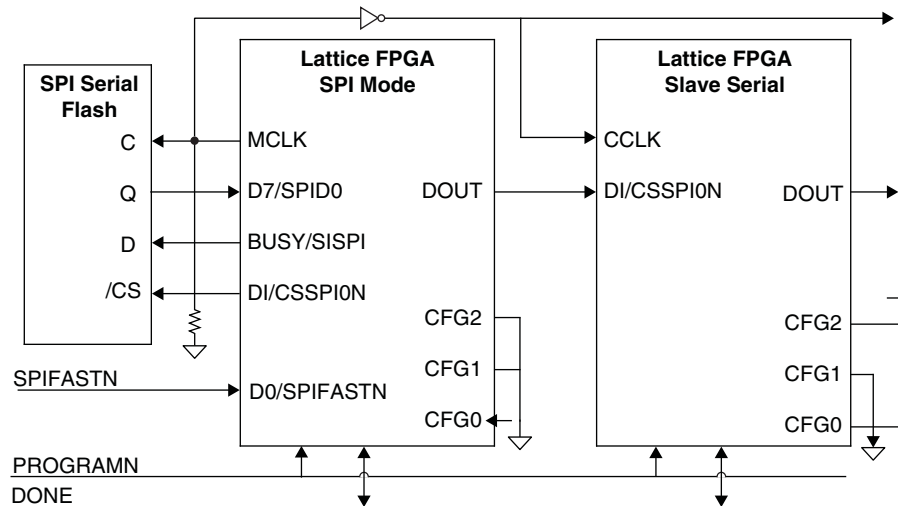


## Combining Configuration Modes

### Multiple FPGAs, One SPI Flash

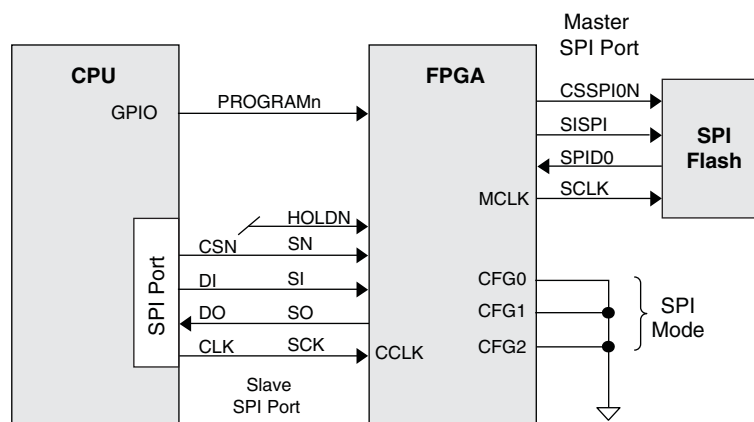
With a sufficiently large SPI Flash, multiple FPGAs can be configured as shown in Figure 15-17. The first FPGA is configured in SPI mode; the following FPGAs are configured in Slave Serial mode. Full details on using SPI mode combined with Slave Serial mode are provided in TN1222 [LatticeECP3 Slave SPI Port User's Guide](#).

Figure 15-17. Multiple FPGAs, One SPI Serial Flash



Note: The inverter for MCLK guarantees the hold time for data input to the daisy chained FPGAs.

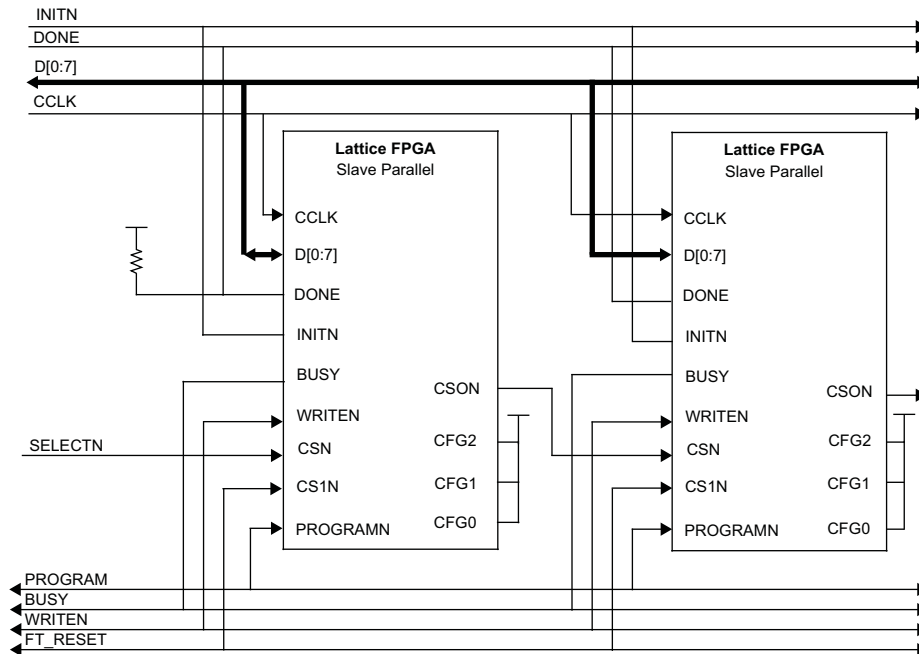
Figure 15-18. Slave SPI Example 1



The system diagram shown in Figure illustrates one application of the Slave SPI interface, where the FPGA selects the SPI Flash as the primary boot source. The modern CPU has the capability to program the SPI Flash boot PROM as well as to command the FPGA to re-boot from the SPI Flash by toggling the PROGRAMN pin. This requirement can only be met if the CPU drives the CCLK, and the MCLK is driven by the FPGA for the SPI Flash boot PROM as shown in Figure .

**CS1N/HOLDN:** When Slave SPI mode is used, this pin is an asynchronous active Low Input that tri-states the serial read out data of the SPI port and sets the device to the suspend state by ignoring the clock. Set the SSPI PERSISTENT to on to retain the pin as HOLDN pin to access the Slave SPI port in user mode.

Figure 15-19. Slave Parallel with Flowthrough



## Chain Mode Options

The LatticeECP3 can be one of many FPGAs in a chain that each need to get configuration data. The Bypass and Flowthrough options control how each FPGA in the chain of devices pass configuration bits to the other devices in the chain. Successful configuration of a chain of FPGAs depends on a thorough understanding of the Bypass and Flowthrough features.

### Bypass Option

This option is used when you are configuring a chain of FPGAs in either parallel or serial daisy chain configurations. The Bypass option, when enabled, adds an additional command to the end of the configuration bitstream being sent to the LatticeECP3. The LatticeECP3 receives all of the configuration bits, and upon reception of the BYPASS command it enables a serial bypass register. This bypass register passes all incoming configuration bits to the DOUT pin for use by the next FPGA in the chain. Prior to the LatticeECP3 receiving the BYPASS command the internal bypass register is initialized to '1'. Any FPGA receiving data from the DOUT pin will see a long string of ones until the BYPASS command is accepted by the LatticeECP3.

The following conditions must be met when Bypass is enabled:

- The PERSISTENT option must be set to Slave Parallel mode
- The bitstream can not be encrypted
- The LatticeECP3 can not be in SPI mode

The Bypass Option is DISABLED by default.

### Flowthrough Option

The Flowthrough option is used when you are configuring a chain of FPGAs in Slave Parallel Configuration mode. It is not applicable to any other configuration mode. The Flowthrough option, when enabled, adds a FLOWTHROUGH command to the end of the bitstream for the LatticeECP3. The LatticeECP3 receives all of the configuration data over the Slave Parallel data bus. When it receives the FLOWTHROUGH command it asserts the CSON output pin driving the parallel bus chip select input of the next FPGA in the chain. Until reception of the

FLOWTHROUGH command the CSON pin is deasserted high, which prevents any downstream FPGA from loading the incoming data bytes.

The following conditions must be met to use the Flowthrough option:

- The PERSISTENT option must be set to Slave Parallel
- The bitstream can not be encrypted
- The CONFIG\_MODE must be Slave Parallel Configuration mode

The Flowthrough option is DISABLED by default.

#### **Reset Configuration RAM in Reconfiguration (T/F)**

When this switch is set to true, it directs the bitstream to reinitialize the device when configuration starts. When the switch is set to false, it directs the bitstream to program the device to retain the current configuration and allows for additional bitstream configuration. The default is true. Use of the feature requires you to be aware of potential contention issues with the prior configuration loaded into the FPGA.

## **References**

- Federal Information Processing Standard Publication 197, Nov. 26, 2001. Advanced Encryption Standard (AES)

## **Technical Support Assistance**

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
June 2009	01.1	Added SSPI command table. Major changes to SPI support.
October 2009	01.2	Added clarification for dual boot function.
November 2009	01.3	Compression support removed.
January 2010	01.4	Updated SPI Flash mode.
March 2010	01.5	Updated Parallel Port Write Timing diagram.
June 2010	01.6	Updated for Lattice Diamond design software support.
December 2010	01.7	Removed EBR_READ command from the Slave SPI Commands table.
March 2011	01.8	Updated Selectable Master Clock (MCCLK) Frequencies During Configuration (Nominal) table.
August 2011	01.9	Added footnote to "One FPGA, One SPI Serial Flash" figure.
September 2011	02.0	Added new table to Appendix A: Maximum Configuration Bits - Serial and Parallel Mode Bitstream Files.
February 2012	02.1	Updated document with new corporate logo.
August 2012	02.2	Added recommendation to pull up CSSPI0N.
April 2013	02.3	Added pins and footnote to the PERSISTENT Setting and Affected Pins table.
May 2013	02.4	Updated the PERSISTENT Setting and Affected Pins table.
	02.5	Added reference for using SPI mode combined with Slave Serial mode in the Combining Configuration Modes section.
	02.6	Updated the Multiple FPGAs, One SPI Serial Flash diagram.
June 2013	02.7	Changed BAB3 to BFB3 in the General Configuration Flow and Configuration Modes sections,
		Changed 0xBAB3 to 0xBFB3 in the General Configuration Flow and Configuration Modes sections,
		Updated the Encrypted File Format for a Master Mode table contents and changed title to LatticeECP3 Master Serial Encrypted Bitstream File Format table,
		Updated the Encrypted File Format for a Slave Serial Mode table contents and changed title to LatticeECP3 Slave Serial, JTAG Burst, and Slave SPI Encrypted Bitstream File Format.
		Updated the Encrypted File Format for a Slave Parallel Mode contents and changed title to LatticeECP3 Master Parallel Encrypted Bitstream File Format table.
July 2013	02.8	Added Reconfiguration Priority section.
		Added general rules when configuring in the Configuration Modes section.
		Updated MCLK FREQ values in the Global Preferences table.
		Changed $t_{PRGMRJ}$ to $t_{PRGM}$ .
		Updated Technical Support Assistance information.

## Appendix A. Configuration Memory Requirements

*Table 15-15. Bitstream Memory*

Description		Number of Bits
Header		16
Preamble		16
Verify ID		64
Reserved		136
CReg0		64
NOOP		8
Reset address		32
Write inc		32
Data frames (by device)	-35	2067
	-70/95	2819
	-150	3607
Bits per frame (by device)	-35	3416
	-70/95	6728
	-150	8384
Total data frame bits (by device)		Data frames multiplied by bits per frame
CRC bits per frame		16
Stop bits per frame		32
End frame		160
CRC		16
Usercode		64
SED CRC		64
Program security		32
EBR frames		Device and user design specified
Write command		32
EBR data		18432
CRC		16
Stop bits		32
CRC		16
EBR bits per frame		18528
Total EBR frame bits		Equal to EBR Frames multiplied by EBR bits per frame
Program done		48
End		32

**Table 15-16. Maximum Configuration Bits – Serial and Parallel Mode Bitstream Files**

Device	All Modes	Slave Serial Mode	Slave Parallel Mode	Units
	Unencrypted Bitstream Size	Encrypted Bitstream Size	Encrypted Bitstream Size	
LatticeECP3-17 No EBR	3.88	5.84	19.60	Mb
LatticeECP3-17 Max EBR	4.41	6.64	22.26	Mb
LatticeECP3-35 No EBR	6.83	10.28	34.38	Mb
LatticeECP3-35 Max EBR	8.10	12.18	40.74	Mb
LatticeECP3-95 No EBR	18.22	27.36	91.32	Mb
LatticeECP3-95 Max EBR	22.46	33.72	112.53	Mb
LatticeECP3-150 No EBR	29.01	43.54	145.27	Mb
LatticeECP3-150 Max EBR	35.58	53.40	178.13	Mb

---

## Appendix B. Lattice Diamond Usage Overview

This appendix discusses the use of Lattice Diamond design software for projects that include the LatticeECP2M SERDES/PCS module .

For general information about the use of Lattice Diamond, refer to the Lattice Diamond Tutorial.

If you have been using ispLEVER software for your FPGA design projects, Lattice Diamond may look like a big change. But if you look closer, you will find many similarities because Lattice Diamond is based on the same toolset and work flow as ispLEVER. The changes are intended to provide a simpler, more integrated, and more enhanced user interface.

### Converting an ispLEVER Project to Lattice Diamond

Design projects created in ispLEVER can easily be imported into Lattice Diamond. The process is automatic except for the ispLEVER process properties, which are similar to the Diamond strategy settings, and PCS modules. After importing a project, you need to set up a strategy for it and regenerate any PCS modules.

### Importing an ispLEVER Design Project

Make a backup copy of the ispLEVER project or make a new copy that will become the Diamond project.


1. In Diamond, choose **File > Open > Import ispLEVER Project**.
2. In the ispLEVER Project dialog box, browse to the project's .syn file and open it.
3. If desired, change the base file name or location for the Diamond project. If you change the location, the new Diamond files will go into the new location, but the original source files will not move or be copied. The Diamond project will reference the source files in the original location.

The project files are converted to Diamond format with the default strategy settings.

### Adjusting PCS Modules

PCS modules created with IPexpress have an unusual file structure and need additional adjustment when importing a project from ispLEVER. There are two ways to do this adjustment. The preferred method is to regenerate the module in Diamond. However this may upgrade the module to a more recent version. An upgrade is usually desirable but if, for some reason, you do not want to upgrade the PCS module, you can manually adjust the module by copying its .txt file into the implementation folder. If you use this method, you must remember to copy the .txt file into any future implementation folders.

### Regenerate PCS Modules

1. Find the PCS module in the Input Files folder of File List view. The module may be represented by an .lpc, .v, or .vhd file.
2. If the File List view shows the Verilog or VHDL file for the module, and you want to regenerate the module, import the module's .lpc file:
  - a. In the File List view, right-click the implementation folder (  ) and choose **Add > Existing File**.
  - b. Browse for the module's .lpc file, **<module\_name>.lpc**, and select it.
  - c. Click **Add**. The .lpc file is added to the File List view.
  - d. Right-click the module's Verilog or VHDL file and choose **Remove**.
3. In File List, double-click the module's .lpc file. The module's IPexpress dialog box opens.
4. In the bottom of the dialog box, click **Generate**. The Generate Log tab is displayed. Check for errors and close.

In File List, the .lpc file is replaced with an .lpx file. The IPexpress manifest (.lpx) file is new with Diamond. The .lpx file keeps track of the files needed for complex modules.

### Using IPexpress with Lattice Diamond

Using IPexpress with Lattice Diamond is essentially same as with ispLEVER.

The configuration GUI tabs are all the same except for the Generation Options tab. Figure 15-20 shows the Generation Options tab window.

Figure 15-20. Generation Options Tab

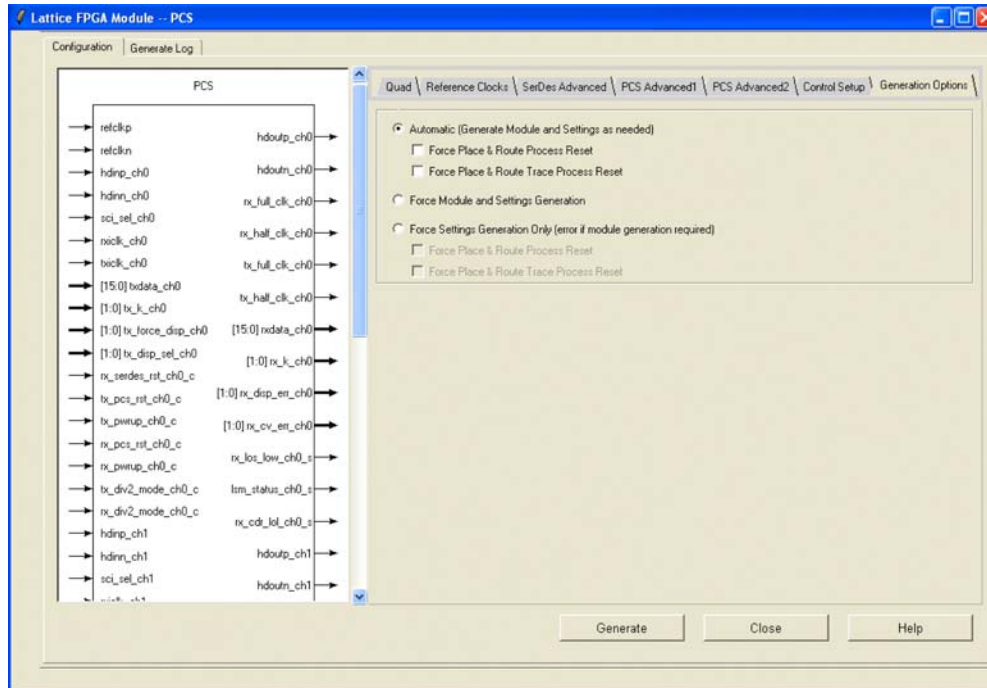


Table 15-17. SERDES\_PCS GUI Attributes – Generation Options Tab

GUI Text	Description
Automatic	Automatically generates the HDL and configuration(.txt) files as needed. Some changes do not require regenerating both files.
Force Module and Settings Generation	Generates both the HDL and configuration files.
Force Settings Generation Only	Generates only the attributes file. You get an error message if the HDL file also needs to be generated.
Force Place & Route Process Reset	Resets the Place & Route Design process, forcing it to be run again with the newly generated PCS module.
Force Place & Route Trace Process Reset	Resets the Place & Route Trace process, forcing it to be run again with the newly generated PCS module.

Note:

Automatic is set as the default option. If either Automatic or Force Settings Generation Only and no sub-options (Process Reset Options) are checked and the HDL module is not generated, the reset pointer is set to Bitstream generation automatically.

After the Generation is finished, the reset marks in the process window will be reset accordingly.



---

## Creating a New Simulation Project Using Simulation Wizard

This section describes how to use the Simulation Wizard to create a simulation project (.spf) file so you can import it into a standalone simulator.

1. In Project Navigator, click **Tools > Simulation Wizard**. The Simulation Wizard opens.
2. In the Preparing the Simulator Interface page click **Next**.
3. In the Simulator Project Name page, enter the name of your project in the Project Name text box and browse to the file path location where you want to put your simulation project using the Project Location text box and Browse button.

When you designate a project name in this wizard page, a corresponding folder will be created in the file path you choose. Click **Yes** in the popup dialog that asks you if you wish to create a new folder.

4. Click either the Active-HDL® or ModelSim® simulator check box and click **Next**.
5. In the Process Stage page choose which type of Process Stage of simulation project you wish to create. Valid types are RTL, Post-Synthesis Gate-Level, Post-Map Gate-Level, and Post-Route Gate-level+Timing. Only those process stages that are available are activated.

Note that you can make a new selection for the current strategy if you have more than one defined in your project.

The software supports multiple strategies per project implementation which allow you to experiment with alternative optimization options across a common set of source files. Since each strategy may have been processed to different stages, this dialog allows you to specify which stage you wish to load.

6. In the Add Source page, select from the source files listed in the Source Files list box or use the browse button on the right to choose another desired source file. Note that if you wish to keep the source files in the local simulation project directory you just created, check the **Copy Source to Simulation Directory** option.
7. Click **Next** and a Summary page appears and provides information on the project selections including the simulation libraries. By default, the Run Simulator check box is enabled and will launch the simulation tool you chose earlier in the wizard in the Simulator Project Name page.
8. Click **Finish**.

The Simulation Wizard Project (.spf) file and a simulation script DO file are generated after running the wizard. You can import the DO file into your current project if desired. If you are using Active-HDL, the wizard will generate an .ado file and if you are using ModelSim, it creates and .mdo file.

*Note: PCS configuration file, (.txt) must be added in step 6.*

## Setting Global Preferences in Diamond

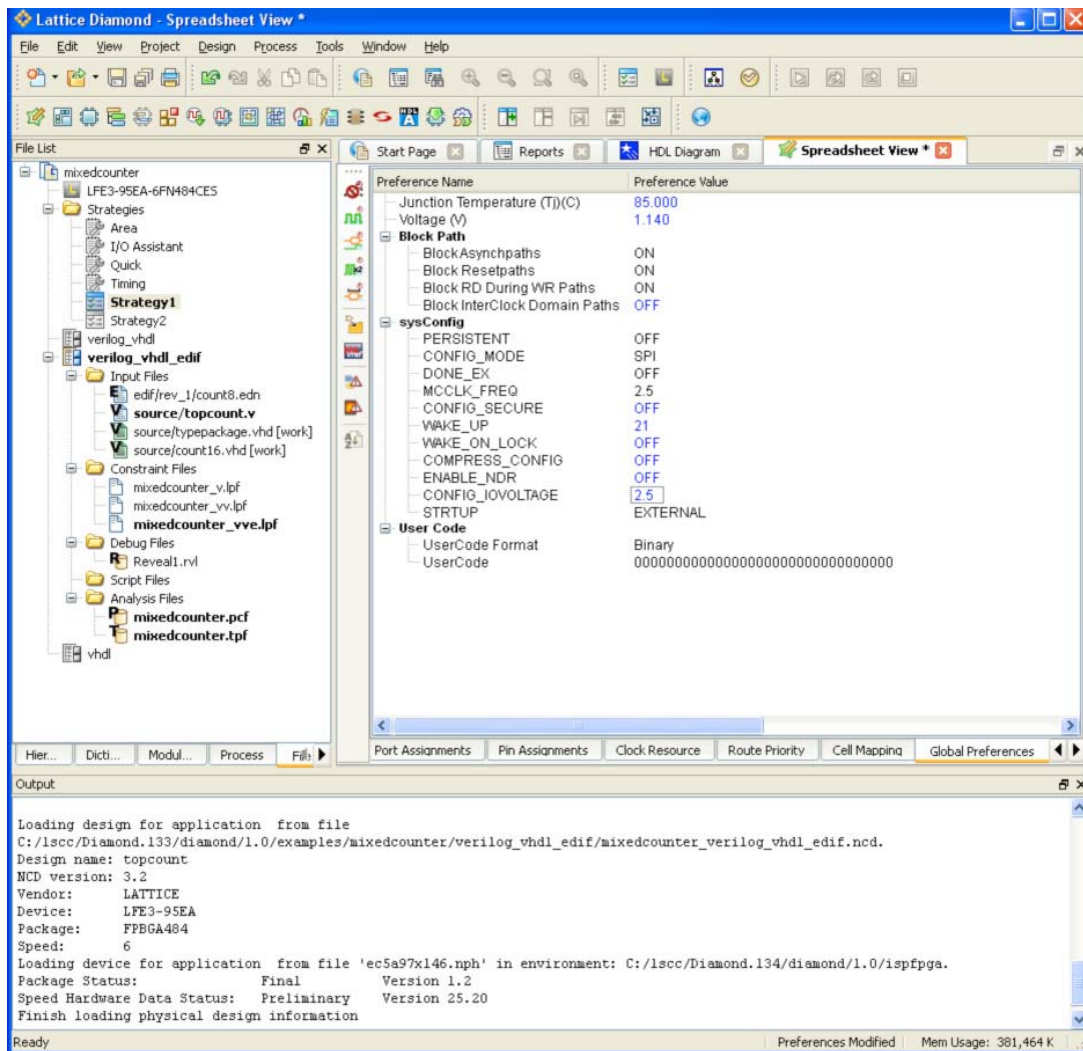
To set any of the Global preferences in Table 15-18, do the following in Diamond:

- Invoke the Spreadsheet View by selecting **Tools > Spreadsheet View**.
- Select the **Global Preferences Tab** beneath the Spreadsheet View pane as shown in Figure 15-21.
- Right-click on the **Preference Value** to be set. In the drop-down menu, select the desired value.

**Table 15-18. Global Preferences**

Preference Name	Values
PERSISTENT	OFF SLAVE_PARALLEL SSPI
CONFIG_MODE	SPI SLAVE_SERIAL JTAG SLAVE_PARALLEL SPIm MASTER_PARALLEL SSPI
DONE_EX	OFF ON
MCCLK_FREQ	2.5 4.3 5.4 6.9 8.1 9.2 10 13 15 20 26 33
CONFIG_SECURE	OFF ON
WAKE_UP	1 4 6 7 10 14 17 21 22 23 24 25
WAKE_ON_LOCK	OFF ON
ENABLE_NDR	OFF ON
CONFIG_IOVOLTAGE	2.5 1.2 1.5 1.8 3.3
STRTUP	EXTERNAL TCLK CCLK MCLK

Figure 15-21. Global Preferences Tab



## Setting Bitstream Generation Options in Diamond

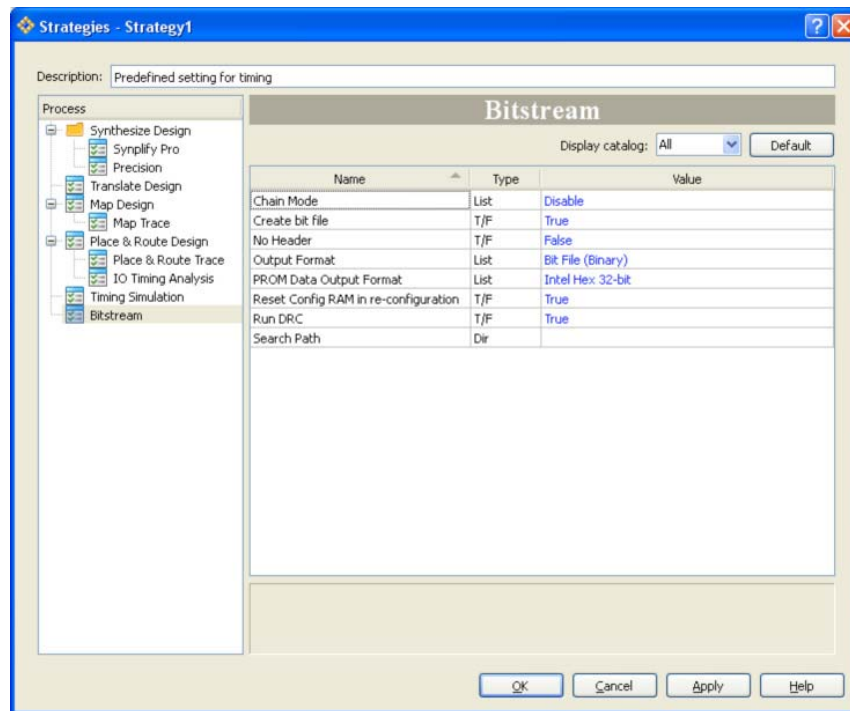
To set any of the Bitstream Generation options listed in Table 15-19, do the following:

- In the **File List** pane, double-click the left mouse button on a **Strategy** to invoke the Strategy settings window.
- In the **Process** pane, left-click on **Bitstream**. All options related to generating a bitstream can be set in this window.

**Table 15-19. Bitstream Generation Options**

Preference Name	Values
Chain Mode	Bypass Disable Flowthrough
Create bit file	True False
No Header	False True
Output Format	Bit File (Binary) Mask and Readback File (ASCII) Mask and Radback File (Binary) Raw Bit file (ASCII)
PROM Data Output Format	Intel Hex 32-bit Motorola Hex 32-bit
Reset Config RAM in re-configuration	True False
Run DRC	True False
Search Path	(Enter a value or browse to specify the search path)

**Figure 15-22. Bitstream Options**



- Double-click the left mouse button on the **Value** you want to set. Select the desired value from the drop-down menu.  
*Note: An explanation of the option is displayed at the bottom of the window. The **Help** button also invokes online help for the option*
- Select **OK**. You can then run the Bitstream File process.

## Setting Security Options in Diamond

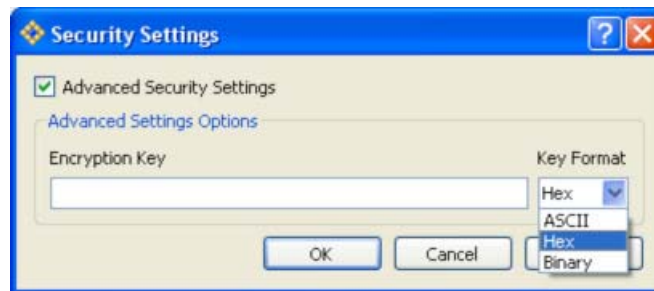
Prior to setting security options in Diamond, you must have installed the Encryption Control Pack. You must also have selected an encrypted device in your project.

To Set Security Settings, do the following:

- Select the **Tools > Security Setting** option. The following dialog box appears:



- If desired, select **Change** and enter a password.
- Select **OK**. A dialog window appears to enter an encryption key.
- If you do not want to enable an encryption key, select **OK**.
- If you do want to enable an encryption key, select the **Advanced Security Settings** checkbox, enter the **Key Format**, and then enter the **Encryption Key**.



- Select **OK** to create the encryption files.

## Introduction

Soft errors occur when high-energy charged particles alter the stored charge in a memory cell in an electronic circuit. The phenomenon first became an issue in DRAM, requiring error detection and correction for large memory systems in high-reliability applications. As device geometries have continued to shrink, the probability of soft errors in SRAM has become significant for some systems. Designers are using a variety of approaches to minimize the effects of soft errors on system behavior.

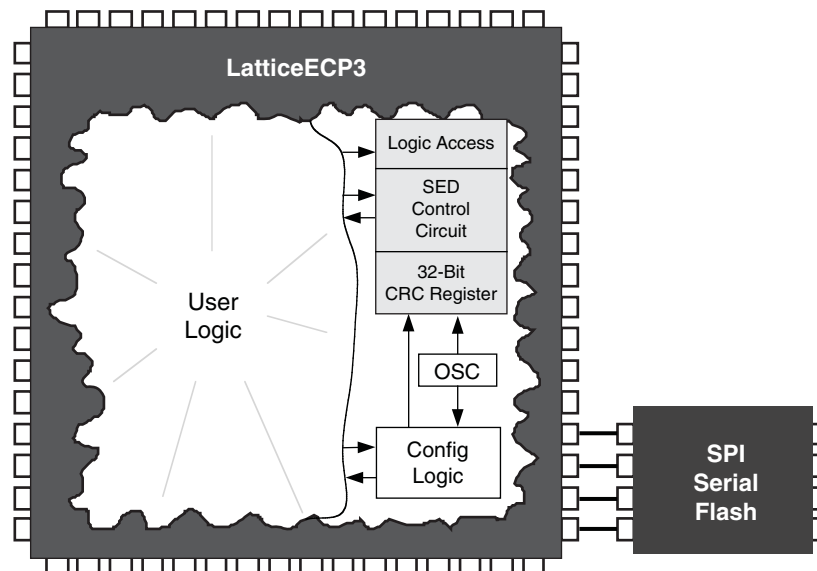
SRAM-based FPGAs store logic configuration data in SRAM cells. As the number and density of SRAM cells in an FPGA increase, the probability that a soft error will alter the programmed logical behavior of the system increases. A number of approaches have been taken to address this issue, but most involve Intellectual Property (IP) cores that the user instantiates into the logic of their design, using valuable resources and possibly affecting design performance.

This document describes the hardware based soft error detect (SED) approach taken by Lattice Semiconductor for LatticeECP3™ FPGAs.

## SED Overview

The SED hardware in the LatticeECP3 devices consists of an access point to FPGA configuration memory, a controller circuit, and a 32-bit register to store the CRC for a given bitstream (see Figure 16-1). The SED hardware reads serial data from the FPGA's configuration memory and calculates a CRC. The calculated CRC is then compared with the expected CRC that was stored in the 32-bit register. If the CRC values match it indicates that there has been no configuration memory corruption, but if the values differ an error signal is generated. SED checking does not impact the performance or operation of the user logic.

**Figure 16-1. System Block Diagram<sup>1</sup>**



1. Any kind of configuration memory can be used, including the SPI configuration shown.

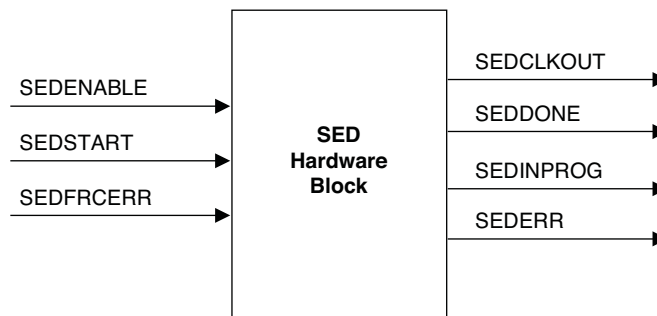
Note that the calculated CRC is based on the particular arrangement of configuration memory for a particular design. Consequently, the expected CRC results cannot be specified until after the design is placed and routed. The ispLEVER® bitstream generation software analyzes the configuration of a placed and routed design and updates the 32-bit SED CRC register contents during bitstream generation. EBR and distributed memory contents are ignored in CRC generation.

The following sections describe the LatticeECP3 SED implementation and flow, along with some sample code to get started with.

## Hardware Description

As shown in Figure 16-2, the LatticeECP3 SED hardware has several inputs and outputs that allow the user to control, and monitor, SED behavior.

Figure 16-2. Signal Block Diagram



## Signal Descriptions

Table 16-1. SED Signal Descriptions

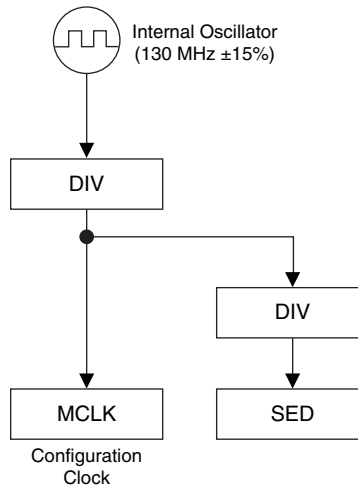
Signal Name	Direction	Active	Description
SEDENABLE	Input	High	SED enable
SEDCLKOUT	Output	N/A	Output clock
SEDSTART	Input	High	Start SED cycle
SEDINPROG	Output	High	SED cycle is in progress
SEDDONE	Output	High	SED cycle is complete
SEDFRCERR	Input	High	Force an SED error flag
SEDERR	Output	High	SED error flag

### SEDCLK

Clock input to the SED hardware.

This clock is derived from the LatticeECP3's on-chip oscillator. The on-chip oscillator's output goes through a divider to create MCCLK. MCCLK goes through another divider to create SEDCLK. This is shown in Figure 16-3.

**Figure 16-3. Divider Diagram**



The software default for MCCLK is 2.5 MHz with a MCCLK divider of 1, but this can be modified using the MCCLK\_FREQ global preference in the ispLEVER pre-map Design Planner (see TN1169, [LatticeECP3 sysCONFIG Usage Guide](#), for possible values of MCCLK).

The divider for SEDCLK can be set to 1, 2, 4, 8, 16 or 32. The default is 1, so the default SEDCLK frequency is 2.5 MHz. The divider value can be set using a parameter, see the example code at the end of this document. Care must be taken to ensure that the SEDCLK setting is above 2.5 MHz.

**Figure 16-4. Possible MCLK Values (MHz)**

2.5	10	33
4.3	13	
5.4	15	
6.9	20	
8.1	26	
9.2	30	

## SEDENABLE

Active high input to the SED hardware.

State	Description
1	Enables output of SEDCLKOUT, arms SED hardware.
0	Aborts SED and forces all SED hardware outputs low.

## SEDCLKOUT

Gated version of SEDCLK, SEDCLKOUT is gated by SEDENABLE, therefore you can use this signal to synchronize the inputs.



## SEDSTART

Active high input to the SED hardware. If sedstart is tied high, it will keep rechecking until it is brought low and the current cycle finishes. It can also be pulsed, and can be brought low after SEDINPROG goes high, if desired. It is a level-sensitive signal.

State	Description
1	Start error detection. Must be high a minimum of one SEDCLKIN period.
0	No action.

## SEDFRCERR

Active high input to the SED hardware. As long as this signal is held low, SEDERR will stay active. It can be asserted at any time.

State	Description
1	Forces SEDERR high, simulating an SED error.
0	No action.

## SEDINPROG

Active high output from the SED hardware.

State	Description
1	SED checking is in progress, goes high on the clock following SEDSTART high.
0	SED checking is not active.

## SEDDONE

Active high output from the SED hardware.

State	Description
1	SED checking is complete. Reset by a high on SEDSTART or a low on SEDENABLE.
0	SED checking is not complete.

## SEDERR

Active high output from the SED hardware. SEDERR indicates that a CRC mismatch was found between the FPGA's configuration bits and the CRC value held in the 32-bit CRC register. It performs no other action than flagging the mismatch. It is up to the designer to evaluate and react to the assertion of SEDERR.

State	Description
1	SED has detected an error. Reset by SEDENABLE going low.
0	SED has not detected an error.

## AUTODONE

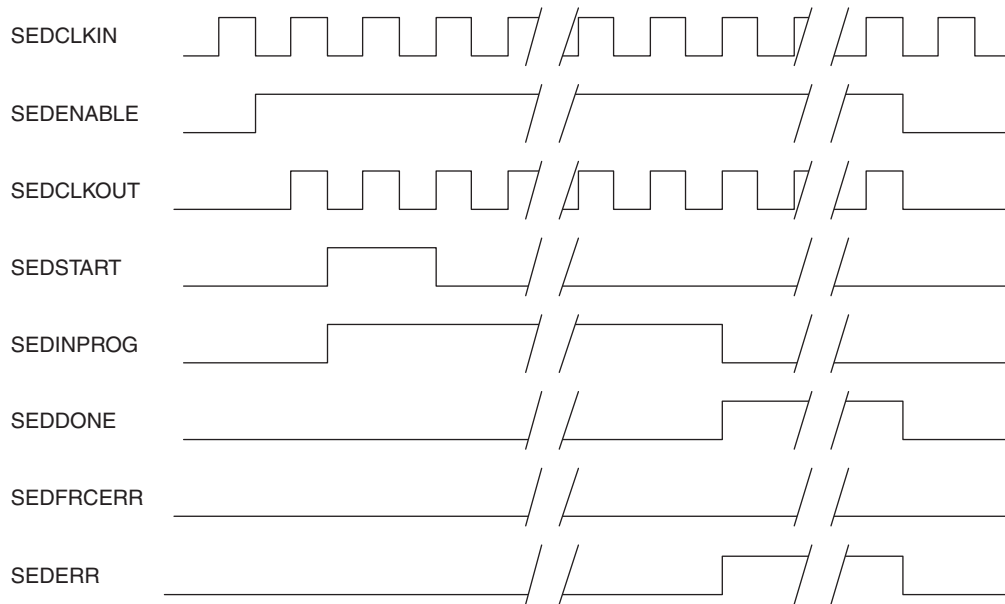
Reserved for future use.

## MCCLK\_freq Attribute

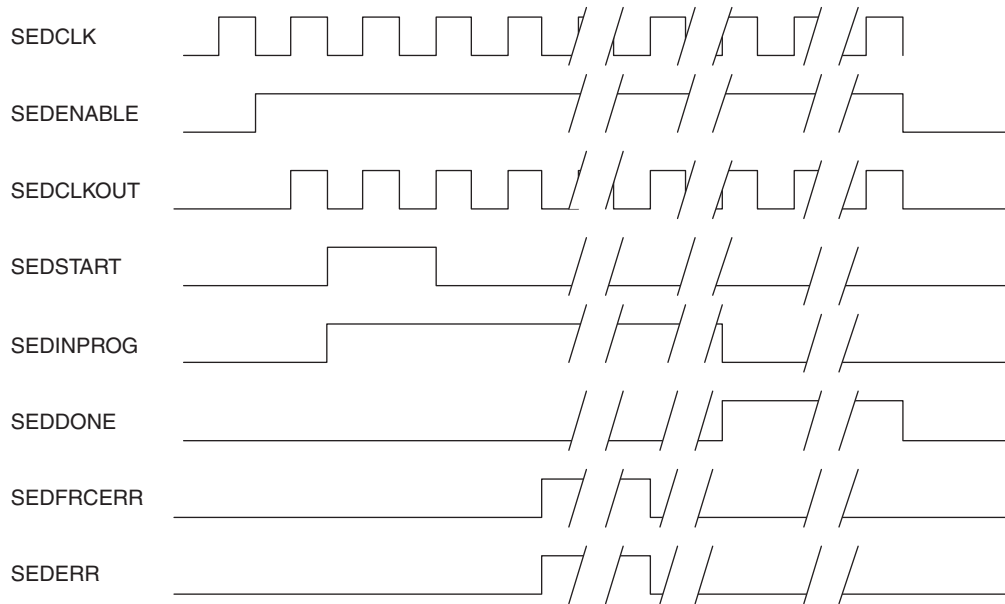
Frequency attribute which specifies how fast MCCLK is for simulation purposes. For design implementation purposes, MCCLK must be set using ispLEVER Design Planner. Both the simulation and Design Planner setting should match or else there will be a DRC error.

## SED Flow

Figure 16-5. Timing Diagram



Normal Failure



Failure Forced With SEDFRCERR

Note: SED, by its inherent nature, has limited simulation support. In simulation, use the SEDFRCERR signal to force errors.

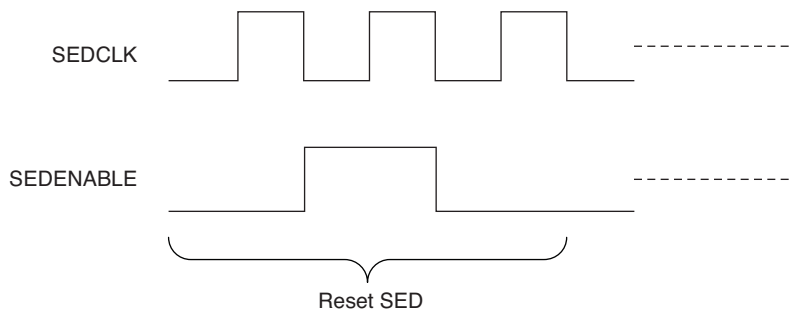
LatticeECP3 SED flow:

1. Toggle SEDENABLE after each time the LatticeECP3 is reprogrammed:
  - a. Deassert SEDENABLE for one clock cycle.
  - b. Assert SEDENABLE for one clock cycle.
  - c. Deassert SEDENABLE for one clock cycle.

This initializes the SED system and must be done prior to testing for soft errors. Following this sequence prevents spurious assertion of the SEDERR output.

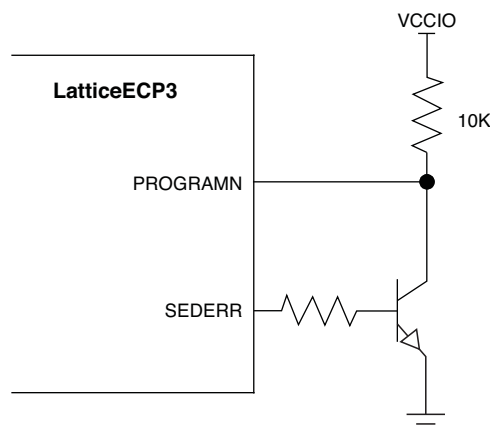
2. After initializing the SED logic (step 1 above) SED is ready to use. First, assert SEDENABLE.
3. Assert SEDSTART (active high). SEDINPROG will go high. If SEDDONE is already high it will go low.
4. SED will read back the SRAM data and calculate the CRC.
5. The calculated CRC is compared with the stored CRC and SEDERR is updated, SEDINPROG goes low, and SEDDONE goes high.
6. When SEDERR is high, it can be reset by bringing SEDENABLE low or reconfiguring the FPGA.

**Figure 16-6. SED Initialization**



To trigger reconfiguration upon error detection, Figure 16-7 shows one possible solution.

**Figure 16-7. Example Schematic**



## SED Run Time

The amount of time needed to perform an SED check depends on the density of the device and the frequency of SEDCLK. There will also be some overhead time for calculation, but it is fairly short in comparison. An approximation of the time required can be found by using the following formula:

Time (max) = Num\_configuration\_bits (Mb) / (SEDCLK \* 0.85) # 15% low side clock

Time (min) = Num\_configuration\_bits (Mb) / (SEDCLK \* 1.15) # 15% high side clock

The density of the FPGA determines the value of the Num\_configuration\_bits value. Table 16-2 provides information on the number of configuration bits for LatticeECP3 FPGAs. SEDCLK is frequency in MHz. Time is in seconds.

For example, if the design uses a LatticeECP3 with 95K look-up tables and the SEDCLK is the software default of 2.5 MHz:

Time (max) = 19 Mbits / (2.5 MHz \* 0.85) = 8.95 seconds

Time (min) = 19 Mbits / (2.5 MHz \* 1.15) = 6.6 seconds

In this example, SED checking will take between 6.6 and ~9 seconds. Remember that this happens in the background and does not affect user logic performance.

Note that the internal oscillator used to generate SEDCLK can vary by ±15%, as shown in the min/max calculations.

**Table 16-2. SED Run Time**

Density	Bitstream Size (Mb)	Run Time <sup>1</sup> (Seconds)	Run Time (Max.)	Run Time (Min.)
ECP3-95	19	7.6	9	6.6

1. Based on SEDCLK = 2.5 MHz.

## Sample Code

The following simple example code shows how to instantiate the SED. In the example the outputs of the SED hardware have been routed to FPGA output pins.

Note that the SEDCA primitive is part of ispLEVER 7.2 or later.

## VHDL Example

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity SEDCA_VHDL_Test is
port(
    SED_Done : out std_logic;
    SED_In_Prog : out std_logic;
    SED_Clk_Out : out std_logic;
    SED_Out : out std_logic;
    SEDENABLEx : in std_logic;
    SEDSTARTx : in std_logic);
end;

architecture behavioral of SEDCA_VHDL_Test is
    component SEDCA
        generic (OSC_DIV : integer :=1 ;
                CHECKALWAYS : string := "DISABLED";
                AUTORECONFIG: string := "OFF" ;

```

```
        MCCLK_FREQ : string := "2.5" ;
        DEV_DENSITY : string := "95K" );

port (
    SEDENABLE : in std_logic;
    SEDSTART  : in std_logic;
    SEDFRCERR : in std_logic;
    SEDERR    : out std_logic;
    SEDDONE   : out std_logic;
    SEDINPROG : out std_logic;
    SEDCLKOUT : out std_logic;
end component;

begin
    inst1: SEDCA
    port map (
        SEDENABLE => SEDENABLEx,
        SEDSTART  => SEDSTARTx,
        SEDFRCERR => '0',
        SEDERR    => SED_Out,
        SEDDONE   => SED_Done,
        SEDINPROG => SED_In_Prog,
        SEDCLKOUT => SED_Clk_Out
    );
end behavioral;
```

## Verilog Example

```
module SEDCA_Verilog_Test( SEDFRCERR, SEDERR, SEDDONE, SEDINPROG, SEDCLKOUT,
    SEDENABLEx, SEDSTARTx);

    input SEDFRCERR, SEDENABLEx, SEDSTARTx;
    output SEDERR, SEDDONE, SEDINPROG, SEDCLKOUT;

    SEDCA
        # (.CHECKALWAYS("DISABLED"),
        .AUTORECONFIG("OFF"),
        .OSC_DIV(1),
        .DEV_DENSITY("95K"),
        .MCCLK_FREQ("2.5"))
    sed_ip (
        .SEDENABLE(SEDENABLEx),
        .SEDSTART(SEDSTARTx),
        .SEDFRCERR(SEDFRCERR),
        .SEDERR(SEDERR),
        .SEDDONE(SEDDONE),
        .SEDINPROG(SEDINPROG),
        .SEDCLKOUT(SEDCLKOUT)
    );
endmodule
```

---

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
November 2009	01.1	Updated SED flow.
		Updated Example Schematic diagram.
		Updated VHDL and Verilog sample code.
January 2011	01.2	Removed references to AUTODONE in Sample Code section.
February 2012	01.3	Updated document with new corporate logo.
July 2013	01.4	Updated Possible MCLK Values (MHz) table.
		Updated Technical Support Assistance information.

## Introduction

The LatticeECP3™ and LatticeECP2M™ families are low-cost FPGA product lines offering high-end features such as high-speed, embedded SERDES (SERializer/DESerializer) interfaces. These devices feature up to 16 SERDES channels with data rates of up to 3.125 Gbps.

This technical note outlines two experiments that measure the SERDES backplane transmission performance thresholds of the LatticeECP3 and LatticeECP2M devices. These experiments include:

- **Eye Diagram Experiment:** Uses eye diagrams to explore FPGA performance over a standard reference backplane.
- **Data Rate Experiment:** Uses bit error rate measurement techniques to verify FPGA backplane data rate limits at varying speeds and trace lengths.

Both experiments use a bit error rate tester (BERT) for pattern generation, a Tyco HM-Zd as the backplane and a LatticeECP3 or LatticeECP2M FPGA as the device under test. Both experiments collect data at 2.5 Gbps and 3.125 Gbps and use pre-emphasis adjustments to optimize transmitter performance.

## Eye Diagram Experiment

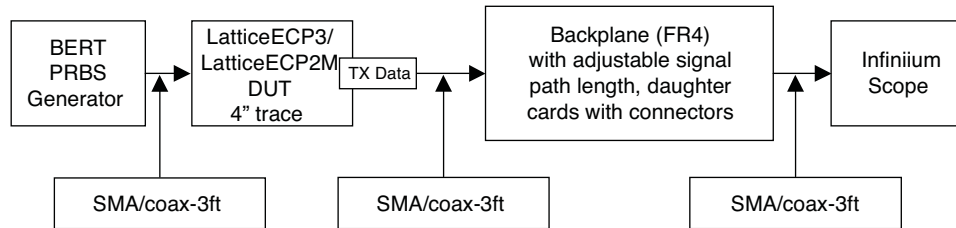
The eye diagram experiment checks the ability of the LatticeECP3 or LatticeECP2M FPGA to drive SERDES signals through a backplane at different pre-emphasis levels and data rates. It provides a visual, qualitative measurement of link performance by persistently sampling the signal at the receive end of the backplane.

In this configuration, a pseudo-random bit sequence (PRBS) pattern is generated and then sent into the LatticeECP3 or LatticeECP2M DUT. The PRBS is looped back and transmitted by the DUT into the backplane. The signal is routed out of the backplane, where it is sampled on an oscilloscope and eye diagrams are assembled. This experiment was performed at 2.5 Gbps and 3.125 Gbps with different levels of pre-emphasis. See Figure 17-1 for an illustration.

The equipment used in this test includes:

- Tyco HM-Zd Quad Route Test backplane with two daughter cards
- Agilent Infiniium DSO81304B 13 GHz oscilloscope
- High-quality coaxial cabling (rated for >3.125 Gbps) with SMA connectors
- Agilent 81250 3.7 GHz Parallel Bit Error Tester (ParBERT)
- Agilent 81130A function/pulse generator
- Agilent E3610A power supply
- Thermonics Thermostream for temperature control
- Internal LatticeECP3 and LatticeECP2M evaluation boards

**Figure 17-1. Eye Diagram Experiment Setup**



## Backplane Specifications

- Tyco Electronics HM-Zd Quad Route Test backplane with daughter cards
- Backplane – 200 mils thick with 14 layers, made from Nalco 4000-FR4 material
- Signal layers 10 mil wide (1/2 copper thickness) designed for 100-ohm differential impedance traces
- Daughter cards – 93 mil thick with 14 layers
- Daughter cards – 6 mil wide, 100-ohm differential impedance traces
- Backplane trace length 40 inches

## Test Setup Parameters

- DUT (LatticeECP3 or LatticeECP2M fpBGA)
- SERDES VCC supply values: 1.2V -5%
- Ambient temperature: 125°C
- Data pattern = PRBS (7-bit polynomial)
- Socketed, nominal device

## Eye Diagram Measurements

Receiver end eye diagrams are an excellent measurement of expected link performance. This experiment tested a 40-inch length trace path, whereas most applications will have a 12 to 24 inch path. Eye diagrams were taken at 2.5 Gbps and 3.125 Gbps at varied pre-emphasis levels.

Pre-emphasis compensates for signal losses that occur with higher speeds and longer trace lengths. In general, increasing pre-emphasis will increase the signal quality for an improved eye diagram. The FPGA provides eight programmable pre-emphasis levels, available on a per-channel basis. To illustrate the progressive advantages of increased pre-emphasis, Tables 17-1 and 17-2 show eye diagrams taken at six of these pre-emphasis settings, each sampled at both 2.5 Gbps and 3.125 Gbps. Eye opening widths (measured in pS) and peak-to-peak voltage swings (in mV) are listed with each sample. For each of these measurements, larger is better.



Table 17-1. LatticeECP3 Eye Diagram Measurements

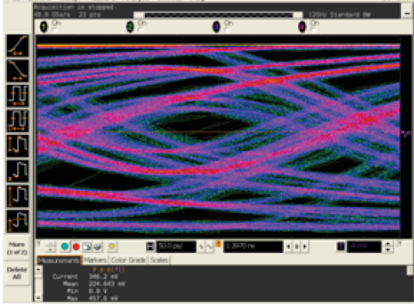
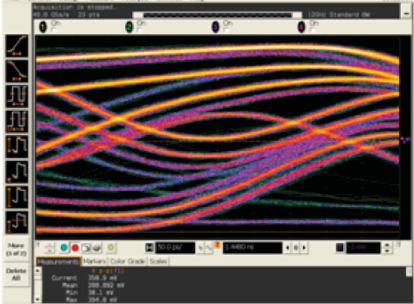
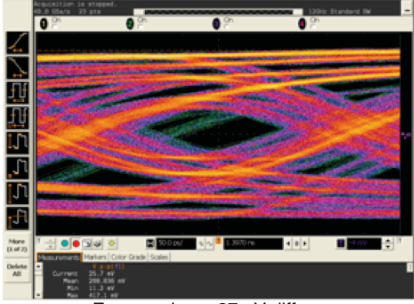
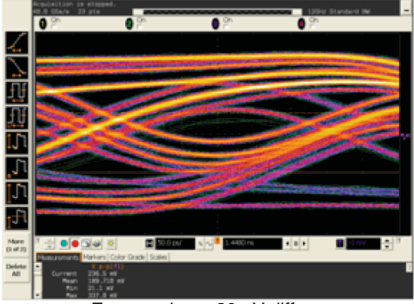
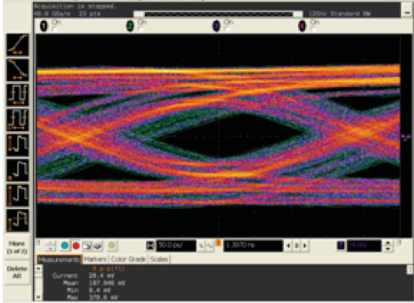
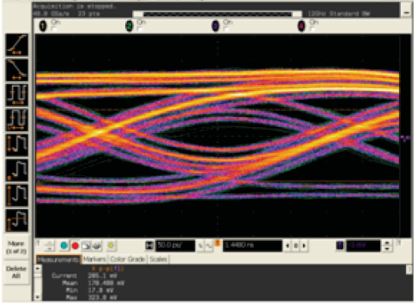
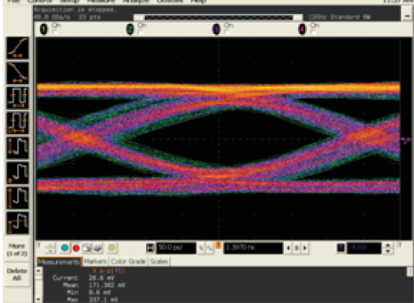
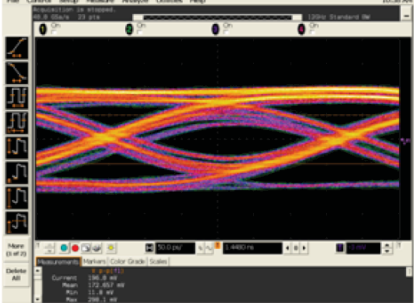
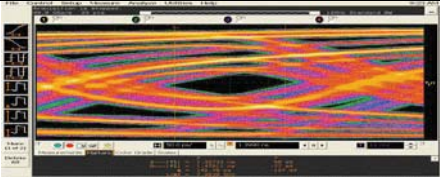
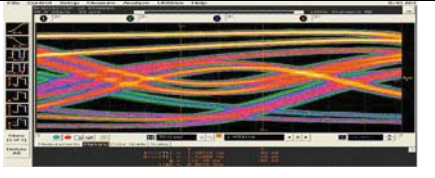
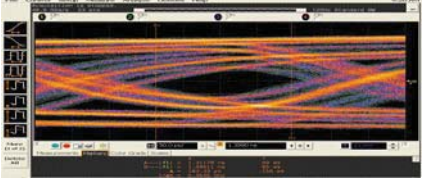
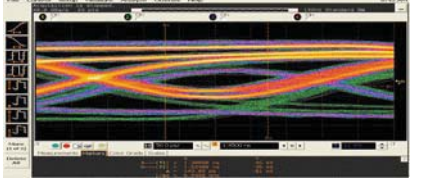
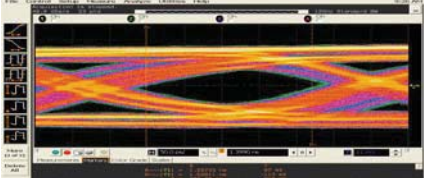
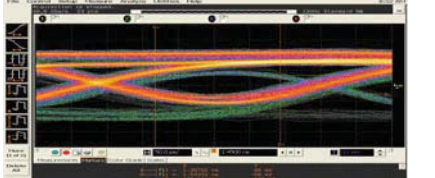
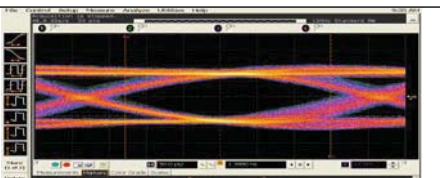
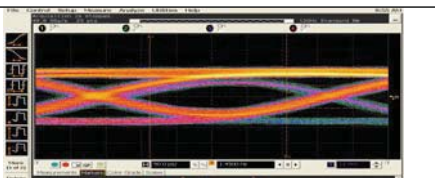
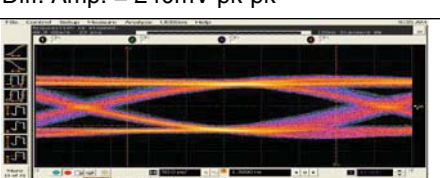
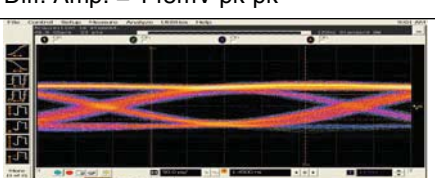
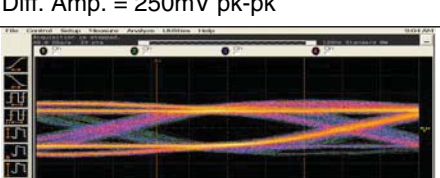
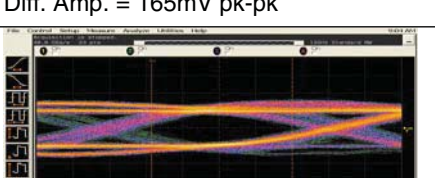
Pre-emphasis	2.5 Gbps	3.125 Gbps
<p>Disabled 0%</p>	 <p>No detectable eye</p>	 <p>No detectable eye</p>
<p>1 (5%)</p>	 <p>Eye opening = 87mV diff p-p</p>	 <p>Eye opening = 30mV diff p-p</p>
<p>2 (12%)</p>	 <p>Eye opening = 146mV diff p-p</p>	 <p>Eye opening = 51mV diff p-p</p>
<p>3 (18%)</p>	 <p>Eye opening = 203mV diff p-p</p>	 <p>Eye opening = 103mV diff p-p</p>

Table 17-2. LatticeECP2M Eye Diagram Measurements

Pre-Emphasis Setting	2.5 Gbps	3.125 Gbps
Pre-Emphasis Disabled	 <p>Eye opening= 140pS, Diff. Amp. = 130mV pk-pk</p>	 <p>Eye opening= 110pS, Diff. Amp. = 45mV pk-pk</p>
1 (16%)	 <p>Eye opening= 180pS, Diff. Amp. = 155mV pk-pk</p>	 <p>Eye opening= 140pS, Diff. Amp. = 80mV pk-pk</p>
2 (36%)	 <p>Eye opening= 220pS, Diff. Amp. = 210mV pk-pk</p>	 <p>Eye opening= 176pS, Diff. Amp. = 123mV pk-pk</p>
4 (44%)	 <p>Eye opening= 265pS, Diff. Amp. = 240mV pk-pk</p>	 <p>Eye opening= 194pS, Diff. Amp. = 145mV pk-pk</p>
5 (56%)	 <p>Eye opening= 270pS, Diff. Amp. = 250mV pk-pk</p>	 <p>Eye opening= 210pS, Diff. Amp. = 165mV pk-pk</p>
6 (80%)	 <p>Eye opening= 270pS, Diff. Amp. = 210mV pk-pk</p>	 <p>Eye opening= 195pS, Diff. Amp. = 160mV pk-pk</p>

## Results and Conclusion

Tables 17-1 and 17-2 show that for both data rates the eye opening width and the pk-pk height were maximized at a pre-emphasis of 5. However, if the receiver sensitivity requirements are met, the user might choose a lower pre-emphasis setting in the interest of power savings and lower EM radiation. For example, the LatticeECP3 and LatticeECP2M SERDES receive ports have a minimum input differential sensitivity of 100 mV, so even with pre-emphasis disabled, this requirement would be met at 2.5 Gps (130 mV pk-pk).

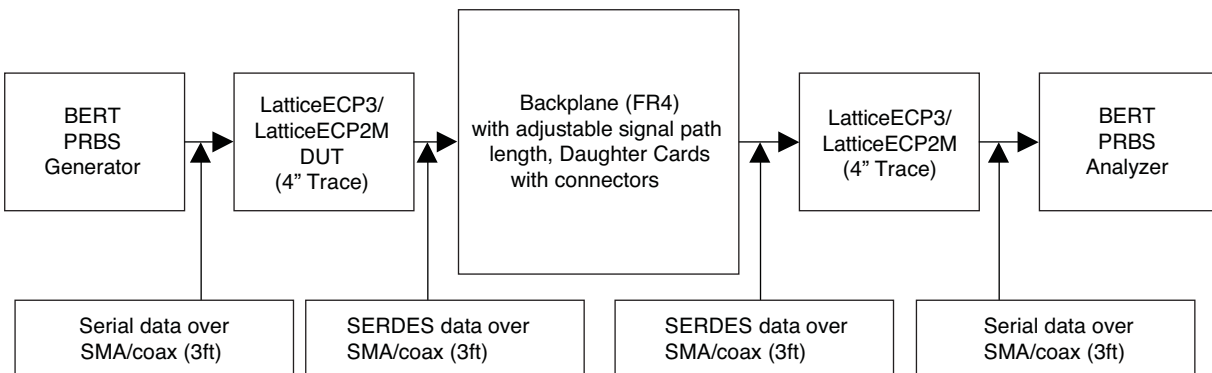
The eye diagrams in this experiment demonstrate that the LatticeECP3 and LatticeECP2M effectively provide high-quality signaling in a long trace, backplane design. These diagrams also show that pre-emphasis settings can be used to optimize SERDES performance.

## Data Rate Experiment

The data rate experiment checks the FPGA SERDES transmission performance using a Bit Error Rate Tester (BERT) for expected/received data comparisons in a pass/fail context. The configuration begins by generating a PRBS sequence at a BERT, then sending the pattern into a LatticeECP3 or LatticeECP2M DUT. The FPGA loops the data back to the backplane as a SERDES bitstream. The SERDES data then exits the backplane and is received by the LatticeECP3 or LatticeECP2M (not necessarily the same device as the earlier DUT), which loops the data back to a BERT analyzer. Finally, the BERT compares the incoming bitstream to an expected data pattern and keeps track of how many mismatches are found. A typical expected bit error rate (BER) is less than  $1 \times 10^{-12}$  errors per second. See Figure 17-2 for an illustration.

The equipment used for this test was the same as used for the eye diagram experiment.

**Figure 17-2. Data Rate Experiment**



## Backplane Specifications

- Backplane specifications are the same as for the eye diagram experiment
- Total trace length: 60 inches for 2.5 Gbps testing and 40 inches for 3.125 Gbps testing

## Test Setup Parameters

- DUT (LatticeECP3 or LatticeECP2M fpBGA)
- Ambient temperature: 125°C
- SERDES VCC supply values: 1.2V -5%
- Socketed device
- All process variations
- Pre-emphasis setting: 4
- Equalization: 8 db

---

## Data Rate Measurements

The data rate experiment was performed against samples from five process variations. Each sample was tested at 2.5 Gbps and 3.125 Gbps. The pass criterion was BER of less than  $1 \times 10^{-12}$  errors per second.

## Results and Conclusions

For all process splits, the FPGA samples achieved a BER of better than  $1 \times 10^{-12}$  at 3.125 Gbps and 2.5 Gbps. This indicates that with a pre-emphasis setting of 4, the LatticeECP3 and LatticeECP2M can drive SERDES data error-free up to 40 inches of backplane at 3.125 Gbps and 40 inches of backplane with margin at 2.5 Gbps.

## Conclusions and Design Guidelines

The experiments detailed in this technical note measured the ability of the LatticeECP3 and LatticeECP2M to reliably transmit SERDES data streams in a typical backplane design. The data-rate experiment used a statistical pass/fail scenario, whereas the eye-diagram experiment provided a visual, qualitative measurement. Both experiments concluded that the LatticeECP3 and LatticeECP2M deliver high-quality SERDES transmission over long backplane distances and over a broad range of data rates.

In both experiments, pre-emphasis was used to optimize LatticeECP3 and LatticeECP2M backplane performance. The eye diagram experiment went further to demonstrate that increased pre-emphasis provides a larger eye opening. Pre-emphasis settings are available on a per-channel basis.

Both experiments proved LatticeECP3 and LatticeECP2M performance at 2.5 Gbps and 3.125 Gbps. The data-rate experiment went further to show that devices from all process variations meet these performance goals.

Due to the number of factors in a high-speed PCB design, it is difficult to predict system performance in all scenarios. The data rate experiment illustrated robust performance at 3.125 Gbps over 40 inches in typical conditions, even for the slowest speed grades of the devices. The maximum trace length that can be implemented for an individual system environment may vary and should be evaluated by the individual designer.

The following suggestions will help designers optimize their high-speed LatticeECP3 and LatticeECP2M applications:

1. It is critical that all SERDES path cables and connectors be carefully selected. Cabling should be high-quality coaxial and connectors should be SMA. Both should be characterized for the intended frequency range. The designer should pay close attention to the parasitic performance of these devices.
2. Backplane and port cards should be implemented with good high-speed design practices in mind. For more details see TN1033, [High-Speed PCB Design Considerations](#).
3. To improve the performance of receivers, use the LatticeECP3 or LatticeECP2M programmable equalization settings. For example, 8 db was used in the data rate experiment in this technical note.
4. Use analog circuit simulation tools to assess backplane performance signal integrity issues prior to building models. Contact Lattice for information about obtaining LatticeECP3 or LatticeECP2M SERDES HSPICE models for your simulations.
5. Early lab experimentation of long paths are recommended prior to full system model design in order to reduce technical risk. Eye diagram and bit error rate experiments are recommended.

---

## References

- TN1118, [LatticeSC High-Speed Backplane Measurements](#)
- TN1176, [LatticeECP3 SERDES/PCS Usage Guide](#)
- TN1124, [LatticeECP2M SERDES/PCS Usage Guide](#)
- TN1033, [High-Speed PCB Design Considerations](#)
- TN1114, [Electrical Recommendations for Lattice SERDES](#)

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2007	01.0	Initial release.
March 2007	01.1	Updated LatticeECP2M Eye-Diagram Measurements table.
April 2007	01.2	Updated Eye-Diagram Measurements section.
May 2010	01.3	Added LatticeECP3 FPGA data.
February 2012	01.4	Updated document with new corporate logo.
June 2013	01.5	Updated Technical Support Assistance information.



## Introduction

When designing complex hardware using the LatticeECP3™ FPGA, designers must pay special attention to critical hardware configuration requirements. This technical note steps through these critical hardware implementation items relative to the LatticeECP3 device. The document does not provide detailed step-by-step instructions but gives a high-level summary checklist to assist in the design process.

The device family consists of FPGA LUT densities ranging from 17K to 150K. This technical note assumes that the reader is familiar with the LatticeECP3 device features as described in the [LatticeECP3 Family Data Sheet](#). The data sheet includes the functional specification for the device. Topics covered in the data sheet include but are not limited to the following:

- High-level functional overview
- Pinouts and packaging information
- Signal descriptions
- Device-specific information about peripherals and registers
- Electrical specifications

Please refer to the LatticeECP3 Family Data Sheet for the device-specific details.

- DS1021 – [LatticeECP3 Family Data Sheet](#)

The critical hardware areas covered in this technical note are:

- Power supplies as they relate to the LatticeECP3 power supply rails and how to connect them to the PCB and the associated system
- Configuration mode selection for proper power-up behavior
- Device I/O interface and critical signals

**Important:** Users should refer to the following documents for detailed recommendations.

- TN1114, [Electrical Recommendations for Lattice SERDES](#)
- TN1177, [LatticeECP3 sysIO Usage Guide](#)
- TN1169, [LatticeECP3 sysCONFIG™ Usage Guide](#)
- TN1180, [LatticeECP3 High-Speed I/O Interface](#)
- TN1033, [High-Speed PCB Design Considerations](#)
- TN1176, [LatticeECP3 SERDES/PCS Usage Guide](#)
- TN1068, [Decoupling and Bypass Filtering for Programmable Devices](#)
- TN1084, [LatticeSC SERDES Jitter](#)
- TN1195, LatticeECP3 SERDES Characterization Report (available under NDA, contact your local Lattice sales representative)
- HSPICE SERDES CML simulation package and die models in RLGC format (available under NDA, contact the license administrator at [lic\\_admin@latticesemi.com](mailto:lic_admin@latticesemi.com))
- [LatticeECP3-related pinout information](#) can be found on the Lattice web site.

## Power Supplies

All supplies including  $V_{CC}$ ,  $V_{CCAUX}$  and  $V_{CCIO8}$  power supplies determine the LatticeECP3 internal “power good” condition. These supplies need to be at a valid and stable level before the device can become operational. Several other supplies including  $V_{CCPLL}$ ,  $V_{CCIB}$ , and  $V_{CCOB}$  are used in conjunction with on-board SERDES and phase-locked loops. Table 18-1 shows the power supplies and the appropriate voltage levels for each supply.

**Table 18-1. LatticeECP3 FPGA Power Supplies**

Supply	Voltage (Nominal Value)	Description
VCC	1.2V	FPGA core power supply.
VCCA	1.2V	Power supplies analog SERDES blocks. Should be isolated and “clean” from excessive noise.
VCCPLL[L:R]	3.3V	Power supply for PLL. Should be isolated and “clean” from excessive noise.
VCCAUX	3.3V	Auxiliary power supply
VCCIO[0-3] <sup>1</sup> & VCCIO[6-8] <sup>1</sup>	1.2V to 3.3V	I/O power supply. Seven general-purpose I/O banks and each bank has its own supply $V_{CCIO0}$ to $V_{CCIO3}$ and $V_{CCIO6}$ to $V_{CCIO8}$ . $V_{CCIO8}$ is used in conjunction with the pins dedicated and shared with device configuration. $V_{CCIO0}$ to $V_{CCIO3}$ and $V_{CCIO6}$ to $V_{CCIO8}$ are optionally used based on per bank usage of I/O
VCCJ	1.2V to 3.3V	JTAG power supply for the TAP controller port.
VCCIB	1.2V to 1.5V	CML input termination voltage
VCCOB	1.2V to 1.5V	CML output termination voltage

1. Banks 4 and 5 do not exist on the LatticeECP3. Therefore, VCCIO4 and VCCIO5 are not available.

The LatticeECP3 FPGA device has a power-up reset state machine that depends on various power supplies.

These supplies should come up monotonically. A power-up reset counter will begin to count after all of the approximate conditions are met:

- VCC reaches 0.8V or above
- VCCAUX reaches 2.7V or above
- VCCIO[8] reaches 0.8V or above

Initialization of the device will not proceed until the last power supply has reached its minimum operating voltage.

## LatticeECP3 SERDES/PCS Power Supplies

Supplies dedicated to the operation of the SERDES/PCS include VCCA, VCCIB, VCCOB. All VCCA supply pins must always be powered to the recommended operating voltage range regardless of the SERDES use.

VCCIB and VCCOB can be left floating for unused SERDES channels. Unused channel outputs are tristated, with approximately 10 KOhm internal resistor connecting between the differential output pair.

When using the SERDES with 1.5V VCCIB or VCCOB, the SERDES should not be left in a steady state condition with the 1.5V power applied and the VCCA 1.2V power not applied. Both the 1.2V and the 1.5V power should be applied to the SERDES at nominally the same time. The normal variation in ramp-up times of power supplies and voltage regulators is not a concern.

It is very important that the VCCA supply be low-noise and isolated from heavily loaded switching supplies. Please refer to TN1114, [Electrical Recommendations for Lattice SERDES](#), for recommendations.

## Power Estimation

Once the LatticeECP3 device density, package and logic implementation is decided, power estimation for the system environment should be determined based on the software Power Calculator provided as part of the ispLEVER® design tool. When estimating power, the designer should keep two goals in mind:

1. Power supply budgeting should be based on the maximum of the power-up in-rush current, configuration current or maximum DC and AC current for the given system's environmental conditions.
2. The ability for the system environment and LatticeECP3 device packaging to be able to support the specified maximum operating junction temperature. By determining these two criteria, LatticeECP3 power requirements are taken into consideration early in the design phase.

## Configuration Considerations

The LatticeECP3 includes provisions to program the FPGA via a JTAG interface or through several modes utilizing the sysCONFIG port. The JTAG port includes a 4-pin interface. The interface requires the following PCB considerations.

**Table 18-2. JTAG Pin Recommendations**

JTAG Pin	PCB Recommendation
TDI	4.7K Pull-up to VCCJ
TMS	4.7K Pull-up to VCCJ
TDO	4.7K Pull-up to VCCJ
TCK	4.7K Pull-down

Every PCB should have easy access to FPGA JTAG pins. This enables debugging in the final system. For best results, route the TCK, TMS, TDI and TDO signals to a common test header along with VCCJ and ground.

Using other programming modes requires the use of the CFG[2:0] input pins. For JTAG, the MODE pins are not used. The CFG[2:0] pins include weak internal pull-ups. It is recommended that 5-10K external resistors be used when using the sysCONFIG modes. Pull-up resistors should be connected to VCCIO8.

The use of external resistors is always needed if the configuration signals are being used to handshake to other devices. Recommended 4.7K pull-up resistors to VCCIO8 and pull-down to board ground should be used on the following pins.

**Table 18-3. Pull-up/Pull-down Recommendations for Configuration Pins**

Pin	PCB Connection
PROGRAMN	Pull-up
INITN	Pull-up
CCLK	Pull-down
CFG[0:2]	See Table 18-4. 1 = 4.7K pull-up, 0 = GND.



**Table 18-4. Configuration Pins Needed per Programming Mode**

Configuration Mode	Bus Size	Dedicated CFG[0:2]	sysCONFIG Pin Mapping + Dedicated Pins			Dedicated Pins
			Clock Pin	I/O	Shared Pins	
SPI Master (Fast/Slow)	1 bit	000	MCLK	O	SPISD0, CSSPIN, SPISI, DOUT, D0, CONT1N, CONT2N	PROGRAMN, INITN, DONE
		010	MCLK	O	SPISD[0,1], CSSPI[0,1]N, SPISI, D0, CONT1N, CONT2N	
Burst Flash	16 bits	001	MCLK	O	D[0:7], XD[8:15], AVDN, OEN, RDY	PROGRAMN, INITN, DONE
MPCM	8 bits	011	MCLK	O	D[0:7], CSN, CS1N, WRITEN, BUSY	PROGRAMN, INITN, DONE
Slave SPI	1 bit	100	CCLK	I	SO, SN, SI, DOUT,HOLDN	PROGRAMN, INITN, DONE
SCM	1 bit	101	CCLK	I	DI and DOUT	PROGRAMN, INITN, DONE
Parallel	8 bits	111	CCLK	I	D[0:7], CSN, CS1N, WRITEN, BUSY	PROGRAMN, INITN, DONE
JTAG	1 bit	XXX	TCK	I	NA	TCK, TMS, TDI, TDO

## I/O Pin Assignments

The VCCA provides a “quiet” supply for the internal PLLs and critical SERDES blocks. For the best jitter performance, careful pin assignment will keep “noisy” I/O pins away from “sensitive” pins. The leading causes of PCB related SERDES crosstalk is related to FPGA outputs located in close proximity to the sensitive SERDES power supplies. These supplies require cautious board layout to insure noise immunity to the switching noise generated by FPGA outputs. Guidelines are provided to build quiet filtered supplies for the VCCA, however robust PCB layout is required to insure that noise does not infiltrate into these analog supplies.

Although coupling has been reduced in the device packages of the LatticeECP3 devices where little crosstalk is generated, the PCB board can cause significant noise injection from any I/O pin adjacent to SERDES data, reference clock, and power pins as well as other critical I/O pins such as clock signals. TN1114, [Electrical Recommendations for Lattice SERDES](#), provides detailed guidelines for optimizing the hardware to reduce the likelihood of crosstalk to the analog supplies. PCB traces running in parallel for long distances need careful analysis. Simulate any suspicious traces using a PCB crosstalk simulation tool to determine if they will cause problems.

It is common practice for designers to select pinouts for their system very early in the design cycle. For the FPGA designer, this requires a detailed knowledge of the targeted FPGA device. Designers often use a spreadsheet program to initially capture the list of the design I/Os. Lattice provides detailed pinout information that can be downloaded from the Lattice website in .csv format for designers to use as a resource to create pinout information. For example, by navigating to [www.latticesemi.com/documents/ecp3\\_17\\_pinout.csv](http://www.latticesemi.com/documents/ecp3_17_pinout.csv) the user can gather the pinout details for all the different package offerings of the ECP3-17 device family, including I/O banking, differential pairing, and input and output details.

## Dedicated FPGA Inputs (Non-configuration)

Pins annotated E\_A/B/C/D (example PR43E\_A, PR43E\_B, etc.) are dedicated input pins. The primary purpose of these pins is to provide a dedicated input to the FPGA PLLs. They are also available for use as general inputs into the FPGA fabric when not used with a PLL. However, they are available as inputs only. These pins cannot be an output or bidirectional.

## Pinout Considerations

The LatticeECP3 supports many applications with high-speed interfaces. These include various rule-based pinouts that need to be understood prior to implementation of the PCB design. The pinout selection must be completed with an understanding of the interface building blocks of the FPGA fabric. These include IOLOGIC blocks such as DDR, clock resource connectivity, and PLL and DLL usage. Refer to TN1180, [LatticeECP3 High-Speed I/O Interface](#) for rules pertaining to these interface types.

## LVDS Pin Assignments

True-LVDS outputs are available on I/O pins on the left and right sides of the device. LVDS output differential drivers are not supported in banks on the top and bottom. Emulated LVDS outputs are available on any A and B pair around the device, but this will require external termination resistors. This is described in TN1177, [LatticeECP3 sysIO Usage Guide](#).

LVDS inputs are available on any A and B pair of all I/O cells around the entire device. The LatticeECP3 device includes differential input terminations with a common mode connection to the bank VTT pin which must be left floating.

## HSTL and SSTL Pin Assignments

These externally referenced I/O standards require an external reference voltage. The  $V_{REF}$  pin(s) should get high priority when assigning pins on the PCB. Each bank includes a separate VREF voltage. VREF1 sets the threshold for the referenced input buffers. In the LatticeECP3 devices, any I/O pin in a bank can also be configured to be a dedicated reference voltage supply pin. However, the predefined VREF pins provide the best case. Each I/O is individually configurable based on the bank's supply and reference voltages.

In addition, there are dedicated Terminating Supply ( $V_{TT}$ ) pins to be used as terminating voltage for one of the two ways to perform parallel terminations. These  $V_{TT}$  pins are located in banks 2, 3, 7 and 6 and may not be available in some packages. Unused  $V_{TT}$  pins should be left not connected.

A calibration resistor is used to compensate output drivers. A 10Kohm +/-1% resistor connected between the XRES pin and PCB ground is needed.

## XRES Pin

The XRES pin provides a PCB connected resistor (10K-ohm +/-1%) reference to the internal band gap circuit used by PLLs. Careful routing of the XRES pin is required to maintain a stable current source. The PCB should maintain a very short connection to the XRES resistor which must be connected directly to the PCB ground plane.

This XRES pin can also be protected by careful pin selection of adjacent signals in the design. Any "switching or noisy" signals on adjacent pins can increase the PLL output jitter due to cross coupling noise from the aggressor pin to the XRES pin.

It is strongly recommended to tie aggressor pins to PCB ground. If the user has to assign a function due to pin constraints, it is recommended that the pin should be a static or low-frequency control signal as opposed to a high-speed data signal. These aggressor pins are defined in Table 18-5.

**Table 18-5. XRES Aggressor Pin Listing**

Package	XRES Aggressor Pins
1156-ball BGA	AN29, AM31

## SERDES Pin Considerations

High-speed signaling requires careful PCB design. Maintaining good transmission line characteristics is a requirement. A continuous ground reference should be maintained with high-speed routing. This includes tightly matched differential routing with very few discontinuities. Please refer to TN1033, [High-Speed PCB Design Considerations](#), for suggested methods and guidance.

When operating at 2.5 Gbps or above, use of the following FPGA I/O pins can cause increased jitter. Extra care must be given to these pins when used in combination with the high-speed SERDES interface. High-speed switching output assignments should be minimized or avoided on these pins when the SERDES interface is in use. Only static output or input configuration is recommended.

If using the PCSC quad on the 1156 package (for either LatticeECP3-70, LatticeECP3-95 or LatticeECP3-150), the following pins are affected: AE26, AF26, AG26, AH26, AH27, AJ27, AK27, AL27, AM27, AN27, AP27, AH28, AJ28, AK28, AL28, AM28, AN28, AP28, AK29, AL29, AM29, AN29, AP29.

If using the PCSD quad on the 1156 package (for LatticeECP3-150), the following pins are affected: AE9, AF9, AG9, AH9, AJ8, AK8, AL8, AM8, AN8, AP8, AH7, AJ7, AK7, AL7, AM7, AN7, AP7, AJ6, AK6, AL6, AM6, AN6, AP6.

The above mentioned aggressor pin list can only impact SERDES quads PCSC and PCSD. Quads PCSA and PCSB have no suggested aggressor pins due to the physical layout of the device.

There are no known aggressor I/O pins for any other LatticeECP3 device/package combinations other than the beforementioned devices.

**Figure 1.1a. Hardware Checklist**

	Item	OK	NA
<b>1</b>	<b>FPGA Power Supplies</b>		
1.1	VCC core @ 1.2V +/-5%		
1.1.1	Use a PCB plane for VCC core with proper decoupling		
1.1.2	VCC core sized to meet power requirement calculation from software		
1.2	VCCA @ 1.2V +/-5%		
1.2.1	VCCA “quiet” and isolated”		
1.2.2	VCCA pins should be ganged together and a solid PCB plane is recommended. This plane should not have adjacent non-SERDES signals passing above or below. It should also be isolated from the VCC core power plane.		
1.3	All VCCIO[1-8] 1.2V to 3.3V		
1.3.1	VCCIO8 used with configuration interfaces (i.e. memory devices). Need to match specifications.		
1.3.2	VCCIO[1:7] used based on user design		
1.4	VCCAUX @ 3.3V +/-5%		
1.4.1	VCCPLL @ 3.3V +/-5%		
1.4.2	VCCPLL “quiet” and isolated” @ 3.3V +/-5%		
1.5	VCCJ 1.2V to 3.3V		
1.6	Power estimation		
1.7	10K-ohm +/-1% pull-down on XRES pin		
1.7.1	XRES pin uses short connection to resistor. Resistor connected directly to PCB ground plane.		
1.7.2	Follow XRES aggressor pin recommendation.		
<b>2</b>	<b>SERDES Power Supplies</b>		
2.3	VCCIB and VCCOB connected for USED SERDES channels		
2.3.1	VCCIB and VCCOB 1.2V-1.5V nominal +/-5%		
<b>3</b>	<b>Configuration</b>		
3.1	Pull-ups and pull-downs on configuration specific pins		
3.2	VCCIO8 bank voltage matches sysCONFIG peripheral devices such as SPI Flash		
3.3	Pull-up or pull-down on SPIFASTN pin		
<b>4</b>	<b>SERDES</b>		
4.1	Dedicated reference clock input from clock source meets the DC and AC requirements		
4.1.1	External AC coupling caps may be required for compatibility to common-mode levels		
4.1.2	Ref clock termination resistors may be needed for compatible signaling levels		
4.2	Maintain good high-speed transmission line routing		
4.2.1	Continuous ground reference plane to serial channels		

**Figure 1.1a. Hardware Checklist (Continued)**

	Item	OK	NA
4.2.2	Tightly length matched differential traces		
4.2.3	Do not pass other signals on the PCB above or below the high-speed SERDES without isolation.		
4.2.4	Keep non-SERDES signal traces from passing above or below the 1.2V VCCA power plane without isolation.		
4.2.5	Avoid the aggressor pins mentioned previously in this document.		
<b>5</b>	<b>Special Pin Assignments</b>		
5.2	VREF assignments followed for single-ended HSTL or SSTL inputs		
5.2.1	Properly decouple the VREF source		
5.3	VTT pins needed for on-die termination for HSTL or SSTL terminated I/O		
5.3.1	All VTT need to be connected to termination power supply if used for VTT. VTT pins do not need to be connected if ODT (on-die termination) is not used in the design. VTT pins can be left floating when not used for ODT.		
5.3.2	VTT power connections (for SSTL or HSTL terminations) needs to be a low-impedance PCB plane and properly decoupled.		
5.3.3	The bank VTT pin must float when using differential input terminations.		
<b>6</b>	<b>Critical Pinout Selection</b>		
6.1	Pinout has been chosen to address FPGA resource connections to I/O logic and clock resources per TN1180, <a href="#">LatticeECP3 High-Speed I/O Interface</a> .		
6.2	Dedicated FPGA inputs are used as inputs only to the FPGA PLL or fabric. Not output or bi-directional.		
<b>7</b>	<b>JTAG</b>		
7.1	Pulldown on TCK. See Table 18-2.		
7.2	Pullups on TDI, TMS, TDO. See Table 18-2.		
<b>8</b>	<b>DDR3 Interface Requirements</b>		
8.1	DQ, DM, and DQS signals should be routed in a data group and should have similar routing and matched via counts. Using more than three vias is not recommended in the route between the FPGA controller and memory device.		
8.2	Maintain a maximum of $\pm 50$ mil between any DQ/DM and its associated DQS strobe within a DQ group. Use careful serpentine routing to meet this requirement.		
8.3	All data groups must reference a ground plane within the stack-up.		
8.4	DDR trace reference must be solid without slots or breaks. It should be continuous between the FPGA and the memory.		
8.5	Provide a separation of 3W spacing between a data group and any other unrelated signals to avoid crosstalk issues. Use a minimum of 2W spacing between all DDR traces excluding differential CK and DQS signals.		
8.6	Assigned FPGA I/O within a data group can be swapped to allow clean layout. Do not swap DQS assignments.		
8.7	Differential pair of DQS to DQS_N trace lengths should be matched at $\pm 10$ mil.		
8.8	Resistor terminations (DQ) placed in a fly-by fashion at the FPGA is highly recommended. Stub fashion terminations, if used, should not include a stub longer than 600 mil.		
8.9	LDQS/LDQS_N and UDQS/UDQS_N trace lengths should be matched within $\pm 100$ mil.		
8.10	Address/control signals and the associated CK and CK_N differential FPGA clock should be routed with a control trace matching $\pm 100$ mil.		
8.11	CK to CK_N trace lengths must be matched to within $\pm 10$ mil.		
8.12	Address and control signals can be referenced to a power plane if a ground plane is not available. Ground reference is preferred.		
8.13	Address and control signals should be kept on a different routing layer from DQ, DQS, and DM to isolate crosstalk between the signals.		

**Figure 1.1a. Hardware Checklist (Continued)**

	Item	OK	NA
8.14	Differential terminations used by the CLK/CLKN pair must be located as close as possible to the memory.		
8.15	Address and control terminations placed after the memory component using a fly-by technique are highly recommended. Stub fashion terminations, if used, should not include a stub longer than 600 mils.		

## Technical Support Assistance

 e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

 Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2009	01.0	Initial release.
March 2009	01.1	Updated Hardware Checklist table.
July 2009	01.2	Added Dedicated FPGA Inputs (Non-configuration) and Pinout Considerations text sections. Added Critical Pinout Selection section to Hardware Checklist table.
July 2009	01.3	LatticeECP3 FPGA Power Supplies table - Updated voltage values for VCCIB and VCCOB supplies.
August 2009	01.4	Updated SERDES Pin Considerations text section with information regarding pins that can cause increased jitter when operating at 2.5Gpbs and above.
November 2009	01.5	Updated FPGA LUT densities in the Introduction section. Updated SERDES Pin Considerations text section.
March 2010	01.6	Updated reference documents list.
December 2010	01.7	Updated Hardware Checklist.
June 2011	01.8	Added Pull-up or pull-down on SPIFASTN pin to Hardware Checklist.
July 2011	01.9	Added DDR3 Interface Requirements section to Hardware Checklist.
July 2011	02.0	Added XRES pin information.
February 2012	02.1	Updated document with new corporate logo.
July 2013	02.2	Updated Technical Support Assistance information.



**Section III. LatticeECP3 Family Handbook Revision History**

---

## Revision History

Date	Handbook Revision Number	Change Summary
February 2009	01.0	Initial release.
April 2009	01.1	Technical note TN1176 updated to version 01.1.
		Technical note TN1189 updated to version 01.1.
June 2009	01.2	LatticeECP3 Family Data Sheet updated to version 01.1.
		Technical note TN1169 updated to version 01.1.
		Technical note TN1176 updated to version 01.2.
		Technical note TN1177 updated to version 01.1.
		Technical note TN1179 updated to version 01.2.
		Technical note TN1182 updated to version 01.1.
		Technical note TN1189 updated to version 01.1.
July 2009	01.3	Technical note TN1178 updated to version 01.1.
		Technical note TN1179 updated to version 01.3.
		Technical note TN1180 updated to version 01.1.
July 2009	01.4	LatticeECP3 Family Data Sheet updated to version 01.2.
August 2009	01.5	LatticeECP3 Family Data Sheet updated to version 01.3.
		Technical note TN1177 updated to version 01.2.
		Technical note TN1189 updated to version 01.4.
September 2009	01.6	LatticeECP3 Family Data Sheet updated to version 01.4.
		Technical note TN1178 updated to version 01.2.
November 2009	01.7	LatticeECP3 Family Data Sheet updated to version 01.5.
		Technical note TN1176 updated to version 01.3.
		Technical note TN1179 updated to version 01.4.
		Technical note TN1180 updated to version 01.3.
		Technical note TN1169 updated to version 01.3.
March 2010	01.8	Technical note TN1189 updated to version 01.5.
		Technical note TN1176 updated to version 01.4.
		Technical note TN1178 updated to version 01.4.
		Technical note TN1179 updated to version 01.5.
		Technical note TN1189 updated to version 01.6.
March 2010	01.9	LatticeECP3 Family Data Sheet updated to version 01.6.
March 2010	02.0	Technical note TN1180 updated to version 01.4.
April 2010	02.1	Technical note TN1176 updated to version 01.5.
April 2010	02.2	Technical note TN1177 updated to version 01.3.
		Technical note TN1178 updated to version 01.5.
		Technical note TN1180 updated to version 01.5.
May 2010	02.3	Technical note TN1149 added to document.

<b>Date</b>	<b>Handbook Revision Number</b>	<b>Change Summary</b>
June 2010	02.4	Technical note TN1169 updated to version 01.6.
		Technical note TN1176 updated to version 01.6.
		Technical note TN1177 updated to version 01.4.
		Technical note TN1178 updated to version 01.6.
		Technical note TN1180 updated to version 01.6.
		Technical note TN1182 updated to version 01.2.
September 2010	02.5	Technical note TN1189 updated to version 01.7.
January 2011	02.6	LatticeECP3 Family Data Sheet updated to version 01.7EA.
		Technical note TN1169 updated to version 01.7.
		Technical note TN1176 updated to version 01.7.
		Technical note TN1177 updated to version 01.5.
		Technical note TN1184 updated to version 01.2.
February 2011	02.7	Technical note TN1189 updated to version 01.7.
February 2011	02.7	Technical note TN1178 updated to version 01.8.
March 2011	02.8	Technical note TN1177 updated to version 01.6.
March 2011	02.9	Technical note TN1169 updated to version 01.8.
April 2011	03.0	LatticeECP3 Family Data Sheet updated to version 01.8EA.
April 2011	03.1	Technical note TN1177 updated to version 01.7.
		Technical note TN1180 updated to version 01.7.
June 2011	03.2	Technical note TN1178 updated to version 01.9.
July 2011	03.3	LatticeECP3 Family Data Sheet updated to version 01.9EA.
		Technical note TN1169 updated to version 01.8.
		Technical note TN1189 updated to version 01.9.
July 2011	03.4	Technical note TN1176 updated to version 01.8.
		Technical note TN1179 updated to version 01.6.
		Technical note TN1189 updated to version 02.0.
September 2011	03.5	Technical note TN1180 updated to version 01.8.
September 2011	03.6	Technical note TN1180 updated to version 02.0.
September 2011	03.7	Technical note TN1176 updated to version 01.9.
November 2011	03.8	Technical note TN1176 updated to version 02.0.
November 2011	03.9	LatticeECP3 Family Data Sheet updated to version 02.0EA.
December 2011	04.0	Technical note TN1180 updated to version 01.9.
January 2012	04.1	Technical note TN1178 updated to version 02.0.
February 2012	04.2	LatticeECP3 Family Data Sheet updated to version 02.1EA.
		Technical note TN1176 updated to version 02.1.
		Technical note TN1177 updated to version 01.8.
		Technical note TN1178 updated to version 02.1.
		Technical note TN1179 updated to version 01.7.
		Technical note TN1180 updated to version 02.0.
		Technical note TN1181 updated to version 01.1.
		Technical note TN1182 updated to version 01.3.
		Technical note TN1169 updated to version 02.1.
		Technical note TN1184 updated to version 01.3.
Technical note TN1149 updated to version 01.4.		



Date	Handbook Revision Number	Change Summary
February 2012 (cont.)	04.2 (cont.)	Technical note TN1189 updated to version 02.1.
March 2012	04.3	Technical note TN1177 updated to version 01.9.
April 2012	04.4	LatticeECP3 Family Data Sheet updated to version 02.2EA.
		Technical note TN1176 updated to version 02.2.
		Technical note TN1178 updated to version 02.2.
May 2012	04.5	Technical note TN1176 updated to version 02.3.
May 2012	04.6	Technical note TN1178 updated to version 02.3.
June 2012	04.7	Technical note TN1177 updated to version 02.0.
August 2012	04.8	Technical note TN1180 updated to version 02.1.
August 2012	04.9	Technical note TN1169 updated to version 02.2.
		Technical note TN1176 updated to version 02.4.
September 2012	05.0	Technical note TN1178 updated to version 02.4.
November 2012	05.1	Technical note TN1180 updated to version 02.2.
July 2013	05.2	Technical note TN1149 updated to version 01.5.
		Technical note TN1169 updated to version 02.8.
		Technical note TN1176 updated to version 02.5.
		Technical note TN1177 updated to version 02.2.
		Technical note TN1178 updated to version 02.5.
		Technical note TN1179 updated to version 01.8.
		Technical note TN1180 updated to version 02.4.
		Technical note TN1181 updated to version 01.2
		Technical note TN1182 updated to version 01.4
		Technical note TN1184 updated to version 01.4
Technical note TN1189 updated to version 02.2		

Note: For detailed revision changes, please refer to the revision history for each document.