



## **LatticeECP2/M Family Handbook**

---

HB1003 Version 02.0, September 2006



# LatticeECP2/M Family Handbook

## Table of Contents

---

September 2006

### Section I. LatticeECP2/M Family Data Sheet

#### Introduction

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

#### Architecture

Architecture Overview .....	2-1
PFU Blocks .....	2-3
Slice .....	2-3
Modes of Operation.....	2-5
Routing.....	2-6
sysCLOCK Phase Locked Loops (GPLL/SPLL) .....	2-6
General Purpose PLL (GPLL) .....	2-6
Standard PLL (SPLL) .....	2-7
Delay Locked Loops (DLL).....	2-8
DLL_DEL Delay Block.....	2-9
PLL/DLL Cascading .....	2-9
GPLL/SPLL/GDLL PIO Input Pin Connections (LatticeECP2M Family Only) .....	2-10
Clock Dividers .....	2-10
Clock Distribution Network .....	2-11
Primary Clock Sources.....	2-11
Secondary Clock/Control Sources .....	2-13
Edge Clock Sources.....	2-14
Primary Clock Routing .....	2-15
Dynamic Clock Select (DCS) .....	2-15
Secondary Clock/Control Routing.....	2-15
Slice Clock Selection.....	2-17
Edge Clock Routing .....	2-17
sysMEM Memory .....	2-18
sysMEM Memory Block.....	2-18
Bus Size Matching .....	2-19
RAM Initialization and ROM Operation .....	2-19
Memory Cascading .....	2-19
Single, Dual and Pseudo-Dual Port Modes.....	2-19
Memory Core Reset.....	2-19
sysDSP™ Block .....	2-20
sysDSP Block Approach Compare to General DSP .....	2-20
sysDSP Block Capabilities .....	2-21
MULT sysDSP Element .....	2-22
MAC sysDSP Element .....	2-23
MULTADDSSUB sysDSP Element .....	2-24
MULTADDSSUM sysDSP Element .....	2-25
Clock, Clock Enable and Reset Resources .....	2-25
Signed and Unsigned with Different Widths.....	2-26
OVERFLOW Flag from MAC .....	2-26
IPexpress™ .....	2-27
Optimized DSP Functions .....	2-27
Resources Available in the LatticeECP2/M Family .....	2-27
LatticeECP2/M DSP Performance .....	2-28

Programmable I/O Cells (PIC) .....	2-28
PIO .....	2-30
Input Register Block .....	2-30
Output Register Block .....	2-32
Tristate Register Block .....	2-34
Control Logic Block .....	2-34
DDR Memory Support.....	2-34
Left and Right Edges.....	2-34
Bottom Edge .....	2-34
Top Edge.....	2-35
DLL Calibrated DQS Delay Block .....	2-36
Polarity Control Logic .....	2-38
DQSFER.....	2-39
sysIO Buffer .....	2-39
sysIO Buffer Banks .....	2-39
Typical sysIO I/O Behavior During Power-up.....	2-42
Supported sysIO Standards .....	2-42
Hot Socketing.....	2-44
Power-up During Configuration.....	2-44
SERDES and PCS (Physical Coding Sublayer).....	2-45
SERDES Block.....	2-45
PCS.....	2-46
SCI (SERDES Client Interface) Bus.....	2-46
IEEE 1149.1-Compliant Boundary Scan Testability.....	2-47
Device Configuration.....	2-47
Software Error Detect (SED) Support .....	2-48
External Resistor.....	2-48
On-Chip Oscillator.....	2-48
Density Shifting .....	2-49
<b>DC and Switching Characteristics</b>	
Absolute Maximum Ratings' .....	3-1
Recommended Operating Conditions .....	3-1
Hot Socketing Specifications.....	3-2
DC Electrical Characteristics.....	3-3
LatticeECP2 Supply Current (Standby).....	3-4
LatticeECP2M Supply Current (Standby).....	3-4
LatticeECP2 Initialization Supply Current .....	3-5
LatticeECP2M Initialization Supply Current .....	3-6
SERDES Power Supply Requirements (LatticeECP2M Family Only) .....	3-7
SERDES Power (LatticeECP2M Family Only) .....	3-7
sysIO Recommended Operating Conditions .....	3-8
sysIO Single-Ended DC Electrical Characteristics .....	3-9
sysIO Differential Electrical Characteristics .....	3-10
LVDS.....	3-10
Differential HSTL and SSTL.....	3-10
LVDS25E .....	3-11
BLVDS .....	3-12
LVPECL .....	3-13
RSDS .....	3-14
MLVDS.....	3-15
Typical Building Block Function Performance.....	3-16
Pin-to-Pin Performance (LVCMS25 12mA Drive) .....	3-16
Register-to-Register Performance .....	3-16
Derating Timing Tables .....	3-17

---

LatticeECP2/M External Switching Characteristics.....	3-18
LatticeECP2/M Internal Switching Characteristics.....	3-22
Timing Diagrams .....	3-24
LatticeECP2/M Family Timing Adders .....	3-26
sysCLOCK GPLL Timing .....	3-29
sysCLOCK SPPLL Timing.....	3-30
DLL Timing.....	3-31
SERDES High Speed Data Transmitter (LatticeECP2M Family Only) .....	3-32
SERDES High Speed Data Receiver (LatticeECP2M Family Only) .....	3-33
Input Data Jitter Tolerance.....	3-33
Input Eye Mask Characterization .....	3-33
SERDES External Reference Clock (LatticeECP2M Family Only) .....	3-35
RESET Pulse Specification (LatticeECP2M Family Only) .....	3-36
Power-Down/Power-Up Specification .....	3-36
LatticeECP2/M sysCONFIG Port Timing Specifications .....	3-37
JTAG Port Timing Specifications .....	3-42
Switching Test Conditions.....	3-43
<b>Pinout Information</b>	
Signal Descriptions .....	4-1
PICs and DDR Data (DQ) Pins Associated with the DDR Strobe (DQS) Pin .....	4-4
LatticeECP2 Pin Information Summary.....	4-5
LatticeECP2M Pin Information Summary.....	4-7
Available Device Resources by Package.....	4-8
LatticeECP2 Power Supply and NC .....	4-9
ECP2-12E Logic Signal Connections: 144 TQFP .....	4-11
ECP2-12E Logic Signal Connections: 208 PQFP .....	4-15
ECP2-12E Logic Signal Connections: 256 fpBGA .....	4-20
ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA.....	4-27
ECP2M-35 Logic Signal Connections: 484 fpBGA .....	4-38
ECP2-50 Logic Signal Connections: 672 fpBGA .....	4-53
ECP2M35 Logic Signal Connections: 672 fpBGA.....	4-69
<b>Ordering Information</b>	
LatticeECP2 Part Number Description.....	5-1
LatticeECP2M Part Number Description.....	5-1
Ordering Information .....	5-2
<b>Supplemental Information</b>	
For Further Information .....	6-1
<b>LatticeECP2/M Family Data Sheet Revision History</b>	
<b>Section II. LatticeECP2/M Family Technical Notes</b>	
<b>LatticeECP2M SERDES/PCS Usage Guide</b>	
Features .....	8-1
Introduction to PCS .....	8-1
Architecture Overview .....	8-2
PCS Quad .....	8-2
PCS Channel .....	8-3
Per Channel PCS/FPGA Interface Ports.....	8-4
SCI (SERDES Client Interface) Bus.....	8-6
Using This Technical Note .....	8-6
SERDES .....	8-7
I/O Definitions.....	8-8
Reference Clock Usage .....	8-10
Configuration GUIs.....	8-11
Configuration File Description .....	8-21

---

---

PCS.....	8-23
LatticeECP2M PCS Quad Module .....	8-23
PCS Functional Setup.....	8-24
Auto-Configuration File .....	8-24
LatticeECP2M PCS 8-bit Modes .....	8-24
8-bit Mode Pin Description .....	8-26
Optional Pin Description.....	8-30
8-bit Mode Detailed Description .....	8-31
Clocks & Resets.....	8-31
Transmit Data.....	8-31
Receive Data.....	8-33
LatticeECP2M PCS in PCI Express Mode .....	8-47
PCI Express Mode Pin Description .....	8-48
PCI Express Mode Detailed Description .....	8-50
LatticeECP2M PCS in PIPE Mode.....	8-52
PIPE Mode Pin Description .....	8-54
LatticeECP2M PCS 10-bit Modes .....	8-57
10-bit Mode Pin Description .....	8-58
PCS Loopback Testing .....	8-58
Serial Far-end Loopback Mode.....	8-58
PCS Parallel Far-end Loopback Mode.....	8-59
PCS Parallel Far-end Loopback Mode .....	8-60
SERDES Client Interface (SCI).....	8-61
Interrupts and Status.....	8-63
Dynamic Configuration of SERDES/PCS Quad.....	8-64
SERDES Debug Capabilities .....	8-64
Memory Map .....	8-65
Configuration Register Definition .....	8-65
Per Quad Register Overview .....	8-66
Per Quad PCS Control Register Details .....	8-67
Per Quad SERDES Control Register Details .....	8-70
Per Quad Reset and Clock Control Register Details .....	8-72
Per Quad PCS Status Register Details.....	8-73
Per Quad SERDES Status Register Details .....	8-75
Per Channel Register Overview.....	8-76
Per Channel SERDES Control Register Details .....	8-79
Per Channel PCS Status Register Details .....	8-81
Per Channel SERDES Status Register Details .....	8-82
Technical Support Assistance.....	8-84
Revision History .....	8-84
Appendix A. 8b/10b Symbol Codes .....	8-85
Appendix B. Attribute Cross-Reference Table .....	8-86
<b>LatticeECP2/M sysIO Usage Guide</b>	
Introduction .....	9-1
sysIO Buffer Overview .....	9-1
Supported sysIO Standards .....	9-1
sysIO Banking Scheme.....	9-3
$V_{CCIO}$ (1.2V/1.5V/1.8V/2.5V/3.3V) .....	9-3
$V_{CCAUX}$ (3.3V) .....	9-4
$V_{CCJ}$ (1.2V/1.5V/1.8V/2.5V/3.3V).....	9-4
Input Reference Voltage ( $V_{REF1}$ , $V_{REF2}$ ).....	9-4
$V_{REF1}$ for DDR Memory Interface .....	9-4
Mixed Voltage Support in a Bank.....	9-4
sysIO Standards Supported by Bank .....	9-5

---

LVCMOS Buffer Configurations .....	9-6
Bus Maintenance Circuit .....	9-6
Programmable Drive .....	9-6
Programmable Slew Rate .....	9-6
Open-Drain Control .....	9-6
Differential SSTL and HSTL support .....	9-6
PCI Support with Programmable PCICLAMP .....	9-7
Programmable Input Delay .....	9-7
Software sysIO Attributes.....	9-7
IO_TYPE .....	9-7
OPENDRAIN .....	9-8
DRIVE .....	9-8
PULLMODE .....	9-9
PCICLAMP .....	9-9
SLEWRATE .....	9-9
FIXEDDELAY .....	9-10
INBUF .....	9-10
DIN/DOUT .....	9-10
LOC .....	9-10
Design Considerations and Usage.....	9-10
Banking Rules .....	9-10
Differential I/O Rules .....	9-10
Assigning V <sub>REF1</sub> /V <sub>REF2</sub> Groups for Referenced Inputs.....	9-11
Differential I/O Implementation.....	9-11
LVDS .....	9-11
BLVDS .....	9-11
RSDS .....	9-11
LVPECL .....	9-12
Differential SSTL and HSTL .....	9-12
Technical Support Assistance .....	9-12
Revision History .....	9-12
Appendix A. HDL Attributes for Synplicity® and Precision® RTL Synthesis.....	9-13
VHDL Synplicity/Precision RTL Synthesis .....	9-13
Verilog Synplicity .....	9-15
Verilog Precision .....	9-16
Appendix B. sysIO Attributes Using the Preference Editor User Interface.....	9-17
Appendix C. sysIO Attributes Using Preference File (ASCII File) .....	9-18
IOBUF .....	9-18
LOCATE .....	9-18
USE DIN CELL .....	9-19
USE DOUT CELL .....	9-19
PGROUP VREF .....	9-19
<b>LatticeECP2/M sysCLOCK PLL/DLL Design and Usage Guide</b>	
Introduction .....	10-1
Clock/Control Distribution Network .....	10-1
LatticeECP2/M Top Level View.....	10-2
Primary Clocks .....	10-3
Secondary Clocks .....	10-3
Edge Clocks .....	10-3
Note on Primary Clocks .....	10-4
Specifying Clocks in the Design Tools .....	10-5
Primary-Pure and Primary-DCS.....	10-5
Global Primary Clock and Quadrant Primary Clock .....	10-5
Note on Edge Clocks .....	10-5

---

sysCLOCK PLL .....	10-6
Functional Description.....	10-6
PLL Divider and Delay Blocks.....	10-6
PLL Inputs and Outputs .....	10-7
PLL Attributes.....	10-8
LatticeECP2/M PLL Primitive Definitions .....	10-9
Dynamic Delay Adjustment (EHXPLL Only).....	10-10
Dynamic Phase/Duty Mode .....	10-11
Dynamic Phase Adjustment/Duty Cycle Select.....	10-11
Optional External Capacitor .....	10-12
PLL Usage in IPexpress.....	10-13
Configuration Tab .....	10-14
Modes .....	10-14
Frequency Calculation .....	10-15
DLL Primitive I/Os .....	10-16
PLL Modes of Operation .....	10-16
PLL Clock Injection Removal .....	10-16
PLL Clock Phase Adjustment.....	10-16
sysCLOCK DLL.....	10-17
DLL Overview.....	10-17
DLL Inputs and Outputs .....	10-18
DLL Attributes .....	10-18
DLL Primitives Definitions .....	10-19
DLL Primitive I/Os .....	10-20
DLL Modes of Operation.....	10-20
DLL Usage in IPexpress .....	10-22
Clock Dividers (CLKDIV).....	10-23
CLKDIV Primitive Definition .....	10-23
CLKDIV Declaration in VHDL Source Code.....	10-24
CLKDIV Usage with Verilog - Example .....	10-25
CLKDIV Example Circuits .....	10-25
Release Behavior.....	10-26
DLLDEL (Slave Delay Line) .....	10-26
DQS DLL and DQS DEL.....	10-28
DCS (Dynamic Clock Select) .....	10-29
DCS Primitive Definition.....	10-29
DCS Timing Diagrams .....	10-29
DCS Usage with VHDL - Example .....	10-32
DCS Usage with Verilog - Example .....	10-32
Oscillator (OSCD) .....	10-32
OSC Primitive Symbol (OSCD).....	10-33
OSC Usage with VHDL - Example .....	10-33
OSC Usage with Verilog - Example .....	10-33
Input Clock Sharing.....	10-34
Setting Clock Preferences.....	10-35
Power Supplies .....	10-35
Technical Support Assistance.....	10-36
Revision History .....	10-36
Appendix A. Primary Clock Sources and Distribution .....	10-37
Appendix B. PLL, DLL, CLKIDV and ECLK Locations and Connectivity .....	10-39
Appendix C. Clock Preferences .....	10-40
<b>LatticeECP2/M Memory Usage Guide</b>	
Introduction .....	11-1
Memories in LatticeECP2/M Devices.....	11-1

---

Utilizing IPExpress.....	11-2
IPExpress Flow.....	11-2
Memory Modules.....	11-6
Single Port RAM (RAM_DQ) – EBR Based .....	11-6
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, FIFO_DC) – EBR Based.....	11-24
Distributed Single Port RAM (Distributed_SPRAM) – PFU Based.....	11-28
Distributed Dual Port RAM (Distributed_DPRAM) – PFU Based.....	11-30
Distributed ROM (Distributed_ROM) – PFU Based .....	11-32
Initializing Memory .....	11-34
Initialization File Format .....	11-34
Binary File .....	11-34
Hex File .....	11-35
Addressed Hex.....	11-35
Technical Support Assistance.....	11-35
Revision History .....	11-35
Appendix A: Attribute Definitions.....	11-36
DATA_WIDTH.....	11-36
REGMODE.....	11-36
RESETMODE .....	11-36
CSDECODE.....	11-36
WRITEMODE.....	11-36
GSR .....	11-36
<b>LatticeECP2/M High-Speed I/O Interface</b>	
Introduction .....	12-1
DDR and DDR2 SDRAM Interfaces Overview.....	12-1
Implementing DDR Memory Interfaces with LatticeECP2/M Devices.....	12-3
DQS Grouping.....	12-3
DDR Software Primitives.....	12-5
Memory Read Implementation .....	12-12
DLL Compensated DQS Delay Elements .....	12-12
DQS Transition Detect or Automatic Clock Polarity Select.....	12-12
Data Valid Module.....	12-12
DDR I/O Register Implementation.....	12-12
Memory Read Implementation in Software .....	12-13
Read Timing Waveforms.....	12-14
Memory Write Implementation .....	12-17
Generic High Speed DDR Implementation .....	12-20
Generic DDR Software Primitives .....	12-21
DDR Generic Usage In IPExpress .....	12-26
Configuration Tab.....	12-27
Design Rules/Guidelines.....	12-28
FCRAM (“Fast Cycle Random Access Memory”) Interface .....	12-29
Board Design Guidelines .....	12-29
References.....	12-29
Technical Support Assistance .....	12-29
Revision History .....	12-30
<b>Power Estimation and Management for LatticeECP2/M Devices</b>	
Introduction .....	13-1
Power Supply Sequencing .....	13-1
Power-Up Sequencing .....	13-1
Power-Down Sequencing.....	13-1

Power Sequencing Recommendations .....	13-1
Power Calculator Hardware Assumptions .....	13-2
Static Power or DC Power .....	13-2
Power Calculator .....	13-3
Power Calculation Equations .....	13-3
Activity Factor Calculation .....	13-4
Ambient and Junction Temperatures and Airflow .....	13-5
Managing Power Consumption .....	13-5
Power Calculator Assumptions .....	13-6
Technical Support Assistance .....	13-6
Revision History .....	13-6
<b>LatticeECP2/M sysDSP Usage Guide</b>	
Introduction .....	14-1
sysDSP Block Hardware .....	14-1
sysDSP Block Software .....	14-2
Overview .....	14-2
Targeting sysDSP Block Using IPExpress .....	14-2
Targeting the sysDSP Block by Inference .....	14-9
sysDSP Blocks in the Report File .....	14-11
MAP Report File .....	14-11
Post PAR Report File .....	14-12
Targeting the sysDSP Block Using Simulink .....	14-13
Simulink Overview .....	14-13
Targeting the sysDSP Block by Instantiating Primitives .....	14-14
sysDSP Block Control Signal and Data Signal Descriptions .....	14-14
Technical Support Assistance .....	14-14
Revision History .....	14-15
Appendix A. DSP Block Primitives .....	14-16
MULT18X18B .....	14-16
MULT18X18ADDSUBB .....	14-16
MULT18X18ADDSUBSUMB .....	14-17
MULT18X18MACB .....	14-19
MULT36X36B .....	14-20
MULT9X9B .....	14-21
MULT9X9ADDSUBB .....	14-21
MULT9X9ADDSUBSUMB .....	14-22
<b>LatticeECP2/M sysCONFIG Usage Guide</b>	
Introduction .....	15-1
General Configuration Flow .....	15-1
Configuration Pins .....	15-1
Dedicated Control Pins .....	15-2
Dual-Purpose sysCONFIG Pins .....	15-4
ispJTAG Pins .....	15-6
Configuration Modes .....	15-7
SPI Mode .....	15-7
SPlm Mode .....	15-9
Programming SPI Serial Flash .....	15-12
Slave Serial Mode .....	15-12
Slave Parallel Mode .....	15-13
ispJTAG Mode .....	15-15
Configuration Options .....	15-16
Device Wake-Up .....	15-18
Synchronizing Wake-Up .....	15-20
Configuration FAQs .....	15-20

---

General .....	15-20
Mode Specific.....	15-21
Technical Support Assistance.....	15-22
Revision History .....	15-22
<b>LatticeECP2/M Configuration Encryption Usage Guide</b>	
Introduction .....	16-1
General Configuration Process.....	16-1
Encryption/Decryption Flow .....	16-2
Encrypting the Bitstream .....	16-2
Programming the 128-bit Key .....	16-4
Verifying a Configuration.....	16-6
File Formats .....	16-6
Decryption Flow .....	16-10
Reference Documents .....	16-11
Technical Support Assistance.....	16-11
Revision History .....	16-11
<b>LatticeECP2/M Soft Error Detection (SED) Usage Guide</b>	
Introduction .....	17-1
SED Overview.....	17-1
Hardware Description.....	17-2
Signal Description .....	17-2
SEDCLKIN .....	17-2
SEDENABLE.....	17-3
SEDCLKOUT .....	17-3
SEDSTART .....	17-3
SEDFRCERR.....	17-3
SEDINPROG.....	17-3
SEDDONE .....	17-4
SEDERR.....	17-4
SED Flow .....	17-4
SED Run Time .....	17-5
Sample Code .....	17-5
VHDL Example.....	17-6
Verilog Example .....	17-7
Technical Support Assistance.....	17-7
Revision History .....	17-7
<b>LatticeECP2/M Family Handbook Revision History .....</b>	<b>18-1</b>



## **Section I. LatticeECP2/M Family Data Sheet**

---

Version 02.1, September 2006

September 2006

Advance Data Sheet DS1007

### Features

- **High Logic Density for System Integration**
  - 6K to 95K LUTs
  - 90 to 601 I/Os
- **Embedded SERDES (LatticeECP2M Only)**
  - Data Rates 540 Mbps to 3.125 Gbps
  - 270 Mbps with Half Rate mode
  - Up to 16 channels per device
  - PCI Express, Ethernet (1GbE, SGMII), OBSAI, CPRI and Serial RapidIO.
- **sysDSP™ Block**
  - 3 to 42 blocks for high performance multiply and accumulate
  - Each block supports
    - One 36x36, four 18X18 or eight 9X9 multipliers
- **Flexible Memory Resources**
  - 55Kbits to 5308Kbits sysMEM™ Embedded Block RAM (EBR)
    - 18-Kbit block
    - Single, pseudo dual and true dual port
    - Byte Enable Mode support
  - 12K to 202Kbits distributed RAM
    - Single port and pseudo dual port
- **sysCLOCK Analog PLLs And DLLs**
  - Two GPLPs and up to six SPLLs per device
    - Clock multiply, divide, phase & delay adjust
    - Dynamic PLL adjustment
  - Two general purpose DLLs per device

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

- DDR registers in I/O cells
- Dedicated gearing logic
- Source synchronous standards support
  - SPI4.2, SFI4, XGMII
  - High Speed ADC/DAC devices
- Dedicated DDR and DDR2 memory support
  - DDR1/DDR2 400 (200MHz)
- Dedicated DQS support

### ■ Programmable sysIO™ Buffer Supports Wide Range Of Interfaces

- LVTTL and LVCMS 33/25/18/15/12
- SSTL 3/2/18 I, II
- HSTL15 I and HSTL18 I, II
- PCI and Differential HSTL, SSTL
- LVDS, RSDS, Bus-LVDS, MLVDS, LVPECL

### ■ Flexible Device Configuration

- 1149.1 Boundary Scan compliant
- Dedicated bank for configuration I/Os
- SPI boot flash interface
- Dual boot images supported
- TransFR™ I/O for simple field updates
- Soft Error Detect macro embedded

### ■ Optional Bitstream Encryption

### ■ System Level Support

- ispTRACY™ internal logic analyzer capability
- Onboard oscillator for initialization & general use
- 1.2V power supply

**Table 1-1. LatticeECP2 Family Selection Guide**

Device	ECP2-6	ECP2-12	ECP2-20	ECP2-35	ECP2-50	ECP2-70
LUTs (K)	6	12	21	32	48	68
Distributed RAM (Kbits)	12	24	42	65	96	136
EBR SRAM (Kbits)	55	221	276	332	387	1106
EBR SRAM Blocks	3	12	15	18	21	60
sysDSP Blocks	3	6	7	8	18	22
18x18 Multipliers	12	24	28	32	72	88
GPLL + SPLL + GDLL	2+0+2	2+0+2	2+0+2	2+0+2	2+2+2	2+4+2
<b>Packages and I/O Combinations</b>						
144-pin TQFP (20 x 20 mm)	90	93				
208-pin PQFP (28 x 28 mm)		131	131			
256-ball fpBGA (17 x 17 mm)	190	193	193			
484-ball fpBGA (23 x 23 mm)		297	331	331	339	
672-ball fpBGA (27 x 27 mm)			402	450	500	500
900-ball fpBGA (31 x 31 mm)						583

© 2006 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.

**Table 1-2. LatticeECP2M Family Selection Guide**

Device	ECP2M20	ECP2M35	ECP2M50	ECP2M70	ECP2M100
LUTs (K)	19	34	48	67	95
sysMEM Blocks (18kb)	66	114	225	246	288
Embedded Memory (Kbits)	1217	2101	4147	4534	5308
Distributed Memory (KBits)	41	71	101	145	202
sysDSP Blocks	6	8	22	24	42
18X18 Multipliers	24	32	88	96	168
SPLL+GPLL+DLL	2+6+2	2+6+2	2+6+2	2+6+2	2+6+2
<b>Packages and SERDES / I/O Combinations</b>					
256-ball fpBGA (17 x 17 mm)	4 / 144				
484-ball fpBGA (23 x 23 mm)	4 / 301	4 / 301	4 / 287		
672-ball fpBGA (27 x 27 mm)		4 / 411	8 / 387		
900-ball fpBGA (31 x 31 mm)			8 / 457	16 / 449	16 / 457
1156-ball fpBGA (35 x 35 mm)					16 / 601

## Introduction

The LatticeECP2/M family of FPGA devices has been optimized to deliver high performance features such as advanced DSP blocks, high speed SERDES (LatticeECP2M family only) and high speed source synchronous interfaces in an economical FPGA fabric. This combination was achieved through advances in device architecture and the use of 90nm technology.

The LatticeECP2/M FPGA fabric was optimized for the new technology from the outset with high performance low cost in mind. The LatticeECP2/M devices include LUT-based logic, distributed and embedded memory, Phase Locked Loops (PLLs), Delay Locked Loops (DLLs), pre-engineered source synchronous I/O support, enhanced sysDSP blocks and advanced configuration support, including encryption and dual-boot capabilities.

The LatticeECP2M family of devices features high speed SERDES with PCS. These high jitter tolerance and low transmission jitter SERDES with PCS blocks can be configured to support an array of popular data protocols including PCI Express, Ethernet (1GbE and SGMII), OBSAI and CPRI. Transmit Pre-emphasis and Receive Equalization settings make SERDES suitable for chip to chip and small form factor backplane applications.

The ispLEVER® design tool from Lattice allows large complex designs to be efficiently implemented using the LatticeECP2/M family of FPGA devices. Synthesis library support for LatticeECP2/M is available for popular logic synthesis tools. The ispLEVER tool uses the synthesis tool output along with the constraints from its floor planning tools to place and route the design in the LatticeECP2/M device. The ispLEVER tool extracts the timing from the routing and back-annotates it into the design for timing verification.

Lattice provides many pre-designed IP (Intellectual Property) ispLeverCORE™ modules for the LatticeECP2/M family. By using these IPs as standardized blocks, designers are free to concentrate on the unique aspects of their design, increasing their productivity.

### Architecture Overview

Each LatticeECP2/M 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 blocks as shown in the ECP2-6 in Figure 2-1. In addition, the LatticeECP2M family contain SERDES Quads in one or more of the corners. Figure 2-2 shows the block diagram of ECP2M20 with one quad.

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 LatticeECP2/M devices contain one or more rows of sysMEM EBR blocks. sysMEM EBRs are large dedicated 18K fast memory blocks. Each sysMEM block can be configured in variety of depths and widths of RAM or ROM. In addition, LatticeECP2/M devices contain up to two rows of DSP Blocks. Each DSP block has multipliers and adder/accumulators, which are the building blocks for complex signal processing capabilities.

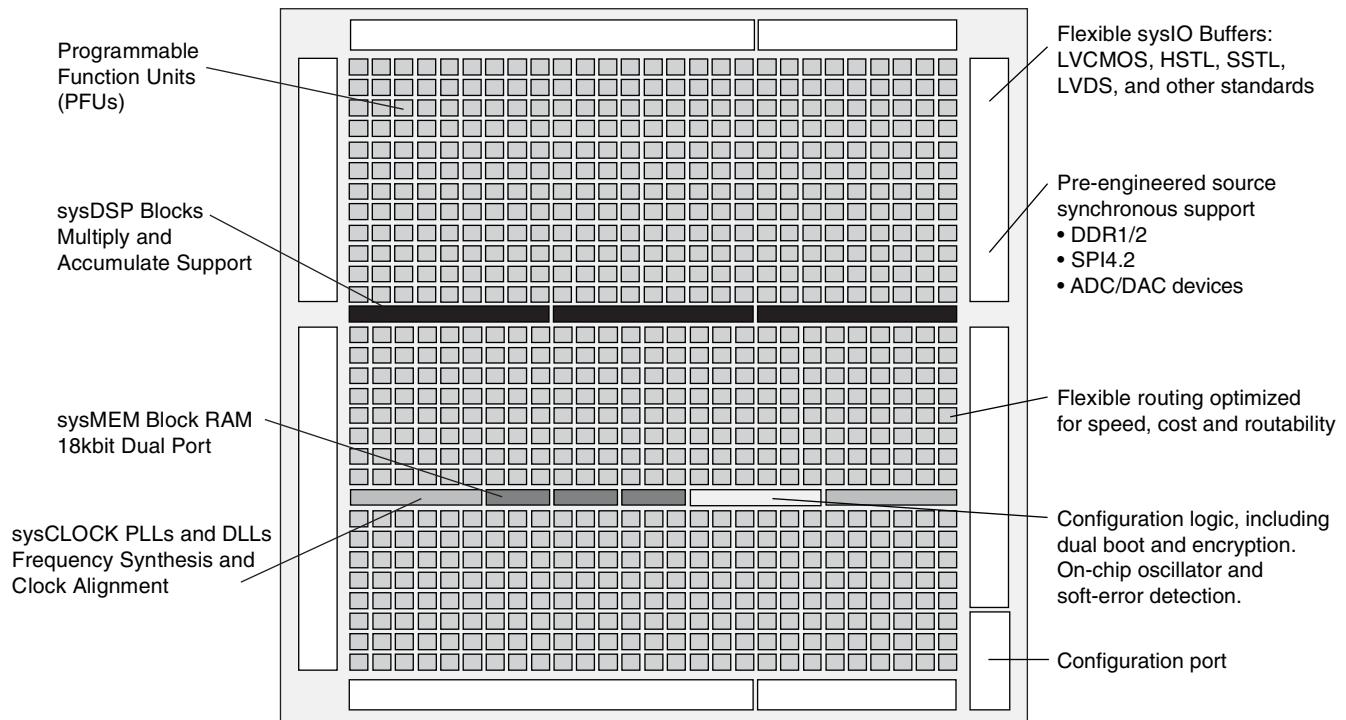
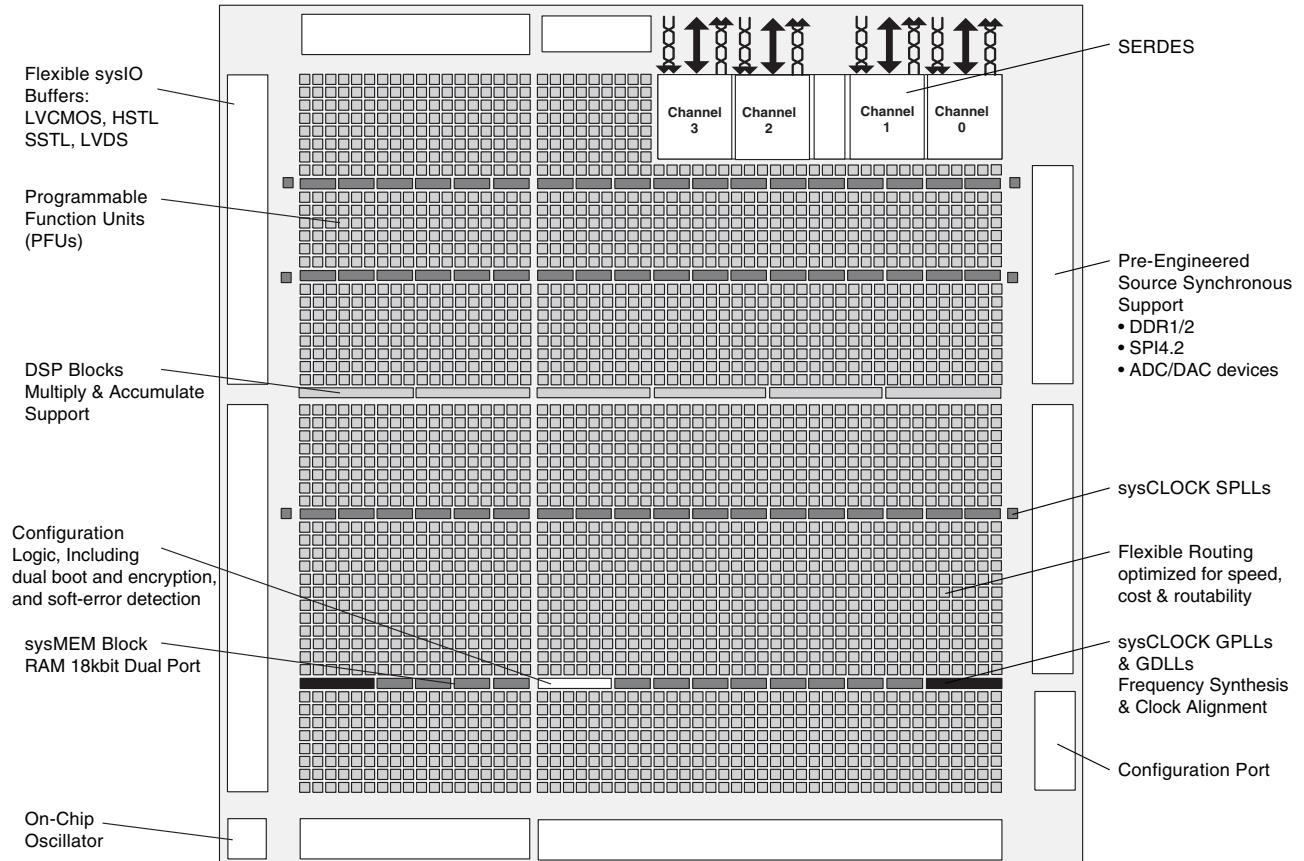
The LatticeECP2M devices feature up to 16 embedded 3.125Gbps SERDES (Serializer / Deserializer). Each SERDES Channel contains independent 8b/10b encoding / decoding, polarity adjust and elastic buffer logic. Each group of four SERDES along with its Physical Coding Sub-layer (PCS) block creates a Quad. The functionality of the SERDES/PCS Quads can be controlled by memory cells set during device configuration or by registers addressable during device operation. The registers in every quad can be programmed by a soft IP interface, referred to as the SERDES Client Interface (SCI). These quads (up to four) are located at the corners of the devices.

Each PIC block encompasses two PIOs (PIO pairs) with their respective sysIO buffers. The sysIO buffers of the LatticeECP2/M devices are arranged into eight 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. 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 the high speed source synchronous standards such as SPI4.2 along with memory interfaces including DDR2.

Other blocks provided include PLLs, DLLs and configuration functions. The LatticeECP2/M architecture provides two General PLLs (GPLL) and up to six Standard PLLs (SPLL) per device. In addition, each LatticeECP2/M family member provides two DLLs per device. The GPLPs and DLLs blocks are located in pairs at the end of the bottom-most EBR row; the DLL block located towards the edge of the device. The SPLL blocks are located at the end of the other EBR/DSP rows.

The configuration block that supports features such as configuration bit-stream de-encryption, transparent updates and dual boot support is located toward the center of this EBR row. Every device in the LatticeECP2/M family supports a sysCONFIG™ port located in the corner between banks four and five, 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 LatticeECP2/M devices use 1.2V as their core voltage.

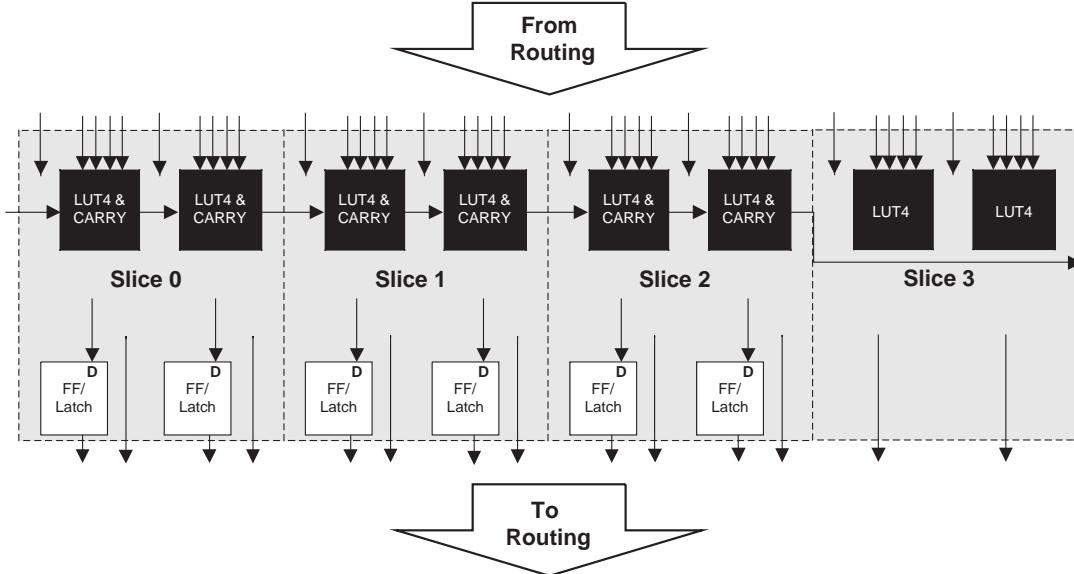
**Figure 2-1. Simplified Block Diagram, ECP2-6 Device (Top Level)****Figure 2-2. Simplified Block Diagram, ECP2M20 Device (Top Level)**

## PFU Blocks

The core of the LatticeECP2/M 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 2-3. All the interconnections to and from PFU blocks are from routing. There are 50 inputs and 23 outputs associated with each PFU block.

**Figure 2-3. PFU Diagram**



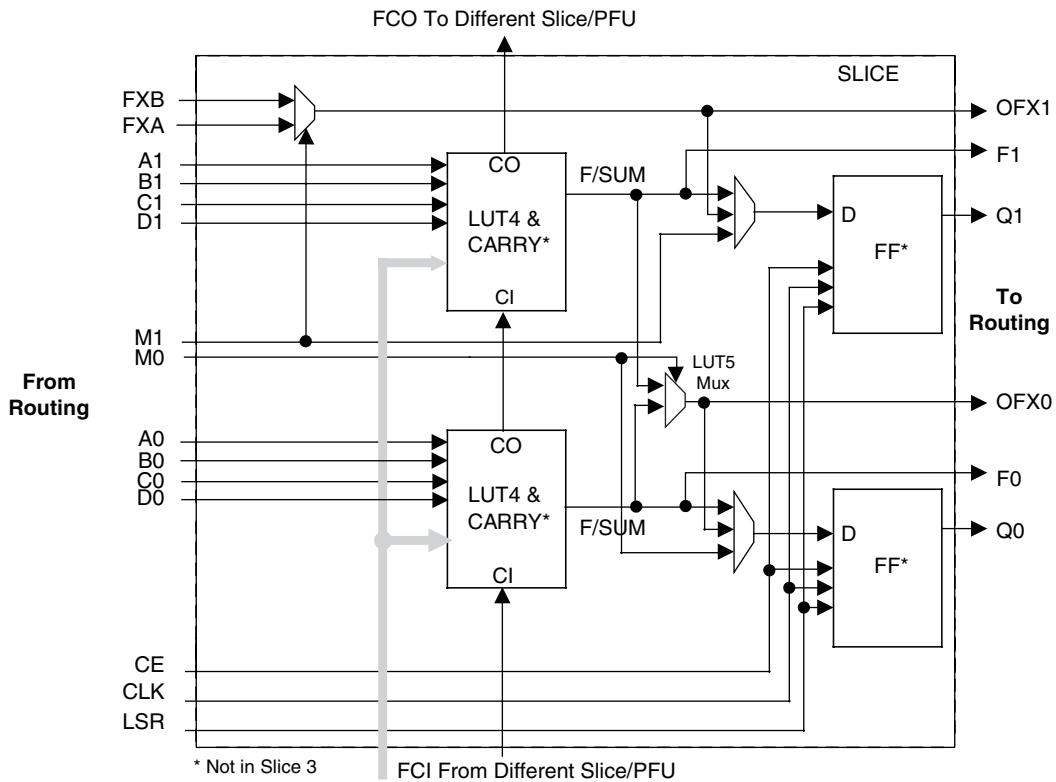
## Slice

Slice 0 through Slice 2 contain two LUT4s feeding two registers, whereas Slice 3 contains two LUT4s only. For PFUs, Slice 0 and Slice 2 can also 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 some 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. Figure 2-4 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.

**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, 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

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 13 input signals from routing and four signals to routing. Table 2-2 lists the signals associated with Slice 0 to Slice 2.

**Figure 2-4. Slice Diagram**

For Slices 0 and 2, memory control signals are generated from Slice 1 as follows:

- WCK is CLK
- WRE is from LSR
- DI[3:2] for Slice 2 and DI[1:0] for Slice 0 data
- WAD [A:D] is a 4bit address from slice 1 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 2-4 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 allows 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 Async 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

Two additional signals: Carry Generate and Carry Propagate are generated per slice in this mode, allowing 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 2 as a 16x1-bit memory. Slice 1 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.

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 on using RAM in LatticeECP2/M devices, please see details of additional technical documentation at the end of this data sheet.

**Table 2-3. Number of Slices Required For Implementing 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 the ROM mode. Preloading is accomplished through the programming interface during PFU configuration.

**Routing**

There are many resources provided in the LatticeECP2/M 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 inter-PFU connections are made with x1 (spans two PFU), x2 (spans three PFU) and x6 (spans seven PFU). The x1 and x2 connections provide fast and efficient connections in horizontal and vertical directions. The x2 and x6 resources are buffered allowing both short and long connections routing between PFUs.

The LatticeECP2/M family has an enhanced routing architecture that produces a compact design. The ispLEVER design tool takes the output of the synthesis tool and places and routes the design. Generally, the place and route tool is completely automatic, although an interactive routing editor is available to optimize the design.

**sysCLOCK Phase Locked Loops (GPLL/SPLL)**

The sysCLOCK PLLs provide the ability to synthesize clock frequencies. All the devices in the LatticeECP2/M family support two General Purpose PLLs (GPLLs) which are full-featured PLLs. In addition, some of the larger devices have two to six Standard PLLs (SPLLs) that have a subset of GPLL functionality.

**General Purpose PLL (GPLL)**

The architecture of the GPLL is shown in Figure 2-5. A description of the GPLL functionality follows.

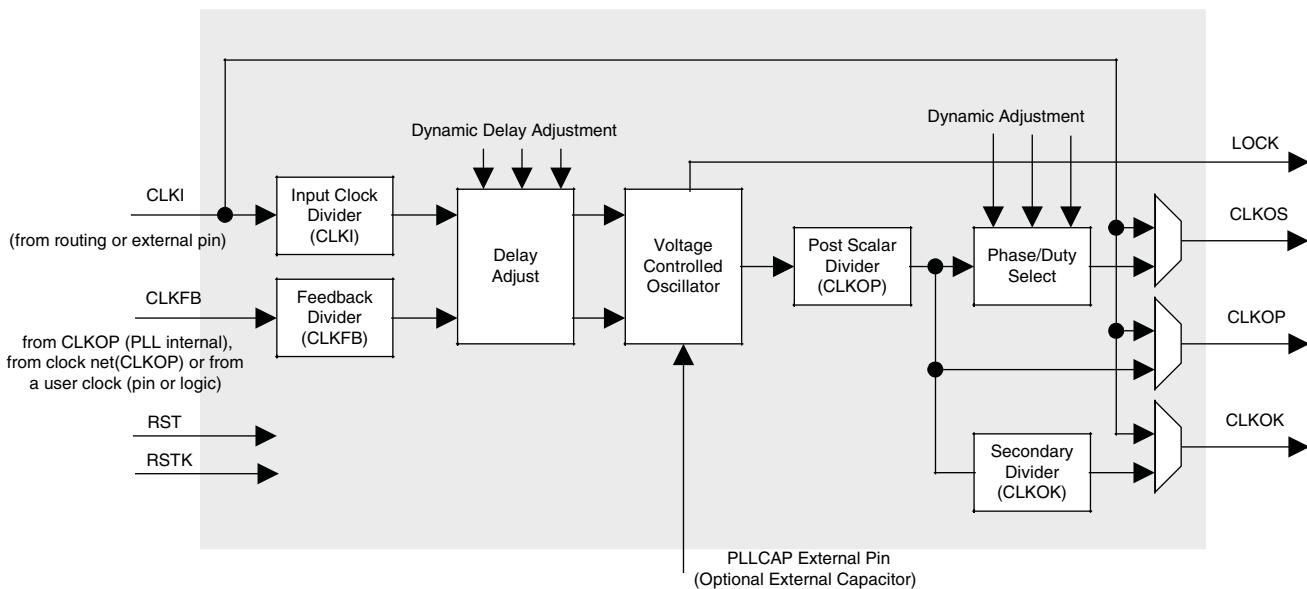
CLKI is the reference frequency (generated either from the pin or from routing) for the PLL. CLKI feeds into the Input Clock Divider block. The CLKFB is the feedback signal (generated from CLKOP or from a user clock PIN/ logic). This signal feeds into the Feedback Divider. The Feedback Divider is used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

The Delay Adjust Block adjusts either the delays of the reference or feedback signals. The Delay Adjust Block can either be programmed during configuration or can be adjusted dynamically. The setup, hold or clock-to-out times of the device can be improved by programming a delay in the feedback or input path of the PLL which will advance or delay the output clock with reference to the input clock.

Following the Delay Adjust Block, both the input path and feedback signals enter 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 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. LatticeECP2/M devices have two dedicated pins on the left and right edges of the device for connecting optional external capacitors to the VCO. This allows the PLLs to operate at a lower frequency. This is a shared resource which can only be used by one PLL (GPLL or SPLL) per side.

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

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

**Figure 2-5. General Purpose PLL (GPLL) Diagram**

### Standard PLL (SPLL)

Some of the larger devices have two to six Standard PLLs (SPLLs). SPLLs have the same features as GPLPs but without delay adjustment capability. SPLLs also provide different parametric specifications. For more information, please see details of additional technical documentation at the end of this data sheet.

Table 2-4 provides a description of the signals in the GPLP and SPLL blocks.

**Table 2-4. GPLP and SPLL Blocks Signal Descriptions**

Signal	I/O	Description
CLKI	I	Clock input from external pin or routing
CLKFB	I	PLL feedback input from CLKOP (PLL internal), from clock net (CLKOP) 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
CLKOS	O	PLL output clock to clock tree (phase shifted/duty cycle changed)
CLKOP	O	PLL output clock to clock tree (no phase shift)
CLKOK	O	PLL output to clock tree through secondary clock divider
LOCK	O	"1" indicates PLL LOCK to CLKI
DDAMODE <sup>1</sup>	I	Dynamic Delay Enable. "1": Pin control (dynamic), "0": Fuse Control (static)
DDAIZR <sup>1</sup>	I	Dynamic Delay Zero. "1": delay = 0, "0": delay = on
DDAILAG <sup>1</sup>	I	Dynamic Delay Lag/Lead. "1": Lead, "0": Lag
DDAIDEL[2:0] <sup>1</sup>	I	Dynamic Delay Input
DPA MODES	I	DPA (Dynamic Phase Adjust/Duty Cycle Select) mode
DPHASE [3:0]	I	DPA Phase Adjust inputs
DDDUTY [3:0]	—	DPA Duty Cycle Select inputs

1. These signals are not available in SPLL.

## Delay Locked Loops (DLL)

In addition to PLLs, the LatticeECP2/M family of devices has two DLLs per device.

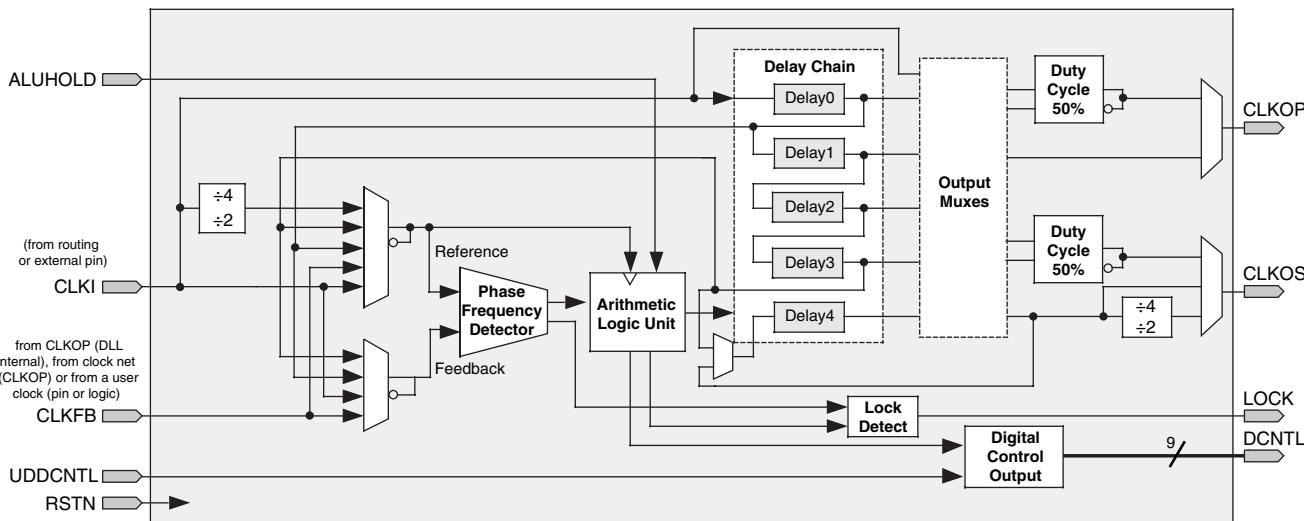
CLKI is the input frequency (generated either from the pin or routing) for the DLL. CLKI feeds into the output muxes block to bypass the DLL, directly to the DELAY CHAIN block and (directly or through divider circuit) to the reference input of the Phase Frequency Detector (PFD) input mux. The reference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

The PFD produces a binary number proportional to the phase and frequency difference between the reference and feedback signals. This binary output of the PFD is feed into a Arithmetic Logic Unit (ALU). 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 DLL\_DEL delay block. 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 independent clock outputs, CLKOP and CLKOS. These outputs can individually select one of the outputs from the tapped delay line. The CLKOS has optional fine phase shift and divider blocks to allow this output to be further modified, if required. The fine phase 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 2-6 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. For more information on the DLL, please see details of additional technical documentation at the end of this data sheet.

**Figure 2-6. Delay Locked Loop Diagram (DLL)**

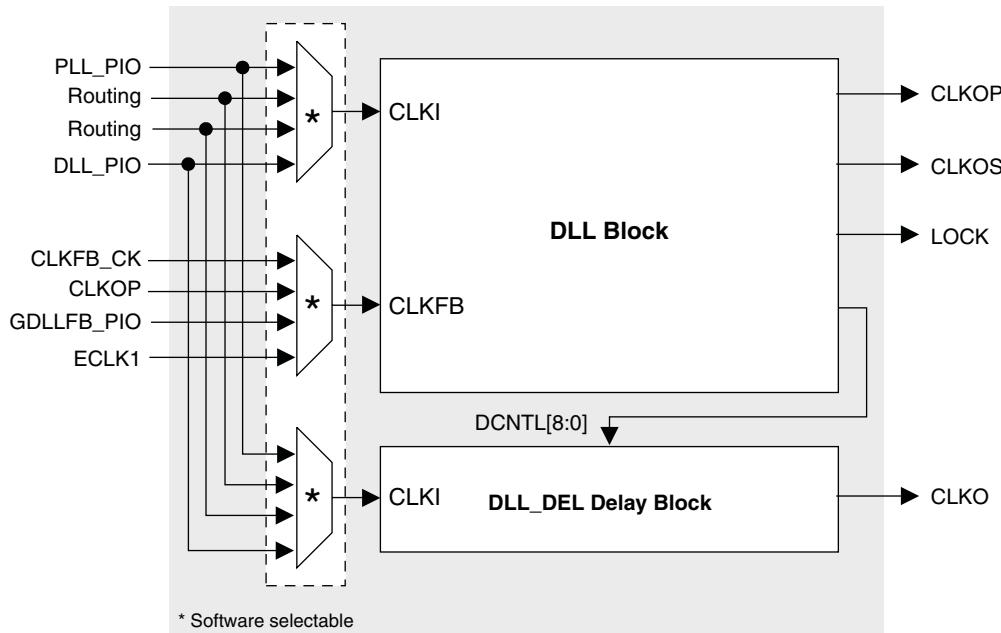


**Table 2-5. DLL Signals**

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
DCNTL[8:0]	O	Encoded digital control signals for PIC INDEL and slave delay calibration
CLKOP	O	The primary clock output
CLKOS	O	The secondary clock output with fine phase shift and/or division by 2 or by 4
LOCK	O	Active high phase lock indicator

### DLL\_DEL Delay Block

Closely associated with each DLL is a DLL\_DEL block. This is a delay block consisting of a delay line with taps and a selection scheme that selects one of the taps. The DCNTL[8:0] bus controls the delay of the CLKO signal. Typically this is the delay setting that the DLL uses to achieve phase alignment. This results in the delay providing a calibrated 90° phase shift that is useful in centering a clock in the middle of a data cycle for source synchronous data. The CLKO signal feeds the edge clock network. Figure 2-7 shows the connections between the DLL block and the DLL\_DEL delay block. For more information, please see details of additional technical documentation at the end of this data sheet.

**Figure 2-7. DLL\_DEL Delay Block**

### PLL/DLL Cascading

LatticeECP2/M devices have been designed to allow certain combinations of PLL (GPLL and SPLL) and DLL cascading. The allowable combinations are as follows:

- PLL to PLL supported
- PLL to DLL supported

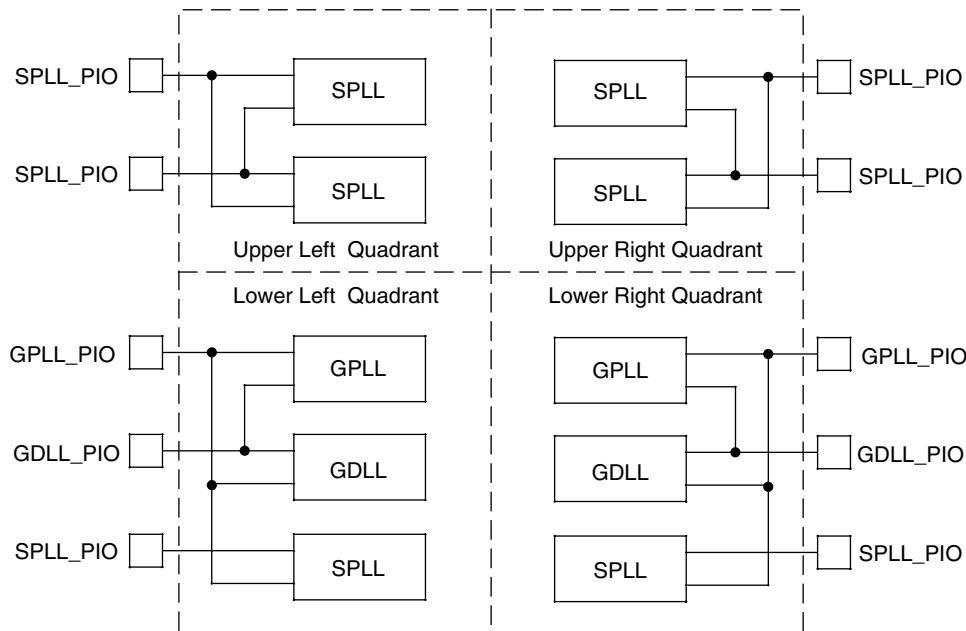
The DLLs in the LatticeECP2/M are used to shift the clock in relation to the data for source synchronous inputs. PLLs are used for frequency synthesis and clock generation for source synchronous interfaces. Cascading PLL and DLL blocks allows applications to utilize the unique benefits of both DLLs and PLLs.

For further information on the DLL, please see details of additional technical documentation at the end of this data sheet.

## GPLL/SPLL/GDLL PIO Input Pin Connections (LatticeECP2M Family Only)

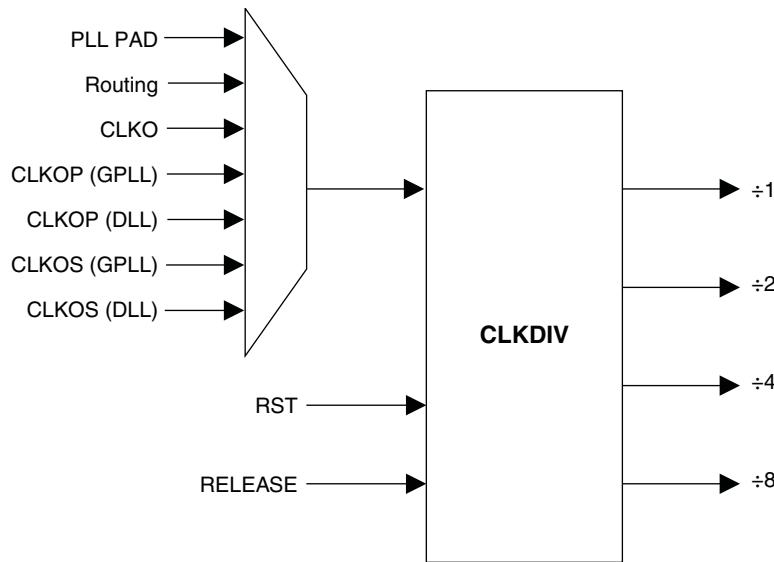
All LatticeECP2M devices contain two GDLLs, two GPLPs and six SPLLs, arranged in quadrants as shown in Figure 2-8. In the LatticeECP2M devices GPLPs, SPLLs and GDLLs share their input pins. Figure 2-8 shows the sharing of SPLLs input pin connections in the upper two quadrants and the sharing of GDLL, GPLP and SPLL input pin connections in the lower two quadrants.

**Figure 2-8. Sharing of PIO Pins by GPLP, SPLL and GDLL in LatticeECP2M Devices**



## Clock Dividers

LatticeECP2/M 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, DLL\_DEL delay blocks, 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 synchronously forces all outputs to low. The RELEASE signal releases outputs synchronously to the input clock. For further information on clock dividers, please see details of additional technical documentation at the end of this data sheet. Figure 2-9 shows the clock divider connections.

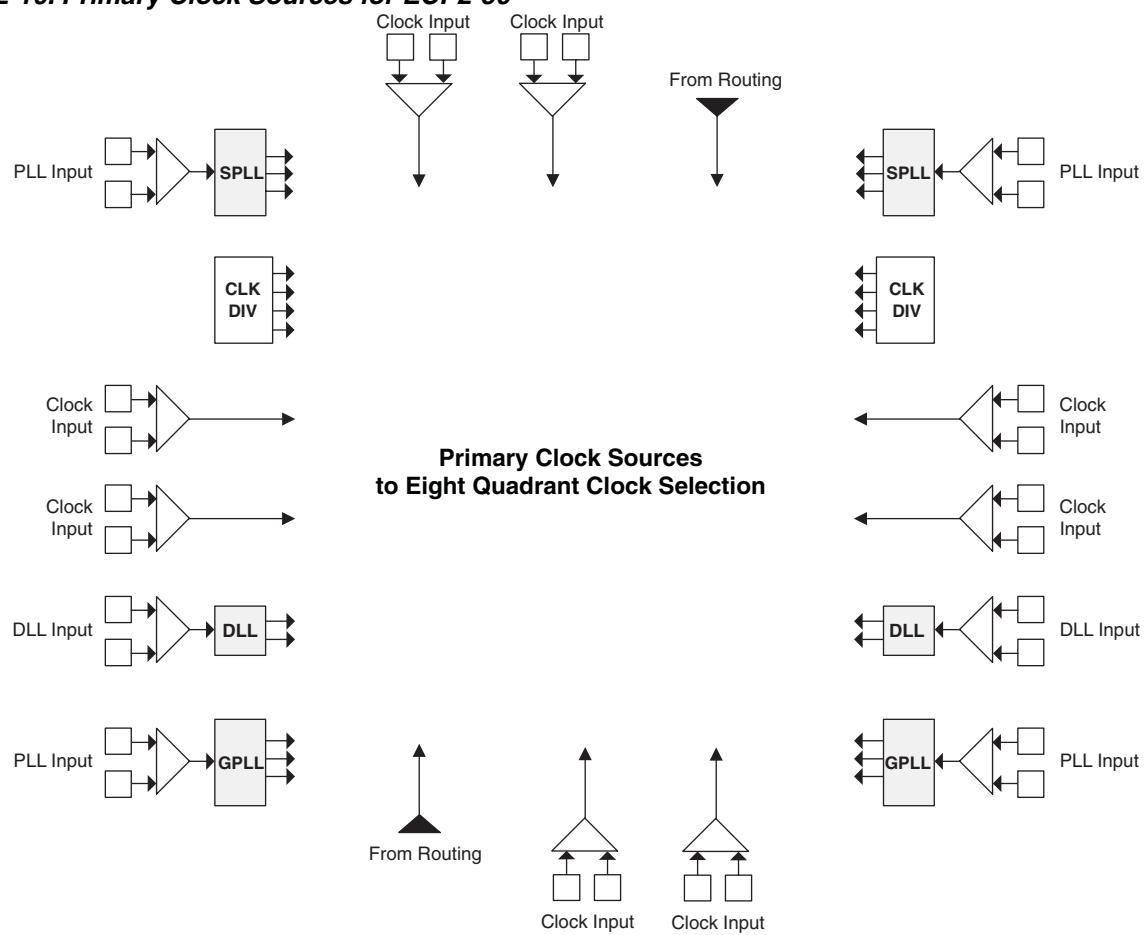
**Figure 2-9. Clock Divider Connections**

## Clock Distribution Network

LatticeECP2/M devices have eight quadrant-based primary clocks and eight flexible region-based secondary clocks/control signals. Two high performance edge clocks are available on each edge of the device to support high speed interfaces. These clock inputs are selected from external I/Os, the sysCLOCK PLLs, DLLs or routing. These clock inputs are fed throughout the chip via a clock distribution system.

### Primary Clock Sources

LatticeECP2/M devices derive clocks from five primary sources: PLL (GPLL and SPLL) outputs, DLL outputs, CLKDIV outputs, dedicated clock inputs and routing. LatticeECP2/M devices have two to eight sysCLOCK PLLs and two DLLs, located on the left and right sides of the device. There are eight dedicated clock inputs, two on each side of the device. Figure 2-10 shows the primary clock sources.

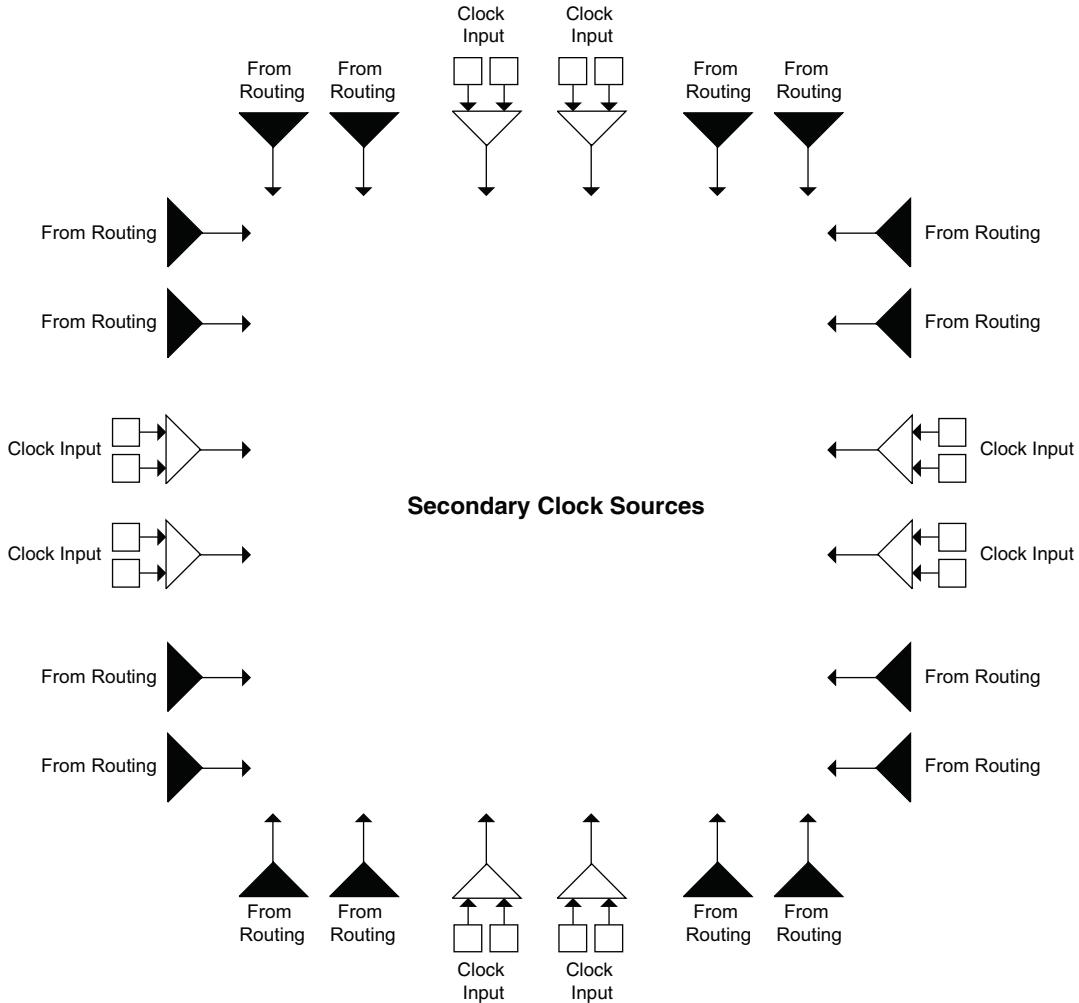
**Figure 2-10. Primary Clock Sources for ECP2-50**

Note: This diagram shows sources for the ECP2-50 device. Smaller LatticeECP2 devices have fewer SPLLs. All LatticeECP2M device have six SPLLs.

## Secondary Clock/Control Sources

LatticeECP2/M devices derive secondary clocks (SC0 through SC7) from eight dedicated clock input pads and the rest from routing. Figure 2-11 shows the secondary clock sources.

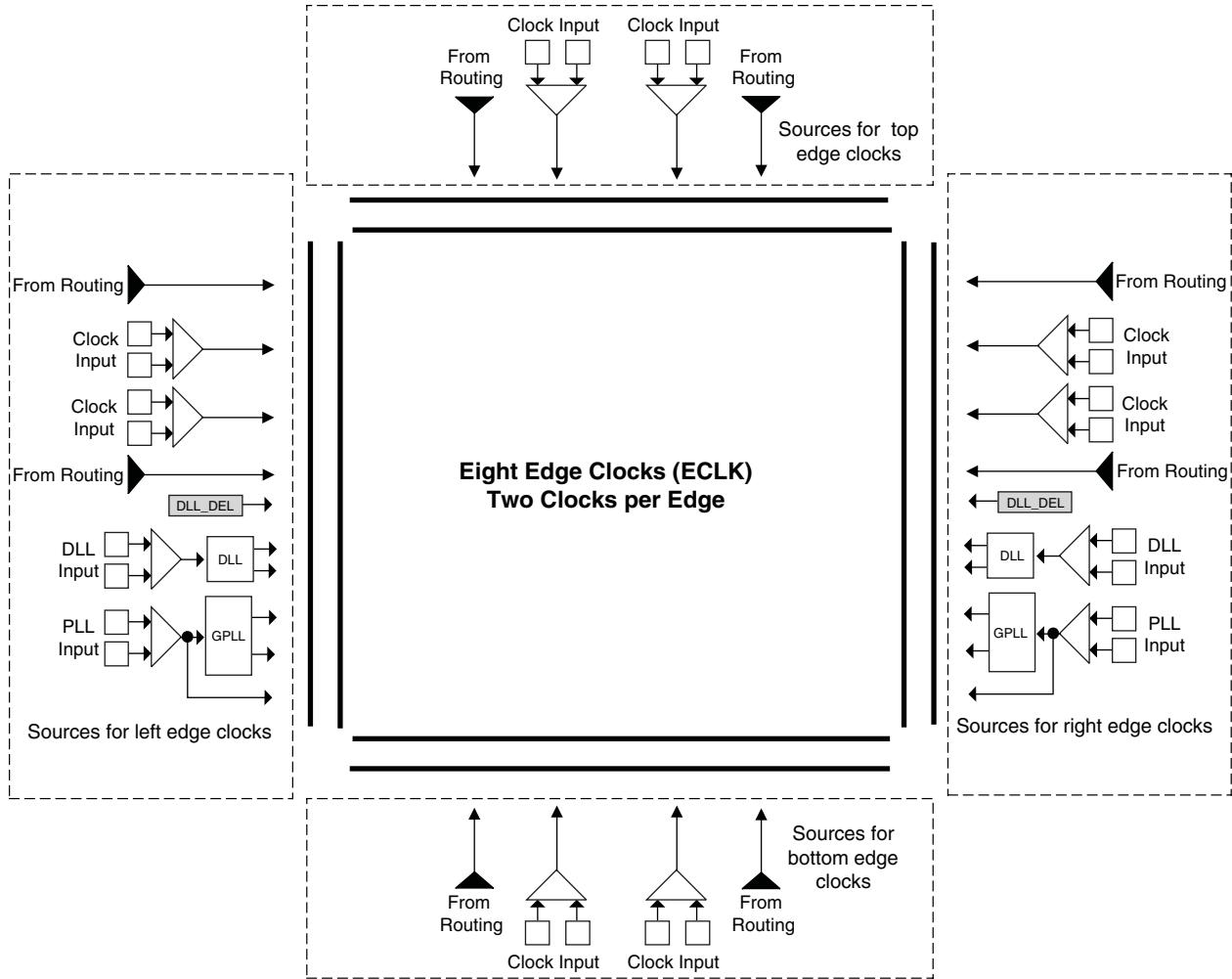
**Figure 2-11. Secondary Clock Sources**



## 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 and clock dividers as shown in Figure 2-12.

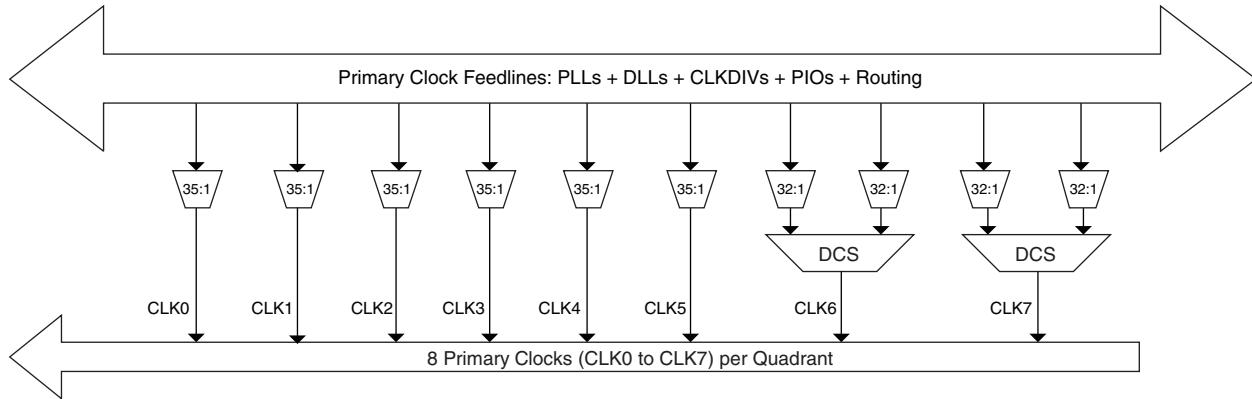
**Figure 2-12. Edge Clock Sources**



## Primary Clock Routing

The clock routing structure in LatticeECP2/M 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 2-13 shows the clock routing for one quadrant. Each quadrant mux is identical. If desired, any clock can be routed globally.

**Figure 2-13. Per Quadrant Primary Clock Selection**

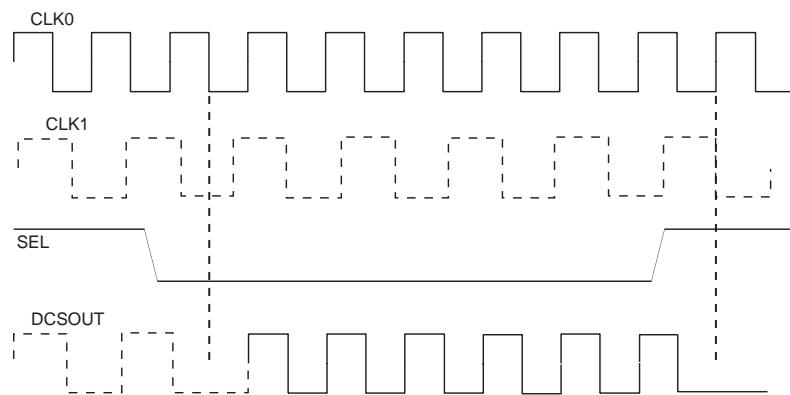


## Dynamic Clock Select (DCS)

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 irrespective of when the select signal is toggled. There are two DCS blocks per quadrant; in total, 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 2-13).

Figure 2-14 shows the timing waveforms of the default DCS operating mode. The DCS block can be programmed to other modes. For more information on the DCS, please see details of additional technical documentation at the end of this data sheet.

**Figure 2-14. DCS Waveforms**

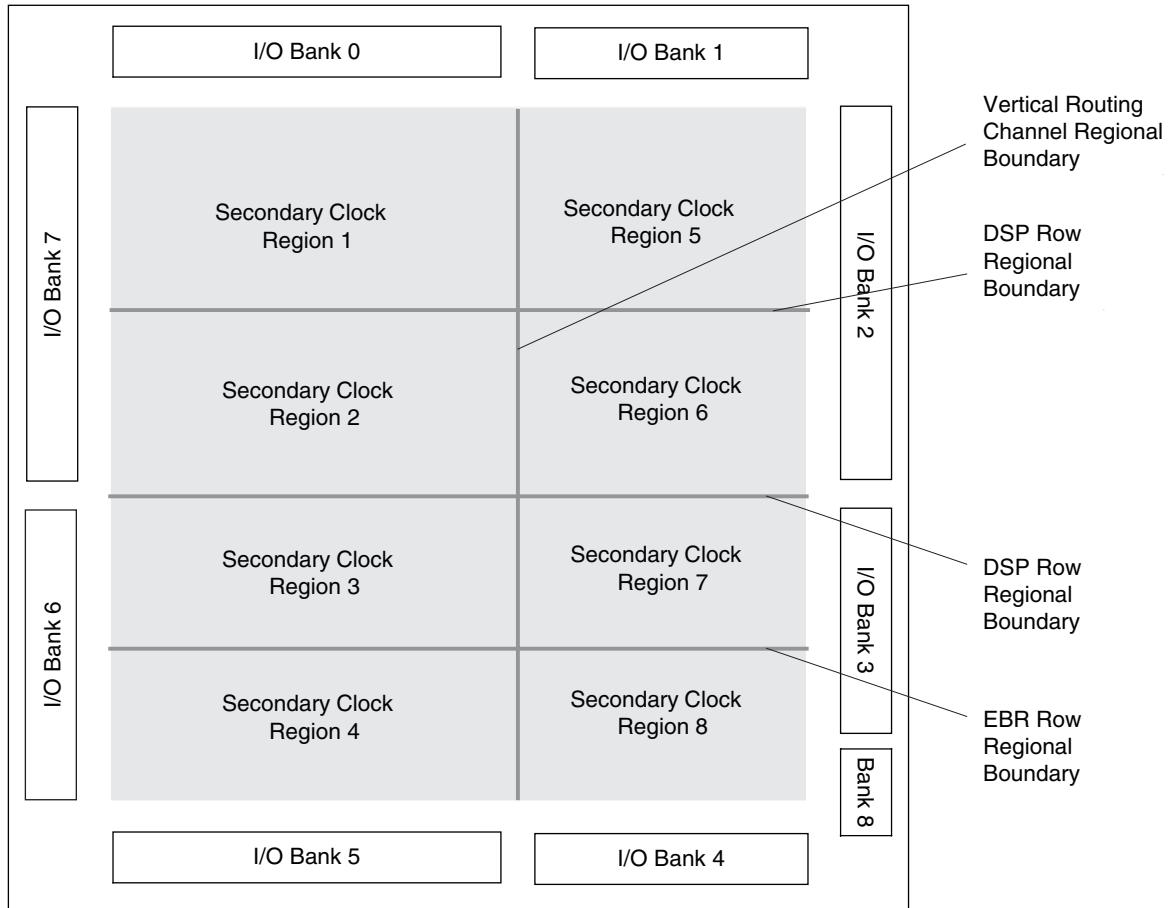


## Secondary Clock/Control Routing

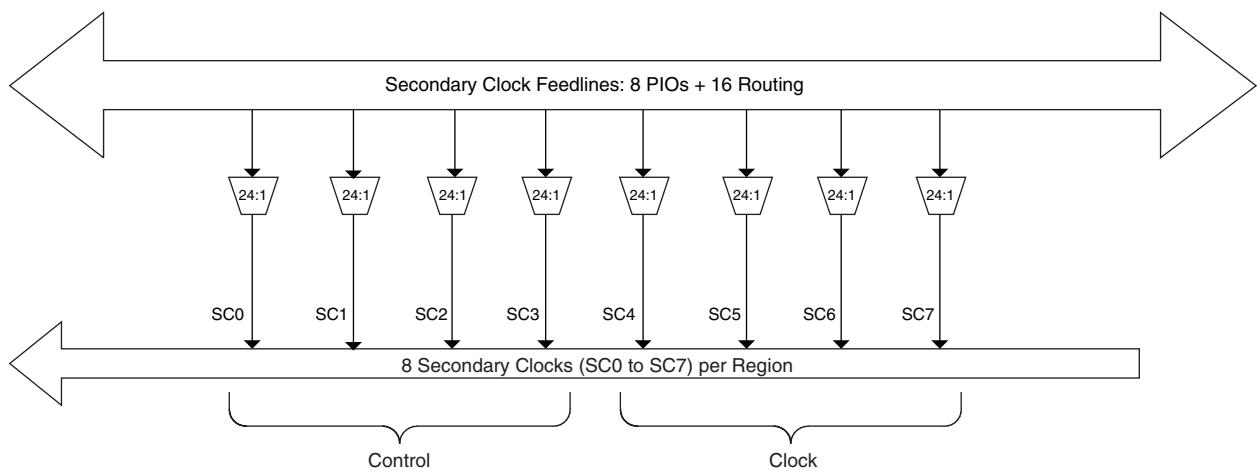
Secondary clocks in the LatticeECP2 devices are region-based resources. EBR/DSP rows and a special vertical routing channel bound the secondary clock regions. This special vertical routing channel aligns with either the left edge of the center DSP block in the DSP row or the center of the DSP row. Figure 2-15 shows this special vertical routing channel and the eight secondary clock regions for the ECP2-50. LatticeECP2 devices have eight secondary clock resources per region (SC0 to SC7).

The secondary clock muxes are located in the center of the device. Figure 2-16 shows the mux structure of the secondary clock routing. Secondary clocks SC0 to SC3 are used for high fan-out control and SC4 to SC7 are used for clock signals.

**Figure 2-15. Secondary Clock Regions ECP2-50**



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

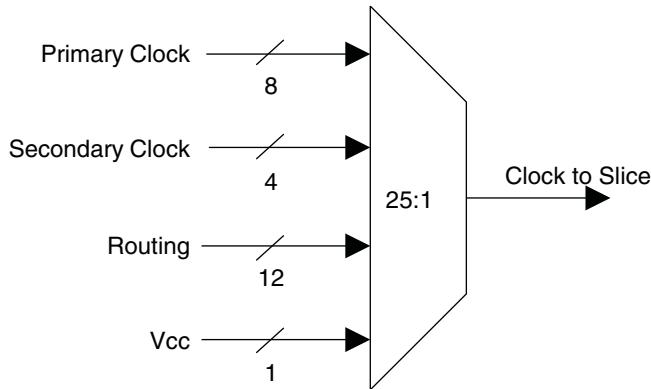


## Slice Clock Selection

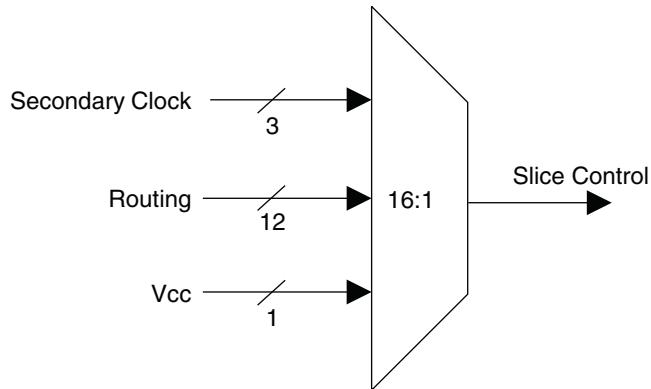
Figure 2-17 shows the clock selections and Figure 2-18 shows the control selections for Slice0 through Slice2. All the primary clocks and the four secondary clocks are routed to this clock selection mux. Other signals via routing can be used as a clock input to the slices. Slice controls are generated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

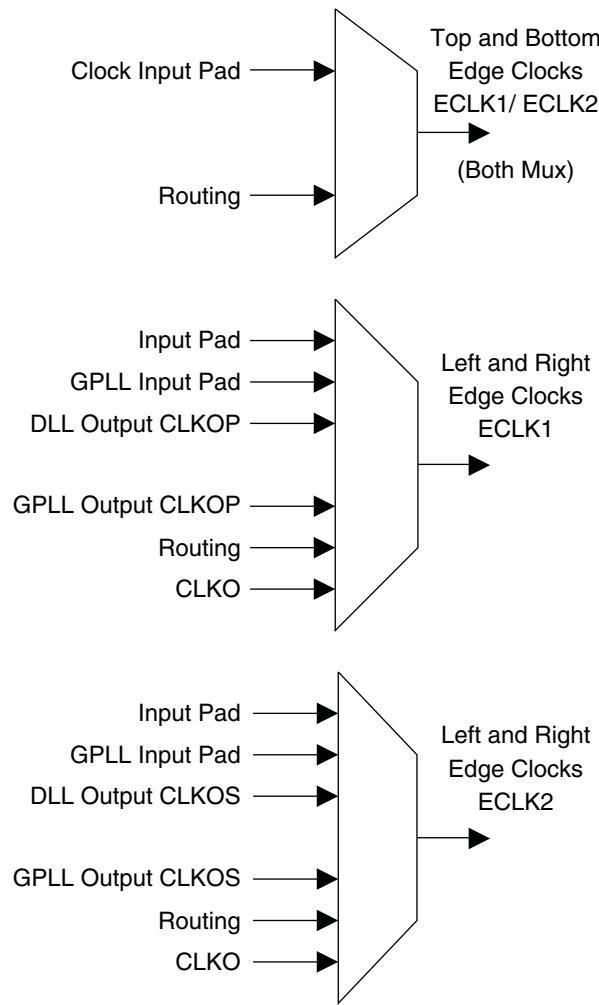


**Figure 2-18. Slice0 through Slice2 Control Selection**



## Edge Clock Routing

LatticeECP2/M 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 eight edge clocks per device: two edge clocks per edge. Different PLL and DLL outputs are routed to the two muxes on the left and right sides of the device. In addition, the CLKO signal (generated from the DLL\_DEL block) is routed to all the edge clock muxes on the left and right sides of the device. Figure 2-19 shows the selection muxes for these clocks.

**Figure 2-19. Edge Clock Mux Connections**

## sysMEM Memory

LatticeECP2/M devices contain a number of sysMEM Embedded Block RAM (EBR). The EBR consists of an 18-Kbit RAM with dedicated input and output registers.

### 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 2-6. 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.

**Table 2-6. 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 RAMs 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

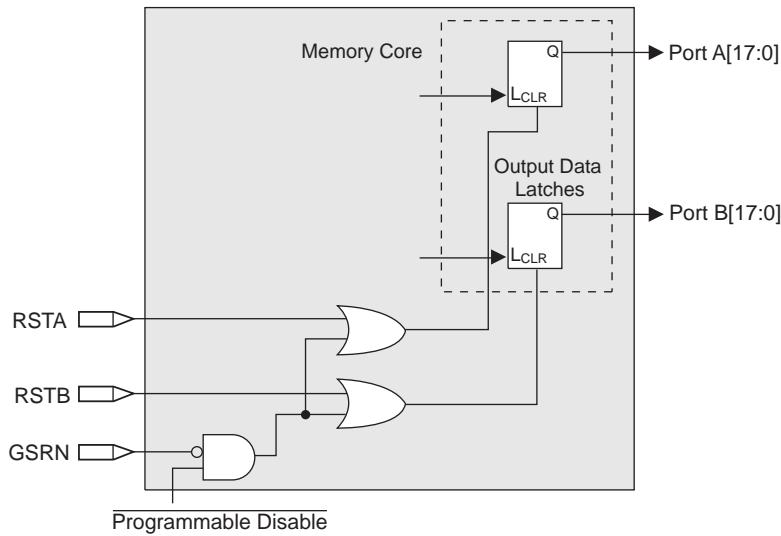
In all the sysMEM RAM modes the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

EBR memory supports three 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 – When new data is being 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 resets both ports. The output data latches and associated resets for both ports are as shown in Figure 2-20.

**Figure 2-20. Memory Core Reset**

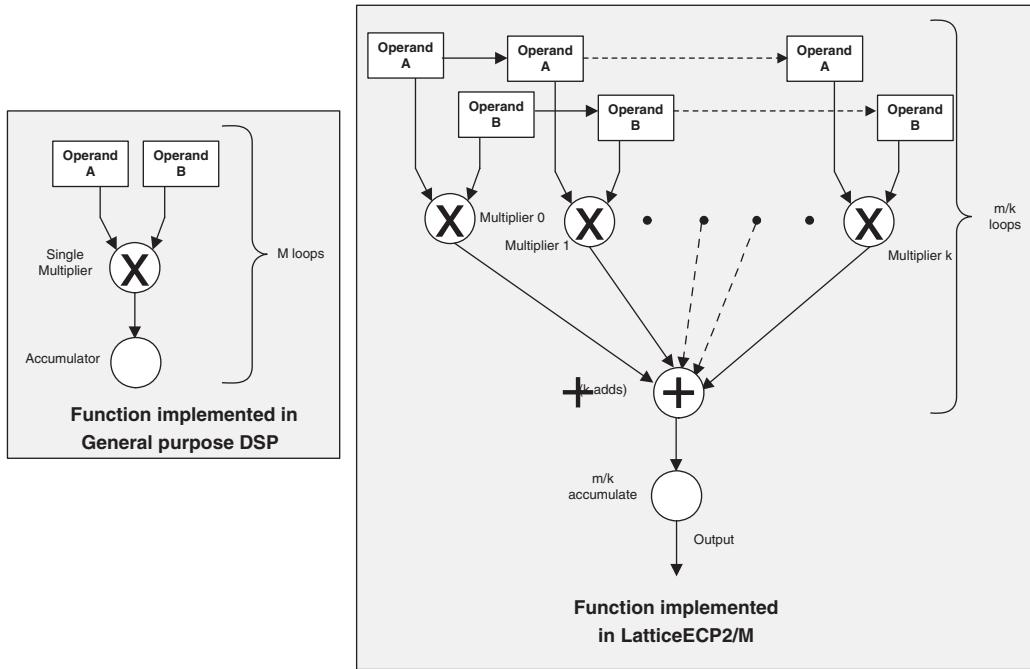
For further information on the sysMEM EBR block, please see the details of additional technical documentation at the end of this data sheet.

## sysDSP™ Block

The LatticeECP2/M family provides a sysDSP block 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 Block Approach Compare 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 LatticeECP2/M, on the other hand, has many DSP blocks that support different data-widths. This allows the designer to use highly parallel implementations of DSP functions. The designer can optimize the DSP performance vs. area by choosing appropriate level of parallelism. Figure 2-21 compares the fully serial and the mixed parallel and serial implementations.

**Figure 2-21. Comparison of General DSP and LatticeECP2/M Approaches**

### sysDSP Block Capabilities

The sysDSP block in the LatticeECP2/M family supports four functional elements in three 9, 18 and 36 data path widths. The user selects a function element for a DSP block and then selects the width and type (signed/unsigned) of its operands. The operands in the LatticeECP2/M family sysDSP Blocks can be either signed or unsigned but not mixed within a function element. Similarly, the operand widths cannot be mixed within a block. In LatticeECP2/M family of devices the DSP elements can be concatenated.

The resources in each sysDSP block can be configured to support the following four elements:

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

The number of elements available in each block depends in the width selected from the three available options x9, x18, and x36. A number of these elements are concatenated for highly parallel implementations of DSP functions. Table 2-7 shows the capabilities of the block.

**Table 2-7. Maximum Number of Elements in a Block**

Width of Multiply	x9	x18	x36
MULT	8	4	1
MAC	2	2	—
MULTADDSUB	4	2	—
MULTADDSUBSUM	2	1	—

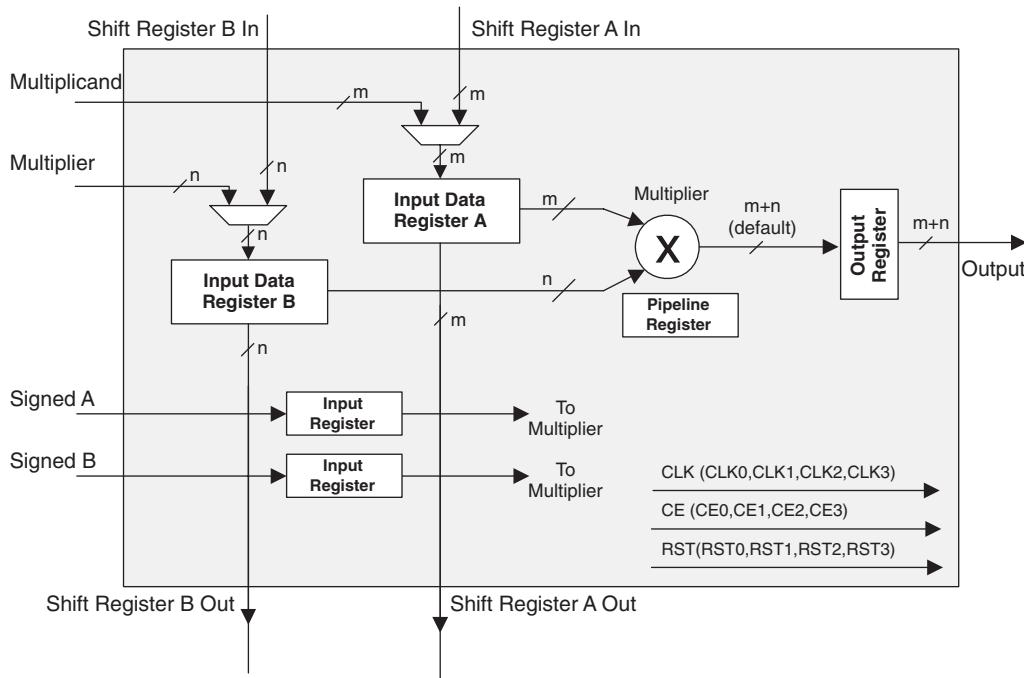
Some options are available in four elements. The input register in all the elements can be directly loaded or can be loaded as shift register from previous operand registers. By selecting 'dynamic operation' the following operations are possible:

- In the ‘Signed/Unsigned’ options the operands can be switched between signed and unsigned on every cycle.
- 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.

## MULT sysDSP Element

This multiplier element implements a multiply with no addition or accumulator nodes. The two operands, A and B, are multiplied and the result is available at the output. The user can enable the input/output and pipeline registers. Figure 2-22 shows the MULT sysDSP element.

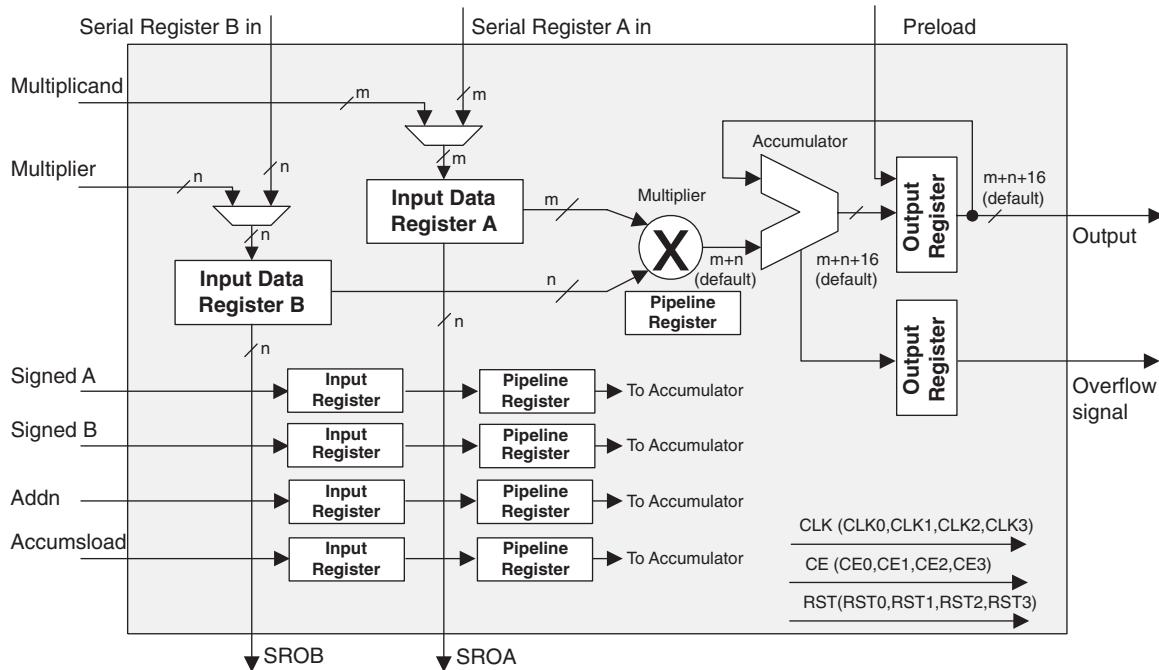
**Figure 2-22. MULT sysDSP Element**



## MAC sysDSP Element

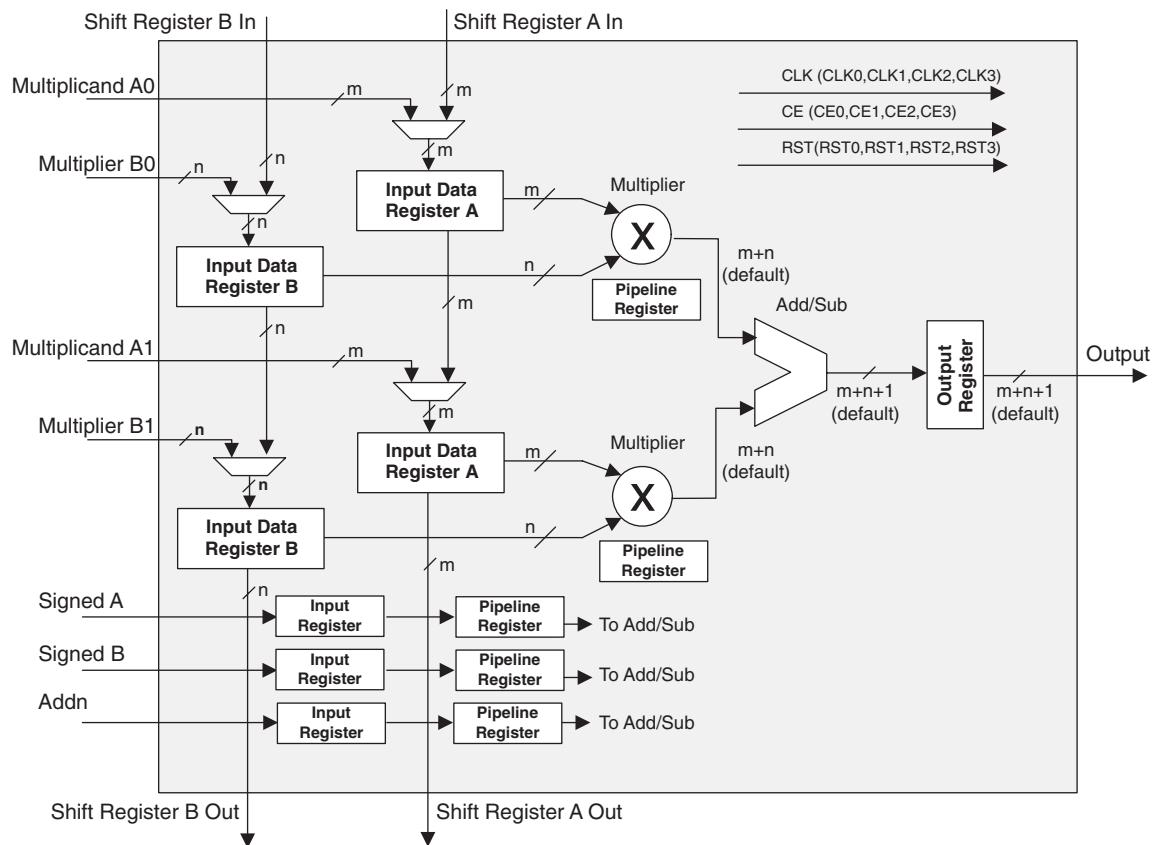
In this case, the two operands, A and B, are multiplied and the result is added with the previous accumulated value. This accumulated value is available at the output. The user can enable the input and pipeline registers but the output register is always enabled. The output register is used to store the accumulated value. The Accumulators in the DSP blocks in LatticeECP2/M family can be initialized dynamically. A registered overflow signal is also available. The overflow conditions are provided later in this document. Figure 2-23 shows the MAC sysDSP element.

**Figure 2-23. MAC sysDSP**



**MULTADDSSUB sysDSP Element**

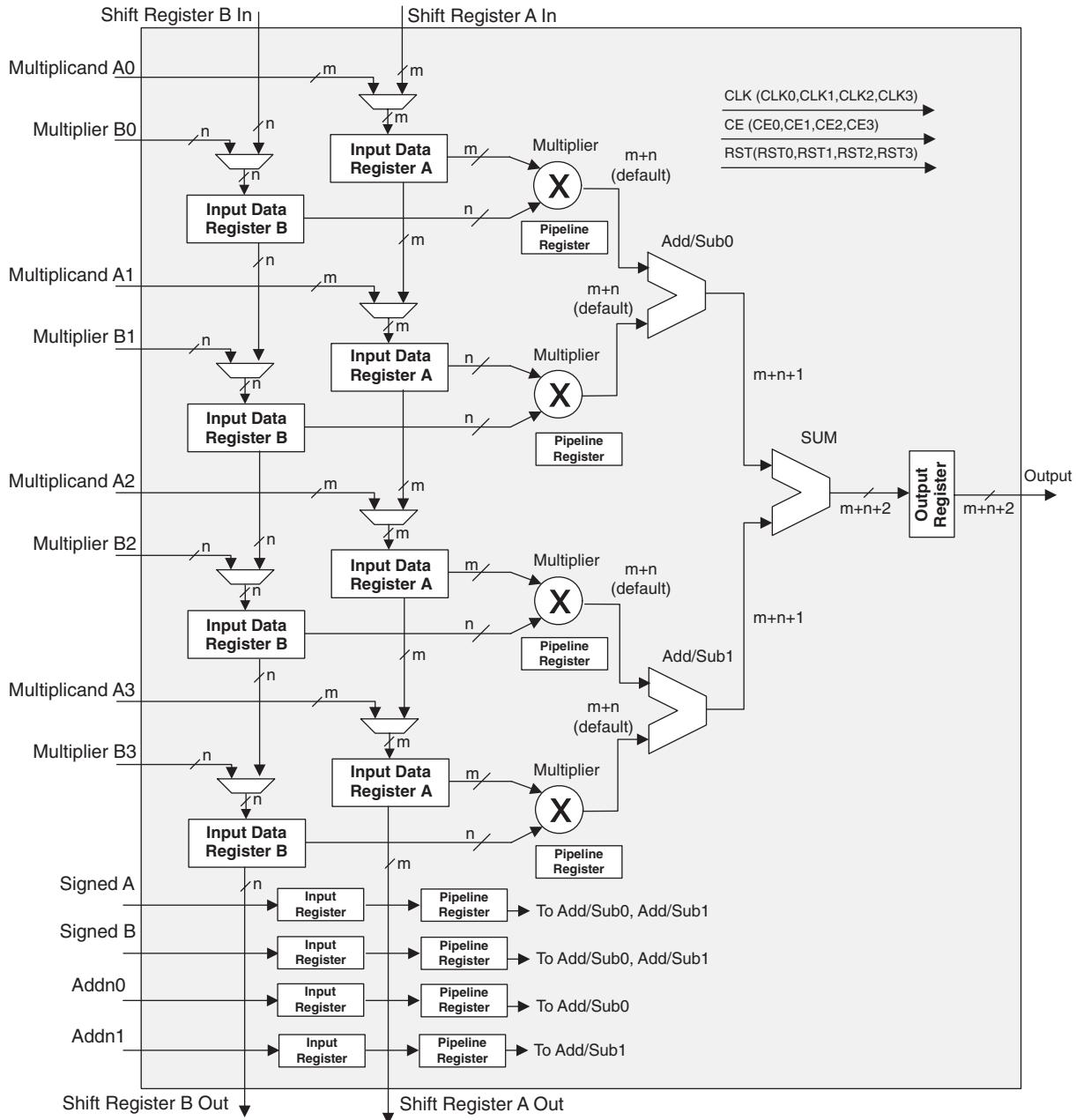
In this case, the operands A0 and B0 are multiplied and the result is added/subtracted with the result of the multiplier operation of operands A1 and A2. The user can enable the input, output and pipeline registers. Figure 2-24 shows the MULTADDSSUB sysDSP element.

**Figure 2-24. MULTADDSSUB**

## MULTADDSSUBSUM sysDSP Element

In this case, the operands A0 and B0 are multiplied and the result is added/subtracted with the result of the multiplier operation of operands A1 and B1. Additionally the operands A2 and B2 are multiplied and the result is added/subtracted with the result of the multiplier operation of operands A3 and B3. The result of both addition/subtraction are added in a summation block. The user can enable the input, output and pipeline registers. Figure 2-25 shows the MULTADDSSUBSUM sysDSP element.

**Figure 2-25. MULTADDSSUBSUM**



## Clock, Clock Enable and Reset Resources

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

Reset (RST) are selected from their four respective sources (CE0, CE1, CE2, CE3 and RST0, RST1, RST2, RST3) at each input register, pipeline register and output register.

### Signed and Unsigned with Different Widths

The DSP block supports different widths of signed and unsigned multipliers besides x9, x18 and x36 widths. For unsigned operands, unused upper data bits should be filled to create a valid x9, x18 or x36 operand. For signed two's complement operands, sign extension of the most significant bit should be performed until x9, x18 or x36 width is reached. Table 2-8 provides an example of this.

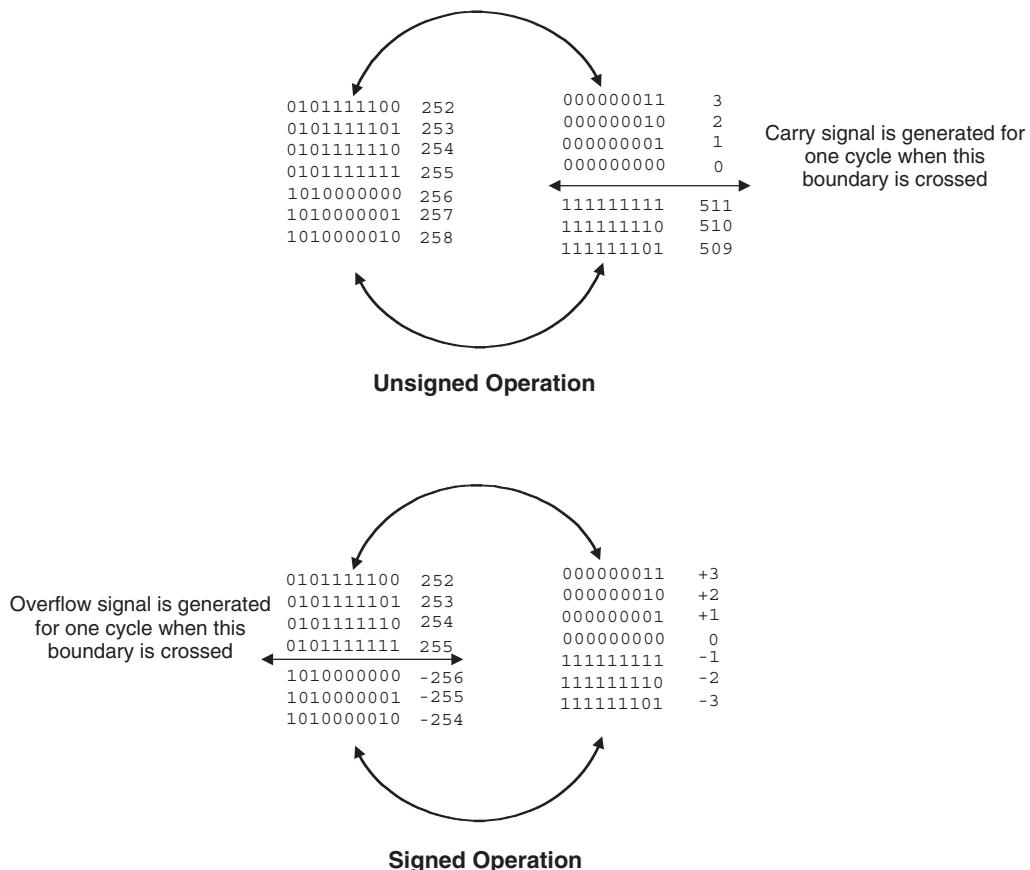
**Table 2-8. Sign Extension Example**

Number	Unsigned	Unsigned 9-bit	Unsigned 18-bit	Signed	Two's Complement Signed 9 Bits	Two's Complement Signed 18 Bits
+5	0101	000000101	0000000000000000101	0101	000000101	0000000000000000101
-6	N/A	N/A	N/A	1010	111111010	111111111111111010

### OVERFLOW Flag from MAC

The sysDSP block provides an overflow output to indicate that the accumulator has overflowed. When two unsigned numbers are added and the result is a smaller number than the accumulator, “roll-over” is said to have occurred and an overflow signal is indicated. When two positive numbers are added with a negative sum and when two negative numbers are added with a positive sum, then the accumulator “roll-over” is said to have occurred and an overflow signal is indicated. Note that when overflow occurs the overflow flag is present for only one cycle. By counting these overflow pulses in FPGA logic, larger accumulators can be constructed. The conditions overflow signal for signed and unsigned operands are listed in Figure 2-26.

**Figure 2-26. Accumulator Overflow/Underflow**



**IPexpress™**

The user can access the sysDSP block via the ispLEVER IPexpress tool which provides the option to configure each DSP module (or group of modules) or by direct HDL instantiation. In addition, Lattice has partnered with The MathWorks® to support instantiation in the Simulink® tool, a graphical simulation environment. Simulink works with ispLEVER to dramatically shorten the DSP design cycle in Lattice FPGAs.

**Optimized DSP Functions**

Lattice provides a library of optimized DSP IP functions. Some of the IP cores planned for the LatticeECP2/M DSP include the Bit Correlator, Fast Fourier Transform, Finite Impulse Response (FIR) Filter, Reed-Solomon Encoder/Decoder, Turbo Encoder/Decoder and Convolutional Encoder/Decoder. Please contact Lattice to obtain the latest list of available DSP IP cores.

**Resources Available in the LatticeECP2/M Family**

Table 2-9 shows the maximum number of multipliers for each member of the LatticeECP2/M family. Table 2-10 shows the maximum available EBR RAM Blocks in each LatticeECP2/M 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 Blocks in the LatticeECP2/M Family**

Device	DSP Block	9x9 Multiplier	18x18 Multiplier	36x36 Multiplier
ECP2-6	3	24	12	3
ECP2-12	6	48	24	6
ECP2-20	7	56	28	7
ECP2-35	8	64	32	8
ECP2-50	18	144	72	18
ECP2-70	22	176	88	22
ECP2M20	6	48	24	6
ECP2M35	8	64	32	8
ECP2M50	22	176	88	22
ECP2M70	24	192	96	24
ECP2M100	42	336	168	42

**Table 2-10. Embedded SRAM in the LatticeECP2/M Family**

Device	EBR SRAM Block	Total EBR SRAM (Kbits)
ECP2-6	3	55
ECP2-12	12	221
ECP2-20	15	277
ECP2-35	18	332
ECP2-50	21	387
ECP2-70	60	1106
ECP2M20	66	1216
ECP2M35	114	2101
ECP2M50	225	4146
ECP2M70	246	4533
ECP2M100	288	5307

## LatticeECP2/M DSP Performance

Table 2-11 lists the maximum performance in millions of MAC operations per second (MMAC) for each member of the LatticeECP2/M family.

**Table 2-11. DSP Performance**

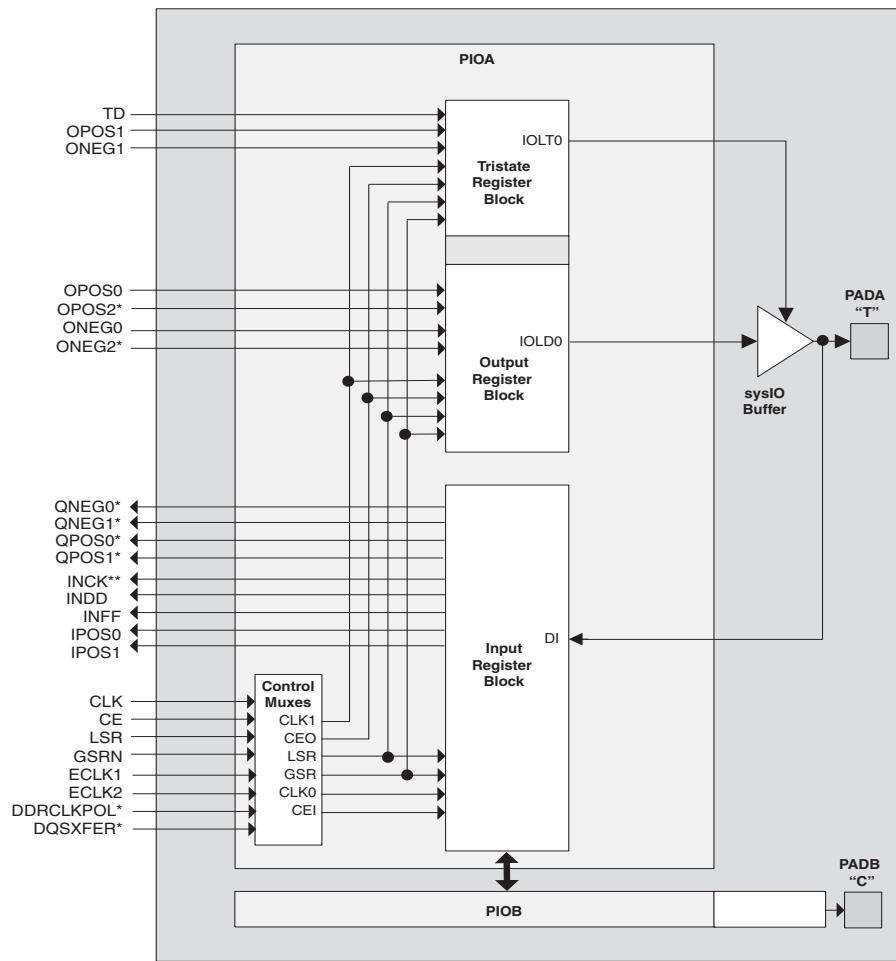
Device	DSP Block	DSP Performance GMAC
ECP2-6	3	3.9
ECP2-12	6	7.8
ECP2-20	7	9.1
ECP2-35	8	10.4
ECP2-50	18	23.4
ECP2-70	22	28.6
ECP2M20	6	7.8
ECP2M35	8	10.4
ECP2M50	22	28.6
ECP2M70	24	31.2
ECP2M100	42	54.6

For further information on the sysDSP block, please see details of additional technical information at the end of this data sheet.

## Programmable I/O Cells (PIC)

Each PIC contains two PIOs connected to their respective sysIO buffers as shown in Figure 2-27. The PIO Block supplies the output data (DO) and the tri-state control signal (TO) to the sysIO buffer and receives input from the buffer. Table 2-14 provides the PIO signal list.

Figure 2-27. PIC Diagram



Two adjacent PIOs can be joined to provide a differential I/O pair (labeled as “T” and “C”) as shown in Figure 2-27. 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 inputs.

**Table 2-12. PIO Signal List**

Name	Type	Description
CE0, CE1	Control from the core	Clock enables for input and output block flip-flops
CLK0, CLK1	Control from the core	System clocks for input and output blocks
ECLK1, ECLK2	Control from the core	Fast edge clocks
LSR	Control from the core	Local Set/Reset
GSRN	Control from routing	Global Set/Reset (active low)
INCK	Input to the core	Input to Primary Clock Network or PLL reference inputs
DQS	Input to PIO	DQS signal from logic (routing) to PIO
INDD	Input to the core	Unregistered data input to core
INFF	Input to the core	Registered input on positive edge of the clock (CLK0)
IPOS0, IPOS1	Input to the core	Double data rate registered inputs to the core
QPOS0 <sup>1</sup> , QPOS1 <sup>1</sup>	Input to the core	Gearbox pipelined inputs to the core
QNEG0 <sup>1</sup> , QNEG1 <sup>1</sup>	Input to the core	Gearbox pipelined inputs to the core
OPOS0, ONEG0, OPOS2, ONEG2	Output data from the core	Output signals from the core for SDR and DDR operation
OPOS1 ONEG1	Tristate control from the core	Signals to Tristate Register block for DDR operation
DEL[3:0]	Control from the core	Dynamic input delay control bits
TD	Tristate control from the core	Tristate signal from the core used in SDR operation
DDRCLKPOL	Control from clock polarity bus	Controls the polarity of the clock (CLK0) that feed the DDR input block
DQSXFER	Control from core	Controls signal to the Output block

1. Signals available on left/right/bottom only.

2. Selected I/O.

## 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 PIOs in left, right and bottom 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 2-28 shows the diagram of the input register block for left, right and bottom edges. The input register block for the top edge contains one memory element to register the input signal as shown in Figure 2-29. The following description applies to the input register block for PIOs in left, right and bottom edges of the device.

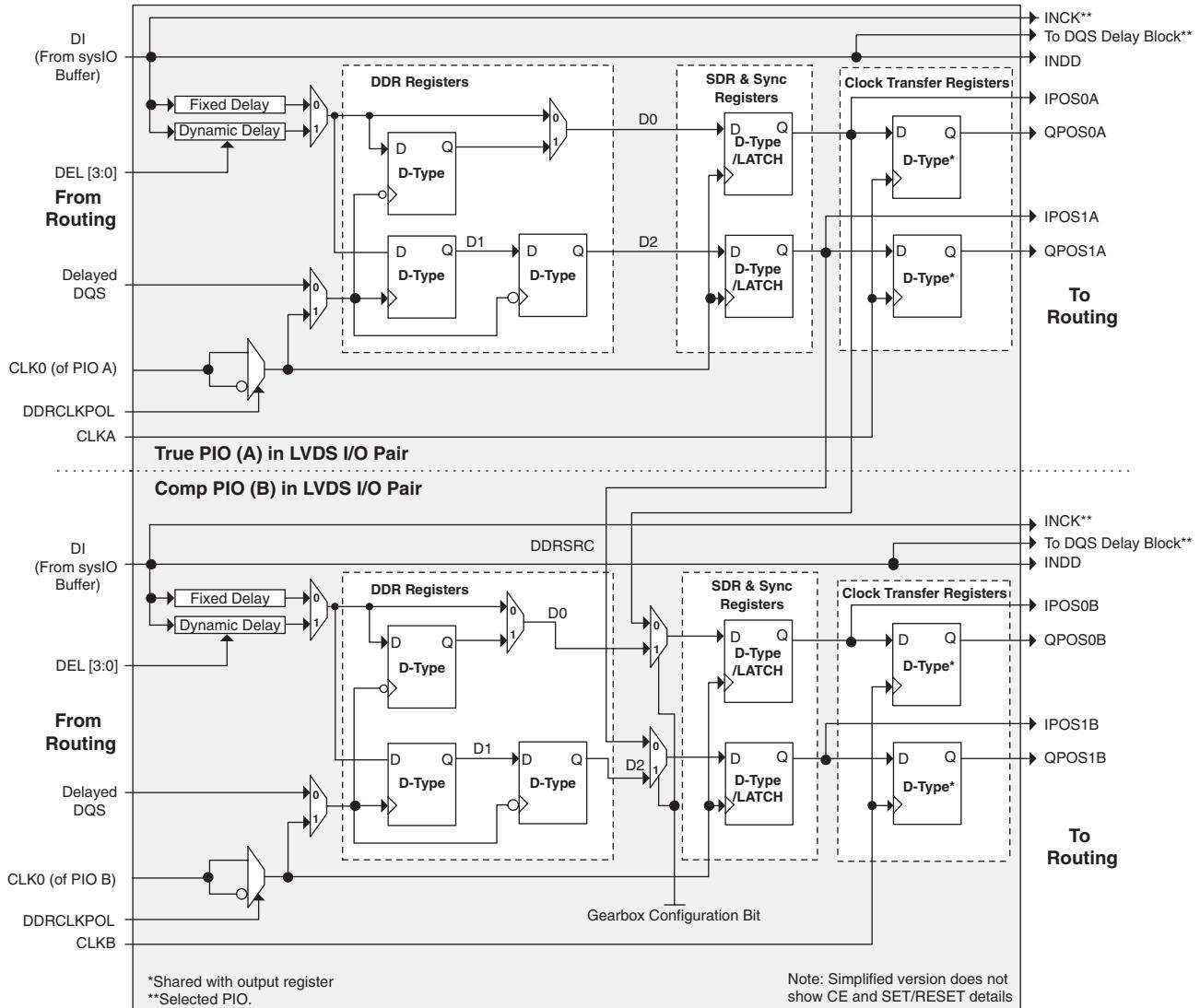
Input signals are fed from the sysIO buffer to the input register block (as signal DI). If desired, the input signal can bypass the register and delay elements and be used directly as a combinatorial signal (INDD), a clock (INCK) and, in selected blocks, the input to the DQS delay block. If an input 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.

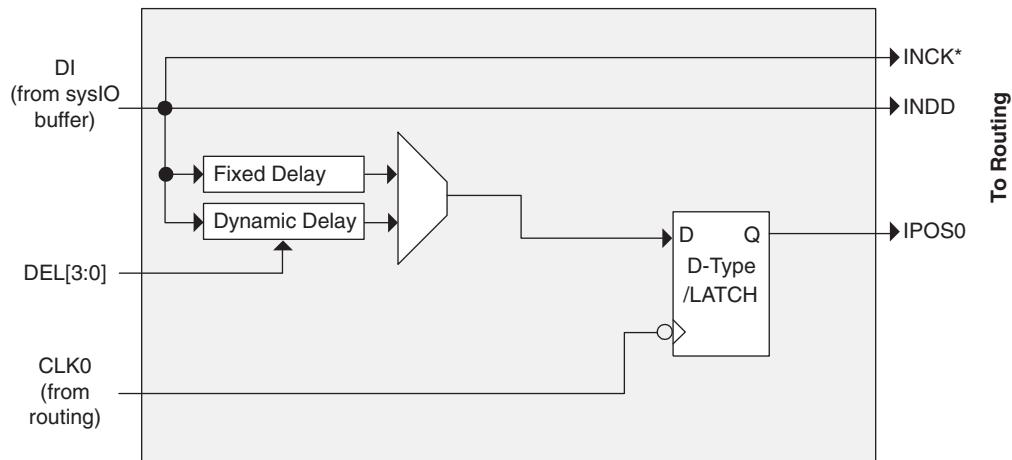
The input block allows three modes of operation. In the single data rate (SDR) the data is registered, by one of the registers in the single data rate sync register block, with the system clock. In DDR Mode, two registers are used to sample the data on the positive and negative edges of the DQS signal, creating two data streams, D0 and D1. These two data streams are synchronized with the system clock before entering the core. Further discussion on this topic is in the DDR Memory section of this data sheet.

By combining input blocks of the complementary PIOs and sharing some registers from output blocks, a gearbox function can be implemented, that takes a double data rate signal applied to PIOA and converts it as four data streams, IPOS0A, IPOS1A, IPOS0B and IPOS1B. Figure 2-28 shows the diagram using this gearbox function. For more information on this topic, please see information regarding additional documentation at the end of this data sheet.

The signal DDRCLKPOL controls the polarity of the clock used in the synchronization registers. It ensures adequate timing when data is transferred from the DQS to system clock domain. For further discussion on this topic, see the DDR Memory section of this data sheet.

**Figure 2-28. Input Register Block for Left, Right and Bottom Edges**



**Figure 2-29. Input Register Block Top Edge**

Note: Simplified version does not show CE and SET/RESET details.  
 \*On selected blocks.

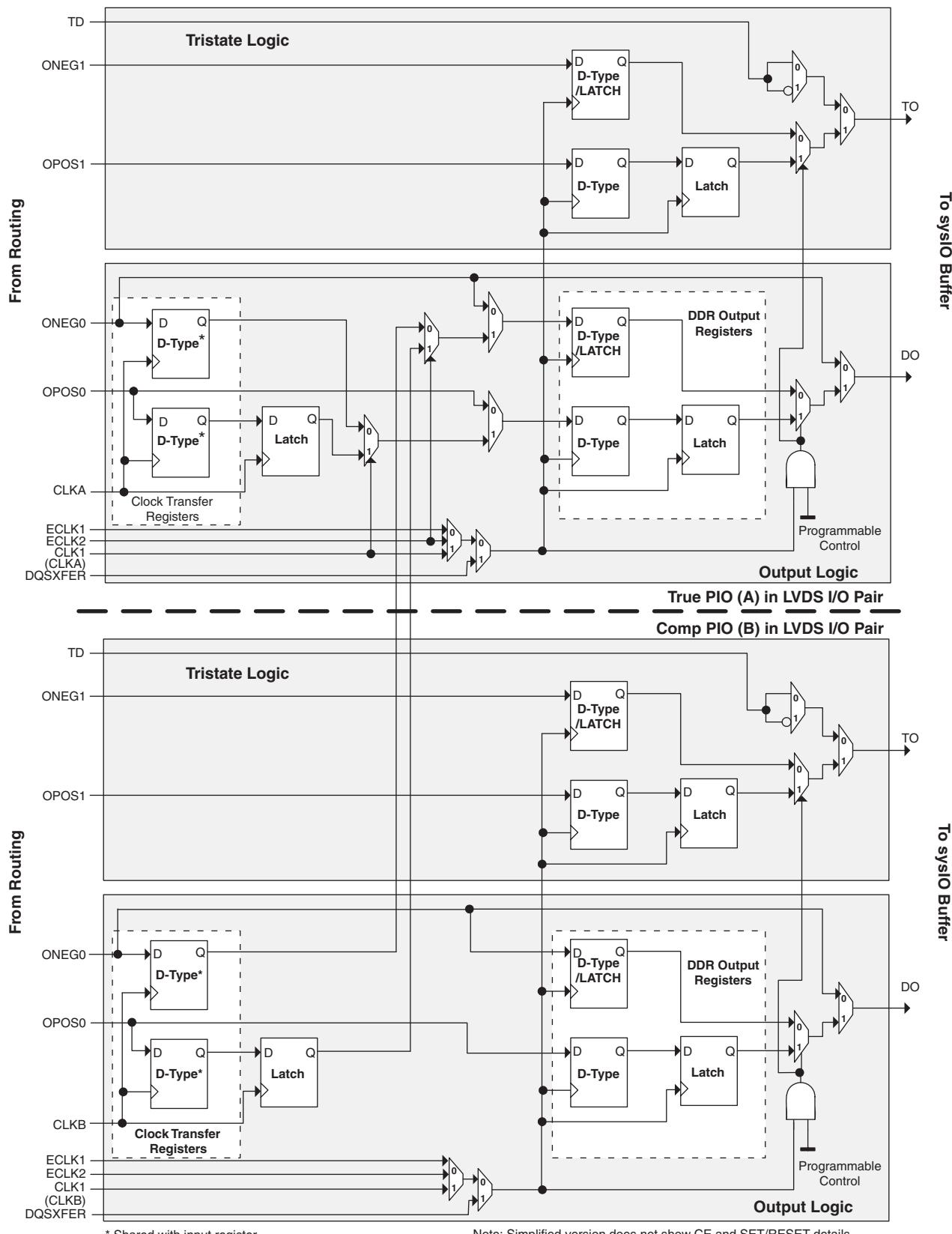
## Output Register Block

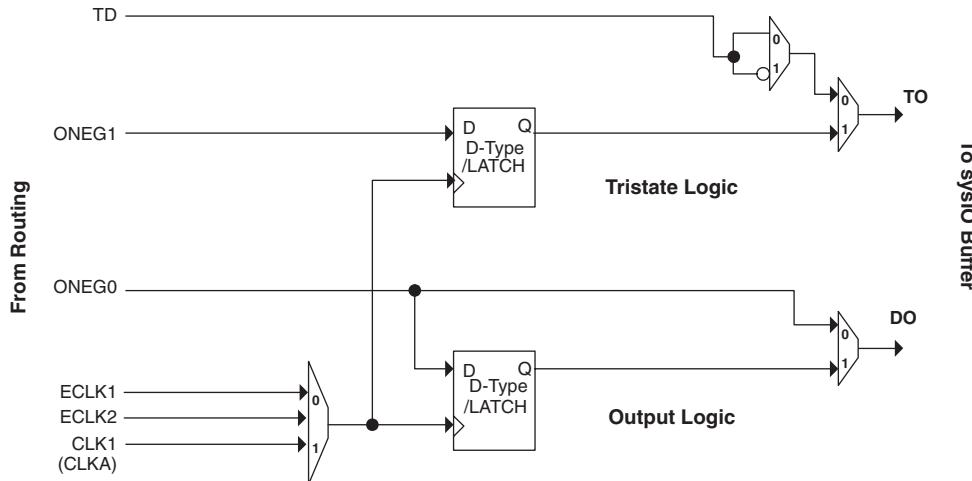
The output register block provides the ability to register signals from the core of the device before they are passed to the sysIO buffers. The blocks on the PIOs on the left, right and bottom contains a register for SDR operation that is combined with an additional latch for DDR operation. Figure 2-30 shows the diagram of the Output Register Block for PIOs on the left, right and the bottom edges. Figure 2-31 shows the diagram of the Output Register Block for PIOs on the top edge of the device.

In SDR mode, ONEG0 feeds one of the flip-flops that then feeds the output. The flip-flop can be configured as a D-type or latch. In DDR mode, ONEG0 and OPOS0 are fed into registers is fed into registers on the positive edge of the clock. Then at the next clock cycle this registered OPOS0 is latched. A multiplexer running off the same clock selects the correct register for feeding to the output (D0).

By combining output blocks of the complementary PIOs and sharing some registers from input blocks, a gearbox function can be implemented, that takes four data streams ONEG0A, ONEG1A, ONEG1B and ONEG1B. Figure 2-31 shows the diagram using this gearbox function. For more information on this topic, please see information regarding additional documentation at the end of this data sheet.

Figure 2-30. Output and Tristate Block for Left, Right and Bottom Edges



**Figure 2-31. Output and Tristate Block, Top Edge**

Note: Simplified version does not show CE and SET/RESET details.

### Tristate Register Block

The tristate register block provides the ability to register tri-state control signals from the core of the device before they are passed to the sysIO buffers. The block contains a register for SDR operation and an additional latch for DDR operation. Figure 2-30 shows the diagram of the Tristate Register Block with the Output Block for the left, right and bottom edges and Figure 2-31 shows the diagram of the Tristate Register Block with the Output Block for the top edge.

In SDR mode, ONEG1 feeds one of the flip-flops that then feeds the output. The flip-flop can be configured a D-type or latch. In DDR mode, ONEG1 and OPOS1 are fed into registers on the positive edge of the clock. Then in the next clock the registered OPOS1 is latched. A multiplexer running off the same clock cycle selects the correct register for feeding to the output (D0).

### Control Logic Block

The control logic block allows the selection and modification of control signals for use in the PIO block. A clock is selected from one of the clock signals provided from the general purpose routing, one of the edge clocks (ECLK1/ECLK2) and a DQS signal provided from the programmable DQS pin and provided to the input register block. The clock can optionally be inverted.

### DDR Memory Support

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

### Left and Right Edges

PICs on these edges have registered elements that support DDR memory interfaces. One of every 16 PIOs contains a delay element to facilitate the generation of DQS signals. The DQS signal feeds the DQS bus which spans the set of 16 PIOs. Figure 2-32 shows the assignment of DQS pins in each set of 16 PIOs.

### Bottom Edge

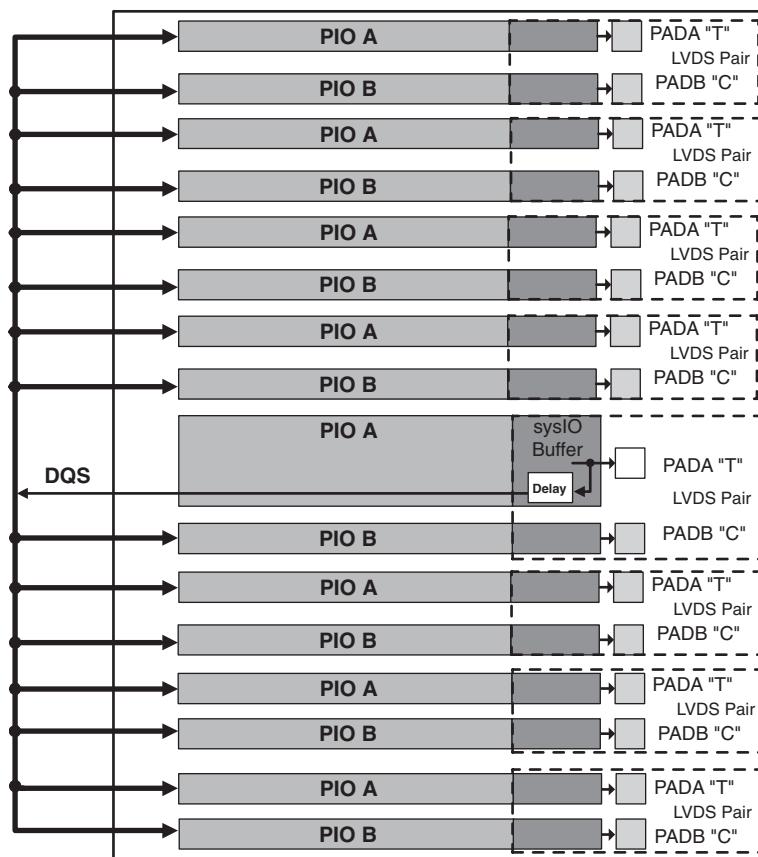
PICs on these edges have registered elements that support DDR memory interfaces. One of every 18 PIOs contains a delay element to facilitate the generation of DQS signals. The DQS signal feeds the DQS bus that spans the set of 18 PIOs. Figure 2-33 shows the assignment of DQS pins in each set of 18 PIOs.

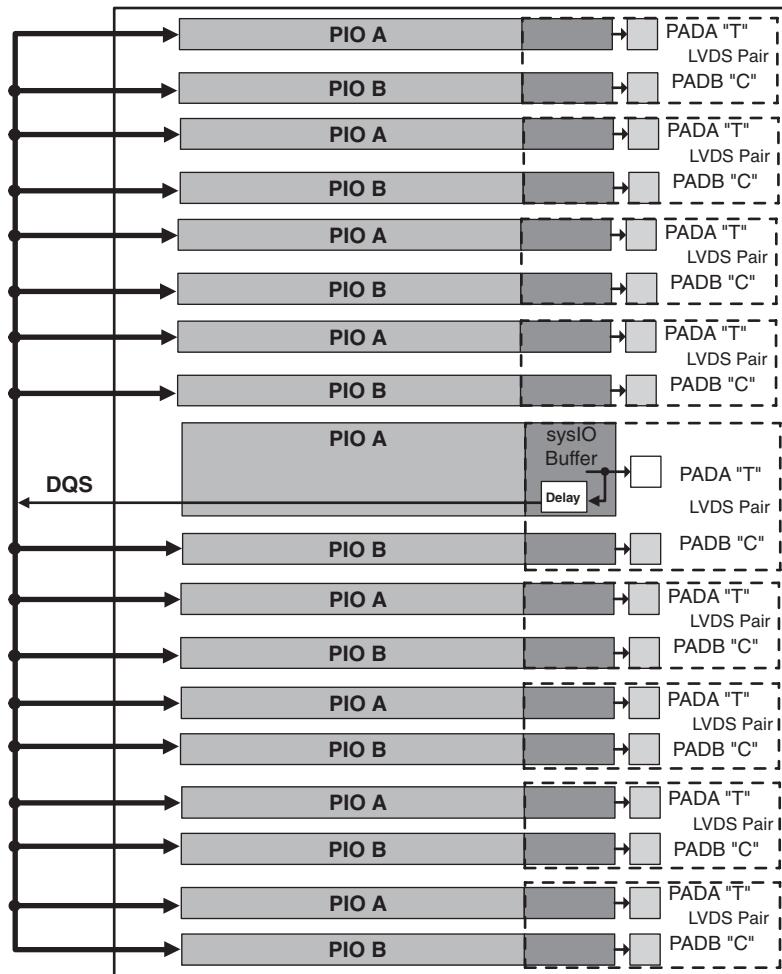
## Top Edge

The PICs on the top edge are different from PIOs on the left, right and bottom edges. PIOs on this edge do not have registers or DQS signals.

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 and right edges are designed for DDR memories that support 16 bits of data, whereas interfaces on the bottom are designed for memories that support 18 bits of data.

**Figure 2-32. DQS Input Routing for the Left and Right Edges of the Device**



**Figure 2-33. DQS Input Routing for the Bottom Edge of the Device**

### 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 DQS signal (selected PIOs only, as shown in Figure 2-34) feeds from the PAD through a DQS delay element to a dedicated DQS routing resource. The DQS signal also feeds polarity control logic which controls the polarity of the clock to the sync registers in the input register blocks. Figure 2-34 and Figure 2-35 show how the DQS transition signals are routed to the PIOs.

The temperature, voltage and process variations of the DQS delay block are compensated by a set of calibration (6-bit bus) signals from two dedicated DLLs (DDR\_DLL) on opposite sides of the device. Each DLL compensates DQS delays in its half of the device as shown in Figure 2-34. The DLL loop is compensated for temperature, voltage and process variations by the system clock and feedback loop.

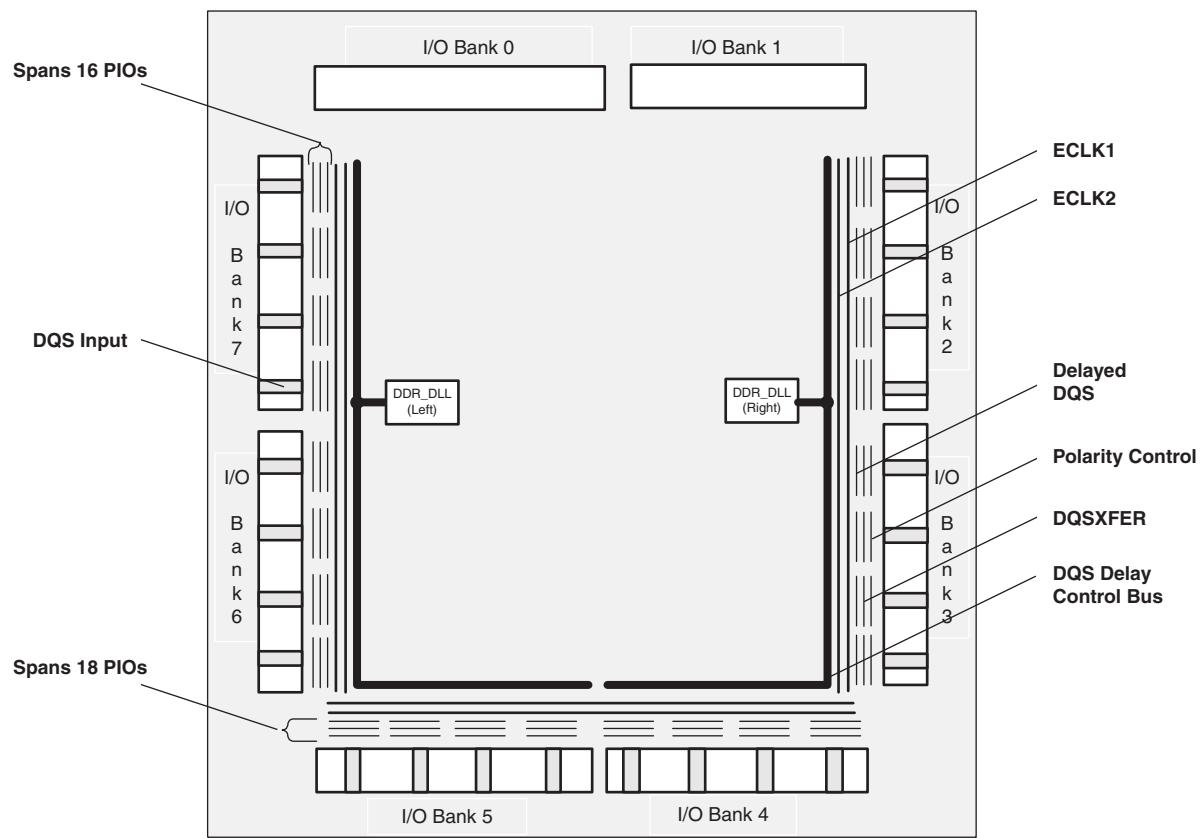
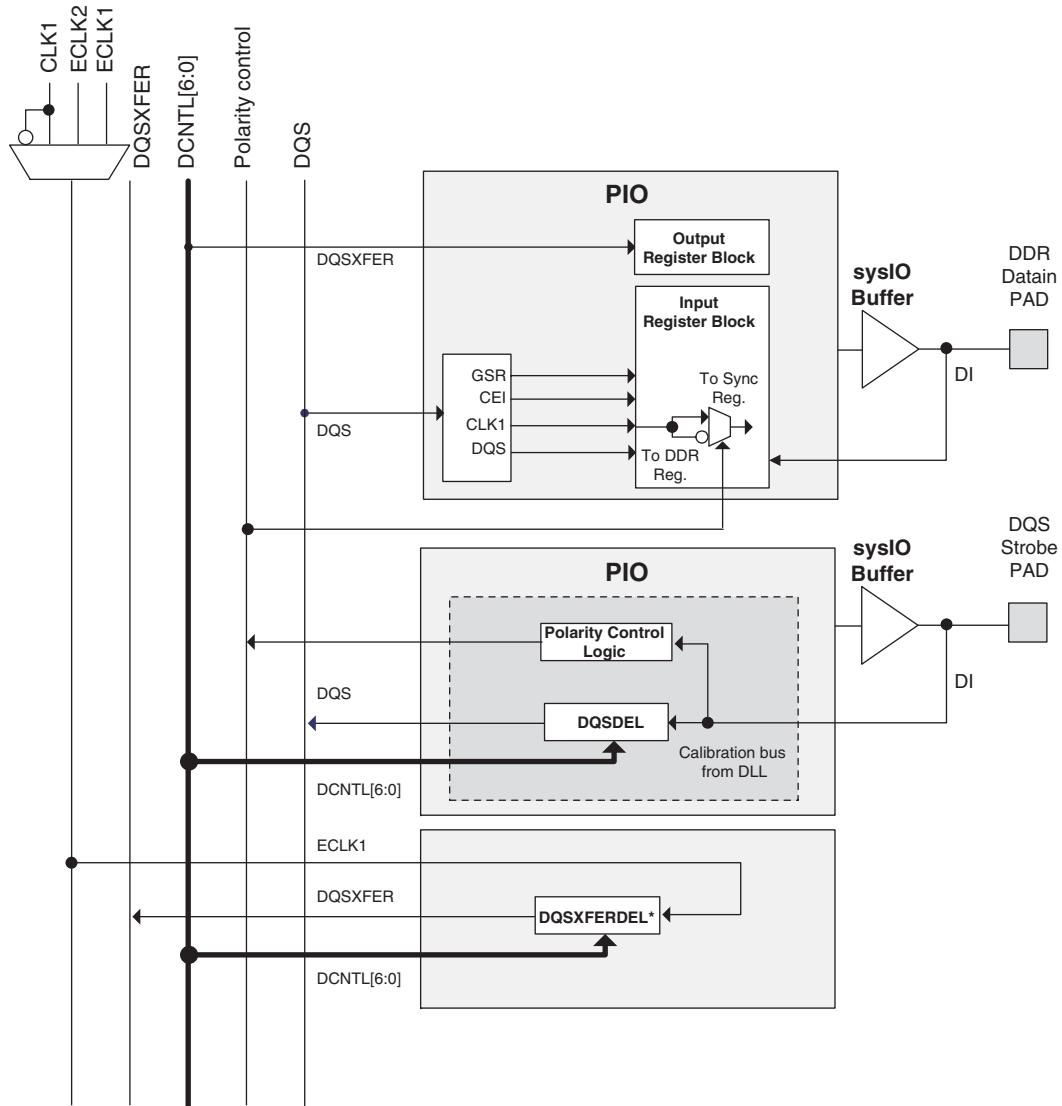
**Figure 2-34. Edge Clock, DLL Calibration and DQS Local Bus Distribution**

Figure 2-35. DQS Local Bus



\*DQSXFERDEL shifts ECLK1 by 90% and is not associated with a particular PIO.

## 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 LatticeECP2/M family contains dedicated circuits to transfer data between these domains. To prevent set-up and hold violations, at the domain transfer between DQS (delayed) and the system clock, a clock polarity selector is used. 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 this transition. This signal is used to control the polarity of the clock to the synchronizing registers.

## DQSXFER

LatticeECP2/M devices provide a DQSXFER signal to the output buffer to assist it in data transfer to DDR memories that require DQS strobe be shifted 90°. This shifted DQS strobe is generated by the DQSDEL block. The DQSXFER signal runs the span of the data bus.

## sysIO Buffer

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

### sysIO Buffer Banks

LatticeECP2/M devices have nine sysIO buffer banks: eight banks for user I/Os arranged two per side. The ninth sysIO buffer bank (Bank 8) is located adjacent to Bank 3 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 sysIO bank has its own I/O supply voltage ( $V_{CCIO}$ ). In addition, each bank, except Bank 8, has voltage references,  $V_{REF1}$  and  $V_{REF2}$ , that allow it to be completely independent from the others. Bank 8 shares two voltage references,  $V_{REF1}$  and  $V_{REF2}$ , with Bank 3. Figure 2-36 shows the nine banks and their associated supplies.

In LatticeECP2/M 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 2-36. LatticeECP2 Banks

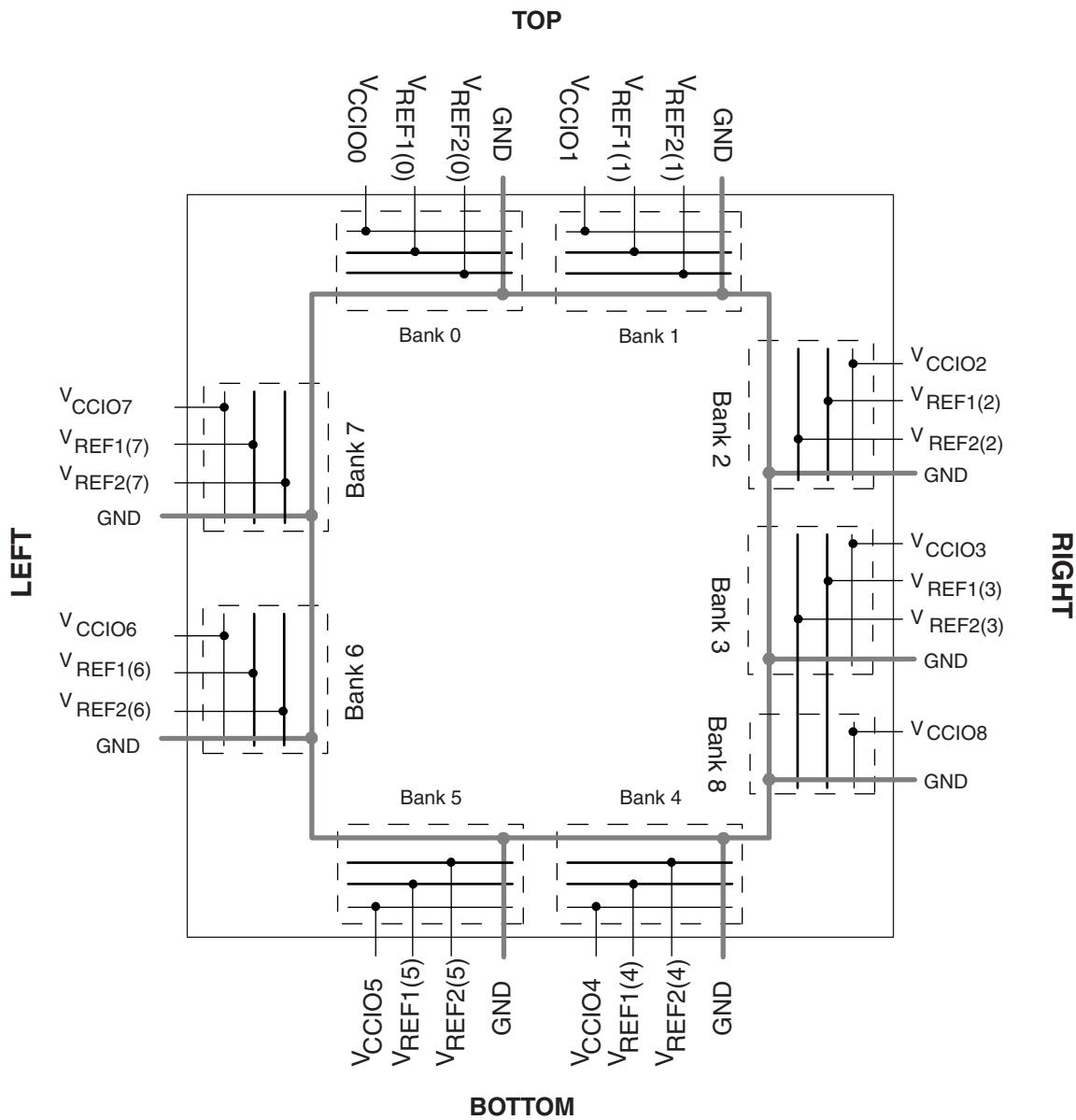
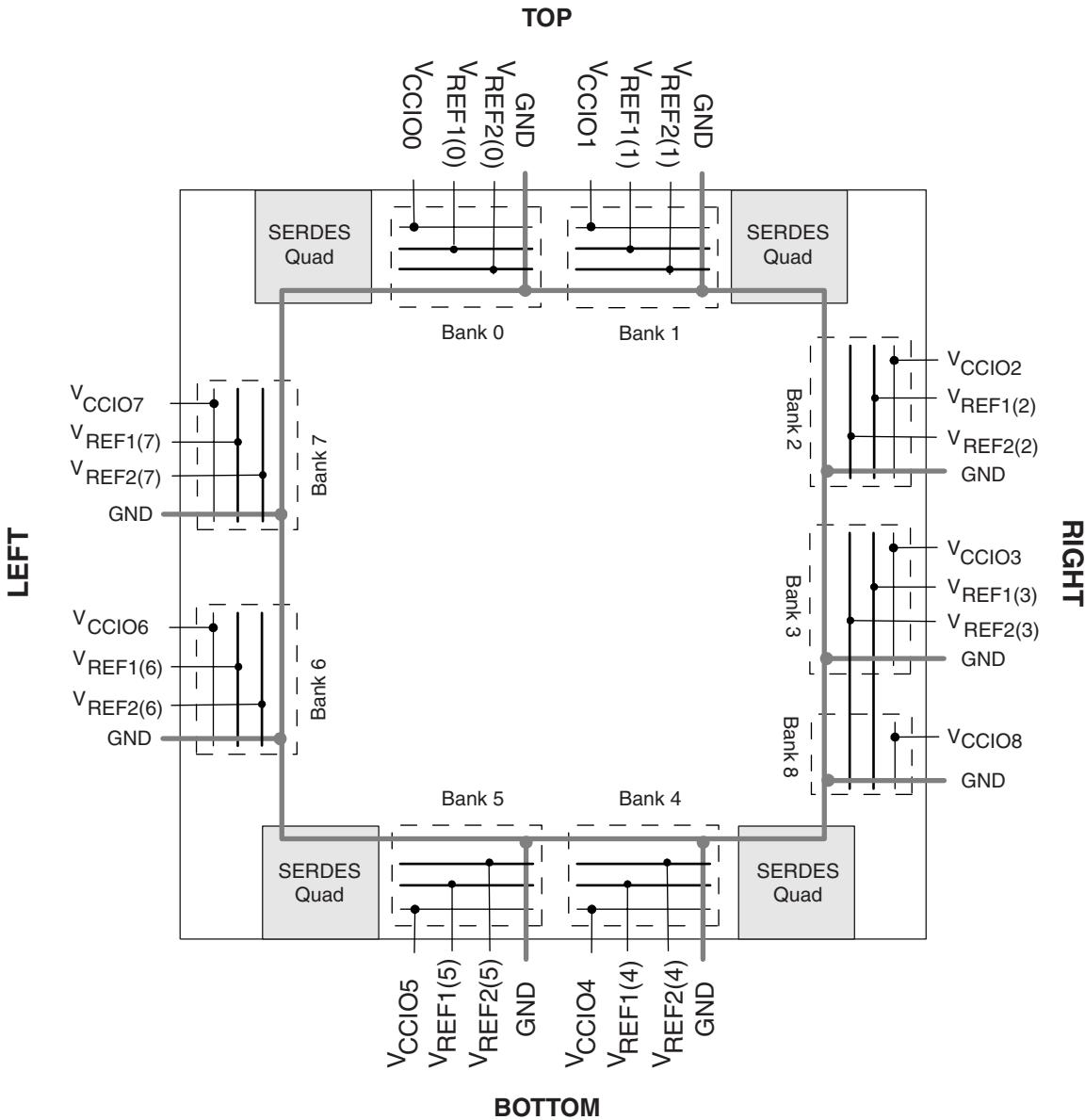


Figure 2-37. LatticeECP2M Banks



LatticeECP2/M devices contain two types of sysIO buffer pairs.

#### 1. Top (Bank 0 and Bank 1) sysIO Buffer Pairs (Single-Ended Outputs Only)

The sysIO 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.

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.

#### 2. Bottom (Bank 4 and Bank 5) sysIO Buffer Pairs (Single-Ended Outputs Only)

The sysIO buffer pairs in the bottom 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.

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.

3. **Left and Right (Banks 2, 3, 6 and 7) sysIO Buffer Pairs (50% Differential and 100% Single-Ended Outputs)**  
The sysIO 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.

LVDS differential output drivers are available on 50% of the buffer pairs on the left and right banks.

4. **Bank 8 sysIO Buffer Pairs (Single-Ended Outputs, Only on Shared Pins When Not Used by Configuration)**

The sysIO buffers in Bank 8 consist of single-ended output drivers and single-ended input buffers (both ratioed and referenced). The referenced input buffer can also be configured as a differential input.

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.

In LatticeECP2 devices, only the I/Os on the bottom banks have programmable PCI clamps. In LatticeECP2M devices, the I/Os on the left and bottom banks have programmable PCI clamps.

## Typical sysIO 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 on controlling the output logic state with valid input logic levels during power-up in LatticeECP2/M devices, see details of additional 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 sysIO Standards

The LatticeECP2/M sysIO buffer supports both single-ended and differential standards. Single-ended standards can be further subdivided into LVCMOS, LVTTL and other standards. The buffers support the LVTTL, LVCMOS 1.2V, 1.5V, 1.8V, 2.5V and 3.3V standards. In the LVCMOS and LVTTL modes, the buffer has individual configuration options for drive strength, 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, MLVDS, BLVDS, LVPECL, RSDS, differential SSTL and differential HSTL. Tables 2-13 and 2-14 show the I/O standards (together with their supply and reference voltages) supported by LatticeECP2/M devices. For further information on utilizing the sysIO buffer to support a variety of standards please see the details of additional technical information at the end of this data sheet.

**Table 2-13. Supported Input Standards**

Input Standard	$V_{REF}$ (Nom.)	$V_{CCIO}^1$ (Nom.)
<b>Single Ended Interfaces</b>		
LV TTL	—	—
LVC MOS33	—	—
LVC MOS25	—	—
LVC MOS18	—	1.8
LVC MOS15	—	1.5
LVC MOS12	—	—
PCI 33	—	3.3
HSTL18 Class I, II	0.9	—
HSTL15 Class I	0.75	—
SSTL3 Class I, II	1.5	—
SSTL2 Class I, II	1.25	—
SSTL18 Class I, II	0.9	—
<b>Differential Interfaces</b>		
Differential SSTL18 Class I, II	—	—
Differential SSTL2 Class I, II	—	—
Differential SSTL3 Class I, II	—	—
Differential HSTL15 Class I	—	—
Differential HSTL18 Class I, II	—	—
LVDS, MLVDS, LVPECL, BLVDS, RS DS	—	—

1 When not specified,  $V_{CCIO}$  can be set anywhere in the valid operating range (page 3-1).

**Table 2-14. Supported Output Standards**

Output Standard	Drive	V <sub>CCIO</sub> (Nom.)
<b>Single-ended Interfaces</b>		
LVTTL	4mA, 8mA, 12mA, 16mA, 20mA	3.3
LVCMOS33	4mA, 8mA, 12mA 16mA, 20mA	3.3
LVCMOS25	4mA, 8mA, 12mA, 16mA, 20mA	2.5
LVCMOS18	4mA, 8mA, 12mA, 16mA	1.8
LVCMOS15	4mA, 8mA	1.5
LVCMOS12	2mA, 6mA	1.2
LVCMOS33, Open Drain	4mA, 8mA, 12mA 16mA, 20mA	—
LVCMOS25, Open Drain	4mA, 8mA, 12mA 16mA, 20mA	—
LVCMOS18, Open Drain	4mA, 8mA, 12mA 16mA	—
LVCMOS15, Open Drain	4mA, 8mA	—
LVCMOS12, Open Drain	2mA, 6mA	—
PCI33	N/A	3.3
HSTL18 Class I, II	N/A	1.8
HSTL15 Class I	N/A	1.5
SSTL3 Class I, II	N/A	3.3
SSTL2 Class I, II	N/A	2.5
SSTL18 Class I, II	N/A	1.8
<b>Differential Interfaces</b>		
Differential SSTL3, Class I, II	N/A	3.3
Differential SSTL2, Class I, II	N/A	2.5
Differential SSTL18, Class I, II	N/A	1.8
Differential HSTL18, Class I, II	N/A	1.8
Differential HSTL15, Class I	N/A	1.5
LVDS	N/A	2.5
MLVDS <sup>1</sup>	N/A	2.5
BLVDS <sup>1</sup>	N/A	2.5
LVPECL <sup>1</sup>	N/A	3.3
RSDS <sup>1</sup>	N/A	2.5

1. Emulated with external resistors. For more detail, please see information regarding additional technical documentation at the end of this data sheet.

## Hot Socketing

LatticeECP2/M devices have been carefully designed to ensure predictable behavior during power-up and power-down. Power supplies can be sequenced in any order. 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 to within specified limits. This allows for easy integration with the rest of the system. These capabilities make the LatticeECP2/M ideal for many multiple power supply and hot-swap applications.

## Power-up During Configuration

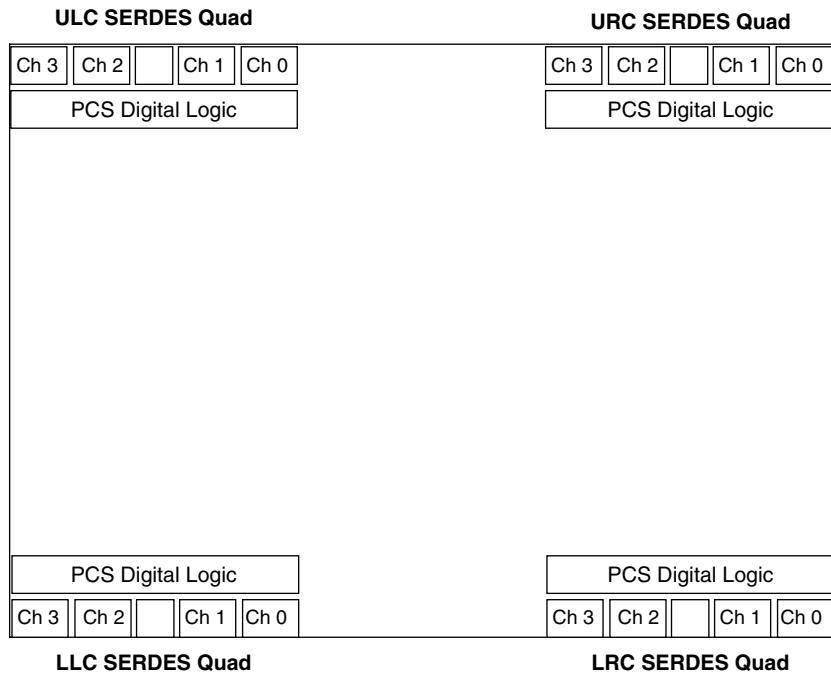
For certain configuration modes there is a power-sequencing requirement. Please see note 3 in the Recommended Operating Conditions table.

## SERDES and PCS (Physical Coding Sublayer)

LatticeECP2M devices feature up to 16 channels of embedded SERDES arranged in quads at the corners of the devices. Figure 2-38 shows the position of the quad blocks in relation to the PFU array for LatticeECP2M70 and LatticeECP2M100 devices. Table 2-15 shows the location of Quads for all the devices.

Each quad contains, four dedicated SERDES (Ch0 to Ch3) for high-speed, full-duplex serial data transfer. Each quad also has PCS block that interfaces to the SERDES channels and contain digital logic to support an array of popular data protocols. PCS also contain logic to interface to FPGA core.

**Figure 2-38. SERDES Quads (LatticeECP2M70/LatticeECP2M100)**



**Table 2-15. Available SERDES Quads per LatticeECP2M Devices**

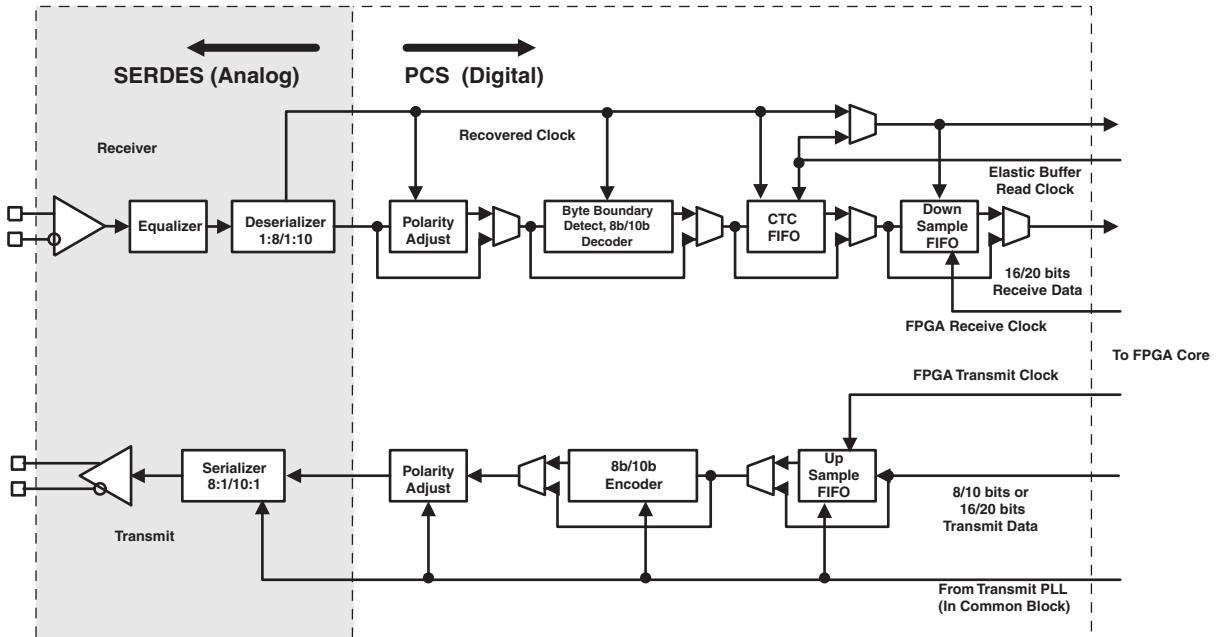
Device	URC Quad	ULC Quad	LRC Quad	LLC Quad
ECP2M20	Available	—	—	—
ECP2M35	Available	—	—	—
ECP2M50	Available	—	Available	—
ECP2M70	Available	Available	Available	Available
ECP2M100	Available	Available	Available	Available

### SERDES Block

A differential receiver receives the serial encoded data stream, equalizes the signal, extracts the buried clock and de-serializes the data-stream before passing the 8- or 10-bit data to the PCS logic. The transmit channel receives the parallel (8- or 10-bit) encoded data, serializes the data and transmits the serial bit stream through the differential buffers. There is a single transmit clock per quad. Figure 2-39 shows a single channel SERDES and its interface to the PCS logic. Each SERDES receiver channel provides a recovered clock to the PCS block and to the FPGA core logic.

Each Transmit and Receive channel has its independent power supplies. The Output and Input buffers of each channel have their own independent power supplies too. In addition, there are separate power supplies for PLL, terminating resistor per quad.

**Figure 2-39. Simplified Channel Block Diagram for SERDES and PCS**



## PCS

As shown in Figure 2-39, the PCS receives the parallel digital data from the deserializer receivers and adjusts the polarity, detects, byte boundary, decodes (8b/10b) and provides Clock Tolerance Compensation (CTC) FIFO for changing the clock domain from receiver clock to the FPGA Clock.

For the transmit channel, the PCS block receives the parallel data from the FPGA core, encodes it with 8b/10b, adjusts 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 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 a soft IP interface that allows the SERDES/PCS Quad block to be controlled by registers as opposed to the configuration memory cells. It is a simple register configuration interface.

The ispLEVER design tools from Lattice 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 ispLEVER, the user can define the mode for each quad in a design.

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

For further information on SERDES, please see details of additional technical documentation at the end of this data sheet.

## IEEE 1149.1-Compliant Boundary Scan Testability

All LatticeECP2/M 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 details on boundary scan test, please see information regarding additional technical documentation at the end of this data sheet.

## Device Configuration

All LatticeECP2/M 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 both byte-wide 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 is a 20-pin interface with six I/Os used as dedicated pins with the remainder used as dual-use pins. See Lattice technical note number TN1108, *LatticeECP2 sysCONFIG Usage Guide* for more information on using the dual-use pins as general purpose I/Os.

There are five ways to configure a LatticeECP2/M device:

1. Industry standard SPI serial memories
2. Industry standard byte wide flash with an ispMACH™ 4000, providing control and addressing
3. System microprocessor to drive a sysCONFIG port or JTAG TAP
4. Industry standard FPGA boot PROM memory
5. JTAG

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.

## Enhanced Configuration Option

LatticeECP2/M devices have enhanced configuration features such as: decryption support, TransFR™ I/O and dual boot image support.

### 1. Decryption Support

LatticeECP2/M devices provide on-chip, One Time Programmable (OTP) non-volatile key storage to support decryption of a 128-bit AES encrypted bitstream, securing designs and deterring design piracy.

### 2. 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 Lattice technical note number TN1087, *Minimizing System Interruption During Configuration Using TransFR Technology*, for details.

### 3. 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 LatticeECP2/M can be re-booted from this new configuration file. If there is a problem such as corrupt data dur-

ing download or incorrect version number with this new boot image, the LatticeECP2/M device can revert back to the original backup configuration and try again. This all can be done without power cycling the system.

For more information on device configuration, please see details of additional technical documentation at the end of this data sheet.

### Software Error Detect (SED) Support

LatticeECP2/M devices have dedicated logic to perform CRC checks. During configuration, the configuration data bitstream can be checked with CRC logic block. In addition the LatticeECP2 device can also be programmed for checking soft errors (SED) in SRAM. This 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 either reload from a known good boot image or generate an error signal.

For further information on Soft Error Detect (SED) support, please see details of additional technical documentation at the end of this data sheet.

### External Resistor

LatticeECP2/M 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 LatticeECP2/M device has an internal CMOS oscillator which is used to derive a Master Clock for configuration. The oscillator and the Master Clock run continuously and are available to user logic after configuration is completed. The software default value of the Master Clock is 2.5MHz. Table 2-16 lists all the available Master Clock frequencies. When a different Master Clock is selected during the design process, the following sequence takes place:

1. Device powers up with a 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 Master Clock frequency of 2.5MHz.

This internal 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 details of additional technical documentation at the end of this data sheet.

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

CCLK (MHz)	CCLK (MHz)	CCLK (MHz)
2.5 <sup>1</sup>	—	45
—	15	51
5.4	20	55
—	26	60
—	30	—
—	34	—
10.0	41	—

1. Software default frequency.

## Density Shifting

The LatticeECP2/M 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 likely success in each case. Design migration between LatticeECP2 and LatticeECP2M families is not possible.

September 2006

Advance Data Sheet DS1007

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

Supply Voltage V <sub>CC</sub> .....	-0.5 to 1.32V
Supply Voltage V <sub>CCAUX</sub> .....	-0.5 to 3.75V
Supply Voltage V <sub>CCJ</sub> .....	-0.5 to 3.75V
Output Supply Voltage V <sub>CCIO</sub> .....	-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 <sub>j</sub> ) .....	+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<sub>IHMAX</sub> + 2) volts is permitted for a duration of <20ns.

### Recommended Operating Conditions

Symbol	Parameter	Min.	Max.	Units
V <sub>CC</sub>	Core Supply Voltage	1.14	1.26	V
V <sub>CCAUX</sub>	Auxiliary Supply Voltage	3.135	3.465	V
V <sub>CCPLL</sub>	PLL Supply Voltage	1.14	1.26	V
V <sub>CCIO</sub> <sup>1, 2</sup>	I/O Driver Supply Voltage	1.14	3.465	V
V <sub>CCJ</sub> <sup>1</sup>	Supply Voltage for IEEE 1149.1 Test Access Port	1.14	3.465	V
t <sub>JCOM</sub>	Junction Temperature, Commercial Operation	0	85	°C
t <sub>JIND</sub>	Junction Temperature, Industrial Operation	-40	100	°C

#### SERDES External Power Supply (For LatticeECP2M Family Only)

V <sub>CCIB</sub>	Input Buffer Power Supply (1.2V)	1.14	1.26	V
	Input Buffer Power Supply (1.5V)	1.425	1.575	V
V <sub>CCOB</sub>	Output Buffer Power Supply (1.2V)	1.14	1.26	V
	Output Buffer Power Supply (1.5V)	1.425	1.575	V
V <sub>CCAUX33</sub>	Termination Resistor Switching Power Supply	3.135	3.465	V
V <sub>CCRX</sub>	Receive Power Supply	1.14	1.26	V
V <sub>CCTX</sub>	Transmit Power Supply	1.14	1.26	V
V <sub>CCP</sub>	PLL and Reference Clock Buffer Power	1.14	1.26	V

1. If V<sub>CCIO</sub> or V<sub>CCJ</sub> is set to 1.2V, they must be connected to the same power supply as V<sub>CC</sub>. If V<sub>CCIO</sub> or V<sub>CCJ</sub> is set to 3.3V, they must be connected to the same power supply as V<sub>CCAUX</sub>.

2. See recommended voltages by I/O standard in subsequent table.

## Hot Socketing Specifications<sup>1, 2, 3, 4</sup>

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
$I_{DK}$	Input or I/O Leakage Current	$0 \leq V_{IN} \leq V_{IH}$ (MAX.)	—	—	+/-1000	$\mu A$

1.  $V_{CC}, V_{CCAUX}$  and  $V_{CCIO}$  should rise/fall monotonically.
2.  $0 \leq V_{CC} \leq V_{CC}$  (MAX),  $0 \leq V_{CCIO} \leq V_{CCIO}$  (MAX) or  $0 \leq V_{CCAUX} \leq V_{CCAUX}$  (MAX).
3.  $I_{DK}$  is additive to  $I_{PU}$ ,  $I_{PW}$  or  $I_{BH}$ .
4. LVC MOS and LV TTL only.

## DC Electrical Characteristics

### Over Recommended Operating Conditions

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
$I_{IL}, I_{IH}^1$	Input or I/O Low Leakage	$0 \leq V_{IN} \leq V_{IH}$ (MAX)	—	—	10	$\mu A$
$I_{PU}$	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) $\leq V_{IN} \leq V_{IH}$ (MAX)	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	$0 \leq V_{IN} \leq V_{IH}$ (MAX)	—	—	210	$\mu A$
$I_{BHHO}$	Bus Hold High Overdrive Current	$0 \leq V_{IN} \leq V_{IH}$ (MAX)	—	—	-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>	$V_{CCIO} = 3.3V, 2.5V, 1.8V, 1.5V, 1.2V$ , $V_{CC} = 1.2V$ , $V_{IO} = 0$ to $V_{IH}$ (MAX)	—	8	—	pf
C2	Dedicated Input Capacitance <sup>2</sup>	$V_{CCIO} = 3.3V, 2.5V, 1.8V, 1.5V, 1.2V$ , $V_{CC} = 1.2V$ , $V_{IO} = 0$ to $V_{IH}$ (MAX)	—	6	—	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,  $f = 1.0\text{MHz}$ .

**LatticeECP2 Supply Current (Standby)<sup>1, 2, 3, 4</sup>****Over Recommended Operating Conditions**

Symbol	Parameter	Device	Typ. <sup>5</sup>	Max.	Units
$I_{CC}$	Core Power Supply Current	ECP2-6			mA
		ECP2-12			mA
		ECP2-20			mA
		ECP2-35			mA
		ECP2-50	50		mA
		ECP2-70			mA
$I_{CCAUX}$	Auxiliary Power Supply Current	ECP2-6			mA
		ECP2-12			mA
		ECP2-20			mA
		ECP2-35			mA
		ECP2-50	20		mA
		ECP2-70			mA
$I_{CCGPLL}$	GPLL Power Supply Current (per GPLL)		1.0		mA
$I_{CCSPLL}$	SPLL Power Supply Current (per SPLL)		1.0		mA
$I_{CCIO}$	Bank Power Supply Current <sup>6</sup>		10		mA
$I_{CCJ}$	$V_{CCJ}$ Power Supply Current		1.5		mA

1. For further information on supply current, please see details of additional 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  $V_{CCIO}$  or GND.

3. Frequency 0MHz.

4. Pattern represents a "blank" configuration data file.

5.  $T_J = 25^\circ\text{C}$ , power supplies at nominal voltage.

6. Per bank.

**LatticeECP2M Supply Current (Standby)<sup>1, 2, 3, 4</sup>****Over Recommended Operating Conditions**

Symbol	Parameter	Device	Typ. <sup>5</sup>	Max.	Units
$I_{CC}$	Core Power Supply Current	ECP2M20			mA
		ECP2M35			mA
		ECP2M50			mA
		ECP2M70			mA
		ECP2M100			mA
$I_{CCAUX}$	Auxiliary Power Supply Current	ECP2M20			mA
		ECP2M35			mA
		ECP2M50			mA
		ECP2M70			mA
		ECP2M100			mA
$I_{CCGPLL}$	GPLL Power Supply Current (per GPLL)				mA
$I_{CCSPLL}$	SPLL Power Supply Current (per SPLL)				mA
$I_{CCIO}$	Bank Power Supply Current <sup>6</sup>				mA
$I_{CCJ}$	$V_{CCJ}$ Power Supply Current				mA

1. For further information on supply current, please see details of additional 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  $V_{CCIO}$  or GND.

3. Frequency 0MHz.

4. Pattern represents a "blank" configuration data file.

5.  $T_J = 25^\circ\text{C}$ , power supplies at nominal voltage.

6. Per bank.

**LatticeECP2 Initialization Supply Current<sup>1, 2, 3, 4, 5, 6</sup>**

Over Recommended Operating Conditions

Symbol	Parameter	Device	Typ. <sup>5</sup>	Max.	Units
$I_{CC}$	Core Power Supply Current	ECP2-6			mA
		ECP2-12			mA
		ECP2-20			mA
		ECP2-35			mA
		ECP2-50	153		mA
		ECP2-70			mA
$I_{CCAUX}$	Auxiliary Power Supply Current	ECP2-6			mA
		ECP2-12			mA
		ECP2-20			mA
		ECP2-35			mA
		ECP2-50	28		mA
		ECP2-70			mA
$I_{CCGPLL}$	GPLL Power Supply Current (per GPLL)		1.0		mA
$I_{CCSPLL}$	SPLL Power Supply Current (per SPLL)		1.0		mA
$I_{CCIO}$	Bank Power Supply Current <sup>7</sup>		2.0		mA
$I_{CCJ}$	VCCJ Power Supply Current <sup>8</sup>		2.0		mA

1. Until DONE signal is active.
2. For further information on supply current, please see details of additional technical documentation at the end of this data sheet.
3. Assumes all outputs are tristated, all inputs are configured as LVCMOS and held at the  $V_{CCIO}$  or GND.
4. Frequency 0MHz.
5.  $T_J = 25^\circ\text{C}$ , power supplies at nominal voltage.
6. A specific configuration pattern is used that scales with the size of the device; consists of 75% PFU utilization, 50% EBR, and 25% I/O configuration.
7. Per bank, at  $V_{CCO} = 2.5\text{V}$ .
8.  $V_{CCJ} = 2.5\text{V}$ .

**LatticeECP2M Initialization Supply Current<sup>1, 2, 3, 4, 5, 6</sup>**

Over Recommended Operating Conditions

Symbol	Parameter	Device	Typ. <sup>5</sup>	Max.	Units
$I_{CC}$	Core Power Supply Current	ECP2M20			mA
		ECP2M35			mA
		ECP2M50			mA
		ECP2M70			mA
		ECP2M100			mA
$I_{CCAUX}$	Auxiliary Power Supply Current	ECP2M20			mA
		ECP2M35			mA
		ECP2M50			mA
		ECP2M70			mA
		ECP2M100			mA
$I_{CCGPLL}$	GPLL Power Supply Current (per GPLL)				mA
$I_{CCSPLL}$	SPLL Power Supply Current (per SPLL)				mA
$I_{CCIO}$	Bank Power Supply Current <sup>7</sup>				mA
$I_{CCJ}$	VCCJ Power Supply Current <sup>8</sup>				mA

1. Until DONE signal is active.
2. For further information on supply current, please see details of additional technical documentation at the end of this data sheet.
3. Assumes all outputs are tristated, all inputs are configured as LVCMS and held at the  $V_{CCIO}$  or GND.
4. Frequency 0MHz.
5.  $T_J = 25^\circ\text{C}$ , power supplies at nominal voltage.
6. A specific configuration pattern is used that scales with the size of the device; consists of 75% PFU utilization, 50% EBR, and 25% I/O configuration.
7. Per bank, at  $V_{CCIO} = 2.5\text{V}$ .
8.  $V_{CCJ} = 2.5\text{V}$ .

**SERDES Power Supply Requirements (LatticeECP2M Family Only)<sup>1</sup>**

Over Recommended Operating Conditions

Symbol	Description	Typ.	Max.	Units
<b>Standby (Power Down)</b>				
I <sub>CCTX-SB</sub>	V <sub>CCTX</sub> current (per channel)	10		µA
I <sub>CCRX-SB</sub>	V <sub>CCRX</sub> current (per channel)	75		µA
I <sub>CCIB-SB</sub>	Input buffer current (per channel)	0		µA
I <sub>CCOB-SB</sub>	Output buffer current (per channel)	0		µA
I <sub>CCPLLS-SB</sub>	SERDES PLL current (per quad)	30		µA
I <sub>CCAX33-SB</sub>	SERDES termination current (per quad)	10		µA
<b>Operating (Data Rate = 3.125 Gbps)</b>				
I <sub>CCTX-OP</sub>	V <sub>CCTX</sub> current (per channel)	20		mA
I <sub>CCRX-OP</sub>	V <sub>CCRX</sub> current (per channel)	35		mA
I <sub>CCIB-OP</sub>	Input buffer current (per channel)	5		mA
I <sub>CCOB-OP</sub>	Output buffer current (per channel)	15		mA
I <sub>CCPLLS-OP</sub>	SERDES PLL current (per quad)	25		mA
I <sub>CCAX33-OP</sub>	SERDES termination current (per quad)	0.01		mA

1. Equalization enabled, pre-emphasis disabled.

**SERDES Power (LatticeECP2M Family Only)**

Table 3-1 presents the SERDES power for one channel.

**Table 3-1. SERDES Power**

Symbol	Description	Typ.	Max.	Units
P <sub>S-1CH-31</sub>	SERDES power (one channel @ 3.125 Gbps)	90		mW
P <sub>S-1CH-25</sub>	SERDES power (one channel @ 2.5 Gbps)	90		mW
P <sub>S-1CH-12</sub>	SERDES power (one channel @ 1.25 Gbps)	85		mW
P <sub>S-1CH-02</sub>	SERDES power (one channel @ 250 Mbps)	75		mW

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

2. Typ. values measured at 25°C and 1.2V.

**sysIO Recommended Operating Conditions**

Standard	V <sub>CCIO</sub>			V <sub>REF</sub> (V)		
	Min.	Typ.	Max.	Min.	Typ.	Max.
LVC MOS 3.3 <sup>2</sup>	3.135	3.3	3.465	—	—	—
LVC MOS 2.5 <sup>2</sup>	2.375	2.5	2.625	—	—	—
LVC MOS 1.8	1.71	1.8	1.89	—	—	—
LVC MOS 1.5	1.425	1.5	1.575	—	—	—
LVC MOS 1.2 <sup>2</sup>	1.14	1.2	1.26	—	—	—
LV TTL <sup>2</sup>	3.135	3.3	3.465	—	—	—
PCI	3.135	3.3	3.465	—	—	—
SSTL18 <sup>2</sup> Class I, II	1.71	1.8	1.89	0.833	0.9	0.969
SSTL2 <sup>2</sup> Class I, II	2.375	2.5	2.625	1.15	1.25	1.35
SSTL3 <sup>2</sup> Class I, II	3.135	3.3	3.465	1.3	1.5	1.7
HSTL <sup>2</sup> Class I	1.425	1.5	1.575	0.68	0.75	0.9
HSTL <sup>2</sup> 18 Class I, II	1.71	1.8	1.89	0.816	0.9	1.08
LVDS <sup>2</sup>	2.375	2.5	2.625	—	—	—
MLVDS25 <sup>1</sup>	2.375	2.5	2.625	—	—	—
LVPECL33 <sup>1,2</sup>	3.135	3.3	3.465	—	—	—
BLVDS25 <sup>1,2</sup>	2.375	2.5	2.625	—	—	—
RSDS <sup>1,2</sup>	2.375	2.5	2.625	—	—	—
SSTL18D_I <sup>2</sup> , II <sup>2</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. Input on this standard does not depend on the value of V<sub>CCIO</sub>.

**sysIO Single-Ended DC Electrical Characteristics**

Input/Output Standard	V <sub>IL</sub>		V <sub>IH</sub>		V <sub>OL</sub> Max. (V)	V <sub>OH</sub> Min. (V)	I <sub>OL</sub> <sup>1</sup> (mA)	I <sub>OH</sub> <sup>1</sup> (mA)
	Min. (V)	Max. (V)	Min. (V)	Max. (V)				
LVCMOS 3.3	-0.3	0.8	2.0	3.6	0.4	V <sub>CCIO</sub> - 0.4	20, 16, 12, 8, 4	-20, -16, -12, -8, -4
					0.2	V <sub>CCIO</sub> - 0.2	0.1	-0.1
LVTTL	-0.3	0.8	2.0	3.6	0.4	V <sub>CCIO</sub> - 0.4	20, 16, 12, 8, 4	-20, -16, -12, -8, -4
					0.2	V <sub>CCIO</sub> - 0.2	0.1	-0.1
LVCMOS 2.5	-0.3	0.7	1.7	3.6	0.4	V <sub>CCIO</sub> - 0.4	20, 16, 12, 8, 4	-20, -16, -12, -8, -4
					0.2	V <sub>CCIO</sub> - 0.2	0.1	-0.1
LVCMOS 1.8	-0.3	0.35 V <sub>CCIO</sub>	0.65 V <sub>CCIO</sub>	3.6	0.4	V <sub>CCIO</sub> - 0.4	16, 12, 8, 4	-16, -12, -8, -4
					0.2	V <sub>CCIO</sub> - 0.2	0.1	-0.1
LVCMOS 1.5	-0.3	0.35 V <sub>CCIO</sub>	0.65 V <sub>CCIO</sub>	3.6	0.4	V <sub>CCIO</sub> - 0.4	8, 4	-8, -4
					0.2	V <sub>CCIO</sub> - 0.2	0.1	-0.1
LVCMOS 1.2	-0.3	0.35 V <sub>CCIO</sub>	0.65 V <sub>CCIO</sub>	3.6	0.4	V <sub>CCIO</sub> - 0.4	6, 2	-6, -2
					0.2	V <sub>CCIO</sub> - 0.2	0.1	-0.1
PCI	-0.3	0.3 V <sub>CCIO</sub>	0.5 V <sub>CCIO</sub>	3.6	0.1 V <sub>CCIO</sub>	0.9 V <sub>CCIO</sub>	1.5	-0.5
SSTL3 Class I	-0.3	V <sub>REF</sub> - 0.2	V <sub>REF</sub> + 0.2	3.6	0.7	V <sub>CCIO</sub> - 1.1	8	-8
SSTL3 Class II	-0.3	V <sub>REF</sub> - 0.2	V <sub>REF</sub> + 0.2	3.6	0.5	V <sub>CCIO</sub> - 0.9	16	-16
SSTL2 Class I	-0.3	V <sub>REF</sub> - 0.18	V <sub>REF</sub> + 0.18	3.6	0.54	V <sub>CCIO</sub> - 0.62	7.6	-7.6
							12	-12
SSTL2 Class II	-0.3	V <sub>REF</sub> - 0.18	V <sub>REF</sub> + 0.18	3.6	0.35	V <sub>CCIO</sub> - 0.43	15.2	-15.2
							20	-20
SSTL18 Class I	-0.3	V <sub>REF</sub> - 0.125	V <sub>REF</sub> + 0.125	3.6	0.4	V <sub>CCIO</sub> - 0.4	6.7	-6.7
SSTL18 Class II	-0.3	V <sub>REF</sub> - 0.125	V <sub>REF</sub> + 0.125	3.6	0.28	V <sub>CCIO</sub> - 0.28	8	-8
							11	-11
HSTL Class I	-0.3	V <sub>REF</sub> - 0.1	V <sub>REF</sub> + 0.1	3.6	0.4	V <sub>CCIO</sub> - 0.4	4	-4
							8	-8
HSTL18 Class I	-0.3	V <sub>REF</sub> - 0.1	V <sub>REF</sub> + 0.1	3.6	0.4	V <sub>CCIO</sub> - 0.4	8	-8
							12	-12
HSTL18 Class II	-0.3	V <sub>REF</sub> - 0.1	V <sub>REF</sub> + 0.1	3.6	0.4	V <sub>CCIO</sub> - 0.4	16	-16

1. The average DC current drawn by I/Os between GND connections, or between the last GND in an I/O bank and the end of an I/O bank, as shown in the logic signal connections table shall not exceed n \* 8mA, where n is the number of I/Os between bank GND connections or between the last GND in a bank and the end of a bank.

**sysIO Differential Electrical Characteristics****LVDS****Over Recommended Operating Conditions**

Parameter	Description	Test Conditions	Min.	Typ.	Max.	Units
$V_{INP}$ , $V_{INM}$	Input Voltage		0	—	2.4	V
$V_{CM}$	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
$I_{SA}$	Output Short Circuit Current	$V_{OD} = 0V$ Driver Outputs Shorted to Ground	—	—	24	mA
$I_{SAB}$	Output Short Circuit Current	$V_{OD} = 0V$ Driver Outputs Shorted to Each Other	—	—	12	mA

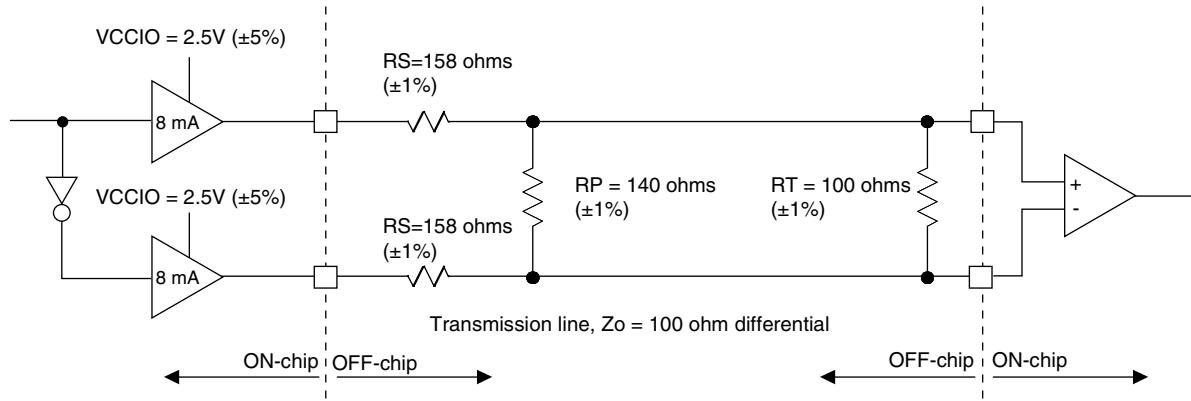
**Differential HSTL and SSTL**

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.

For further information on LVPECL, RSDS, MLVDS, BLVDS and other differential interfaces please see details of additional technical information at the end of this data sheet.

**LVDS25E**

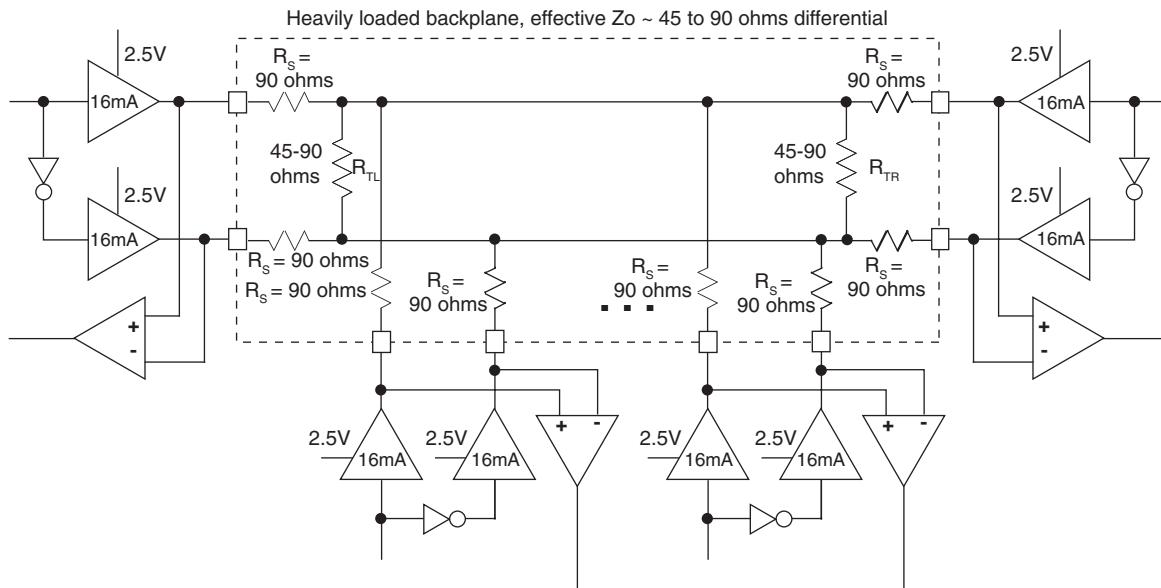
The top and bottom sides of LatticeECP2/M devices support LVDS outputs via emulated complementary LVCMS outputs in conjunction with a parallel resistor across the driver outputs. The scheme shown in Figure 3-1 is one possible solution for point-to-point signals.

**Figure 3-1. LVDS25E Output Termination Example****Table 3-2. 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

**BLVDS**

The LatticeECP2/M devices support the BLVDS standard. This standard is emulated using complementary LVC-MOS 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.

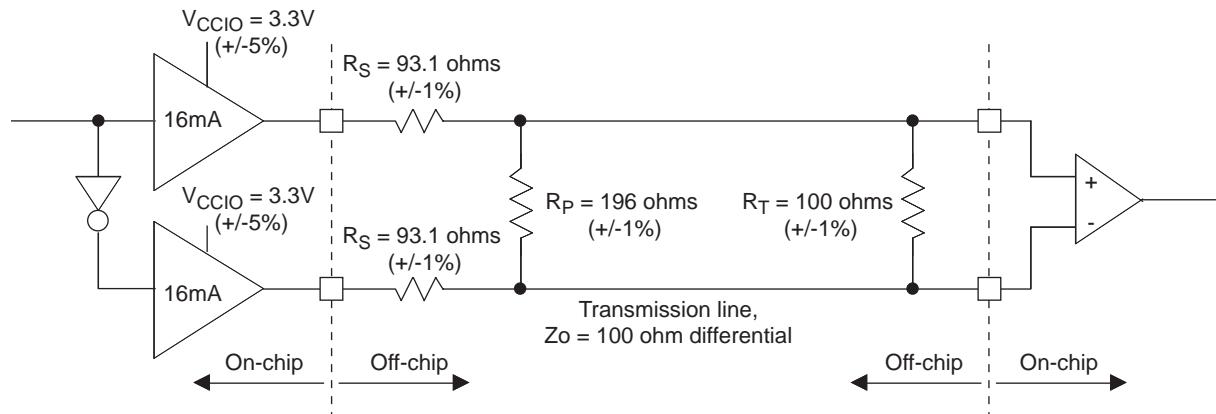
**Figure 3-2. BLVDS Multi-point Output Example****Table 3-3. BLVDS DC Conditions<sup>1</sup>****Over Recommended Operating Conditions**

Parameter	Description	Typical		Units
		Zo = 45Ω	Zo = 45Ω	
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.

**LVPECL**

The LatticeECP2/M devices support the differential LVPECL standard. This standard is emulated using complementary LVCMS 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.

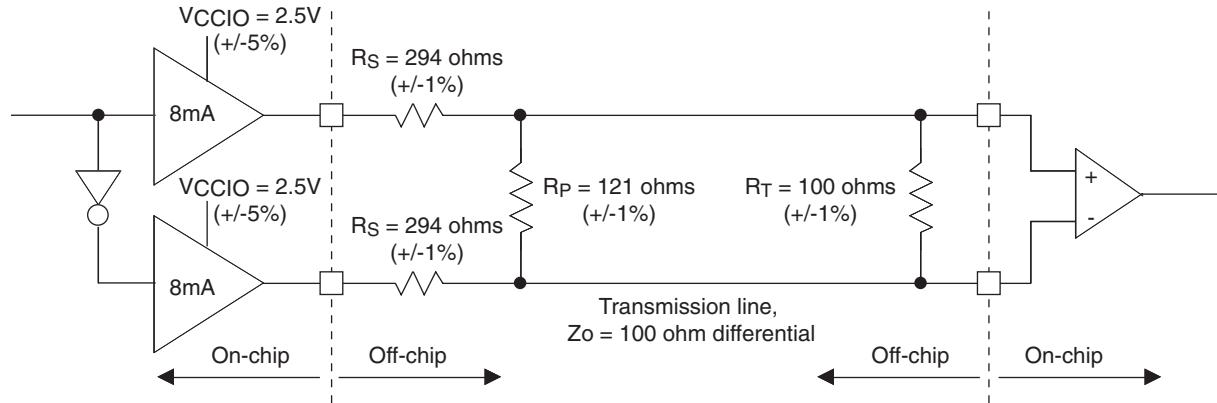
**Figure 3-3. Differential LVPECL****Table 3-4. LVPECL DC Conditions<sup>1</sup>****Over Recommended Operating Conditions**

Parameter	Description	Typical	Units
$V_{CCIO}$	Output Driver Supply (+/-5%)	3.30	V
$Z_{OUT}$	Driver Impedance	10	$\Omega$
$R_S$	Driver Series Resistor (+/-1%)	93	$\Omega$
$R_P$	Driver Parallel Resistor (+/-1%)	196	$\Omega$
$R_T$	Receiver Termination (+/-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.

**RSDS**

The LatticeECP2/M devices support differential RSDS standard. 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.

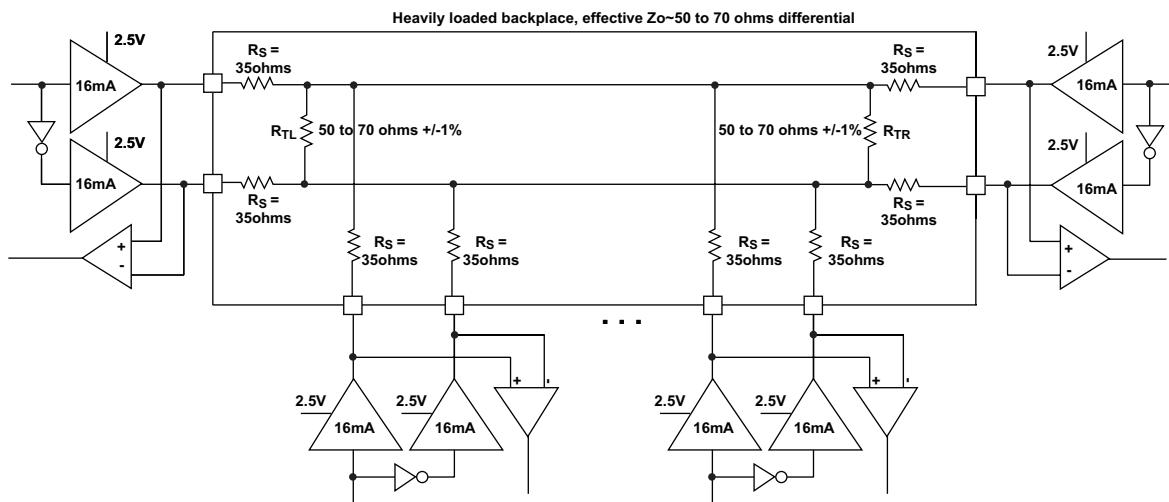
**Figure 3-4. RSDS (Reduced Swing Differential Standard)****Table 3-5. RSDS DC Conditions<sup>1</sup>****Over Recommended Operating Conditions**

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.

**MLVDS**

The LatticeECP2/M devices support the differential MLVDS standard. This standard is emulated using complementary LVCMS 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. MLVDS (Reduced Swing Differential Standard)****Table 3-6. MLVDS 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.

For further information on LVPECL, RSDS, MLVDS, BLVDS and other differential interfaces please see details of additional technical information at the end of this data sheet.

**Typical Building Block Function Performance<sup>1</sup>****Pin-to-Pin Performance (LVCMOS25 12mA Drive)**

Function	-7 Timing	Units
<b>Basic Functions</b>		
16-bit Decoder	4.2	ns
32-bit Decoder	4.6	ns
64-bit Decoder	5.4	ns
4:1 MUX	3.4	ns
8:1 MUX	3.7	ns
16:1 MUX	4.0	ns
32:1 MUX	4.1	ns

**Register-to-Register Performance**

Function	-7 Timing	Units
<b>Basic Functions</b>		
16-bit Decoder	499	MHz
32-bit Decoder	380	MHz
64-bit Decoder	349	MHz
4:1 MUX	796	MHz
8:1 MUX	723	MHz
16:1 MUX	667	MHz
32:1 MUX	543	MHz
8-bit Adder	561	MHz
16-bit Adder	460	MHz
64-bit Adder	238	MHz
16-bit Counter	453	MHz
32-bit Counter	363	MHz
64-bit Counter	253	MHz
64-bit Accumulator	250	MHz
<b>Embedded Memory Functions</b>		
512x36 Single Port RAM, EBR Output Registers	370	MHz
1024x18 True-Dual Port RAM (Write Through or Normal, EBR Output Registers)	370	MHz
1024x18 True-Dual Port RAM (Read-Before-Write, EBR Output Registers)	294	MHz
1024x18 True-Dual Port RAM (Write Through or Normal, PLC Output Registers)	272	MHz
<b>Distributed Memory Functions</b>		
16x4 Pseudo-Dual Port RAM (One PFU)	845	MHz
32x4 Pseudo-Dual Port RAM	457	MHz
64x8 Pseudo-Dual Port RAM	369	MHz
<b>DSP Functions</b>		
18x18 Multiplier (All Registers)	420	MHz
9x9 Multiplier (All Registers)	420	MHz

**Register-to-Register Performance (Continued)**

Function	-7 Timing	Units
36x36 Multiply (All Registers)	372	MHz
18x18 Multiply/Accumulate (Input and Output Registers)	323	MHz
18x18 Multiply-Add/Sub-Sum (All Registers)	420	MHz
<b>DSP IP Functions</b>		
16-Tap Fully-Parallel FIR Filter		MHz
1024-pt, Radix 4, Decimation in Frequency FFT		MHz
8X8 Matrix Multiplication		MHz

1. These timing numbers 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.

Timing v.0.07

**Derating Timing Tables**

Logic timing provided in the following sections of this data sheet and the 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 ispLEVER design tool can provide logic timing numbers at a particular temperature and voltage.

**LatticeECP2/M External Switching Characteristics**

Over Recommended Operating Conditions

Parameter	Description	Device	-7		-6		-5		Units
			Min.	Max.	Min.	Max.	Min.	Max.	
<b>General I/O Pin Parameters (Using Primary Clock without PLL)<sup>1</sup></b>									
t <sub>CO</sub>	Clock to Output - PIO Output Register	ECP2-50	—	3.06	—	3.45	—	3.83	ns
		ECP2M-35	—	3.00	—	3.50	—	3.80	ns
t <sub>SU</sub>	Clock to Data Setup - PIO Input Register	ECP2-50	0.00	—	0.00	—	0.00	—	ns
		ECP2M-35	0.00	—	0.00	—	0.00	—	ns
t <sub>H</sub>	Clock to Data Hold - PIO Input Register	ECP2-50	0.59	—	0.74	—	0.88	—	ns
		ECP2M-35	1.00	—	1.30	—	1.50	—	ns
t <sub>SU_DEL</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP2-50	0.98	—	0.98	—	0.99	—	ns
		ECP2M-35	1.10	—	1.40	—	1.70	—	ns
t <sub>H_DEL</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP2-50	0.00	—	0.00	—	0.00	—	ns
		ECP2M-35	0.00	—	0.00	—	0.00	—	ns
f <sub>MAX_IO</sub>	Clock Frequency of I/O and PFU Register	ECP2-50	—	420	—	336	—	252	MHz
		ECP2M-35	—	420	—	336	—	252	MHz
<b>General I/O Pin Parameters (using Edge Clock without PLL)<sup>1</sup></b>									
t <sub>COE</sub>	Clock to Output - PIO Output Register	ECP2-50	—	2.47	—	2.77	—	3.06	ns
		ECP2M-35	—	2.40	—	2.70	—	3.00	ns
t <sub>SUE</sub>	Clock to Data Setup - PIO Input Register	ECP2-50	0.00	—	0.00	—	0.00	—	ns
		ECP2M-35	0.00	—	0.00	—	0.00	—	ns
t <sub>HE</sub>	Clock to Data Hold - PIO Input Register	ECP2-50	0.62	—	0.69	—	0.76	—	ns
		ECP2M-35	1.00	—	1.20	—	1.40	—	ns
t <sub>SU_DEL_E</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP2-50	0.53	—	0.57	—	0.61	—	ns
		ECP2M-35	1.20	—	1.60	—	1.90	—	ns
t <sub>H_DEL_E</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP2-50	0.00	—	0.00	—	0.00	—	ns
		ECP2M-35	0.00	—	0.00	—	0.00	—	ns
<b>General I/O Pin Parameters (using Primary Clock with PLL)<sup>1</sup></b>									
t <sub>COPLL</sub>	Clock to Output - PIO Output Register	ECP2-50	—	2.12	—	2.36	—	2.60	ns
		ECP2M-35	—	2.10	—	2.40	—	2.60	ns
t <sub>SUPLL</sub>	Clock to Data Setup - PIO Input Register	ECP2-50	0.25	—	0.30	—	0.36	—	ns
		ECP2M-35	0.30	—	0.40	—	0.50	—	ns
t <sub>HPLL</sub>	Clock to Data Hold - PIO Input Register	ECP2-50	0.45	—	0.55	—	0.60	—	ns
		ECP2M-35	0.80	—	1.00	—	1.20	—	ns
t <sub>SU_DELPLL</sub>	Clock to Data Setup - PIO Input Register with Data Input Delay	ECP2-50	0.85	—	0.94	—	1.04	—	ns
		ECP2M-35	1.10	—	1.30	—	1.50	—	ns
t <sub>H_DELPLL</sub>	Clock to Data Hold - PIO Input Register with Input Data Delay	ECP2-50	0.00	—	0.00	—	0.00	—	ns
		ECP2M-35	0.00	—	0.00	—	0.00	—	ns
<b>DDR<sup>2</sup> I/O Pin Parameters</b>									
t <sub>DVADQ</sub>	Data Valid Before DQS (DDR Read)	ECP2-50	—	0.225	—	0.225	—	0.225	UI
		ECP2M-35	—	0.225	—	0.225	—	0.225	UI
t <sub>DVEDQ</sub>	Data Hold After DQS (DDR Read)	ECP2-50	0.640	—	0.640	—	0.640	—	UI
		ECP2M-35	0.640	—	0.640	—	0.640	—	UI
t <sub>DQVBS</sub>	Data Valid Before DQS	ECP2-50	0.250	—	0.250	—	0.250	—	UI
		ECP2M-35	0.250	—	0.250	—	0.250	—	UI

**LatticeECP2/M External Switching Characteristics (Continued)****Over Recommended Operating Conditions**

Parameter	Description	Device	-7		-6		-5		Units
			Min.	Max.	Min.	Max.	Min.	Max.	
$t_{DQVAS}$	Data Valid After DQS	ECP2-50	0.250	—	0.250	—	0.250	—	UI
		ECP2M-35	0.250	—	0.250	—	0.250	—	UI
$f_{MAX\_DDR}$	DDR Clock Frequency <sup>6</sup>	ECP2-50	95	200	95	166	95	133	MHz
		ECP2M-35	95	200	95	166	95	133	MHz
<b>DDR2<sup>3</sup> I/O Pin Parameters</b>									
$t_{DVADQ}$	Data Valid Before DQS (DDR Read)	ECP2-50	—	0.225	—	0.225	—	0.225	UI
		ECP2M-35	—	0.225	—	0.225	—	0.225	UI
$t_{DVEDQ}$	Data Hold After DQS (DDR Read)	ECP2-50	0.640	—	0.640	—	0.640	—	UI
		ECP2M-35	0.640	—	0.640	—	0.640	—	UI
$t_{DQVBS}$	Data Valid Before DQS	ECP2-50	0.250	—	0.250	—	0.250	—	UI
		ECP2M-35	0.250	—	0.250	—	0.250	—	UI
$t_{DQVAS}$	Data Valid After DQS	ECP2-50	0.250	—	0.250	—	0.250	—	UI
		ECP2M-35	0.250	—	0.250	—	0.250	—	UI
$f_{MAX\_DDR2}$	DDR Clock Frequency	ECP2-50	133	200	133	166	133	145	MHz
		ECP2M-35	133	200	133	166	133	145	MHz
<b>Source Synchronous (e.g. SPI4.2 Static Alignment) I/O Parameters<sup>4</sup></b>									
	Maximum Data Rate	ECP2-50	—	750	—	622	—	622	Mbps
		ECP2M-35	—	750	—	622	—	622	Mbps
$t_{DVACLKSPI}$	Data Valid Before CLK	ECP2-50	—	0.26	—	0.26	—	0.26	UI
		ECP2M-35	—	0.26	—	0.26	—	0.26	UI
$t_{DVECLKSPI}$	Data Hold After CLK	ECP2-50	0.74	—	0.74	—	0.74	—	UI
		ECP2M-35	0.78	—	0.78	—	0.78	—	UI
$t_{DIASPI}$	Data Invalid After Clock	ECP2-50	—	220	—	220	—	220	ps
		ECP2M-35	—	260	—	280	—	280	ps
$t_{DIBSPI}$	Data Invalid Before Clock	ECP2-50	—	220	—	220	—	220	ps
		ECP2M-35	—	260	—	280	—	280	ps
<b>XGMII I/O Pin Parameters<sup>5</sup> (312 Mbps)</b>									
$t_{SUXGMII}$	Data Setup Before Read Clock	ECP2-50	480	—	480	—	480	—	ps
		ECP2M-35	480	—	480	—	480	—	ps
$t_{HXGMII}$	Data Hold After Read Clock	ECP2-50	480	—	480	—	480	—	ps
		ECP2M-35	480	—	480	—	480	—	ps
$t_{DQVBSXGMII}$	Data Invalid Before Strobe/Clock	ECP2-50	960	—	960	—	960	—	ps
		ECP2M-35	960	—	960	—	960	—	ps
$t_{DQVASXGMII}$	Data Invalid After Strobe/Clock	ECP2-50	960	—	960	—	960	—	ps
		ECP2M-35	960	—	960	—	960	—	ps
<b>Primary</b>									
$f_{tMAX\_PRI}$	Frequency for Primary Clock Tree	ECP2-50	—	420	-	336	—	252	MHz
		ECP2M-35	—	420	-	336	—	252	MHz
$t_{W\_PRI}$	Clock Pulse Width for Primary Clock	ECP2-50	0.95	—	1.19	—	1.59	—	ns
		ECP2M-35	0.95	—	1.19	—	1.59	—	ns

**LatticeECP2/M External Switching Characteristics (Continued)**

Over Recommended Operating Conditions

Parameter	Description	Device	-7		-6		-5		Units
			Min.	Max.	Min.	Max.	Min.	Max.	
$t_{\text{SKEW\_PRI}}$	Primary Clock Skew Within a Bank	ECP2-50	—	300	—	360	—	420	ps
		ECP2M-35	—	300	—	360	—	420	ps
<b>Edge Clock</b>									
$f_{\text{MAX\_EDGE}}$	Frequency for Edge Clock	ECP2-50	—	420	—	337	—	311	MHz
		ECP2M-35	—	420	—	337	—	311	MHz
$t_{\text{W\_EDGE}}$	Clock Pulse Width for Edge Clock	ECP2-50	0.95	—	1.19	—	1.59	—	ns
		ECP2M-35	0.95	—	1.19	—	1.59	—	ns
$t_{\text{SKEW\_EDGE}}$	Edge Clock Skew Within an Edge of the Device	ECP2-50	—	250	—	300	—	350	ps
		ECP2M-35	—	300	—	360	—	420	ps

1. General timing numbers based on LVCMS 2.5, 12mA, 0pf load.

2. DDR timing numbers based on SSTL25.

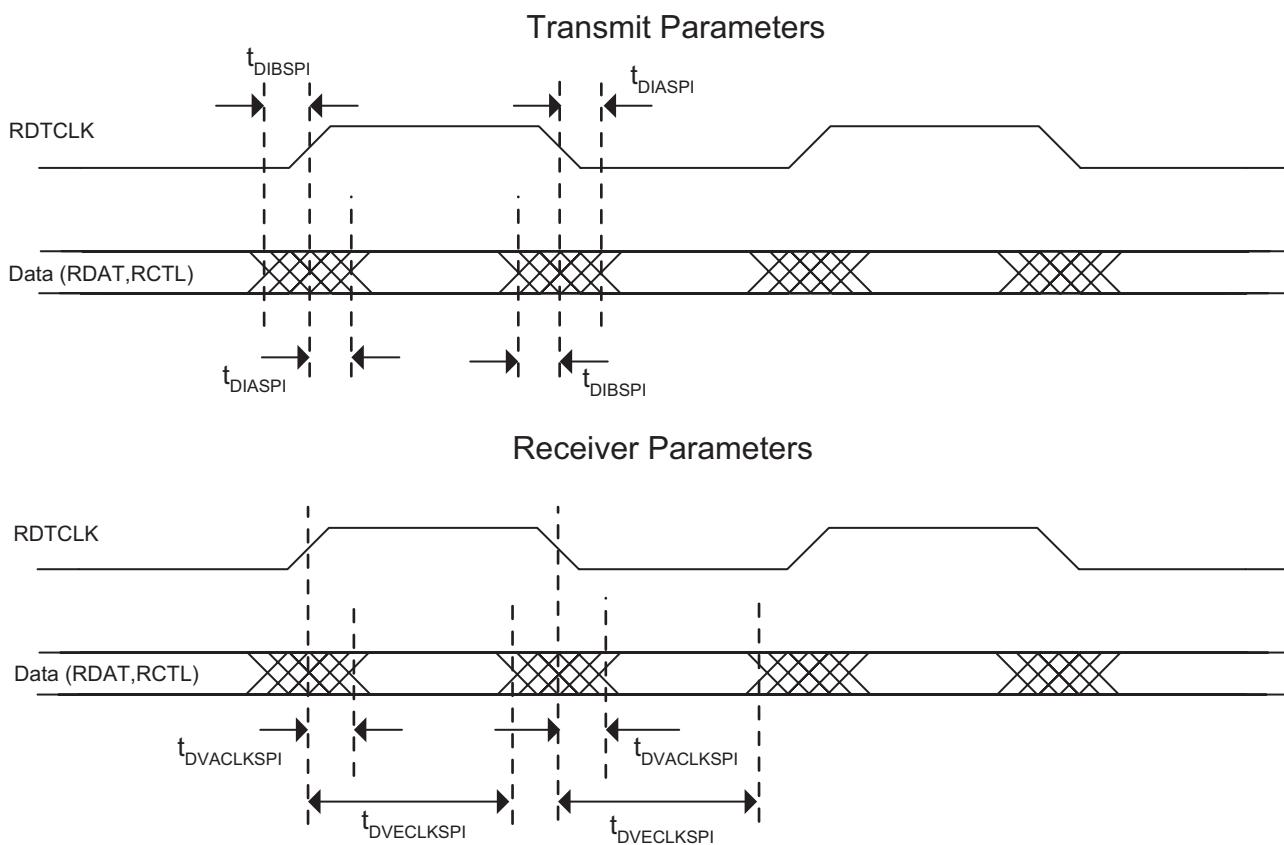
3. DDR2 timing numbers based on SSTL18.

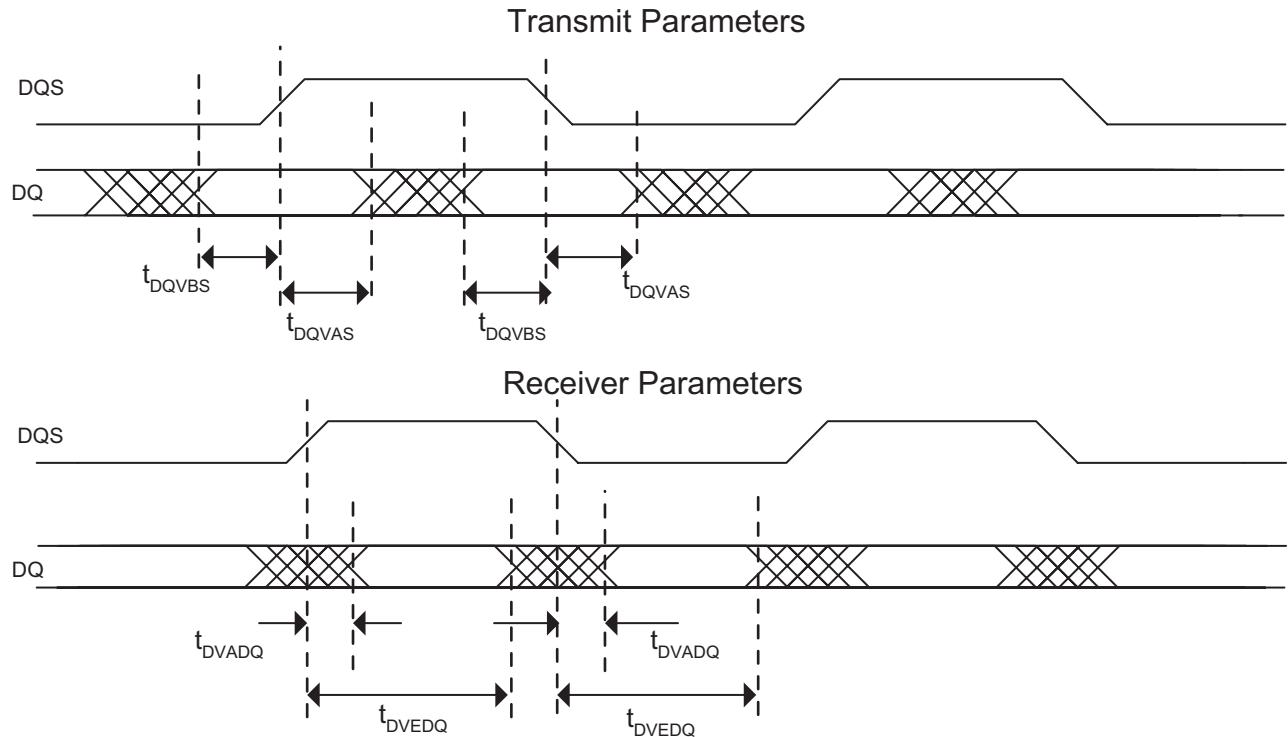
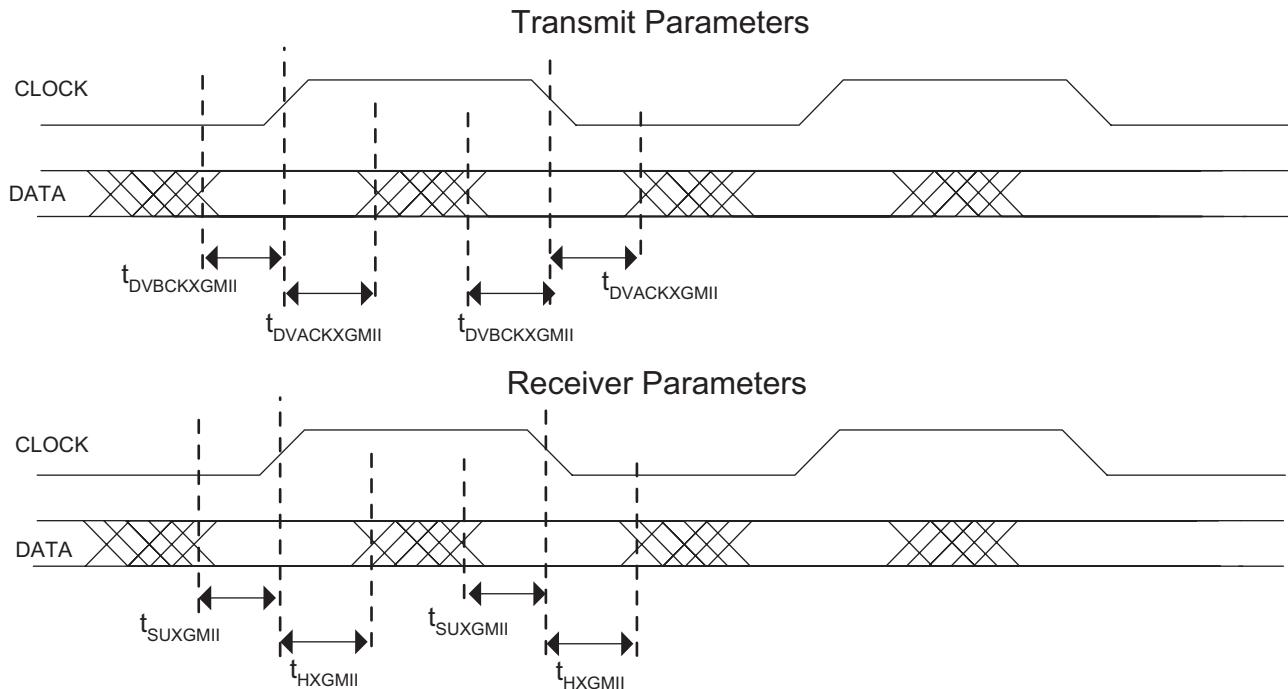
4. SPI4.2 and SFI4 timing numbers based on LVDS25.

5. XGMII timing numbers based on HSTL class I.

6. IP will be used to support DDR and DDR2 memory data rates down to 95MHz for DDR and 133MHz for DDR2. This approach uses a free-running clock and PFU register to sample the data instead of the hardwired DDR memory interface.

Timing v.0.07

**Figure 3-6. SPI4.2 Parameters**

**Figure 3-7. DDR and DDR2 Parameters****Figure 3-8. XGMII Parameters**

**LatticeECP2/M Internal Switching Characteristics<sup>1</sup>**

Over Recommended Operating Conditions

Parameter	Description	-7		-6		-5		Units
		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)	—	0.180	—	0.198	—	0.216	ns
t <sub>LUT6_PFU</sub>	LUT6 delay (A to D inputs to OFX output)	—	0.304	—	0.331	—	0.358	ns
t <sub>LSR_PFU</sub>	Set/Reset to output of PFU (Asynchronous)	—	0.600	—	0.655	—	0.711	ns
t <sub>SUM_PFU</sub>	Clock to Mux (M0,M1) Input Setup Time	0.128	—	0.129	—	0.129	—	ns
t <sub>HM_PFU</sub>	Clock to Mux (M0,M1) Input Hold Time	-0.051	—	-0.049	—	-0.046	—	ns
t <sub>SUD_PFU</sub>	Clock to D input setup time	0.061	—	0.071	—	0.081	—	ns
t <sub>HD_PFU</sub>	Clock to D input hold time	0.002	—	0.003	—	0.003	—	ns
t <sub>CK2Q_PFU</sub>	Clock to Q delay, (D-type Register Configuration)	—	0.285	—	0.309	—	0.333	ns
<b>PFU Dual Port Memory Mode Timing</b>								
t <sub>CORAM_PFU</sub>	Clock to Output (F Port)	—	0.902	—	1.083	—	1.263	ns
t <sub>SUDATA_PFU</sub>	Data Setup Time	-0.172	—	-0.205	—	-0.238	—	ns
t <sub>HDATA_PFU</sub>	Data Hold Time	0.199	—	0.235	—	0.271	—	ns
t <sub>SUADDR_PFU</sub>	Address Setup Time	-0.245	—	-0.284	—	-0.323	—	ns
t <sub>HADDR_PFU</sub>	Address Hold Time	0.246	—	0.285	—	0.324	—	ns
t <sub>SUWREN_PFU</sub>	Write/Read Enable Setup Time	-0.122	—	-0.145	—	-0.168	—	ns
t <sub>HWREN_PFU</sub>	Write/Read Enable Hold Time	0.132	—	0.156	—	0.180	—	ns
<b>PIC Timing</b>								
<b>PIO Input/Output Buffer Timing</b>								
t <sub>IN_PIO</sub>	Input Buffer Delay (LVCMOS25)	—	0.613	—	0.681	—	0.749	ns
t <sub>OUT_PIO</sub>	Output Buffer Delay (LVCMOS25)	—	1.115	—	1.115	—	1.343	ns
<b>IOLOGIC Input/Output Buffer Timing</b>								
t <sub>SUI_PIO</sub>	Input Register Setup Time (Data Before Clock)	0.596	—	0.645	—	0.694	—	ns
t <sub>HI_PIO</sub>	Input Register Hold Time (Data after Clock)	-0.570	—	-0.614	—	-0.658	—	ns
t <sub>COO_PIO</sub>	Output Register Clock to Output Delay	—	0.61	—	0.66	—	0.72	ns
t <sub>SUCE_PIO</sub>	Input Register Clock Enable Setup Time	0.032	—	0.037	—	0.041	—	ns
t <sub>HCE_PIO</sub>	Input Register Clock Enable Hold Time	-0.022	—	-0.025	—	-0.028	—	ns
t <sub>SULSR_PIO</sub>	Set/Reset Setup Time	0.184	—	0.201	—	0.217	—	ns
t <sub>HLSR_PIO</sub>	Set/Reset Hold Time	-0.080	—	-0.086	—	-0.093	—	ns
<b>EBR Timing</b>								
t <sub>CO_EBR</sub>	Clock (Read) to output from Address or Data	—	2.51	—	2.75	—	2.99	ns
t <sub>COO_EBR</sub>	Clock (Write) to output from EBR output Register	—	0.33	—	0.36	—	0.39	ns
t <sub>SUDATA_EBR</sub>	Setup Data to EBR Memory	-0.157	—	-0.181	—	-0.205	—	ns
t <sub>HDATA_EBR</sub>	Hold Data to EBR Memory	0.173	—	0.195	—	0.217	—	ns
t <sub>SUADDR_EBR</sub>	Setup Address to EBR Memory	-0.115	—	-0.130	—	-0.145	—	ns
t <sub>HADDR_EBR</sub>	Hold Address to EBR Memory	0.138	—	0.155	—	0.172	—	ns
t <sub>SUWREN_EBR</sub>	Setup Write/Read Enable to PFU Memory	-0.128	—	-0.149	—	-0.170	—	ns
t <sub>HWREN_EBR</sub>	Hold Write/Read Enable to PFU Memory	0.139	—	0.156	—	0.173	—	ns
t <sub>SUCE_EBR</sub>	Clock Enable Setup Time to EBR Output Register	0.123	—	0.134	—	0.145	—	ns
t <sub>HCE_EBR</sub>	Clock Enable Hold Time to EBR Output Register	-0.081	—	-0.090	—	-0.100	—	ns
t <sub>RSTO_EBR</sub>	Reset To Output Delay Time from EBR Output Register	—	1.03	—	1.15	—	1.26	ns

**LatticeECP2/M Internal Switching Characteristics<sup>1</sup> (Continued)**

Over Recommended Operating Conditions

Parameter	Description	-7		-6		-5		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
t <sub>SUBE_EBR</sub>	Byte Enable Set-Up Time to EBR Output Register	—	—	—	—	—	—	ns
t <sub>HBE_EBR</sub>	Byte Enable Hold Time to EBR Output Register	—	—	—	—	—	—	ns
<b>GPLL Parameters</b>								
t <sub>RSTREC_GPLL</sub>	Reset Recovery to Rising Clock	1.0	—	1.0	—	1.0	—	ns
t <sub>RSTSU_GPLL</sub>	Reset signal setup time	1.8	—	1.8	—	1.8	—	ns
<b>SPLL Parameters</b>								
t <sub>RSTREC_SPLL</sub>	Reset Recovery to Rising Clock	1.0	—	1.0	—	1.0	—	ns
t <sub>RSTSU_SPLL</sub>	Reset signal setup time	1.8	—	1.8	—	1.8	—	ns
<b>DSP Block Timing<sup>2,3</sup></b>								
t <sub>SUI_DSP</sub>	Input Register Setup Time	0.12	—	0.13	—	0.14	—	ns
t <sub>HI_DSP</sub>	Input Register Hold Time	0.02	—	-0.01	—	-0.03	—	ns
t <sub>SUP_DSP</sub>	Pipeline Register Setup Time	2.18	—	2.42	—	2.66	—	ns
t <sub>HP_DSP</sub>	Pipeline Register Hold Time	-0.68	—	-0.77	—	-0.86	—	ns
t <sub>SUO_DSP</sub>	Output Register Setup Time	4.26	—	4.71	—	5.16	—	ns
t <sub>HO_DSP</sub>	Output Register Hold Time	-1.25	—	-1.40	—	-1.54	—	ns
t <sub>COI_DSP</sub>	Input Register Clock to Output Time	—	3.92	—	4.30	—	4.68	ns
t <sub>COP_DSP</sub>	Pipeline Register Clock to Output Time	—	1.87	—	1.98	—	2.08	ns
t <sub>COO_DSP</sub>	Output Register Clock to Output Time	—	0.50	—	0.52	—	0.55	ns
t <sub>SUADDSSUB</sub>	AddSub Input Register Setup Time	-0.24	—	-0.26	—	-0.28	—	ns
t <sub>HADDSSUB</sub>	AddSub Input Register Hold Time	0.27	—	0.29	—	0.32	—	ns

1. Internal parameters are characterized but not tested on every device.

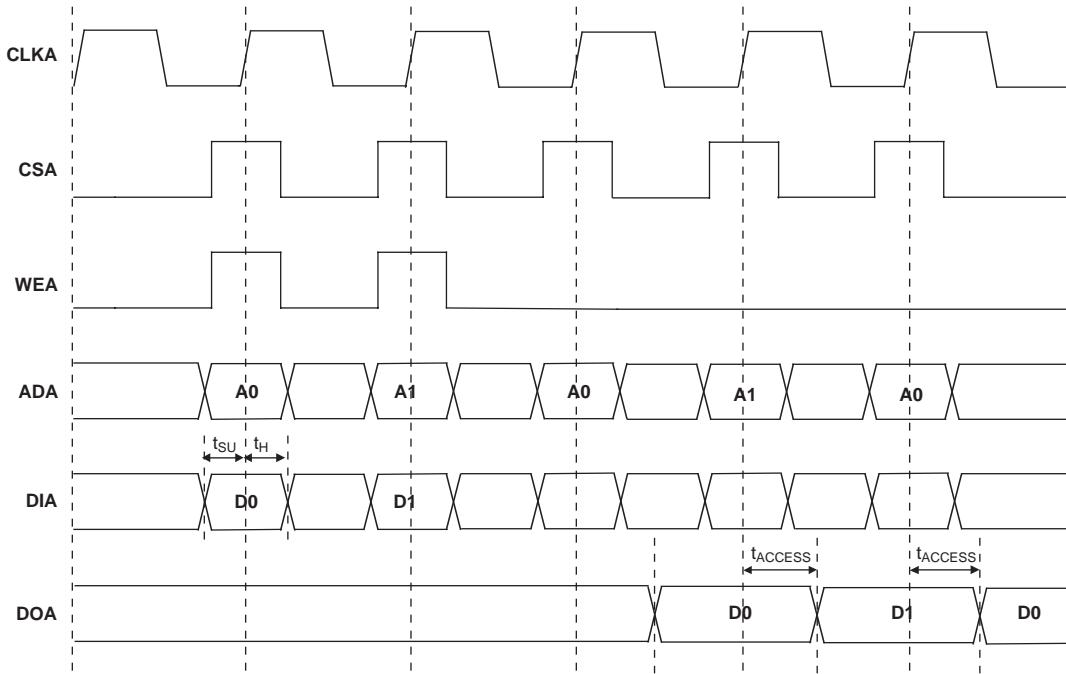
2. These parameters apply to LatticeECP devices only.

3. DSP Block is configured in Multiply Add/Sub 18x18 Mode.

Timing v.0.07

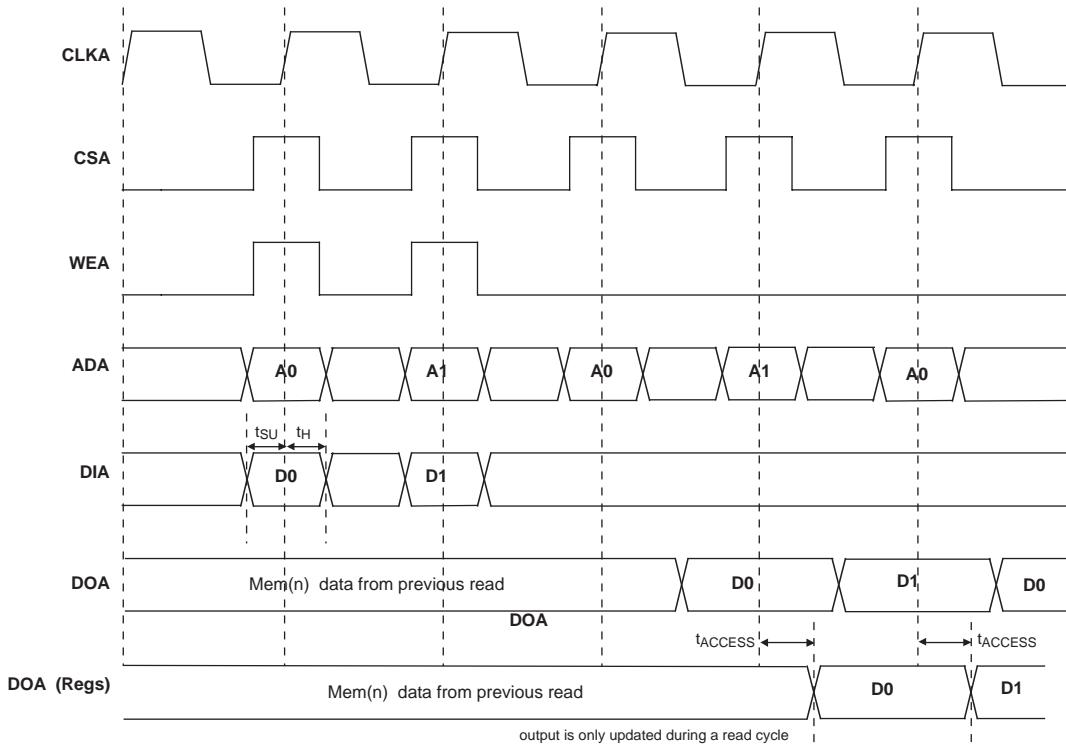
## Timing Diagrams

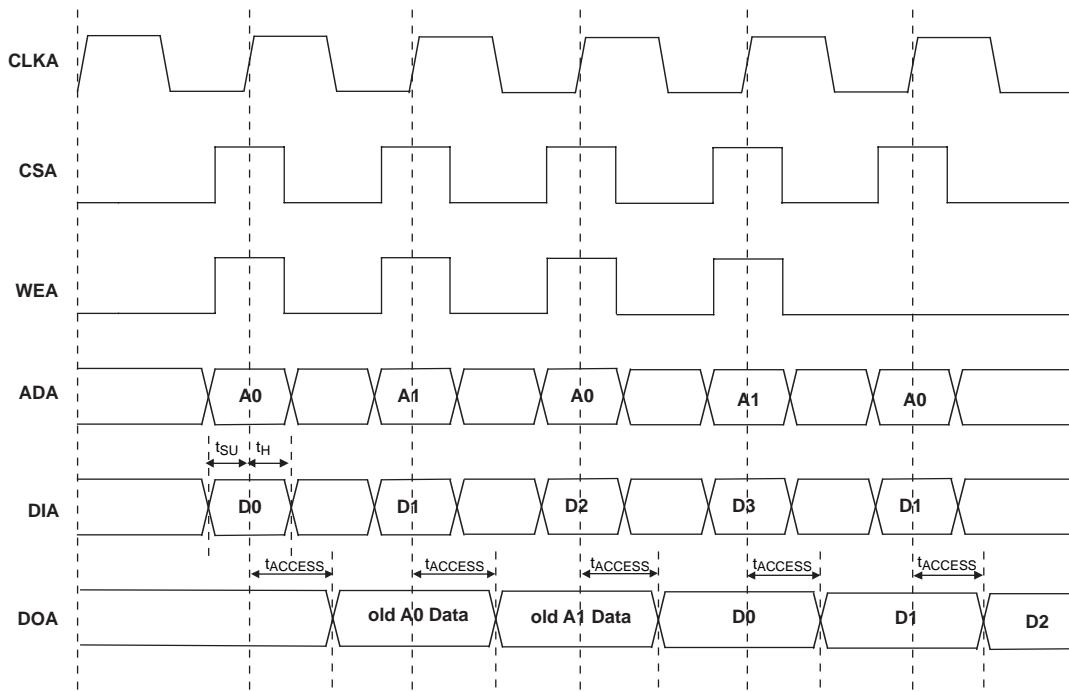
**Figure 3-9. Read/Write Mode (Normal)**



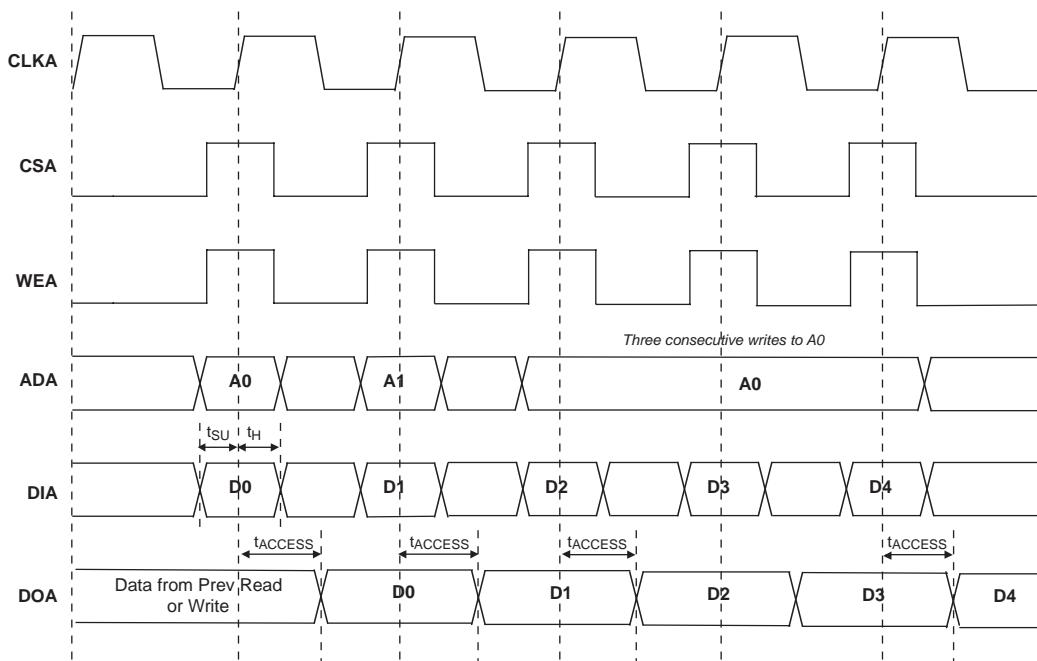
Note: Input data and address are registered at the positive edge of the clock and output data appears after the positive edge of the clock.

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



**Figure 3-11. Read Before Write (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.

**Figure 3-12. 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.

**LatticeECP2/M Family Timing Adders<sup>1, 2, 3</sup>**

Over Recommended Operating Conditions

Buffer Type	Description	-7	-6	-5	Units
<b>Input Adjusters</b>					
LVDS25E	LVDS E	-0.04	-0.07	-0.10	ns
LVDS25	LVDS	-0.04	-0.02	0.00	ns
BLVDS25	BLVDS	-0.04	-0.09	-0.15	ns
MLVDS	LVDS	-0.15	-0.15	-0.15	ns
RSDS	RSDS	0.16	0.01	-0.15	ns
LVPECL33	LVPECL	0.16	0.15	0.13	ns
HSTL18_I	HSTL_18 class I	0.01	-0.01	-0.04	ns
HSTL18_II	HSTL_18 class II	0.01	-0.01	-0.04	ns
HSTL18D_I	Differential HSTL 18 class I	0.01	-0.01	-0.04	ns
HSTL18D_II	Differential HSTL 18 class II	0.01	-0.01	-0.04	ns
HSTL15_I	HSTL_15 class I	0.01	-0.01	-0.04	ns
HSTL15D_I	Differential HSTL 15 class I	0.01	-0.01	-0.04	ns
SSTL33_I	SSTL_3 class I	-0.03	-0.07	-0.10	ns
SSTL33_II	SSTL_3 class II	-0.03	-0.07	-0.10	ns
SSTL33D_I	Differential SSTL_3 class I	-0.03	-0.07	-0.10	ns
SSTL33D_II	Differential SSTL_3 class II	-0.03	-0.07	-0.10	ns
SSTL25_I	SSTL_2 class I	-0.04	-0.07	-0.10	ns
SSTL25_II	SSTL_2 class II	-0.04	-0.07	-0.10	ns
SSTL25D_I	Differential SSTL_2 class I	-0.04	-0.07	-0.10	ns
SSTL25D_II	Differential SSTL_2 class II	-0.04	-0.07	-0.10	ns
SSTL18_I	SSTL_18 class I	-0.01	-0.04	-0.07	ns
SSTL18_II	SSTL_18 class II	-0.01	-0.04	-0.07	ns
SSTL18D_I	Differential SSTL_18 class I	-0.01	-0.04	-0.07	ns
SSTL18D_II	Differential SSTL_18 class II	-0.01	-0.04	-0.07	ns
LVTTL33	LVTTL	-0.16	-0.16	-0.16	ns
LVCMOS33	LVCMOS 3.3	-0.08	-0.12	-0.16	ns
LVCMOS25	LVCMOS 2.5	0.00	0.00	0.00	ns
LVCMOS18	LVCMOS 1.8	-0.16	-0.17	-0.17	ns
LVCMOS15	LVCMOS 1.5	-0.14	-0.14	-0.14	ns
LVCMOS12	LVCMOS 1.2	-0.04	-0.01	0.01	ns
<b>Output Adjusters</b>					
PCI33	PCI	-0.08	-0.12	-0.16	ns
LVDS25E	LVDS 2.5 E	0.25	0.30	0.13	ns
LVDS25	LVDS 2.5	0.10	0.25	0.17	ns
BLVDS25	BLVDS 2.5	0.00	0.10	-0.03	ns
MLVDS	MLVDS 2.5	0.00	0.10	-0.03	ns
RSDS	RSDS 2.5	0.25	0.30	0.13	ns
LVPECL33	LVPECL 3.3	-0.02	0.07	-0.06	ns
HSTL18_I	HSTL_18 class I 8mA drive	-0.19	-0.11	-0.25	ns
HSTL18_II	HSTL_18 class II	-0.30	-0.22	-0.37	ns
HSTL18D_I	Differential HSTL 18 class I 8mA drive	-0.19	-0.11	-0.25	ns

**LatticeECP2/M Family Timing Adders<sup>1,2,3</sup> (Continued)**

Over Recommended Operating Conditions

Buffer Type	Description	-7	-6	-5	Units
HSTL18D_II	Differential HSTL 18 class II	-0.30	-0.22	-0.37	ns
HSTL15_I	HSTL_15 class I 4mA drive	-0.22	-0.13	-0.27	ns
HSTL15D_I	Differential HSTL 15 class I 4mA drive	-0.22	-0.13	-0.27	ns
SSTL33_I	SSTL_3 class I	-0.12	-0.04	-0.18	ns
SSTL33_II	SSTL_3 class II	-0.20	-0.12	-0.27	ns
SSTL33D_I	Differential SSTL_3 class I	-0.12	-0.04	-0.18	ns
SSTL33D_II	Differential SSTL_3 class II	-0.20	-0.12	-0.27	ns
SSTL25_I	SSTL_2 class I 8mA drive	-0.16	-0.08	-0.22	ns
SSTL25_II	SSTL_2 class II 16mA drive	-0.19	-0.11	-0.25	ns
SSTL25D_I	Differential SSTL_2 class I 8mA drive	-0.16	-0.08	-0.22	ns
SSTL25D_II	Differential SSTL_2 class II 16mA drive	-0.19	-0.11	-0.25	ns
SSTL18_I	SSTL_1.8 class I	-0.14	-0.06	-0.20	ns
SSTL18_II	SSTL_1.8 class II 8mA drive	-0.20	-0.11	-0.25	ns
SSTL18D_I	Differential SSTL_1.8 class I	-0.14	-0.06	-0.20	ns
SSTL18D_II	Differential SSTL_1.8 class II 8mA drive	-0.20	-0.11	-0.25	ns
LVTTL33_4mA	LVTTL 4mA drive	0.52	0.52	0.68	ns
LVTTL33_8mA	LVTTL 8mA drive	0.06	0.06	0.09	ns
LVTTL33_12mA	LVTTL 12mA drive	0.04	0.04	0.05	ns
LVTTL33_16mA	LVTTL 16mA drive	0.03	0.03	0.02	ns
LVTTL33_20mA	LVTTL 20mA drive	-0.09	-0.09	-0.10	ns
LVCMOS33_4mA	LVCMOS 3.3 4mA drive, fast slew rate	0.52	0.52	0.68	ns
LVCMOS33_8mA	LVCMOS 3.3 8mA drive, fast slew rate	0.06	0.06	0.09	ns
LVCMOS33_12mA	LVCMOS 3.3 12mA drive, fast slew rate	0.04	0.04	0.05	ns
LVCMOS33_16mA	LVCMOS 3.3 16mA drive, fast slew rate	0.03	0.03	0.02	ns
LVCMOS33_20mA	LVCMOS 3.3 20mA drive, fast slew rate	-0.09	-0.09	-0.10	ns
LVCMOS25_4mA	LVCMOS 2.5 4mA drive, fast slew rate	0.41	0.41	0.53	ns
LVCMOS25_8mA	LVCMOS 2.5 8mA drive, fast slew rate	0.01	0.01	0.00	ns
LVCMOS25_12mA	LVCMOS 2.5 12mA drive, fast slew rate	0.00	0.00	0.00	ns
LVCMOS25_16mA	LVCMOS 2.5 16mA drive, fast slew rate	0.04	0.04	0.04	ns
LVCMOS25_20mA	LVCMOS 2.5 20mA drive, fast slew rate	-0.09	-0.09	-0.11	ns
LVCMOS18_4mA	LVCMOS 1.8 4mA drive, fast slew rate	0.37	0.37	0.43	ns
LVCMOS18_8mA	LVCMOS 1.8 8mA drive, fast slew rate	0.10	0.10	0.13	ns
LVCMOS18_12mA	LVCMOS 1.8 12mA drive, fast slew rate	-0.02	-0.02	-0.02	ns
LVCMOS18_16mA	LVCMOS 1.8 16mA drive, fast slew rate	-0.02	-0.02	-0.03	ns
LVCMOS15_4mA	LVCMOS 1.5 4mA drive, fast slew rate	0.29	0.29	0.32	ns
LVCMOS15_8mA	LVCMOS 1.5 8mA drive, fast slew rate	0.05	0.05	0.06	ns
LVCMOS12_2mA	LVCMOS 1.2 2mA drive, fast slew rate	0.58	0.58	0.79	ns
LVCMOS12_6mA	LVCMOS 1.2 6mA drive, fast slew rate	0.13	0.13	0.26	ns
LVCMOS33_4mA	LVCMOS 3.3 4mA drive, slow slew rate	2.17	2.17	2.71	ns
LVCMOS33_8mA	LVCMOS 3.3 8mA drive, slow slew rate	2.50	2.50	2.83	ns
LVCMOS33_12mA	LVCMOS 3.3 12mA drive, slow slew rate	1.72	1.72	2.05	ns
LVCMOS33_16mA	LVCMOS 3.3 16mA drive, slow slew rate	1.64	1.64	1.62	ns

**LatticeECP2/M Family Timing Adders<sup>1,2,3</sup> (Continued)**

Over Recommended Operating Conditions

Buffer Type	Description	-7	-6	-5	Units
LVCMOS33_20mA	LVCMOS 3.3 20mA drive, slow slew rate	1.33	1.33	1.39	ns
LVCMOS25_4mA	LVCMOS 2.5 4mA drive, slow slew rate	2.18	2.18	2.33	ns
LVCMOS25_8mA	LVCMOS 2.5 8mA drive, slow slew rate	2.19	2.19	2.51	ns
LVCMOS25_12mA	LVCMOS 2.5 12mA drive, slow slew rate	1.50	1.50	1.82	ns
LVCMOS25_16mA	LVCMOS 2.5 16mA drive, slow slew rate	1.60	1.60	1.58	ns
LVCMOS25_20mA	LVCMOS 2.5 20mA drive, slow slew rate	1.43	1.43	1.34	ns
LVCMOS18_4mA	LVCMOS 1.8 4mA drive, slow slew rate	2.22	2.22	2.32	ns
LVCMOS18_8mA	LVCMOS 1.8 8mA drive, slow slew rate	1.93	1.93	2.23	ns
LVCMOS18_12mA	LVCMOS 1.8 12mA drive, slow slew rate	1.43	1.43	1.58	ns
LVCMOS18_16mA	LVCMOS 1.8 16mA drive, slow slew rate	1.47	1.47	1.45	ns
LVCMOS15_4mA	LVCMOS 1.5 4mA drive, slow slew rate	2.32	2.32	2.43	ns
LVCMOS15_8mA	LVCMOS 1.5 8mA drive, slow slew rate	1.84	1.84	2.12	ns
LVCMOS12_2mA	LVCMOS 1.2 2mA drive, slow slew rate	2.52	2.52	2.74	ns
LVCMOS12_6mA	LVCMOS 1.2 6mA drive, slow slew rate	1.69	1.69	1.96	ns
PCI33	PCI33	0.04	0.15	0.04	ns

1. Timing adders are characterized but not tested on every device.
2. LVCMOS timing measured with the load specified in Switching Test Condition table.
3. All other standards tested according to the appropriate specifications.

Timing v. 0.07

**sysCLOCK GPLL Timing****Over Recommended Operating Conditions**

Parameter	Description	Conditions	Min.	Typ.	Max.	Units
$f_{IN}$	Input Clock Frequency (CLKI, CLKFB)	Without external capacitor	25	—	420	MHz
		With external capacitor <sup>5, 6</sup>	3	—	50	MHz
$f_{OUT}$	Output Clock Frequency (CLKOP, CLKOS)	Without external capacitor	25	—	420	MHz
		With external capacitor <sup>5</sup>	5	—	50	MHz
$f_{OUT2}$	K-Divider Output Frequency (CLKOK)	Without external capacitor	0.195	—	210	MHz
		With external capacitor <sup>5</sup>	0.039	—	25	MHz
$f_{VCO}$	PLL VCO Frequency		640	—	1280	MHz
$f_{PFD}$	Phase Detector Input Frequency	Without external capacitor	25	—		MHz
		With external capacitor <sup>5, 6</sup>	3	—		MHz
<b>AC Characteristics</b>						
$t_{DT}$	Output Clock Duty Cycle	Default duty cycle selected <sup>3</sup>	45	50	55	%
$t_{PH}^4$	Output Phase Accuracy		—	—	$\pm 0.05$	UI
$t_{OPJIT}^1$	Output Clock Period Jitter	$f_{OUT} \geq 100$ MHz	—	—	$\pm 125$	ps
		$50 \leq f_{OUT} < 100$ MHz	—	—	0.025	UIPP
		$f_{OUT} < 50$ MHz	—	—	0.04	UIPP
$t_{SK}$	Input Clock to Output Clock Skew	N/M = integer	—	—	$\pm 250$	ps
$t_W$	Output Clock Pulse Width	At 90% or 10%	1	—	—	ns
$t_{LOCK}^2$	PLL Lock-in Time	Without external capacitor	—	—	150	$\mu$ s
		With external capacitor <sup>5</sup>	—	—	500	$\mu$ s
$t_{PA}$	Programmable Delay Unit		85	130	360	ps
$t_{IPJIT}$	Input Clock Period Jitter		—	—	$\pm 200$	ps
$t_{FBKDLY}$	External Feedback Delay		—	—	10	ns
$t_{HI}$	Input Clock High Time	90% to 90%	0.5	—	—	ns
$t_{LO}$	Input Clock Low Time	10% to 10%	0.5	—	—	ns
$t_{RST}$	RST Pulse Width (RESETM/RESETK)		15	—	—	ns
	Reset Signal Pulse Width (CNTRST)	Without external capacitor	500	—	—	ns
		With external capacitor <sup>5</sup>	20	—	—	$\mu$ s

1. Jitter sample is taken over 10,000 samples of the primary PLL output with clean reference clock.

2. Output clock is valid after  $t_{LOCK}$  for PLL reset and dynamic delay adjustment.

3. Using LVDS output buffers.

4. Relative to CLKOP.

5. Value of external capacitor: 5.6 nF  $\pm 20\%$ , NPO dielectric, ceramic chip capacitor, 1206 or smaller package.

6. Operation to 2MHz is possible with a limited output frequency range. Please contact Lattice Semiconductor.

Timing v.0.07

**sysCLOCK SPLL Timing****Over Recommended Operating Conditions**

Parameter	Description	Conditions	Min.	Typ.	Max.	Units
$f_{IN}$	Input Clock Frequency (CLKI, CLKFB)	Without external capacitor	33	—	420	MHz
		With external capacitor <sup>5, 6</sup>	3	—	50	MHz
$f_{OUT}$	Output Clock Frequency (CLKOP, CLKOS)	Without external capacitor	33	—	420	MHz
		With external capacitor <sup>5</sup>	5	—	50	MHz
$f_{OUT2}$	K-Divider Output Frequency (CLKOK)	Without external capacitor	0.258	—	210	MHz
		With external capacitor <sup>5</sup>	0.039	—	25	MHz
$f_{VCO}$	PLL VCO Frequency		640	—	1280	MHz
$f_{PFD}$	Phase Detector Input Frequency	Without external capacitor	33	—	—	MHz
		With external capacitor <sup>6</sup>	3	—	—	MHz

**AC Characteristics**

$t_{DT}$	Output Clock Duty Cycle	Default Duty Cycle Selected <sup>3</sup>	45	50	55	%
$t_{PH}^4$	Output Phase Accuracy		—	—	$\pm 0.05$	UI
$t_{OPJIT}^1$	Output Clock Period Jitter	$f_{OUT} \geq 100$ MHz	—	—	$\pm 125$	ps
		$50 \leq f_{OUT} < 100$ MHz	—	—	0.025	UIPP
		$f_{OUT} < 50$ MHz	—	—	0.04	UIPP
$t_{SK}$	Input Clock to Output Clock Skew	Divider Ratio = Integer	—	—	$\pm 250$	ps
$t_W$	Output Clock Pulse Width	At 90% or 10%	1	—	—	ns
$t_{LOCK}^2$	PLL Lock-in Time	Without external capacitor	—	—	150	$\mu s$
		With external capacitor <sup>5</sup>	—	—	500	$\mu s$
$t_{IPJIT}$	Input Clock Period Jitter		—	—	$\pm 200$	ps
$t_{FBKDLY}$	External Feedback Delay		—	—	10	ns
$t_{HI}$	Input Clock High Time	90% to 90%	0.5	—	—	ns
$t_{LO}$	Input Clock Low Time	10% to 10%	0.5	—	—	ns
$t_{RST}$	RST Pulse Width (RESETM/RESETK)		15	—	—	ns
	Reset Signal Pulse Width (CNTRST)	Without external capacitor	500	—	—	ns
		With external capacitor <sup>5</sup>	20	—	—	$\mu s$

1. Jitter sample is taken over 10,000 samples of the primary PLL output with clean reference clock.

2. Output clock is valid after  $t_{LOCK}$  for PLL reset and dynamic delay adjustment.

3. Using LVDS output buffers.

4. Relative to CLKOP.

5. Value of external capacitor: 5.6 nF  $\pm 20\%$ , NPO dielectric, ceramic chip capacitor, 1206 or smaller package.

6. Operation to 2MHz is possible with a limited output frequency range. Please contact Lattice Semiconductor.

Timing v.0.07

## DLL Timing

### Over Recommended Operating Conditions

Parameter	Description	Conditions	Min.	Typ.	Max.	Units
$f_{REF}$	Input reference clock frequency (on-chip or off-chip)		100	—	500	MHz
$f_{FB}$	Feedback clock frequency (on-chip or off-chip)		100	—	500	MHz
$f_{CLKOP}^1$	Output clock frequency, CLKOP		100	—	500	MHz
$f_{CLKOS}^2$	Output clock frequency, CLKOS		25	—	500	MHz
$t_{PJIT}$	Output clock period jitter (clean input)			—	250	ps p-p
$t_{CYJIT}$	Output clock cycle to cycle jitter (clean input)				250	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)		35		65	%
$t_{DUTYTRD}$	Output clock duty cycle (at 50% levels, arbitrary duty cycle input clock, 50% duty cycle circuit enabled, time reference delay mode)		40		60	%
$t_{DUTYCIR}$	Output clock duty cycle (at 50% levels, arbitrary duty cycle input clock, 50% duty cycle circuit enabled, clock injection removal mode)		40		60	%
$t_{SKEW}^3$	Output clock to clock skew between two outputs with the same phase setting		—	—	100	ps
$t_{PWH}$	Input clock minimum pulse width high (at 80% level)		750	—	—	ps
$t_{PWL}$	Input clock minimum pulse width low (at 20% level)		750	—	—	ps
$t_R, t_F$	Input clock rise and fall time (20% to 80% levels)		—	—	1	ps
$t_{INSTB}$	Input clock period jitter		—	—	+/-250	V/ns
$t_{LOCK}$	DLL lock time		18,500	—	—	cycles
$t_{RSWD}$	Digital reset minimum pulse width (at 80% level)		3	—	—	ns
$t_{PA}$	Delay step size		16.5	42	59.4	ps
$t_{RANGE}^1$	Max. delay setting for single delay block (144 taps)		2.376	6	8.553	ns
$t_{RANGE}^4$	Max. delay setting for four chained delay blocks		9.504	24	34.214	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.

Timing v.0.07

**SERDES High Speed Data Transmitter<sup>1</sup> (LatticeECP2M Family Only)****Table 3-7. Serial Output Timing and Levels**

Symbol	Description	Min.	Typ.	Max.	Units
V <sub>TX-DIFF-P-P-0</sub>	Differential swing (default setting) <sup>1,2</sup>		1.25		V, p-p
V <sub>TX-DIFF-P-P-1</sub>	Differential swing (+10% setting) <sup>1,2</sup>		1.4		V, p-p
V <sub>TX-DIFF-P-P-4</sub>	Differential swing (-30% setting) <sup>1,2</sup>		1.0		V, p-p
V <sub>TX-DIFF-P-P-7</sub>	Differential swing (-10% setting) <sup>1,2</sup>		1.2		V, p-p
V <sub>OCM</sub>	Output common mode voltage		0.8		V
T <sub>TX-R</sub>	Rise time (20% to 80%) <sup>3</sup>		70		ps
T <sub>TX-F</sub>	Fall time (80% to 20%) <sup>3</sup>		70		ps
Z <sub>TX-OI-SE</sub>	Output Impedance (single ended)		50		Ohms
R <sub>LTX-RL</sub>	Return loss (with package)		9		dB

1. Four transmitter pre-emphasis settings are provided: 0%, ±10%, and 30%.

2. All measurements are with 50 ohm impedance.

3. PCIe compliance pattern used.

4. All data collected at 2.5Gbps.

**Table 3-8. Output Jitter<sup>1</sup>**

	Description	Min.	Typ.	Max.	Units
3.125Gbps	Deterministic	—	0.15	—	UI, p-p
	Random	—	0.22	—	UI, p-p
	Total	—	0.33	—	UI, p-p
2.5Gbps <sup>2</sup>	Deterministic	—	0.08	—	UI, p-p
	Random	—	0.17	—	UI, p-p
	Total	—	0.24	—	UI, p-p
1.25 Gbps	Deterministic	—	0.05	—	UI, p-p
	Random	—	0.1	—	UI, p-p
	Total	—	0.15	—	UI, p-p
250 Mbps	Deterministic	—	0.05	—	UI, p-p
	Random	—	0.12	—	UI, p-p
	Total	—	0.15	—	UI, p-p

1. Values are measured with PRBS 2<sup>7</sup>-1, all channels operating.

2. 2.5Gbps is separately tested and passed per PCIe v.1.0a requirement.

**SERDES High Speed Data Receiver (LatticeECP2M Family Only)****Table 3-9. Serial Input Data Specifications**

Symbol	Description	Min.	Typ.	Max.	Units
RX-CIDs	Stream of nontransitions <sup>1</sup> (CID = Consecutive Identical Digits) @ 10 <sup>-12</sup> BER		7 @ 3.125 Gbps 20 @ 1.25 Gbps		Bits
V <sub>RX-DIFF-S</sub>	Differential input sensitivity	100	—	—	mV, p-p
V <sub>RX-IN</sub>	Input levels	0	—	V <sub>CCRX</sub> + 0.3	V
V <sub>RX-CM-DC</sub>	Input common mode range (DC coupled)	0.5	—	1.2	V
V <sub>RX-CM-AC</sub>	Input common mode range (AC coupled) <sup>1</sup>	0		1.5	V
T <sub>RX-RELOCK</sub>	CDR re-lock time <sup>2</sup>	—		—	Bits
Z <sub>RX-TERM</sub>	Input termination 50 Ohm	—	50		Ohms
	Input termination 2K Ohm	—	2K	—	Ohms
RL <sub>RX-RL</sub>	Return loss (without package) <sup>3</sup>	—	9	—	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 8b/10b encoded data

3. AC coupling is used to interface to LVPECL and LVDS.

**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 recently modified specifications to indicate tolerance levels for different jitter types as they relate to specific protocols (e.g. FC, etc.). Sinusoidal jitter is considered to be a worst case jitter type. Table 3-10 shows receiver specifications with sinusoidal jitter injection.

**Table 3-10. Receiver Sinusoidal Jitter Tolerance Specification**

Symbol	Input Data	Conditions	Typ.	Units
J <sub>RX-SJT-TYP-31</sub>	Jitter tolerance @ 3.125 Gbps, typical	175mV differential eye	0.20	UI, p-p
J <sub>RX-SJT-TYP-25</sub>	Jitter tolerance @ 2.5 Gbps, typical	175mV differential eye	0.20	UI, p-p
J <sub>RX-SJT-TYP-12</sub>	Jitter tolerance @ 1.25 Gbps, typical	175mV differential eye	0.25	UI, p-p
J <sub>RX-SJT-TYP-02</sub>	Jitter tolerance @ 270 Mbps, typical	175mV differential eye	TBD	UI, p-p

Note: With PRBS 2<sup>7</sup>-1 data pattern, all channels operating, REFCLK jitter of 30ps, Tj=0°C to 100°C, 1.14 to 1.26V supply, 10MHz sinusoidal jitter injection.

The Clock and Data Recovery (CDR) portion of the SERDES receiver has the ability to filter incoming signal jitter that is below the clock recovery PLL bandwidth (about 3 MHz). The eye mask specifications of Figure 3-13 are for jitter frequencies above the PLL bandwidth of the CDR, which is a worst case condition. When jitter occurs at frequencies below the PLL bandwidth, the receiver jitter tolerance is significantly better. For this case, error-free data detection can occur even with a completely closed eye mask.

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

Input Data	Conditions	Typ.	Units
Jitter tolerance at 3.125 Gbps typical	600 mV differential eye	0.75	UI, p-p

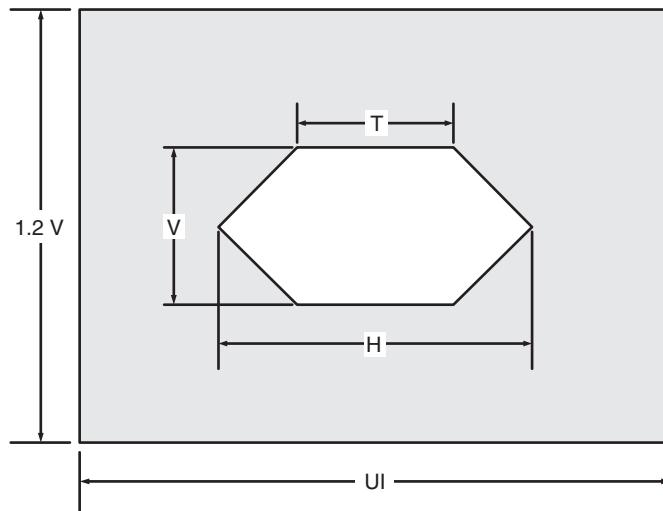
Note: With PRBS 2<sup>7</sup>-1 data pattern, all channels operating, REFCLK jitter of 30ps, Tj=0°C to 100°C, 1.14 to 1.26V supply, 10MHz sinusoidal jitter injection (0.40 UI, DDJ, 0.20 UI RJ, 0.15 UI PJ at 10 MHz).

**Input Eye Mask Characterization**

Figure 3-13 provides an eye mask characterization of the SERDES receiver input. The eye mask is specified below for two different eye mask heights. It provides guidance on a number of input parameters, including signal amplitude and rise time limits, noise and jitter limits, and P and N input skew tolerance. Almost all detrimental character-

istics of transmit signal and the interconnection link design result in eye closure. This, combined with the eye opening limitations of the line receiver, can provide a good indication of a link's ability to transfer data error-free.

**Figure 3-13. Receive Data Eye Diagram Template (Differential)**



**Table 3-12. Receiver Eye Mask Specifications**

Symbol	Input Data	Conditions	Typ.	Units
T <sub>RX-EYE-H-175-31</sub>	Eye opening width (H) @ 3.125 Gbps, with equalization	V = 175 mV p-p differential	0.30	UI, p-p
T <sub>RX-EYE-T-175-31</sub>	Eye opening width (T) @ 3.125 Gbps, with equalization	V = 175 mV p-p differential	0.05	UI, p-p
T <sub>RX-EYE-H-175-25</sub>	Eye opening width (H) @ 2.5 Gbps, with equalization	V = 175 mV p-p differential	0.30	UI, p-p
T <sub>RX-EYE-T-175-25</sub>	Eye opening width (T) @ 2.5 Gbps, with equalization	V = 175 mV p-p differential	0.05	UI, p-p
T <sub>RX-EYE-H-175-12</sub>	Eye opening width (H) @ 1.25 Gbps, with equalization	V = 175 mV p-p differential	0.30	UI, p-p
T <sub>RX-EYE-T-175-12</sub>	Eye opening width (T) @ 1.25 Gbps, with equalization	V = 175 mV p-p differential	0.05	UI, p-p
T <sub>RX-EYE-H-175-02</sub>	Eye opening width (H) @ 270 Mbps, with equalization	V = 175 mV p-p differential		UI, p-p
T <sub>RX-EYE-T-175-02</sub>	Eye opening width (T) @ 270 Mbps, with equalization	V = 175 mV p-p differential		UI, p-p

Note: With PRBS 2<sup>7</sup>-1 data pattern, all channels operating, REFCLK jitter of 30ps, Tj= 0C to 100C, 1.14 to 1.26V supply, 10<sup>-12</sup> maximum bit error rate.

**SERDES External Reference Clock (LatticeECP2M Family Only)**

The external reference clock selection and its interface are a critical part of system applications for this product. Table 3-13 specifies reference clock requirements, over the full range of operating conditions.

**Table 3-13. External Reference Clock Specification ( $refclkp/refclkn$ )**

Symbol	Description	Min.	Typ.	Max.	Units
$F_{REF}$	Frequency range	50	—	320	MHz
$F_{REF-PPM}$	Frequency tolerance	-300	—	300	ppm
$J_{REF-RJ}$	Random jitter <sup>1</sup>	—	30	50	ps, p-p
$V_{REF-IN-DIFF}$	Input swing, differential clock	200	0	2400	mV, p-p differential
$V_{REF-IN-SE}$	Input swing, single-ended clock <sup>2</sup>	100	—	1200	mV, p-p
$V_{REF-IN}$	Input levels	0	—	$V_{CCRX} + 0.3$	V
$V_{REF-CM-DC}$	Input common mode range (DC coupled)	0.5	—	1.2	V
$V_{REF-CM-AC}$	Input common mode range (AC coupled)	0	—	1.5	V
$D_{REF}$	Duty cycle <sup>4</sup>	40	—	60	%
$T_{REF-R}$	Rise time (20% to 80%)		500	1000	ps
$T_{REF-F}$	Fall time (80% to 20%)		500	1000	ps
$Z_{REF-IN-TERM}$	Input termination		50/2K		Ohms
$C_{REF-IN-CAP}$	Input capacitance <sup>5</sup>	—	—	1.5	pF

Notes:

1. Recommended not to exceed 50 ps jitter. Reference clock jitter will add directly to TX jitter generation performance.
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. When AC coupled, the input common mode range is determined by:  
 $(\text{Min input level}) + (\text{Peak-to-peak input swing})/2 \leq (\text{Input common mode voltage}) \leq (\text{Max input level}) - (\text{Peak-to-peak input swing})/2$
4. Measured at 50% amplitude.
5. Input capacitance of 1.5pF is total capacitance, including both device and package.

**RESET Pulse Specification (LatticeECP2M Family Only)****Table 3-14. RESET Pulse Specification**

Symbol	Description	Min.	Units
$t_{PWRONRST}^1$	Power-on-reset high time, pwr_on_RST = 1		μs
$t_{MACRORST}^1$	SERDES reset high time, serdes_RST = 1		μs
$t_{RRST}^2$	Receive channel reset high time, rx_RST_[0-3] = 1		μs
$t_{TRST}^2$	Transmit channel reset high time, tx_RST_[0-3] = 1		μs

1. The power-on reset and SERDES reset signals must be asserted long enough to ensure that the analog circuits are in the correct state prior to deasserting the reset signal.
2. The rx\_RST and tx\_RST signals reset digital logic only and therefore only have to be asserted for a short period of time.

**Power-Down/Power-Up Specification****Table 3-15. Power-Down and Power-Up Specification**

Symbol	Description	Max.	Units
$t_{PWRDN}$	Power-down time after all power down register bits set to '0'	10	μs
$t_{PWRUP}$	Power-up time after all power down register bits set to '1'	5	ms

**LatticeECP2/M sysCONFIG Port Timing Specifications**

Over Recommended Operating Conditions

Parameter	Description	Min.	Max.	Units
<b>sysCONFIG Byte Data Flow</b>				
$t_{SUCBDI}$	Byte D[0:7] Setup Time to CCLK	7	—	ns
$t_{HCBDI}$	Byte D[0:7] Hold Time to CCLK	1	—	ns
$t_{CODO}$	CCLK to DOUT in Flowthrough Mode	—	12	ns
$t_{SUCS}$	CSN[0:1] Setup Time to CCLK	7	—	ns
$t_{HCS}$	CSN[0:1] Hold Time to CCLK	1	—	ns
$t_{SUWD}$	Write Signal Setup Time to CCLK	7	—	ns
$t_{HWD}$	Write Signal Hold Time to CCLK	1	—	ns
$t_{DCB}$	CCLK to BUSY Delay Time	—	12	ns
$t_{CORD}$	CCLK to Out for Read Data	—	12	ns
<b>sysCONFIG Byte Slave Clocking</b>				
$t_{BSCH}$	Byte Slave CCLK Minimum High Pulse	6	—	ns
$t_{BSCL}$	Byte Slave CCLK Minimum Low Pulse	9	—	ns
$t_{BSCYC}$	Byte Slave CCLK Cycle Time	15	—	ns
<b>sysCONFIG Serial (Bit) Data Flow</b>				
$t_{SUSCDI}$	DI Setup Time to CCLK Slave Mode	7	—	ns
$t_{HSCDI}$	DI Hold Time to CCLK Slave Mode	1	—	ns
$t_{CODO}$	CCLK to DOUT in Flowthrough Mode	—	12	ns
$t_{SUMCDI}$	DI Setup Time to CCLK Master Mode	7	—	ns
$t_{HMCDI}$	DI Hold Time to CCLK Master Mode	1	—	ns
<b>sysCONFIG Serial Slave Clocking</b>				
$t_{SSCH}$	Serial Slave CCLK Minimum High Pulse	6	—	ns
$t_{SSCL}$	Serial Slave CCLK Minimum Low Pulse	6	—	ns
<b>sysCONFIG POR, Initialization and Wake-up</b>				
$t_{ICFG}$	Minimum Vcc to INITN High	—	50	ms
$t_{VMC}$	Time from $t_{ICFG}$ to Valid Master CCLK	—	2	us
$t_{PRGMRJ}$	PROGRAMN Pin Pulse Rejection	—	8	ns
$t_{PRGM}$	PROGRAMN Low Time to Start Configuration	25	—	ns
$t_{DINIT}$	PROGRAMN High to INITN High Delay	—	1	ms
$t_{DPPINIT}$	Delay Time from PROGRAMN Low to INITN Low	—	37	ns
$t_{DPPDONE}$	Delay Time from PROGRAMN Low to DONE Low	—	37	ns
$t_{IODISS}$	User I/O Disable from PROGRAMN Low	—	35	ns
$t_{IOENSS}$	User I/O Enabled Time from CCLK Edge During Wake-up Sequence	—	25	ns
$t_{MWC}$	Additional Wake Master Clock Signals after DONE Pin High	120	—	cycles
<b>sysCONFIG SPI Port</b>				
$t_{CFGX}$	INITN High to CCLK Low	—	1	μs
$t_{CSSPI}$	INITN High to CSSPIN Low	—	2	us
$t_{CSCCLK}$	CCLK Low before CSSPIN Low	0	—	ns
$t_{SOCDO}$	CCLK Low to Output Valid	—	15	ns
$t_{SOE}$	CSSPIN[0:1] Active Setup Time	300	—	ns
$t_{CSPID}$	CSSPIN[0:1] Low to First CCLK Edge Setup Time	300+3cyc	600+6cyc	ns

**LatticeECP2/M sysCONFIG Port Timing Specifications (Continued)**

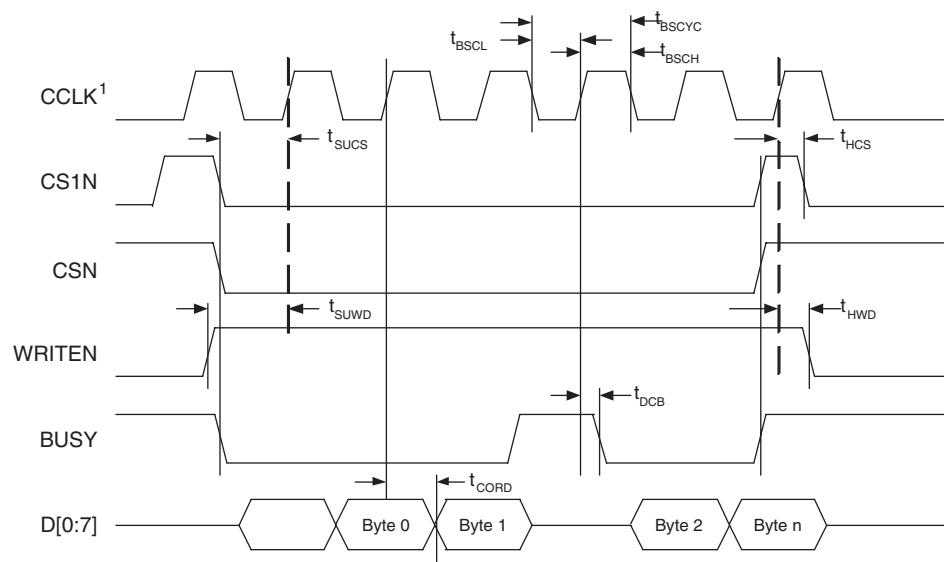
Over Recommended Operating Conditions

Parameter	Description	Min.	Max.	Units
$f_{MAXSPI}$	Max. CCLK Frequency - SPI Flash Read Opcode (0x03) (SPIFASTN = 1)	—	20	MHz
	Max. CCLK Frequency - SPI Flash Fast Read Opcode (0x0B) (SPIFASTN = 0)	—	50	MHz
$t_{SUSPI}$	SOSPI Data Setup Time Before CCLK	7	—	ns
$t_{HSPSI}$	SOSPI Data Hold Time After CCLK	2	—	ns

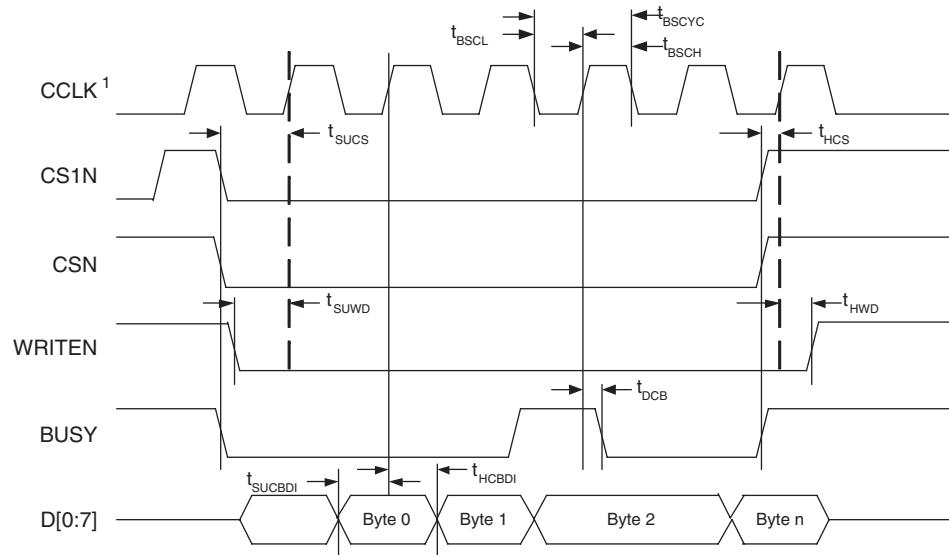
Timing v.0.07

Parameter	Min.	Max.	Units
Master Clock Frequency	Selected value - 30%	Selected value + 30%	MHz
Master Clock Period Jitter			UIPP
Duty Cycle	40	60	%

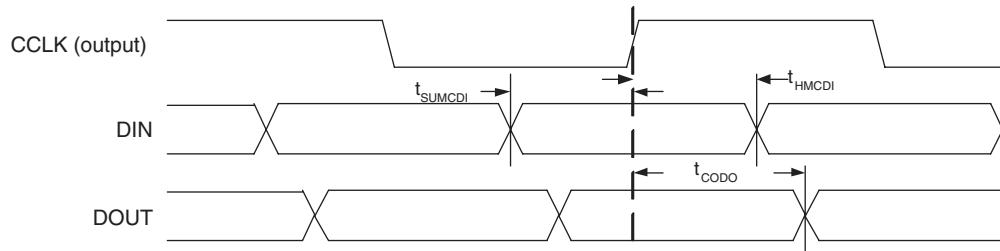
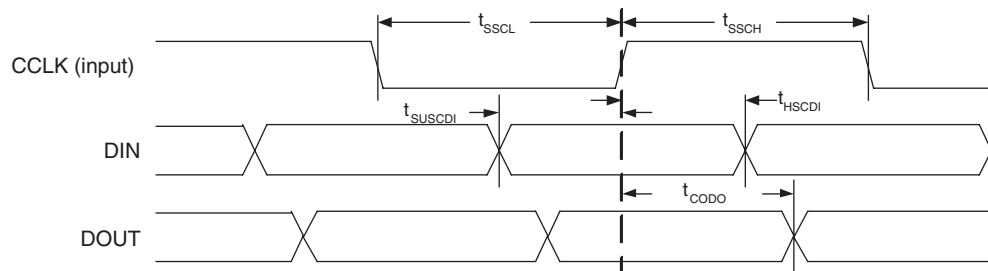
Timing v.0.07

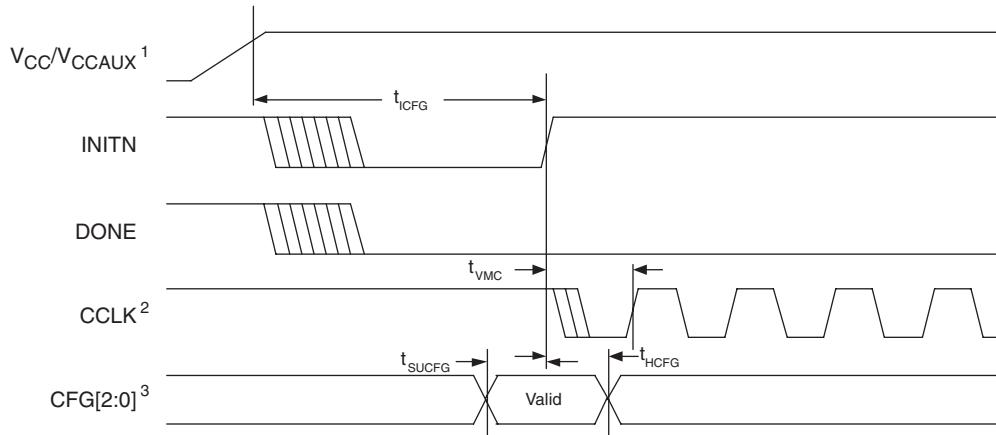
**Figure 3-14. sysCONFIG Parallel Port Read Cycle**

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

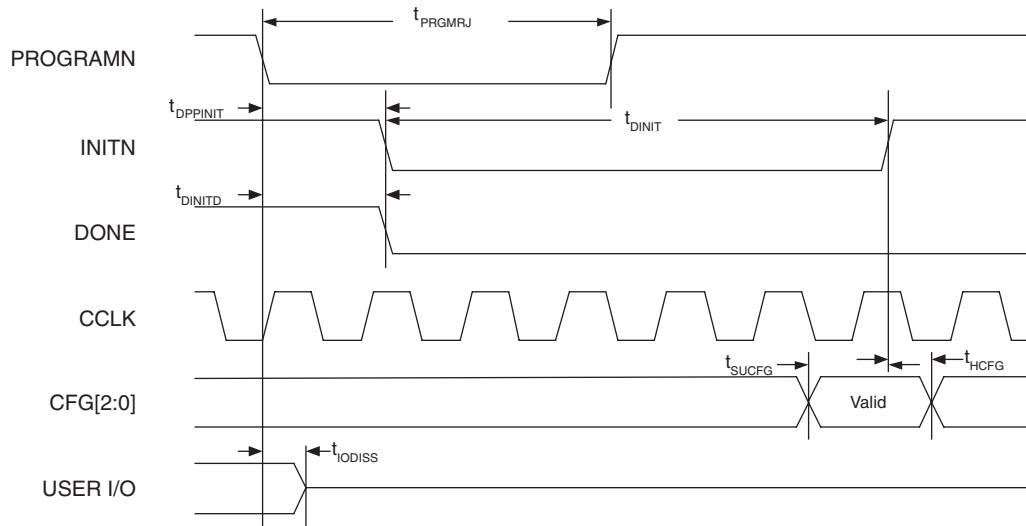
**Figure 3-15. sysCONFIG Parallel Port Write Cycle**

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

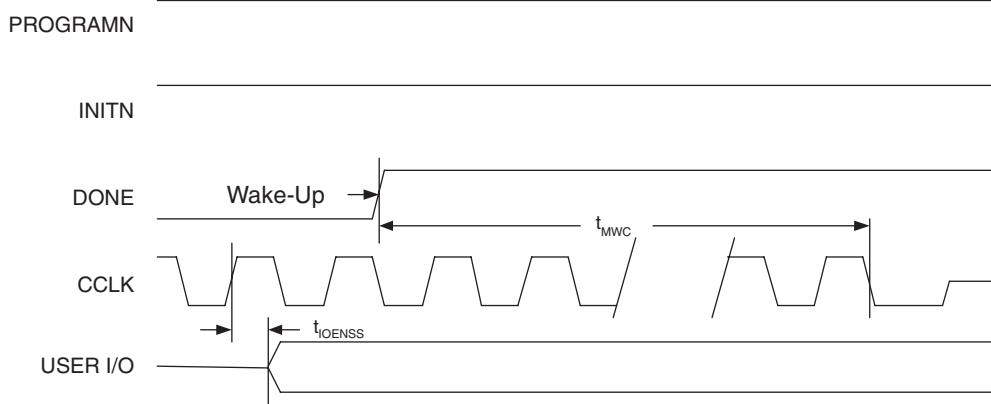
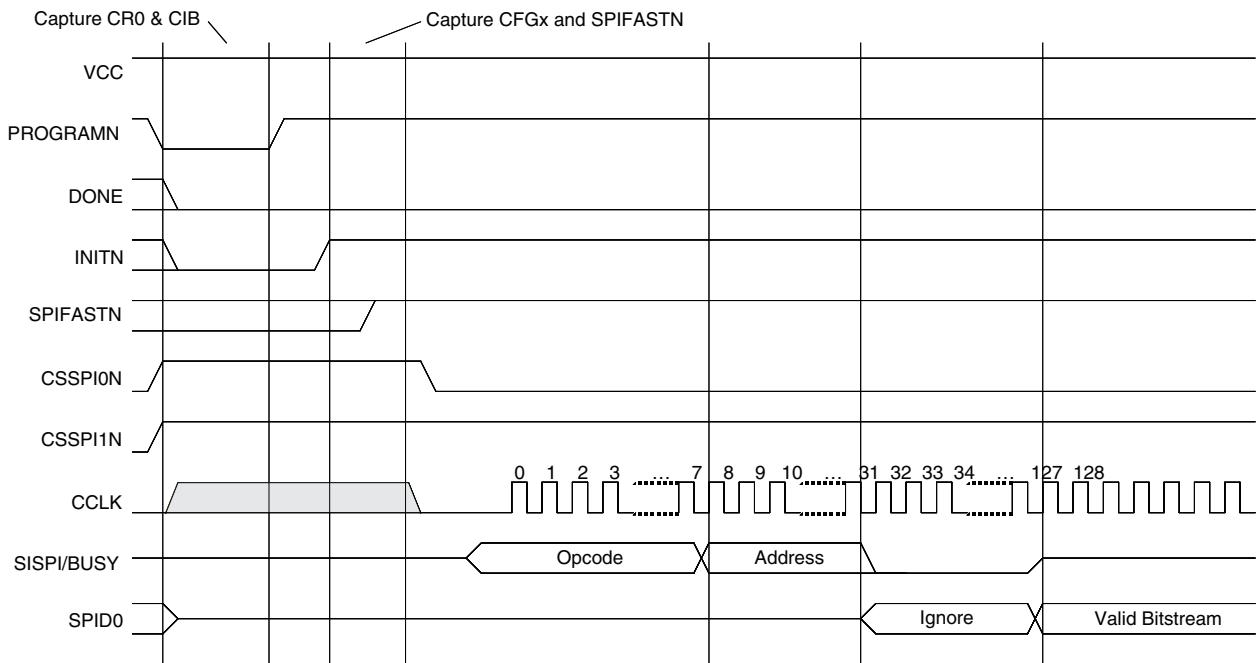
**Figure 3-16. sysCONFIG Master Serial Port Timing****Figure 3-17. sysCONFIG Slave Serial Port Timing**

**Figure 3-18. Power-On-Reset (POR) Timing**

1. Time taken from  $V_{CC}$  or  $V_{CCAUX}$ , whichever is the last to reach its  $V_{MIN}$ .
2. Device is in a Master Mode.
3. The CFG pins are normally static (hard wired).

**Figure 3-19. Configuration from PROGRAMN Timing**

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

**Figure 3-20. Wake-Up Timing****Figure 3-21. SPI/SPI<sub>M</sub> Configuration Waveforms**

**JTAG Port Timing Specifications**

Over Recommended Operating Conditions

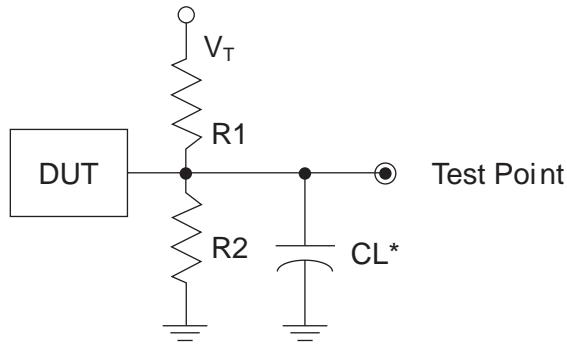
Symbol	Parameter	Min	Max	Units
$f_{MAX}$	TCK clock frequency	—	25	MHz
$t_{BTCP}$	TCK [BSCAN] clock pulse width	40	—	ns
$t_{BTCPH}$	TCK [BSCAN] clock pulse width high	20	—	ns
$t_{BTCPL}$	TCK [BSCAN] clock pulse width low	20	—	ns
$t_{BTS}$	TCK [BSCAN] setup time	8	—	ns
$t_{BTH}$	TCK [BSCAN] hold time	10	—	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
$t_{TCOEN}$	TAP controller falling edge of clock to valid enable	—	10	ns
$t_{TCRS}$	BSCAN test capture register setup time	8	—	ns
$t_{TCRH}$	BSCAN test capture register hold time	25	—	ns
$t_{BUTCO}$	BSCAN test update register, falling edge of clock to valid output	—	25	ns
$t_{BUDIS}$	BSCAN test update register, falling edge of clock to valid disable	—	25	ns
$t_{BUPOEN}$	BSCAN test update register, falling edge of clock to valid enable	—	25	ns

Timing v.0.07

## Switching Test Conditions

Figure 3-22 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 3-16.

**Figure 3-22. Output Test Load, LVTTL and LVCMOS Standards**



\*CL Includes Test Fixture and Probe Capacitance

**Table 3-16. 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>
LVTTL and other LVCMOS settings (L → H, H → L)	$\infty$	$\infty$	0pF	LVCMOS 3.3 = V <sub>CCIO</sub> /2	—
				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)	$\infty$	1MΩ		V <sub>CCIO</sub> /2	—
LVCMOS 2.5 I/O (Z → L)	1MΩ	$\infty$		V <sub>CCIO</sub> /2	V <sub>CCIO</sub>
LVCMOS 2.5 I/O (H → Z)	$\infty$	100		V <sub>OH</sub> - 0.10	—
LVCMOS 2.5 I/O (L → Z)	100	$\infty$		V <sub>OL</sub> + 0.10	V <sub>CCIO</sub>

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

September 2006

Advance Data Sheet DS1007

### Signal Descriptions

Signal Name <sup>1, 2, 3</sup>	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 Row Number. When Edge is L (Left) or R (Right), only need to specify Column 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>
GSRN	I	Global RESET signal (active low). Any I/O pin can be GSRN.
NC	—	No connect.
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>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.
XRES	—	10K ohm +/-1% resistor must be connected between this pad and ground.
<b>PLL, DLL and Clock Functions</b> (Used as user programmable I/O pins when not in use for PLL or clock pins)		
[LOC][num]_V <sub>CCPLL</sub>	—	Power supply pin for PLL: ULM, LLM, URM, LRM, num = row from center.
[LOC][num]_GPLL[T, C]_IN_A	I	General Purpose PLL (GPLL) input pads: ULM, LLM, URM, LRM, num = row from center, T = true and C = complement, index A,B,C...at each side.
[LOC][num]_GPLL[T, C]_FB_A	I	Optional feedback GPLL input pads: ULM, LLM, URM, LRM, num = row from center, T = true and C = complement, index A,B,C...at each side.
[LOC][num]_SPLL[T, C]_IN_A	I	Secondary PLL (SPLL) input pads: ULM, LLM, URM, LRM, num = row from center, T = true and C = complement, index A,B,C...at each side.
[LOC][num]_SPLL[T, C]_FB_A	I	Optional feedback (SPLL) input pads: ULM, LLM, URM, LRM, num = row from center, T = true and C = complement, index A,B,C...at each side.
[LOC][num]_DLL[T, C]_IN_A	I	DLL input pads: ULM, LLM, URM, LRM, num = row from center, T = true and C = complement, index A,B,C...at each side.
[LOC][num]_DLL[T, C]_FB_A	I	Optional feedback (DLL) input pads: ULM, LLM, URM, LRM, num = row from center, T = true and C = complement, index A,B,C...at each side.
PCLK[T, C]_[n:0]_[3:0]	I	Primary Clock pads, T = true and C = complement, n per side, indexed by bank and 0,1,2,3 within bank.
[LOC]DQS[num]	I	DQS input pads: T (Top), R (Right), B (Bottom), L (Left), DQS, num = ball function number. Any pad can be configured to be output.

**Signal Descriptions (Cont.)**

Signal Name <sup>1, 2, 3</sup>	I/O	Description
<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. This is a dedicated pin.
DONE	I/O	Open Drain pin. Indicates that the configuration sequence is complete, and the startup sequence is in progress. This is a dedicated pin.
CCLK	I/O	Configuration Clock for configuring an FPGA in sysCONFIG mode.
BUSY/SISPI	I/O	Read control command in SPI3 or SPIX mode.
CSN	I	sysCONFIG chip select (active low). During configuration, a pull-up is enabled.
CS1N	I	sysCONFIG chip select (active low). During configuration, a pull-up is enabled.
WRITEN	I	Write Data on Parallel port (active low).
D[7:0]/SPID[0:7]	I/O	sysCONFIG Port Data I/O.
DOUT/CS0N (for LatticeECP2 only) DOUT/CS0N/CSSPI1N (for LatticeECP2M)	O	Output for serial configuration data (rising edge of CCLK) when using sysCONFIG port. CSSPI1N is used in SPIm Mode only.
DI/CSSPON	I/O	Input for serial configuration data (clocked with CCLK) when using sysCONFIG port. During configuration, a pull-up is enabled. Output when used in SPI/SPIm modes.
<b>Dedicated SERDES Signals</b>		
[LOC]_SQ_VCCAUX33	—	Termination resistor switching power (3.3V)
[LOC]_SQ_REFCLKN	I	Negative Reference Clock Input
[LOC]_SQ_REFCLKP	I	Positive Reference Clock Input
[LOC]_SQ_VCCP	—	PLL and Reference clock buffer power (1.2V)
[LOC]_SQ_VCCIBm	—	Input buffer power supply, channel m (1.2V/1.5V)
[LOC]_SQ_VCCOBm	—	Output buffer power supply, channel m (1.2V/1.5V)
[LOC]_SQ_HDOUTNm	O	High-speed output, negative channel m
[LOC]_SQ_HDOUTPm	O	High-speed output, positive channel m
[LOC]_SQ_HDINNm	I	High-speed input, negative channel m
[LOC]_SQ_HDINPm	I	High-speed input, positive channel m
[LOC]_SQ_VCCTXm	—	Transmitter power supply, channel m (1.2V)

## Signal Descriptions (Cont.)

Signal Name <sup>1, 2, 3</sup>	I/O	Description
[LOC]_SQ_VCCRXm	—	Receiver power supply, channel m (1.2V)

1. These signals are relevant for LatticeECP2M family.
2. m defines the associated channel in the Quad.
3. These signals are defined in Quads [LOC] indicates the corner SERDES Quad is located: ULC (upper left), URC (upper right), LLC (lower left), LRC (lower right).

**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-4]	A	DQ
	B	DQ
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]DQS <sub>n</sub>
	B	DQ
P[Edge] [n+1]	A	DQ
	B	DQ
P[Edge] [n+2]	A	DQ
	B	DQ
P[Edge] [n+3]	A	DQ
	B	DQ
<b>For Bottom Edge of the Device</b>		
P[Edge] [n-4]	A	DQ
	B	DQ
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]DQS <sub>n</sub>
	B	DQ
P[Edge] [n+1]	A	DQ
	B	DQ
P[Edge] [n+2]	A	DQ
	B	DQ
P[Edge] [n+3]	A	DQ
	B	DQ
P[Edge] [n+4]	A	DQ
	B	DQ

## Notes:

1. "n" is a row PIC number.
2. The DDR interface is designed for memories that support one DQS strobe up to 15 bits of data for the left and right edges and up to 17 bits of data for the bottom edge. In some packages, all the potential DDR data (DQ) pins may not be available. PIC numbering definitions are provided in the "Signal Names" column of the Signal Descriptions table.

**LatticeECP2 Pin Information Summary**

Pin Type		LFE2-12E				LFE2-50E	
		144 TQFP	208 PQFP	256 fpBGA	484 fpBGA	484 fpBGA	672 fpBGA
Single Ended User I/O		93	131	193	297	339	500
Differential Pair User I/O		42	62	96	148	169	249
Configuration	TAP Pins	5	5	5	5	5	5
	Muxed Pins	14	14	14	14	14	14
	Dedicated Pins (Non TAP)	7	7	7	7	7	7
Non Configuration	Muxed Pins	33	40	54	57	68	79
	Dedicated Pins	3	3	3	3	3	3
VCC		8	12	6	16	16	20
VCCAUX		4	8	4	16	16	16
VCCIO	Bank0	1	2	2	4	4	5
	Bank1	1	2	2	4	4	5
	Bank2	1	2	2	4	4	5
	Bank3	1	2	2	4	4	5
	Bank4	1	2	2	4	4	5
	Bank5	1	2	2	4	4	5
	Bank6	1	2	2	4	4	5
	Bank7	1	2	2	4	4	5
	Bank8	1	2	1	2	2	2
GND		12	22	20	60	61	72
NC		1	0	1	44	0	3
Single Ended/ Differential I/O per Bank (including emulated with resistors)	Bank0	8/4	18/9	18/9	50/25	50/25	67/33
	Bank1	18/6	18/9	34/17	46/23	46/23	66/33
	Bank2	4/2	11/5	20/10	24/12	38/19	56/28
	Bank3	8/4	11/5	12/6	16/8	22/11	48/24
	Bank4	18/9	19/9	32/16	46/23	46/23	62/31
	Bank5	10/5	18/9	17/8	46/23	46/23	68/34
	Bank6	9/4	18/8	26/13	32/16	40/20	64/32
	Bank7	12/6	12/6	20/10	23/11	37/18	55/27
	Bank8	6/2	6/2	14/7	14/7	14/7	14/7
True LVDS I/O per Bank	Bank0 (Top Edge)						
	Bank1 (Top Edge)						
	Bank2 (Right Edge)			5	6	9	13
	Bank3 (Right Edge)			3	4	5	12
	Bank4 (Bottom Edge)						
	Bank5 (Bottom Edge)						
	Bank6 (Left Edge)			7	8	10	16
	Bank7 (Left Edge)			5	5	8	12
	Bank8 (Right Edge)						

**LatticeECP2 Pin Information Summary (Cont.)**

Pin Type	LFE2-12E				LFE2-50E	
	144 TQFP	208 PQFP	256 fpBGA	484 fpBGA	484 fpBGA	672 fpBGA
Available DDR-Interfaces per I/O Bank <sup>1</sup>	Bank0					
	Bank1					
	Bank2		1	1	2	3
	Bank3					3
	Bank4		2	3	3	4
	Bank5		1	3	3	4
	Bank6		1	1	1	4
	Bank7		1	1	2	3
	Bank8					
PCI Capable I/Os per Bank	Bank0					
	Bank1					
	Bank2					
	Bank3					
	Bank4		32	46	46	62
	Bank5		14	46	46	68
	Bank6					
	Bank7					
	Bank8					

1. Minimum requirement to implement a fully functional 8-bit wide DDR bus. Available DDR interface consists of at least 12 I/Os (1 DQS + 1 DQSB + 8 DQs + 1 DM + Bank VREF1).

**LatticeECP2M Pin Information Summary**

Pin Type		ECP2M35	
		484 fpBGA	672 fpBGA
Single Ended User I/O		317	424
Differential Pair User I/O	Normal	124	162
Configuration	High speed	36	52
	Dedicated	5	5
	Muxed	13	13
	TAP	4	4
Dedicated Inputs (without supplies)		3	3
V <sub>CC</sub>		16	29
V <sub>CCAUX</sub>		8	17
V <sub>CCPLL</sub>		4	8
VCCIO	Bank0	3	5
	Bank1	4	4
	Bank2	4	5
	Bank3	4	5
	Bank4	4	4
	Bank5	4	5
	Bank6	4	5
	Bank7	4	5
GND, GND0 to GND7		57	80
NC		12	37
Single Ended / Differential I/O per Bank	Bank0	36/18	63/31
	Bank1	18/18	18/18
	Bank2	30/15	50/25
	Bank3	36/18	43/21
	Bank4	62/31	50/25
	Bank5	28/14	60/30
	Bank6	39/19	52/26
	Bank7	40/20	60/30
V <sub>CCJ</sub>		1	1

**Available Device Resources by Package**

Resource	Device	256 fpBGA	484 fpBGA	672 fpBGA
<b>LatticeECP2</b>				
PLL/DLL	ECP2-12E	4	4	—
	ECP2-50	—	6	6
<b>LatticeECP2M</b>				
PLL/DLL	ECP2M-35	—	10	10

**LatticeECP2 Power Supply and NC**

Signals	144 TQFP	208 PQFP	256 fpBGA	484 fpBGA	672 fpBGA
VCC	16, 29, 48, 54, 83, 102, 128, 135	12, 19, 40, 74, 80, 97, 116, 140, 146, 171, 188, 198	G7, G9, H7, J10, K10, K8	J10, J11, J12, J13, K14, K9, L14, L9, M14, M9, N14, N9, P10, P11, P12, P13	M8, L20, L12, L13, L14, L15, M11, M12, M15, M16, N11, N16, P11, P16, R11, R12, R15, R16, T12, T13, T14, T15
VCCIO0	139	195, 206	C5, E7	G10, G9, H8, H9	D11, D6, G9, J12, K12
VCCIO1	117	162, 170	C12, E10	G11, G12, G13, G14	D16, D21, G18, J15, K15
VCCIO2	106	143, 148	E14, G12	H14, H15, J15, K16	F23, J20, L23, M17, M18
VCCIO3	89	123, 135	K12, M14	L16, M16, N16, P16	AA23, R17, R18, T23, V20
VCCIO4	64	93, 100	M10, P12	R14, T12, T13, T14	AC16, AC21, U15, V15, Y18
VCCIO5	42	55, 63	M7, P5	R9, T10, T11, T9	AC11, AC6, U12, V12, Y9
VCCIO6	31	38, 44	K5, M3	N7, P7, P8, R8	AA4, R10, R9, T4, V7
VCCIO7	9	10, 14	E3, G5	J8, K7, L7, M7	F4, J7, L4, M10, M9
VCCIO8	85	113, 118	T15	P15, R15	AE25, V18
VCCJ	35	51	K7	T8	AB5
VCCAUX	6, 39, 90, 142	7, 30, 70, 86, 125, 151, 174, 190	G8, H10, J7, K9	G5, K5, R5, V7, V11, V8, V13, V15, M17, P17, E17, G18, D11, F13, C5, E6	J10, J11, J16, J17, K18, L18, T18, U18, V16, V17, V10, V11, T9, U9, K9, L9
GND <sup>1</sup>	11, 21, 30, 47, 51, 61, 81, 95, 105, 120, 133, 138	5, 13, 17, 25, 32, 42, 60, 68, 77, 81, 89, 102, 115, 122, 139, 145, 159, 169, 175, 184, 192, 201	A1, A16, B12, B5, C8, E15, E2, H14, H8, H9, J3, J8, J9, M15, M2, P9, R12, R5, T1, T16	A22, AA19, AA4, AB1, AB22, B19, B4, C14, C9, D2, D21, F17, F6, H10, H11, H12, H13, J14, J20, J3, J9, K10, K11, K12, K13, K15, K8, L10, L11, L12, L13, L15, L8, M10, M11, M12, M13, M15, M8, N10, N11, N12, N13, N15, N8, P14, P20, P3, P9, R10, R11, R12, R13, U17, U6, W2, W21, Y14, Y9, A1	A2, A25, AA18, AA24, AA3, AA9, AD11, AD16, AD21, AD6, AE1, AE26, AF2, AF25, B1, B26, C11, C16, C21, C6, F18, F24, F3, F9, J13, F14, J21, J6, K10, K11, K13, K14, K16, K17, L10, L11, L16, L17, L24, L3, M13, M14, N10, N12, N13, N14, N15, N17, P10, P12, P13, P14, P15, P17, R13, R14, T10, T11, T16, T17, T24, T3, U10, U11, U13, U14, U16, U17, V13, V14, V21, V6
NC <sup>2</sup>	LFE2-12E: 127	None	LFE2-12E: None	LFE2-12E: E3, F3, F1, H4, F2, H5, G1, G3, G2, G4, K6, N1, M2, N2, M1, N3, N5, N4, P5, N19, M19, J22, L22, H22, K22, J16, D22, F21, E21, E22, H19, G20, G19, F20, C21, C22, H6, J6, H3, H2, H17, H16, H20, H18	

1. All grounds must be electrically connected at the board level. For fpBGA packages, the total number of GND balls is less than the actual number of GND logic connections from the die to the common package GND plane.
2. NC pins should not be connected to any active signals, VCC or GND.

**LatticeECP2M Power Supply and NC**

Signals	484 fpBGA	672 fpBGA
VCC	J10, J11, J12, J13, K14, K9, L14, L9, M14, M9, N14, N9, P10, P11, P12, P13	AD13, AD14, AD16, AD17, AD19, AD21, AD22, AD24, AD25, L12, L13, L14, L15, M11, M12, M15, M16, N11, N16, P11, P16, R11, R12, R15, R16, T12, T13, T14, T15
VCCIO0	B5, B9, E7, H9	B12, B7, F11, J13, K12
VCCIO1	D13, E16, H14	D18, F16, J14, K15
VCCIO2	E21, G18, J15, K19	G25, L21, M17, M25, N18
VCCIO3	N19, P15, T18, V21	P18, R17, R25, T21, Y25
VCCIO4	AA18, R14, V16, W13	AA16, AC18, U15, V14
VCCIO5	AA5, R9, V7, W10	AA11, AE12, AE7, U12, V13
VCCIO6	N4, P8, T5, V2	P9, R10, R2, T6, Y2
VCCIO7	E2, G5, J8, K4	G2, L6, M10, M2, N9
VCCJ	W4	AA7
VCCAUX	H11, H12, L15, L8, M15, M8, R11, R12	AE19, J11, J12, J15, J16, L18, L9, M18, M9, R18, R9, T18, T9, V11, V12, V15, V16
VCCPLL	H8, H15, R8, R15	G19, H7, J17, K6, P7, P20, R8, V18
GND, GND0 to GND7	A1, A10, A16, A22, AA19, AA4, AB1, AB22, B13, B19, B4, D16, D2, D21, D7, G19, G4, H10, H13, J14, J9, K10, K11, K12, K13, K15, K20, K3, K8, L10, L11, L12, L13, M10, M11, M12, M13, N10, N11, N12, N13, N15, N20, N3, N8, P14, P9, R10, R13, T19, T4, W16, W2, W21, W7, Y10, Y13	A13, A19, A2, A25, AA2, AA25, AB18, AB22, AB5, AB9, AE1, AE11, AE16, AE22, AE26, AE6, AF13, AF19, AF2, AF25, B1, B11, B16, B22, B26, B6, E18, E22, E5, E9, F2, F25, G11, G16, J22, J5, K11, K13, K14, K16, L10, L11, L16, L17, L2, L20, L25, L7, M13, M14, N10, N12, N13, N14, N15, N17, P10, P12, P13, P14, P15, P17, R13, R14, T10, T11, T16, T17, T2, T20, T25, T7, U11, U13, U14, U16, V22, V5, Y11, Y16
NC	D14, D15, E14, E15, F13, F14, F15, G12, G13, G14, G15, U6	AB3, AB4, AC1, AC2, AD15, AD18, AD20, AD23, AE13, AE25, AF16, AF22, B4, B5, C26, D20, D21, D22, D23, D24, D25, D26, E20, E21, E25, E26, F20, G20, K10, K17, R4, U10, U23, V10, W7, N7, V7

**ECP2-12E Logic Signal Connections: 144 TQFP**

Pin Number	Pin Function	Bank	Dual Function	Differential
1	PL2A	7	VREF2_7	T*
2	PL2B	7	VREF1_7	C*
3	PL4A	7		T*
4	PL4B	7		C*
5	PL6A	7		T*
6	VCCAUX	-		
7	PL6B	7		C*
8	PL8A	7		T*
9	VCCIO7	7		
10	PL8B	7		C*
11	GND	-		
12	PL12A	7		T*
13	PL12B	7		C*
14	PL13A	7	PCLKT7_0	T
15	PL13B	7	PCLKC7_0	C
16	VCC	-		
17	PL15A	6	PCLKT6_0	T*
18	PL15B	6	PCLKC6_0	C*
19	PL16A	6	VREF2_6	T
20	PL16B	6	VREF1_6	C
21	GND	-		
22	LLM0_VCCPLL	6		
23	PL18A	6	LLM0_GDLLT_FB_A	T
24	PL18B	6	LLM0_GDLLC_FB_A	C
25	LLM0_PLLCAP	6		
26	PL20A	6	LLM0_GPLLT_IN_A	T*
27	PL20B	6	LLM0_GPLLC_IN_A	C*
28	PL22A	6		
29	VCC	-		
30	GND	-		
31	VCCIO6	6		
32	TCK	-		
33	TDI	-		
34	TDO	-		
35	VCCJ	-		
36	TMS	-		
37	PB2A	5	VREF2_5	T
38	PB2B	5	VREF1_5	C
39	VCCAUX	-		
40	PB6A	5	BDQS6	T
41	PB6B	5		C
42	VCCIO5	5		
43	PB12A	5		T

**ECP2-12E Logic Signal Connections: 144 TQFP (Cont.)**

Pin Number	Pin Function	Bank	Dual Function	Differential
44	PB12B	5		C
45	PB16A	5		T
46	PB16B	5		C
47	GND	-		
48	VCC	-		
49	PB26A	5	PCLKT5_0	T
50	PB26B	5	PCLKC5_0	C
51	GND	-		
52	PB31A	4	PCLKT4_0	T
53	PB31B	4	PCLKC4_0	C
54	VCC	-		
55	PB34A	4		T
56	PB34B	4		C
57	PB40A	4		T
58	PB40B	4		C
59	PB44A	4		T
60	PB44B	4		C
61	GND	-		
62	PB48A	4		T
63	PB48B	4		C
64	VCCIO4	4		
65	PB50A	4		T
66	PB50B	4		C
67	PB52A	4		T
68	PB52B	4		C
69	PB54A	4		T
70	PB54B	4		C
71	PB55A	4	VREF2_4	T
72	PB55B	4	VREF1_4	C
73	CFG1	8		
74	CFG2	8		
75	PROGRAMN	8		
76	INITN	8		
77	CFG0	8		
78	CCLK	8		
79	DONE	8		
80	PR29A	8	D0	
81	GND	-		
82	PR26A	8	D6	
83	VCC	-		
84	PR25B	8	D7	C
85	VCCIO8	8		
86	PR25A	8	DI	T
87	PR24B	8	DOUT,CS0N	C

**ECP2-12E Logic Signal Connections: 144 TQFP (Cont.)**

Pin Number	Pin Function	Bank	Dual Function	Differential
88	PR24A	8	BUSY	T
89	VCCIO3	3		
90	VCCAUX	-		
91	PR20B	3	RLM0_GPLLIC_IN_A	C*
92	PR20A	3	RLM0_GPLLT_IN_A	T*
93	RLM0_PLLCAP	3		
94	RLM0_VCCPLL	3		
95	GND	-		
96	PR17B	3	RLM0_GDLLC_IN_A	C*
97	PR17A	3	RLM0_GDLLT_IN_A	T*
98	PR16B	3	VREF2_3	C
99	PR16A	3	VREF1_3	T
100	PR15B	3	PCLKC3_0	C*
101	PR15A	3	PCLKT3_0	T*
102	VCC	-		
103	PR13B	2	PCLKC2_0	C
104	PR13A	2	PCLKT2_0	T
105	GND	-		
106	VCCIO2	2		
107	PR2B	2	VREF2_2	C*
108	PR2A	2	VREF1_2	T*
109	PT55B	1	VREF2_1	C
110	PT55A	1	VREF1_1	T
111	PT54B	1		C
112	PT54A	1		T
113	PT52B	1		C
114	PT52A	1		T
115	PT50B	1		C
116	PT50A	1		T
117	VCCIO1	1		
118	PT48B	1		C
119	PT48A	1		T
120	GND	-		
121	PT44B	1		C
122	PT44A	1		T
123	PT40B	1		C
124	PT40A	1		T
125	PT34B	1		C
126	PT34A	1		T
127	NC	1		
128	VCC	-		
129	PT30B	1	PCLKC1_0	C
130	PT30A	1	PCLKT1_0	T
131	PT28B	0	PCLKC0_0	C

**ECP2-12E Logic Signal Connections: 144 TQFP (Cont.)**

Pin Number	Pin Function	Bank	Dual Function	Differential
132	XRES	0		
133	GND	-		
134	PT28A	0	PCLKT0_0	T
135	VCC	-		
136	PT16B	0		C
137	PT16A	0		T
138	GND	-		
139	VCCIO0	0		
140	PT6B	0		C
141	PT6A	0		T
142	VCCAUX	-		
143	PT2B	0	VREF2_0	C
144	PT2A	0	VREF1_0	T

\*Supports dedicated LVDS outputs.

**ECP2-12E Logic Signal Connections: 208 PQFP**

Pin Number	Pin Function	Bank	Dual Function	Differential
1	PL2A	7	VREF2_7	T*
2	PL2B	7	VREF1_7	C*
3	PL4A	7		T*
4	PL4B	7		C*
5	GND	-		
6	PL6A	7		T*
7	VCCAUX	-		
8	PL6B	7		C*
9	PL8A	7		T*
10	VCCIO7	7		
11	PL8B	7		C*
12	VCC	-		
13	GND	-		
14	VCCIO7	7		
15	PL12A	7		T*
16	PL12B	7		C*
17	GND	-		
18	PL13A	7	PCLKT7_0	T
19	VCC	-		
20	PL13B	7	PCLKC7_0	C
21	PL15A	6	PCLKT6_0	T*
22	PL15B	6	PCLKC6_0	C*
23	PL16A	6	VREF2_6	T
24	PL16B	6	VREF1_6	C
25	GND	-		
26	PL17A	6	LLM0_GDLLT_IN_A	T*
27	PL17B	6	LLM0_GDLLC_IN_A	C*
28	LLM0_VCCPLL	6		
29	LLM0_PLLCAP	6		
30	VCCAUX	-		
31	PL20A	6	LLM0_GPLLT_IN_A	T*
32	GND	-		
33	PL21A	6	LLM0_GPLLT_FB_A	T
34	PL20B	6	LLM0_GPLLC_IN_A	C*
35	PL21B	6	LLM0_GPLLC_FB_A	C
36	PL23A	6		
37	PL24A	6		T*
38	VCCIO6	6		
39	PL24B	6		C*
40	VCC	-		
41	PL26A	6		T*
42	GND	-		
43	PL26B	6		C*

**ECP2-12E Logic Signal Connections: 208 PQFP (Cont.)**

Pin Number	Pin Function	Bank	Dual Function	Differential
44	VCCIO6	6		
45	PL28A	6	LDQS28	T*
46	PL28B	6		C*
47	PL30A	6		
48	TCK	-		
49	TDI	-		
50	TDO	-		
51	VCCJ	-		
52	TMS	-		
53	PB2A	5	VREF2_5	T
54	PB2B	5	VREF1_5	C
55	VCCIO5	5		
56	PB6A	5	BDQS6	T
57	PB6B	5		C
58	PB8A	5		T
59	PB8B	5		C
60	GND	-		
61	PB12A	5		T
62	PB12B	5		C
63	VCCIO5	5		
64	PB16A	5		T
65	PB16B	5		C
66	PB18A	5		T
67	PB18B	5		C
68	GND	-		
69	PB20A	5		T
70	VCCAUX	-		
71	PB20B	5		C
72	PB22A	5		T
73	PB22B	5		C
74	VCC	-		
75	PB26A	5	PCLKT5_0	T
76	PB26B	5	PCLKC5_0	C
77	GND	-		
78	PB31A	4	PCLKT4_0	T
79	PB31B	4	PCLKC4_0	C
80	VCC	-		
81	GND	-		
82	PB34A	4		T
83	PB34B	4		C
84	PB36A	4		T
85	PB36B	4		C
86	VCCAUX	-		
87	PB40A	4		T

**ECP2-12E Logic Signal Connections: 208 PQFP (Cont.)**

Pin Number	Pin Function	Bank	Dual Function	Differential
88	PB40B	4		C
89	GND	-		
90	PB42A	4	BDQS42	T
91	PB42B	4		C
92	PB44A	4		T
93	VCCIO4	4		
94	PB44B	4		C
95	PB48A	4		T
96	PB48B	4		C
97	VCC	-		
98	PB52A	4		T
99	PB52B	4		C
100	VCCIO4	4		
101	PB54A	4		
102	GND	-		
103	PB55A	4	VREF2_4	T
104	PB55B	4	VREF1_4	C
105	CFG1	8		
106	PROGRAMN	8		
107	CFG2	8		
108	INITN	8		
109	CFG0	8		
110	CCLK	8		
111	DONE	8		
112	PR29A	8	D0	
113	VCCIO8	8		
114	PR26A	8	D6	
115	GND	-		
116	VCC	-		
117	PR25B	8	D7	C
118	VCCIO8	8		
119	PR25A	8	DI	T
120	PR24B	8	DOUT,CS0N	C
121	PR24A	8	BUSY	T
122	GND	-		
123	VCCIO3	3		
124	PR21A	3	RLM0_GPLLFB_A	
125	VCCAUX	-		
126	PR20B	3	RLM0_GPLLC_IN_A	C*
127	PR20A	3	RLM0_GPLLFB_A	T*
128	RLM0_PLLCAP	3		
129	RLM0_VCCPLL	3		
130	PR18B	3	RLM0_GDLLC_FB_A	C
131	PR18A	3	RLM0_GDLFB_A	T

**ECP2-12E Logic Signal Connections: 208 PQFP (Cont.)**

Pin Number	Pin Function	Bank	Dual Function	Differential
132	PR17B	3	RLM0_GDLLC_IN_A	C*
133	PR17A	3	RLM0_GDLLT_IN_A	T*
134	PR16B	3	VREF2_3	C
135	VCCIO3	3		
136	PR16A	3	VREF1_3	T
137	PR15B	3	PCLKC3_0	C*
138	PR15A	3	PCLKT3_0	T*
139	GND	-		
140	VCC	-		
141	PR13B	2	PCLKC2_0	C
142	PR13A	2	PCLKT2_0	T
143	VCCIO2	2		
144	PR12A	2		
145	GND	-		
146	VCC	-		
147	PR8B	2		C*
148	VCCIO2	2		
149	PR8A	2		T*
150	PR6B	2		C*
151	VCCAUX	-		
152	PR6A	2		T*
153	PR4B	2		C*
154	PR4A	2		T*
155	PR2B	2	VREF2_2	C*
156	PR2A	2	VREF1_2	T*
157	PT55B	1	VREF2_1	C
158	PT55A	1	VREF1_1	T
159	GND	-		
160	PT54B	1		C
161	PT54A	1		T
162	VCCIO1	1		
163	PT52B	1		C
164	PT52A	1		T
165	PT50B	1		C
166	PT50A	1		T
167	PT48B	1		C
168	PT48A	1		T
169	GND	-		
170	VCCIO1	1		
171	VCC	-		
172	PT40B	1		C
173	PT40A	1		T
174	VCCAUX	-		
175	GND	-		

**ECP2-12E Logic Signal Connections: 208 PQFP (Cont.)**

Pin Number	Pin Function	Bank	Dual Function	Differential
176	PT36B	1		C
177	PT36A	1		T
178	PT34B	1		C
179	PT34A	1		T
180	PT30B	1	PCLKC1_0	C
181	PT30A	1	PCLKT1_0	T
182	XRES	1		
183	PT28B	0	PCLKC0_0	C
184	GND	-		
185	PT28A	0	PCLKT0_0	T
186	PT26B	0		C
187	PT26A	0		T
188	VCC	-		
189	PT20B	0		C
190	VCCAUX	-		
191	PT20A	0		T
192	GND	-		
193	PT18B	0		C
194	PT18A	0		T
195	VCCIO0	0		
196	PT16B	0		C
197	PT16A	0		T
198	VCC	-		
199	PT12B	0		C
200	PT12A	0		T
201	GND	-		
202	PT8B	0		C
203	PT8A	0		T
204	PT6B	0		C
205	PT6A	0		T
206	VCCIO0	0		
207	PT2B	0	VREF2_0	C
208	PT2A	0	VREF1_0	T

\*Supports dedicated LVDS outputs.

**ECP2-12E Logic Signal Connections: 256 fpBGA**

Ball Number	Ball Function	Bank	Dual Function	Differential
C3	PL2A	7	VREF2_7	T*
C2	PL2B	7	VREF1_7	C*
D3	PL5A	7		T
D4	PL4A	7		T*
D2	PL5B	7		C
GND	GNDIO	7		
E4	PL4B	7		C*
B1	PL7A	7		T
C1	PL7B	7		C
F5	PL9A	7		T
F4	PL8A	7		T*
G6	PL9B	7		C
G4	PL8B	7		C*
D1	PL10A	7	LDQS10	T*
GND	GNDIO	7		
E1	PL10B	7		C*
F3	PL11A	7		T
G3	PL11B	7		C
F2	PL12A	7		T*
F1	PL12B	7		C*
GND	GNDIO	7		
G2	PL13A	7	PCLKT7_0	T
G1	PL13B	7	PCLKC7_0	C
H6	PL15A	6	PCLKT6_0	T*
H5	PL15B	6	PCLKC6_0	C*
H4	PL16A	6	VREF2_6	T
GND	GNDIO	6		
H3	PL16B	6	VREF1_6	C
H2	PL17A	6	LLM0_GDLLT_IN_A	T*
H1	PL17B	6	LLM0_GDLLC_IN_A	C*
G10	VCCPLL	-		
J4	PL18A	6	LLM0_GDLLT_FB_A	T
J5	PL18B	6	LLM0_GDLLC_FB_A	C
J6	LLM0_PLLCAP	6		
K4	PL20A	6	LLM0_GPLLT_IN_A	T*
GND	GNDIO	6		
J1	PL21A	6	LLM0_GPLLT_FB_A	T
K3	PL20B	6	LLM0_GPLLC_IN_A	C*
J2	PL21B	6	LLM0_GPLLC_FB_A	C
GND	GNDIO	6		
L2	PL24A	6		T*
K2	PL25A	6		T
L3	PL24B	6		C*

**ECP2-12E Logic Signal Connections: 256 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
K1	PL25B	6		C
L4	PL26A	6		T*
L1	PL27A	6		T
L5	PL26B	6		C*
M1	PL27B	6		C
GND	GNDIO	6		
N1	PL29A	6		T
N2	PL28A	6	LDQS28	T*
P1	PL29B	6		C
P2	PL28B	6		C*
R1	PL30A	6		T*
GND	GNDIO	6		
R2	PL30B	6		C*
N4	TDI	-		
M4	TCK	-		
P3	TDO	-		
N3	TMS	-		
K7	VCCJ	-		
M5	PB2A	5	VREF2_5	T
K6	PB3A	5		
M6	PB2B	5	VREF1_5	C
R3	PB5A	5		T
P4	PB5B	5		C
N5	PB21A	5		T
N6	PB21B	5		C
T2	PB22A	5		T
P6	PB23A	5		T
T3	PB22B	5		C
R6	PB23B	5		C
R4	PB24A	5	BDQS24	T
L6	PB25A	5		T
T4	PB24B	5		C
L7	PB25B	5		C
N7	PB26A	5	PCLKT5_0	T
M8	PB26B	5	PCLKC5_0	C
P7	PB31A	4	PCLKT4_0	T
R8	PB31B	4	PCLKC4_0	C
T5	PB32A	4		T
T6	PB32B	4		C
T8	PB33A	4	BDQS33	T
R7	PB34A	4		T
T9	PB33B	4		C
T7	PB34B	4		C
L8	PB35A	4		T

**ECP2-12E Logic Signal Connections: 256 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
P8	PB36A	4		T
L9	PB35B	4		C
N8	PB36B	4		C
R9	PB37A	4		T
R10	PB37B	4		C
N9	PB47A	4		T
T10	PB48A	4		T
M9	PB47B	4		C
R11	PB48B	4		C
P10	PB49A	4		T
N11	PB50A	4		T
N10	PB49B	4		C
P11	PB50B	4		C
T11	PB51A	4	BDQS51	T
M11	PB52A	4		T
T12	PB51B	4		C
L11	PB52B	4		C
T13	PB53A	4		T
R13	PB54A	4		T
T14	PB53B	4		C
P13	PB54B	4		C
N12	PB55A	4	VREF2_4	T
M12	PB55B	4	VREF1_4	C
R15	CFG2	8		
N14	CFG1	8		
N13	PROGRAMN	8		
N15	CFG0	8		
P15	PR30B	8	WRITEN	C
L12	INITN	8		
N16	PR29B	8	CSN	C
GND	GNDIO	8		
R14	CCLK	8		
P14	PR30A	8	CS1N	T
M13	DONE	8		
R16	PR28B	8	D1	C
M16	PR29A	8	D0	T
P16	PR28A	8	D2	T
L15	PR27B	8	D3	C
GND	GNDIO	8		
L14	PR26A	8	D6	T
L16	PR27A	8	D4	T
L10	PR25B	8	D7	C
L13	PR26B	8	D5	C
K11	PR25A	8	DI	T

**ECP2-12E Logic Signal Connections: 256 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
K14	PR24B	8	DOUT,CS0N	C
K13	PR24A	8	BUSY	T
GND	GNDIO	8		
K15	PR21B	3	RLM0_GPLL0_FB_A	C
K16	PR21A	3	RLM0_GPLLT_FB_A	T
GND	GNDIO	3		
J16	PR20B	3	RLM0_GPLL0_IN_A	C*
J15	PR20A	3	RLM0_GPLLT_IN_A	T*
J14	RLM0_PLLCAP	3		
J13	PR18B	3	RLM0_GDLL0_FB_A	C
J12	PR18A	3	RLM0_GDLLT_FB_A	T
H12	PR17B	3	RLM0_GDLL0_IN_A	C*
GND	GNDIO	3		
H13	PR17A	3	RLM0_GDLLT_IN_A	T*
H15	PR16B	3	VREF2_3	C
H16	PR16A	3	VREF1_3	T
H11	PR15B	3	PCLKC3_0	C*
J11	PR15A	3	PCLKT3_0	T*
G16	PR13B	2	PCLKC2_0	C
GND	GNDIO	2		
G15	PR13A	2	PCLKT2_0	T
F15	PR11B	2		C
G11	PR12B	2		C*
F14	PR11A	2		T
F12	PR12A	2		T*
G14	PR10B	2		C*
G13	PR10A	2	RDQS10	T*
GND	GNDIO	2		
F16	PR8B	2		C*
F9	PR9B	2		C
E16	PR8A	2		T*
F10	PR9A	2		T
D16	PR7B	2		C
D15	PR7A	2		T
C15	PR4B	2		C*
C16	PR5B	2		C
GND	GNDIO	2		
D14	PR4A	2		T*
B16	PR5A	2		T
F13	PR2B	2	VREF2_2	C*
E13	PR2A	2	VREF1_2	T*
F11	PT55B	1	VREF2_1	C
E11	PT55A	1	VREF1_1	T
A15	PT54B	1		C

**ECP2-12E Logic Signal Connections: 256 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
E12	PT53B	1		C
B15	PT54A	1		T
D12	PT53A	1		T
B14	PT52B	1		C
C14	PT51B	1		C
A14	PT52A	1		T
D13	PT51A	1		T
C13	PT50B	1		C
A13	PT49B	1		C
B13	PT50A	1		T
A12	PT49A	1		T
B11	PT48B	1		C
D11	PT47B	1		C
A11	PT48A	1		T
C11	PT47A	1		T
D10	PT37B	1		C
C10	PT37A	1		T
B10	PT36B	1		C
A9	PT35B	1		C
A10	PT36A	1		T
B9	PT35A	1		T
A8	PT34B	1		C
D9	PT33B	1		C
B8	PT34A	1		T
C9	PT33A	1		T
B7	PT32B	1		C
E9	PT31B	1		C
A7	PT32A	1		T
D8	PT31A	1		T
A6	PT30B	1	PCLKC1_0	C
B6	PT30A	1	PCLKT1_0	T
F8	PT28B	0	PCLKC0_0	C
E6	XRES	1		
E8	PT28A	0	PCLKT0_0	T
A5	PT27B	0		C
A3	PT26B	0		C
A4	PT27A	0		T
B3	PT26A	0		T
A2	PT25B	0		C
C7	PT24B	0		C
B2	PT25A	0		T
D7	PT24A	0		T
D6	PT23B	0		C
F7	PT22B	0		C

**ECP2-12E Logic Signal Connections: 256 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
C6	PT23A	0		T
F6	PT22A	0		T
C4	PT21B	0		C
B4	PT21A	0		T
D5	PT2B	0	VREF2_0	C
E5	PT2A	0	VREF1_0	T
G7	VCC	-		
G9	VCC	-		
H7	VCC	-		
J10	VCC	-		
K10	VCC	-		
K8	VCC	-		
G8	VCCAUX	-		
H10	VCCAUX	-		
J7	VCCAUX	-		
K9	VCCAUX	-		
C5	VCCIO0	0		
E7	VCCIO0	0		
C12	VCCIO1	1		
E10	VCCIO1	1		
E14	VCCIO2	2		
G12	VCCIO2	2		
K12	VCCIO3	3		
M14	VCCIO3	3		
M10	VCCIO4	4		
P12	VCCIO4	4		
M7	VCCIO5	5		
P5	VCCIO5	5		
K5	VCCIO6	6		
M3	VCCIO6	6		
E3	VCCIO7	7		
G5	VCCIO7	7		
T15	VCCIO8	8		
A1	GND	-		
A16	GND	-		
B12	GND	-		
B5	GND	-		
C8	GND	-		
E15	GND	-		
E2	GND	-		
H14	GND	-		
H8	GND	-		
H9	GND	-		
J3	GND	-		

**ECP2-12E Logic Signal Connections: 256 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
J8	GND	-		
J9	GND	-		
M15	GND	-		
M2	GND	-		
P9	GND	-		
R12	GND	-		
R5	GND	-		
T1	GND	-		
T16	GND	-		

\*Supports dedicated LVDS outputs.

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
E4	PL2A	7	VREF2_7	T*	E4	PL2A	7	VREF2_7	T*
E5	PL2B	7	VREF1_7	C*	E5	PL2B	7	VREF1_7	C*
E3	NC	-			E3	PL12A	7		T*
F4	PL3A	7		T	F4	PL13A	7		T
F3	NC	-			F3	PL12B	7		C*
F5	PL3B	7		C	F5	PL13B	7		C
E2	PL4A	7		T*	E2	PL14A	7		T*
G6	PL5A	7		T	G6	PL15A	7		T
E1	PL4B	7		C*	E1	PL14B	7		C*
G7	PL5B	7		C	G7	PL15B	7		C
F1	NC	-			F1	PL17A	7		T
H4	NC	-			H4	PL16A	7	LDQS16	T*
F2	NC	-			F2	PL17B	7		C
H5	NC	-			H5	PL16B	7		C*
G1	NC	-			G1	PL19A	7		T
G3	NC	-			G3	PL18A	7		T*
G2	NC	-			G2	PL19B	7		C
G4	NC	-			G4	PL18B	7		C*
J4	PL7A	7		T	J4	PL38A	7		T
H1	PL6A	7			H1	PL37A	7		
J5	PL7B	7		C	J5	PL38B	7		C
L6	PL9A	7		T	L6	PL40A	7		T
J2	PL8A	7		T*	J2	PL39A	7		T*
L5	PL9B	7		C	L5	PL40B	7		C
J1	PL8B	7		C*	J1	PL39B	7		C*
K3	PL10A	7	LDQS10	T*	K3	PL41A	7	LDQS41	T*
K6	NC	-			K6	LUM0_VCCPLL	7		
K4	PL10B	7		C*	K4	PL41B	7		C*
K2	PL11A	7		T	K2	PL42A	7		T
K1	PL11B	7		C	K1	PL42B	7		C
L4	PL12A	7		T*	L4	PL43A	7		T*
L3	PL12B	7		C*	L3	PL43B	7		C*
L2	PL13A	7	PCLKT7_0	T	L2	PL44A	7	PCLKT7_0	T
L1	PL13B	7	PCLKC7_0	C	L1	PL44B	7	PCLKC7_0	C
M5	PL15A	6	PCLKT6_0	T*	M5	PL46A	6	PCLKT6_0	T*
M6	PL15B	6	PCLKC6_0	C*	M6	PL46B	6	PCLKC6_0	C*
M3	PL16A	6	VREF2_6	T	M3	PL47A	6	VREF2_6	T
M4	PL16B	6	VREF1_6	C	M4	PL47B	6	VREF1_6	C
N1	NC	-			N1	PL49A	6		T
M2	NC	-			M2	PL48A	6		T*
N2	NC	-			N2	PL49B	6		C
M1	NC	-			M1	PL48B	6		C*
N3	NC	-			N3	PL58A	6	LDQS58	T*
N5	NC	-			N5	PL59A	6		T
N4	NC	-			N4	PL58B	6		C*
P5	NC	-			P5	PL59B	6		C

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
P1	PL17A	6	LLM0_GDLLT_IN_A	T*	P1	PL60A	6	LLM0_GDLLT_IN_A	T*
P2	PL17B	6	LLM0_GDLLC_IN_A	C*	P2	PL60B	6	LLM0_GDLLC_IN_A	C*
P4	PL18A	6	LLM0_GDLLT_FB_A	T	P4	PL61A	6	LLM0_GDLLT_FB_A	T
R4	PL18B	6	LLM0_GDLLC_FB_A	C	R4	PL61B	6	LLM0_GDLLC_FB_D	C
N6	LLM0_VCCPLL	6			N6	LLM0_VCCPLL	6		
P6	LLM0_PLLCAP	6			P6	LLM0_PLLCAP	6		
R1	PL20A	6	LLM0_GPLLT_IN_A	T*	R1	PL63A	6	LLM0_GPLLT_IN_A	T*
R3	PL21A	6	LLM0_GPLLT_FB_A	T	R3	PL64A	6	LLM0_GPLLT_FB_A	T
R2	PL20B	6	LLM0_GPLLC_IN_A	C*	R2	PL63B	6	LLM0_GPLLC_IN_A	C*
T4	PL21B	6	LLM0_GPLLC_FB_A	C	T4	PL64B	6	LLM0_GPLLC_FB_A	C
T5	PL23A	6		T	T5	PL66A	6		T
T1	PL22A	6		T*	T1	PL65A	6		T*
T3	PL23B	6		C	T3	PL66B	6		C
T2	PL22B	6		C*	T2	PL65B	6		C*
V1	PL25A	6		T	V1	PL72A	6		T
V2	PL25B	6		C	V2	PL72B	6		C
U1	PL24A	6		T*	U1	PL71A	6		T*
U3	PL27A	6		T	U3	PL74A	6		T
U2	PL24B	6		C*	U2	PL71B	6		C*
U4	PL27B	6		C	U4	PL74B	6		C
R6	PL26A	6		T*	R6	PL73A	6		T*
R7	PL29A	6		T	R7	PL76A	6		T
T7	PL29B	6		C	T7	PL76B	6		C
T6	PL26B	6		C*	T6	PL73B	6		C*
AA2	PL31A	6		T	AA2	PL78A	6		T
Y1	PL28A	6	LDQS28	T*	Y1	PL75A	6	LDQS75	T*
AA1	PL31B	6		C	AA1	PL78B	6		C
W1	PL28B	6		C*	W1	PL75B	6		C*
V3	PL30B	6		C*	V3	PL77B	6		C*
V4	PL30A	6		T*	V4	PL77A	6		T*
U5	TDI	-			U5	TDI	-		
U7	TCK	-			U7	TCK	-		
V6	TDO	-			V6	TDO	-		
V5	TMS	-			V5	TMS	-		
T8	VCCJ	-			T8	VCCJ	-		
W4	PB3A	5		T	W4	PB3A	5		T
Y3	PB2A	5	VREF2_5	T	Y3	PB2A	5	VREF2_5	T
W3	PB3B	5		C	W3	PB3B	5		C
Y2	PB2B	5	VREF1_5	C	Y2	PB2B	5	VREF1_5	C
AB3	PB5A	5		T	AB3	PB5A	5		T
W5	PB4A	5		T	W5	PB4A	5		T
AB2	PB5B	5		C	AB2	PB5B	5		C
W6	PB4B	5		C	W6	PB4B	5		C
AB5	PB7A	5		T	AB5	PB7A	5		T
Y4	PB6A	5	BDQS6	T	Y4	PB6A	5	BDQS6	T
AB4	PB7B	5		C	AB4	PB7B	5		C

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
AA3	PB6B	5		C	AA3	PB6B	5		C
AB6	PB9A	5		T	AB6	PB9A	5		T
AA5	PB8A	5		T	AA5	PB8A	5		T
AA6	PB9B	5		C	AA6	PB9B	5		C
Y5	PB8B	5		C	Y5	PB8B	5		C
Y6	PB12A	5		T	Y6	PB30A	5		T
W7	PB11A	5		T	W7	PB29A	5		T
Y7	PB12B	5		C	Y7	PB30B	5		C
W8	PB11B	5		C	W8	PB29B	5		C
U8	PB14A	5		T	U8	PB32A	5		T
AA7	PB13A	5		T	AA7	PB31A	5		T
U9	PB14B	5		C	U9	PB32B	5		C
AB7	PB13B	5		C	AB7	PB31B	5		C
Y8	PB16A	5		T	Y8	PB34A	5		T
W9	PB15A	5	BDQS15	T	W9	PB33A	5	BDQS33	T
AA8	PB16B	5		C	AA8	PB34B	5		C
V9	PB15B	5		C	V9	PB33B	5		C
AB8	PB18A	5		T	AB8	PB36A	5		T
W10	PB17A	5		T	W10	PB35A	5		T
AA9	PB18B	5		C	AA9	PB36B	5		C
V10	PB17B	5		C	V10	PB35B	5		C
Y10	PB21A	5		T	Y10	PB39A	5		T
AB9	PB20A	5		T	AB9	PB38A	5		T
AA10	PB21B	5		C	AB10	PB38B	5		C
AB10	PB20B	5		C	AA10	PB39B	5		C
AB11	PB23A	5		T	AB11	PB41A	5		T
U10	PB22A	5		T	U10	PB40A	5		T
AA11	PB23B	5		C	AA11	PB41B	5		C
U11	PB22B	5		C	U11	PB40B	5		C
AB12	PB25A	5		T	AB12	PB43A	5		T
Y11	PB24A	5	BDQS24	T	Y11	PB42A	5	BDQS42	T
AA12	PB25B	5		C	AA12	PB43B	5		C
W11	PB24B	5		C	W11	PB42B	5		C
AB13	PB26A	5	PCLKT5_0	T	AB13	PB44A	5	PCLKT5_0	T
AB14	PB26B	5	PCLKC5_0	C	AB14	PB44B	5	PCLKC5_0	C
Y12	PB32A	4		T	Y12	PB50A	4		T
W12	PB32B	4		C	W12	PB50B	4		C
U12	PB31A	4	PCLKT4_0	T	U12	PB49A	4	PCLKT4_0	T
V12	PB31B	4	PCLKC4_0	C	V12	PB49B	4	PCLKC4_0	C
U13	PB34A	4		T	U13	PB52A	4		T
AA13	PB33A	4	BDQS33	T	AA13	PB51A	4	BDQS51	T
U14	PB34B	4		C	U14	PB52B	4		C
Y13	PB33B	4		C	Y13	PB51B	4		C
AB16	PB36A	4		T	AB16	PB54A	4		T
AB15	PB35A	4		T	AB15	PB53A	4		T
AB17	PB36B	4		C	AB17	PB54B	4		C

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
AA14	PB35B	4		C	AA14	PB53B	4		C
W13	PB37A	4		T	W13	PB55A	4		T
W14	PB37B	4		C	W14	PB55B	4		C
AB18	PB39A	4		T	AB18	PB57A	4		T
AB19	PB39B	4		C	AB19	PB57B	4		C
Y15	PB41A	4		T	Y15	PB59A	4		T
V14	PB40A	4		T	V14	PB58A	4		T
AA15	PB41B	4		C	AA15	PB59B	4		C
W15	PB40B	4		C	W15	PB58B	4		C
AB20	PB43A	4		T	AB20	PB61A	4		T
AA16	PB42A	4	BDQS42	T	AA16	PB60A	4	BDQS60	T
AB21	PB43B	4		C	AB21	PB61B	4		C
AA17	PB42B	4		C	AA17	PB60B	4		C
Y16	PB45A	4		T	Y16	PB63A	4		T
U15	PB44A	4		T	U15	PB62A	4		T
W16	PB45B	4		C	W16	PB63B	4		C
U16	PB44B	4		C	U16	PB62B	4		C
AA18	PB46A	4		T	AA18	PB64A	4		T
AA20	PB46B	4		C	AA20	PB64B	4		C
V16	PB49A	4		T	V16	PB76A	4		T
V17	PB49B	4		C	V17	PB76B	4		C
AA21	PB48A	4		T	AA21	PB75A	4		T
Y19	PB51A	4	BDQS51	T	Y19	PB78A	4	BDQS78	T
AA22	PB48B	4		C	AA22	PB75B	4		C
Y20	PB51B	4		C	Y20	PB78B	4		C
Y18	PB50A	4		T	Y18	PB77A	4		T
Y21	PB53A	4		T	Y21	PB80A	4		T
Y17	PB50B	4		C	Y17	PB77B	4		C
Y22	PB53B	4		C	Y22	PB80B	4		C
W17	PB52A	4		T	W17	PB79A	4		T
U18	PB54A	4		T	U18	PB81A	4		T
W18	PB52B	4		C	W18	PB79B	4		C
V18	PB54B	4		C	V18	PB81B	4		C
T15	PB55A	4	VREF2_4	T	T15	PB82A	4	VREF2_4	T
T16	PB55B	4	VREF1_4	C	T16	PB82B	4	VREF1_4	C
W19	CFG2	8			W19	CFG2	8		
V19	CFG1	8			V19	CFG1	8		
V20	PROGRAMN	8			V20	PROGRAMN	8		
W20	CFG0	8			W20	CFG0	8		
U22	PR28B	8	D1	C	U22	PR75B	8	D1	C
V22	INITN	8			V22	INITN	8		
R16	PR30B	8	WRITEN	C	R16	PR77B	8	WRITEN	C
W22	CCLK	8			W22	CCLK	8		
R17	PR30A	8	CS1N	T	R17	PR77A	8	CS1N	T
V21	DONE	8			V21	DONE	8		
U19	PR29B	8	CSN	C	U19	PR76B	8	CSN	C

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
T17	PR26B	8	D5	C	T17	PR73B	8	D5	C
U20	PR29A	8	D0	T	U20	PR76A	8	D0	T
U21	PR28A	8	D2	T	U21	PR75A	8	D2	T
T18	PR26A	8	D6	T	T18	PR73A	8	D6	T
T20	PR27B	8	D3	C	T20	PR74B	8	D3	C
T21	PR25B	8	D7	C	T21	PR72B	8	D7	C
T19	PR27A	8	D4	T	T19	PR74A	8	D4	T
T22	PR25A	8	DI	T	T22	PR72A	8	DI	T
R18	PR24B	8	DOUT,CSON	C	R18	PR71B	8	DOUT,CSON	C
R19	PR24A	8	BUSY	T	R19	PR71A	8	BUSY	T
P18	PR22B	3		C*	P18	PR65B	3		C*
R22	PR23B	3		C	R22	PR66B	3		C
P19	PR22A	3		T*	P19	PR65A	3		T*
R21	PR23A	3		T	R21	PR66A	3		T
R20	PR21B	3	RLM0_GPLLFB_A	C	R20	PR64B	3	RLM0_GPLLFB_A	C
P22	PR21A	3	RLM0_GPLLTFB_A	T	P22	PR64A	3	RLM0_GPLLTFB_A	T
P21	PR20B	3	RLM0_GPLLCIN_A	C*	P21	PR63B	3	RLM0_GPLLCIN_A	C*
N21	PR20A	3	RLM0_GPLLTIN_A	T*	N21	PR63A	3	RLM0_GPLLTIN_A	T*
N17	RLM0_PLLCAP	3			N17	RLM0_PLLCAP	3		
N18	RLM0_VCCPLL	3			N18	RLM0_VCCPLL	3		
N22	PR18B	3	RLM0_GDLLFB_A	C	N22	PR61B	3	RLM0_GDLLFB_A	C
M22	PR17B	3	RLM0_GDLLCIN_A	C*	M22	PR60B	3	RLM0_GDLLCIN_A	C*
N20	PR18A	3	RLM0_GDLLTFB_A	T	N20	PR61A	3	RLM0_GDLLTFB_A	T
M21	PR17A	3	RLM0_GDLLTIN_A	T*	M21	PR60A	3	RLM0_GDLLTIN_A	T*
N19	NC	-			N19	PR59B	3		C
M19	NC	-			M19	PR59A	3		T
J22	NC	-			J22	PR48B	3		C*
L22	NC	-			L22	PR49B	3		C
H22	NC	-			H22	PR48A	3		T*
K22	NC	-			K22	PR49A	3		T
M20	PR16B	3	VREF2_3	C	M20	PR47B	3	VREF2_3	C
L21	PR16A	3	VREF1_3	T	L21	PR47A	3	VREF1_3	T
K21	PR15B	3	PCLKC3_0	C*	K21	PR46B	3	PCLKC3_0	C*
J21	PR15A	3	PCLKT3_0	T*	J21	PR46A	3	PCLKT3_0	T*
M18	PR13B	2	PCLKC2_0	C	M18	PR44B	2	PCLKC2_0	C
L17	PR13A	2	PCLKT2_0	T	L17	PR44A	2	PCLKT2_0	T
L19	PR12B	2		C*	L19	PR43B	2		C*
K18	PR10B	2		C*	K18	PR41B	2		C*
L20	PR12A	2		T*	L20	PR43A	2		T*
K19	PR10A	2	RDQS10	T*	K19	PR41A	2	RDQS41	T*
L18	PR11B	2		C	L18	PR42B	2		C
K17	PR11A	2		T	K17	PR42A	2		T
J16	NC	-			J16	RUM0_VCCPLL	2		
J17	PR8B	2		C*	J17	PR39B	2		C*
G22	PR9B	2		C	G22	PR40B	2		C
J18	PR8A	2		T*	J18	PR39A	2		T*

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
F22	PR9A	2		T	F22	PR40A	2		T
H21	PR6B	2		C*	H21	PR37B	2		C*
K20	PR7B	2		C	K20	PR38B	2		C
G21	PR6A	2		T*	G21	PR37A	2		T*
J19	PR7A	2		T	J19	PR38A	2		T
D22	NC	-			D22	PR18B	2		C*
F21	NC	-			F21	PR19B	2		C
E21	NC	-			E21	PR18A	2		T*
E22	NC	-			E22	PR19A	2		T
H19	NC	-			H19	PR16B	2		C*
G20	NC	-			G20	PR17B	2		C
G19	NC	-			G19	PR16A	2	RDQS16	T*
F20	NC	-			F20	PR17A	2		T
G17	PR5B	2		C	G17	PR15B	2		C
E20	PR4B	2		C*	E20	PR14B	2		C*
F19	PR5A	2		T	F19	PR15A	2		T
D20	PR4A	2		T*	D20	PR14A	2		T*
F18	PR3B	2		C	F18	PR13B	2		C
C21	NC	-			C21	PR12B	2		C*
F16	PR3A	2		T	F16	PR13A	2		T
C22	NC	-			C22	PR12A	2		T*
D19	PR2B	2	VREF2_2	C*	D19	PR2B	2	VREF2_2	C*
E19	PR2A	2	VREF1_2	T*	E19	PR2A	2	VREF1_2	T*
B21	PT55B	1	VREF2_1	C	B21	PT82B	1	VREF2_1	C
B22	PT55A	1	VREF1_1	T	B22	PT82A	1	VREF1_1	T
D18	PT53B	1		C	D18	PT80B	1		C
C20	PT54B	1		C	C20	PT81B	1		C
E18	PT53A	1		T	E18	PT80A	1		T
C19	PT54A	1		T	C19	PT81A	1		T
B20	PT52B	1		C	B20	PT79B	1		C
D17	PT51B	1		C	D17	PT78B	1		C
C18	PT51A	1		T	C18	PT78A	1		T
A19	PT52A	1		T	A19	PT79A	1		T
A18	PT49B	1		C	A18	PT76B	1		C
A21	PT50B	1		C	A21	PT77B	1		C
B18	PT49A	1		T	B18	PT76A	1		T
A20	PT50A	1		T	A20	PT77A	1		T
D16	PT47B	1		C	D16	PT74B	1		C
G16	PT48B	1		C	G16	PT75B	1		C
E16	PT47A	1		T	E16	PT74A	1		T
G15	PT48A	1		T	G15	PT75A	1		T
C17	PT46B	1		C	C17	PT64B	1		C
C16	PT46A	1		T	C16	PT64A	1		T
A17	PT44B	1		C	A17	PT62B	1		C
B17	PT45B	1		C	B17	PT63B	1		C
A16	PT44A	1		T	A16	PT62A	1		T

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
B16	PT45A	1		T	B16	PT63A	1		T
E15	PT42B	1		C	E15	PT60B	1		C
C15	PT43B	1		C	C15	PT61B	1		C
F15	PT42A	1		T	F15	PT60A	1		T
D15	PT43A	1		T	D15	PT61A	1		T
B15	PT40B	1		C	B15	PT58B	1		C
A15	PT40A	1		T	A15	PT58A	1		T
A14	PT39A	1		T	A14	PT57A	1		T
B14	PT39B	1		C	B14	PT57B	1		C
D14	PT37B	1		C	D14	PT55B	1		C
E14	PT36B	1		C	E14	PT54B	1		C
C13	PT37A	1		T	C13	PT55A	1		T
F14	PT36A	1		T	F14	PT54A	1		T
A13	PT35B	1		C	A13	PT53B	1		C
E13	PT34B	1		C	E13	PT52B	1		C
B13	PT35A	1		T	B13	PT53A	1		T
D13	PT34A	1		T	D13	PT52A	1		T
E12	PT33B	1		C	E12	PT51B	1		C
D12	PT33A	1		T	D12	PT51A	1		T
A12	PT31B	1		C	A12	PT49B	1		C
B12	PT30B	1	PCLKC1_0	C	B12	PT48B	1	PCLKC1_0	C
A11	PT31A	1		T	A11	PT49A	1		T
C12	PT30A	1	PCLKT1_0	T	C12	PT48A	1	PCLKT1_0	T
F12	XRES	1			F12	XRES	1		
B10	PT28B	0	PCLKC0_0	C	B10	PT46B	0	PCLKC0_0	C
B11	PT28A	0	PCLKT0_0	T	B11	PT46A	0	PCLKT0_0	T
C11	PT26B	0		C	C11	PT44B	0		C
A10	PT27B	0		C	A10	PT45B	0		C
C10	PT26A	0		T	C10	PT44A	0		T
A9	PT27A	0		T	A9	PT45A	0		T
A8	PT24B	0		C	A8	PT42B	0		C
E11	PT25B	0		C	E11	PT43B	0		C
A7	PT24A	0		T	A7	PT42A	0		T
F11	PT25A	0		T	F11	PT43A	0		T
B8	PT23B	0		C	B8	PT41B	0		C
B9	PT23A	0		T	B9	PT41A	0		T
C8	PT20B	0		C	C8	PT38B	0		C
B7	PT21B	0		C	B7	PT39B	0		C
D8	PT20A	0		T	D8	PT38A	0		T
A6	PT21A	0		T	A6	PT39A	0		T
C7	PT17B	0		C	C7	PT35B	0		C
D10	PT18B	0		C	D10	PT36B	0		C
C6	PT17A	0		T	C6	PT35A	0		T
E10	PT18A	0		T	E10	PT36A	0		T
F10	PT15B	0		C	F10	PT33B	0		C
B6	PT16B	0		C	B6	PT34B	0		C

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
D9	PT15A	0		T	D9	PT33A	0		T
B5	PT16A	0		T	B5	PT34A	0		T
A5	PT13B	0		C	A5	PT31B	0		C
F9	PT14B	0		C	F9	PT32B	0		C
A4	PT13A	0		T	A4	PT31A	0		T
E9	PT14A	0		T	E9	PT32A	0		T
G8	PT11B	0		C	G8	PT29B	0		C
A3	PT12B	0		C	A3	PT30B	0		C
E8	PT11A	0		T	E8	PT29A	0		T
A2	PT12A	0		T	A2	PT30A	0		T
C3	PT10B	0		C	C3	PT10B	0		C
B3	PT10A	0		T	B3	PT10A	0		T
E7	PT8B	0		C	E7	PT8B	0		C
F8	PT9B	0		C	F8	PT9B	0		C
F7	PT8A	0		T	F7	PT8A	0		T
D7	PT9A	0		T	D7	PT9A	0		T
D4	PT6B	0		C	D4	PT6B	0		C
D5	PT7B	0		C	D5	PT7B	0		C
C4	PT6A	0		T	C4	PT6A	0		T
D6	PT7A	0		T	D6	PT7A	0		T
J7	PT4B	0		C	J7	PT4B	0		C
B2	PT5B	0		C	B2	PT5B	0		C
H7	PT4A	0		T	H7	PT4A	0		T
B1	PT5A	0		T	B1	PT5A	0		T
D1	PT2B	0	VREF2_0	C	D1	PT2B	0	VREF2_0	C
D3	PT3B	0		C	D3	PT3B	0		C
C1	PT2A	0	VREF1_0	T	C1	PT2A	0	VREF1_0	T
C2	PT3A	0		T	C2	PT3A	0		T
J10	VCC	-			J10	VCC	-		
J11	VCC	-			J11	VCC	-		
J12	VCC	-			J12	VCC	-		
J13	VCC	-			J13	VCC	-		
K14	VCC	-			K14	VCC	-		
K9	VCC	-			K9	VCC	-		
L14	VCC	-			L14	VCC	-		
L9	VCC	-			L9	VCC	-		
M14	VCC	-			M14	VCC	-		
M9	VCC	-			M9	VCC	-		
N14	VCC	-			N14	VCC	-		
N9	VCC	-			N9	VCC	-		
P10	VCC	-			P10	VCC	-		
P11	VCC	-			P11	VCC	-		
P12	VCC	-			P12	VCC	-		
P13	VCC	-			P13	VCC	-		
G10	VCCIO0	0			G10	VCCIO0	0		
G9	VCCIO0	0			G9	VCCIO0	0		

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
H8	VCCIO0	0			H8	VCCIO0	0		
H9	VCCIO0	0			H9	VCCIO0	0		
G11	VCCIO1	1			G11	VCCIO1	1		
G12	VCCIO1	1			G12	VCCIO1	1		
G13	VCCIO1	1			G13	VCCIO1	1		
G14	VCCIO1	1			G14	VCCIO1	1		
H14	VCCIO2	2			H14	VCCIO2	2		
H15	VCCIO2	2			H15	VCCIO2	2		
J15	VCCIO2	2			J15	VCCIO2	2		
K16	VCCIO2	2			K16	VCCIO2	2		
L16	VCCIO3	3			L16	VCCIO3	3		
M16	VCCIO3	3			M16	VCCIO3	3		
N16	VCCIO3	3			N16	VCCIO3	3		
P16	VCCIO3	3			P16	VCCIO3	3		
R14	VCCIO4	4			R14	VCCIO4	4		
T12	VCCIO4	4			T12	VCCIO4	4		
T13	VCCIO4	4			T13	VCCIO4	4		
T14	VCCIO4	4			T14	VCCIO4	4		
R9	VCCIO5	5			R9	VCCIO5	5		
T10	VCCIO5	5			T10	VCCIO5	5		
T11	VCCIO5	5			T11	VCCIO5	5		
T9	VCCIO5	5			T9	VCCIO5	5		
N7	VCCIO6	6			N7	VCCIO6	6		
P7	VCCIO6	6			P7	VCCIO6	6		
P8	VCCIO6	6			P8	VCCIO6	6		
R8	VCCIO6	6			R8	VCCIO6	6		
J8	VCCIO7	7			J8	VCCIO7	7		
K7	VCCIO7	7			K7	VCCIO7	7		
L7	VCCIO7	7			L7	VCCIO7	7		
M7	VCCIO7	7			M7	VCCIO7	7		
P15	VCCIO8	8			P15	VCCIO8	8		
R15	VCCIO8	8			R15	VCCIO8	8		
C5	VCCAUX	-			C5	VCCAUX	-		
D11	VCCAUX	-			D11	VCCAUX	-		
E17	VCCAUX	-			E17	VCCAUX	-		
E6	VCCAUX	-			E6	VCCAUX	-		
F13	VCCAUX	-			F13	VCCAUX	-		
G18	VCCAUX	-			G18	VCCAUX	-		
G5	VCCAUX	-			G5	VCCAUX	-		
K5	VCCAUX	-			K5	VCCAUX	-		
M17	VCCAUX	-			M17	VCCAUX	-		
P17	VCCAUX	-			P17	VCCAUX	-		
R5	VCCAUX	-			R5	VCCAUX	-		
V11	VCCAUX	-			V11	VCCAUX	-		
V13	VCCAUX	-			V13	VCCAUX	-		
V15	VCCAUX	-			V15	VCCAUX	-		

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
V7	VCCAUX	-			V7	VCCAUX	-		
V8	VCCAUX	-			V8	VCCAUX	-		
A1	GND	-			A1	GND	-		
A22	GND	-			A22	GND	-		
AA19	GND	-			AA19	GND	-		
AA4	GND	-			AA4	GND	-		
AB1	GND	-			AB1	GND	-		
AB22	GND	-			AB22	GND	-		
B19	GND	-			B19	GND	-		
B4	GND	-			B4	GND	-		
C14	GND	-			C14	GND	-		
C9	GND	-			C9	GND	-		
D2	GND	-			D2	GND	-		
D21	GND	-			D21	GND	-		
F17	GND	-			F17	GND	-		
F6	GND	-			F6	GND	-		
H10	GND	-			H10	GND	-		
H11	GND	-			H11	GND	-		
H12	GND	-			H12	GND	-		
H13	GND	-			H13	GND	-		
J14	GND	-			J14	GND	-		
J20	GND	-			J20	GND	-		
J3	GND	-			J3	GND	-		
J9	GND	-			J9	GND	-		
K10	GND	-			K10	GND	-		
K11	GND	-			K11	GND	-		
K12	GND	-			K12	GND	-		
K13	GND	-			K13	GND	-		
K15	GND	-			K15	GND	-		
K8	GND	-			K8	GND	-		
L10	GND	-			L10	GND	-		
L11	GND	-			L11	GND	-		
L12	GND	-			L12	GND	-		
L13	GND	-			L13	GND	-		
L15	GND	-			L15	GND	-		
L8	GND	-			L8	GND	-		
M10	GND	-			M10	GND	-		
M11	GND	-			M11	GND	-		
M12	GND	-			M12	GND	-		
M13	GND	-			M13	GND	-		
M15	GND	-			M15	GND	-		
M8	GND	-			M8	GND	-		
N10	GND	-			N10	GND	-		
N11	GND	-			N11	GND	-		
N12	GND	-			N12	GND	-		
N13	GND	-			N13	GND	-		

**ECP2-12E & ECP2-50E Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential	Ball Number	Ball Function	Bank	Dual Function	Differential
N15	GND	-			N15	GND	-		
N8	GND	-			N8	GND	-		
P14	GND	-			P14	GND	-		
P20	GND	-			P20	GND	-		
P3	GND	-			P3	GND	-		
P9	GND	-			P9	GND	-		
R10	GND	-			R10	GND	-		
R11	GND	-			R11	GND	-		
R12	GND	-			R12	GND	-		
R13	GND	-			R13	GND	-		
U17	GND	-			U17	GND	-		
U6	GND	-			U6	GND	-		
W2	GND	-			W2	GND	-		
W21	GND	-			W21	GND	-		
Y14	GND	-			Y14	GND	-		
Y9	GND	-			Y9	GND	-		
H6	NC	-			H6	PL25A	7	LUM0_SPLLT_IN_A	T
J6	NC	-			J6	PL25B	7	LUM0_SPLL_C_IN_A	C
H3	NC	-			H3	PL26A	7	LUM0_SPLLT_FB_A	T
H2	NC	-			H2	PL26B	7	LUM0_SPLL_C_FB_A	C
H17	NC	-			H17	PR26B	2	RUM0_SPLL_C_FB_A	C
H16	NC	-			H16	PR26A	2	RUM0_SPLLT_FB_A	T
H20	NC	-			H20	PR25B	2	RUM0_SPLL_C_IN_A	C
H18	NC	-			H18	PR25A	2	RUM0_SPLLT_IN_A	T

\*Supports dedicated LVDS outputs.

**ECP2M-35 Logic Signal Connections: 484 fpBGA**

Ball Number	Ball Function	Bank	Dual Function	Differential
D1	PL2A	7		T*
VCCAUX	VCCAUX			
E1	PL2B	7		C*
F1	PL3A	7		T
F2	PL3B	7		C
F5	PL4A	7		T*
G6	PL4B	7		C*
F4	PL5A	7		T
VCC	VCC			
F3	PL5B	7		C
GND	GND	7		
G1	PL6A	7	LDQS6	T*
G2	PL6B	7		C*
H1	PL7A	7		T
H2	PL7B	7		C
H7	PL8A	7		T*
H6	PL8B	7		C*
G3	PL9A	7	VREF2_7	T
H3	PL9B	7	VREF1_7	C
GND	GND	7		
H5	PL11A	7	LUM0_SPLL_IN_A	T*
VCCAUX	VCCAUX			
H4	PL11B	7	LUM0_SPLL_IN_A	C*
J1	PL12A	7	LUM0_SPLL_FB_A	T
J2	PL12B	7	LUM0_SPLL_FB_A	C
J3	PL13A	7		T*
J4	PL13B	7		C*
J7	PL14A	7		T
GND	GND	7		
VCC	VCC			
J6	PL14B	7		C
GND	GND	7		
VCC	VCC			
K1	PL28A	7	LUM1_SPLL_IN_A	T*
K2	PL28B	7	LUM1_SPLL_IN_A	C*
J5	PL29A	7	LUM1_SPLL_FB_A	T
K5	PL29B	7	LUM1_SPLL_FB_A	C
K7	PL30A	7		T*
K6	PL30B	7		C*
VCC	VCC			
L6	PL31A	7		T
L7	PL31B	7		C
GND	GND	7		

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
L1	PL32A	7	LDQS32	T*
L2	PL32B	7		C*
M7	PL33A	7		T
L5	PL33B	7		C
L3	PL34A	7		T*
L4	PL34B	7		C*
M1	PL35A	7	PCLKT7_0	T
GND	GND	7		
M2	PL35B	7	PCLKC7_0	C
GND	GND	7		
VCC	VCC			
VCC	VCC			
M6	PL37A	6	PCLKT6_0	T*
VCCAUX	VCCAUX			
M5	PL37B	6	PCLKC6_0	C*
M3	PL38A	6	VREF2_6	T
M4	PL38B	6	VREF1_6	C
VCC	VCC			
N7	PL41A	6	LLM2_SPLL_IN_A	T*
GND	GND	6		
N6	PL41B	6	LLM2_SPLL_IN_A	C*
N1	PL42A	6	LLM2_SPLL_FB_A	T
N2	PL42B	6	LLM2_SPLL_FB_A	C
VCC	VCC			
GND	GND	6		
P6	PL48A	6	LDQS48	T*
N5	PL48B	6		C*
P1	PL49A	6		T
P2	PL49B	6		C
P3	PL50A	6		T*
P4	PL50B	6		C*
P5	PL51A	6		T
GND	GND	6		
P7	PL51B	6		C
GND	GND	6		
VCC	VCC			
VCCAUX	VCCAUX			
VCC	VCC			
R1	PL57A	6	LLM0_GPLL_IN_A/ LDQS57	T*
GND	GND	6		
R2	PL57B	6	LLM0_GPLL_IN_A	C*
R3	PL58A	6	LLM0_GPLL_FB_A	T
R4	PL58B	6	LLM0_GPLL_FB_A	C

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
R6	PL59A	6	LLM0_GDLLT_IN_A	T*
R5	PL59B	6	LLM0_GDLLC_IN_A	C*
T1	PL60A	6	LLM0_GDLLT_FB_A	T
T2	PL60B	6	LLM0_GDLLC_FB_A	C
GND	GND	6		
VCCAUX	VCCAUX			
R7	LLM0_PLLCAP	6		
T6	PL62A	6		T*
T7	PL62B	6		C*
U1	PL63A	6		T
U2	PL63B	6		C
T3	PL64A	6		T*
U3	PL64B	6		C*
VCC	VCC			
U5	PL65B	6		C
GND	GND	6		
V5	PL66A	6	LDQS66	T*
U4	PL66B	6		C*
V1	PL67A	6		T
V3	PL67B	6		C
W1	PL68A	6		T*
Y1	PL68B	6		C*
AA1	PL69A	6		T
GND	GND	6		
AA2	PL69B	6		C
V4	TCK	9		
Y2	TDI	9		
Y3	TMS	9		
GND	GND	6		
W3	TDO	9		
W4	VCCJ	9		
W5	PB2A	5		T
GND	GND	5		
Y4	PB2B	5		C
W6	PB3A	5		T
V6	PB3B	5		C
AA3	PB4A	5		T
AB2	PB4B	5		C
T8	PB5A	5		T
VCC	VCC			
U7	PB5B	5		C
U8	PB6A	5	BDQS6	T
GND	GND	5		
T9	PB6B	5		C

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
V8	PB7A	5		T
W8	PB7B	5		C
Y6	PB8A	5		T
Y5	PB8B	5		C
AB3	PB9A	5		T
AB4	PB9B	5		C
AB5	PB10A	5		T
GND	GND	5		
AA6	PB10B	5		C
VCCAUX	VCCAUX			
VCC	VCC			
VCC	VCC			
VCCAUX	VCCAUX			
V9	PB31A	5		T
U9	PB31B	5		C
U10	PB32A	5		T
GND	GND	5		
T10	PB32B	5		C
W9	PB33A	5	BDQS33	T
Y8	PB33B	5		C
VCC	VCC			
AA7	PB34A	5	VREF2_5	T
Y7	PB34B	5	VREF1_5	C
AB6	PB35A	5	PCLKT5_0	T
AB7	PB35B	5	PCLKC5_0	C
GND	GND	5		
AA8	PB40A	4	PCLKT4_0	T
AB8	PB40B	4	PCLKC4_0	C
AA9	PB41A	4	VREF2_4	T
VCC	VCC			
Y9	PB41B	4	VREF1_4	C
AB9	PB42A	4	BDQS42	T
GND	GND	4		
AB10	PB42B	4		C
AA10	PB43A	4		T
Y11	PB43B	4		C
GND	GND	4		
V10	PB47A	4		T
U11	PB47B	4		C
VCCAUX	VCCAUX			
V11	PB48A	4		T
W11	PB48B	4		C
AA11	PB49A	4		T
AB11	PB49B	4		C

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
T11	PB50A	4		T
U12	PB50B	4		C
GND	GND	4		
AA12	PB51A	4	BDQS51	T
Y12	PB51B	4		C
VCC	VCC			
V12	PB52A	4		T
W12	PB52B	4		C
AB12	PB53A	4		T
AA13	PB53B	4		C
T12	PB54A	4		T
U13	PB54B	4		C
V13	PB55A	4		T
T13	PB55B	4		C
GND	GND	4		
AB13	PB56A	4		T
GND	GND	4		
AB14	PB56B	4		C
U14	PB57A	4		T
T14	PB57B	4		C
AA14	PB58A	4		T
Y14	PB58B	4		C
W14	PB59A	4		T
VCC	VCC			
V14	PB59B	4		C
AB15	PB60A	4	BDQS60	T
GND	GND	4		
AA15	PB60B	4		C
V15	PB61A	4		T
U15	PB61B	4		C
AB16	PB62A	4		T
AA16	PB62B	4		C
AB17	PB63A	4		T
AA17	PB63B	4		C
Y15	PB64A	4		T
GND	GND	4		
W15	PB64B	4		C
AB20	PB65A	4		T
AB21	PB65B	4		C
VCCAUX	VCCAUX			
AA21	PB66A	4		T
AA20	PB66B	4		C
AB19	PB67A	4		T
AB18	PB67B	4		C

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
Y22	PB68A	4		T
Y21	PB68B	4		C
GND	GND	4		
Y17	PB69A	4	BDQS69	T
Y18	PB69B	4		C
VCC	VCC			
Y16	PB70A	4		T
W17	PB70B	4		C
Y19	PB71A	4		T
Y20	PB71B	4		C
W19	PB72A	4		T
W18	PB72B	4		C
V17	PB73A	4		T
V18	PB73B	4		C
GND	GND	4		
W20	CFG2	8		
V20	CFG1	8		
V19	CFG0	8		
V22	PROGRAMN	8		
W22	CCLK	8		
U18	INITN	8		
U22	DONE	8		
GND	GND	8		
U20	PR68B	8	WRITEN	C
U21	PR68A	8	CS1N	T
U17	PR67B	8	CSN	C
U16	PR67A	8	D0	T
T16	PR66B	8	D1	C
T17	PR66A	8	D2	T
T22	PR65B	8	D3	C
GND	GND	8		
R22	PR65A	8	D4	T
T15	PR64B	8	D5	C
VCC	VCC			
R17	PR64A	8	D6	T
T20	PR63B	8	D7	C
T21	PR63A	8	DI	T
R21	PR62B	8	DOUT_CS0N	C
R20	PR62A	8	BUSY	T
R16	RLM0_PLLCAP	3		
R18	PR60B	3	RLM0_GDLLC_FB_A	C
GND	GND	3		
R19	PR60A	3	RLM0_GDLLT_FB_A	T
P22	PR59B	3	RLM0_GDLLC_IN_A	C*

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
P21	PR59A	3	RLM0_GDLLT_IN_A	T*
P16	PR58B	3	RLM0_GPLLC_IN_A	C
P17	PR58A	3	RLM0_GPLLT_IN_A	T
P20	PR57B	3	RLM0_GPLLC_FB_A	C*
GND	GND	3		
P19	PR57A	3	RLM0_GPLLT_FB_A/ RDQS57	T*
GND	GND	3		
VCC	VCC			
VCCAUX	VCCAUX			
VCC	VCC			
P18	PR51B	3		C
N16	PR51A	3		T
GND	GND	3		
N22	PR50B	3		C*
N21	PR50A	3		T*
N17	PR49B	3		C
N18	PR49A	3		T
M22	PR48B	3		C*
M21	PR48A	3	RDQS48	T*
GND	GND	3		
M16	PR47B	3		C
M17	PR47A	3		T
M20	PR46B	3		C*
VCC	VCC			
M19	PR46A	3		T*
M18	PR45B	3		C
L16	PR45A	3		T
L22	PR44B	3		C*
GND	GND	3		
L21	PR44A	3		T*
K22	PR42B	3	RLM2_SPLLFB_A	C
K21	PR42A	3	RLM2_SPLLTFB_A	T
L17	PR41B	3	RLM2_SPLLFB_IN_A	C*
GND	GND	3		
L18	PR41A	3	RLM2_SPLLTIN_A	T*
GND	GND	3		
L20	PR40B	3		C
L19	PR40A	3		T
VCC	VCC			
K16	PR39B	3		C*
K17	PR39A	3		T*
J16	PR38B	3	VREF2_3	C
K18	PR38A	3	VREF1_3	T

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
J22	PR37B	3	PCLKC3_0	C*
J21	PR37A	3	PCLKT3_0	T*
VCCAUX	VCCAUX			
VCC	VCC			
VCC	VCC			
GND	GND	2		
H22	PR35B	2	PCLKC2_0	C
H21	PR35A	2	PCLKT2_0	T
GND	GND	2		
J17	PR34B	2		C*
J18	PR34A	2		T*
J20	PR33B	2		C
J19	PR33A	2		T
H16	PR32B	2		C*
H17	PR32A	2	RDQS32	T*
GND	GND	2		
G22	PR31B	2		C
GND	GND	2		
G21	PR31A	2		T
H20	PR30B	2		C*
VCC	VCC			
H19	PR30A	2		T*
G16	PR29B	2	RUM1_SPLL_C_FB_A	C
H18	PR29A	2	RUM1_SPLL_T_FB_A	T
F22	PR28B	2	RUM1_SPLL_C_IN_A	C*
F21	PR28A	2	RUM1_SPLL_T_IN_A	T*
GND	GND	2		
G20	PR26B	2		C
F20	PR26A	2		T
GND	GND	2		
G17	PR25B	2		C*
F17	PR25A	2		T*
VCC	VCC			
GND	GND	2		
E22	PR14B	2		C
D22	PR14A	2		T
VCC	VCC			
E20	PR13B	2		C*
D20	PR13A	2		T*
D19	PR12B	2	RUM0_SPLL_C_FB_A	C
E19	PR12A	2	RUM0_SPLL_T_FB_A	T
F18	PR11B	2	RUM0_SPLL_C_IN_A	C*
F19	PR11A	2	RUM0_SPLL_T_IN_A	T*
GND	GND	2		

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
VCCAUX	VCCAUX			
E18	PR9B	2	VREF2_2	C
GND	GND	2		
D18	PR9A	2	VREF1_2	T
GND	GND	2		
VCC	VCC			
VCC	VCC			
F16	XRES			
VCC	VCC			
VCC	VCC			
C22	URC_SQ_VCCRX0	1		
A21	URC_SQ_HDINP0	1		T
B22	URC_SQ_VCCIB0	1		
B21	URC_SQ_HDINN0	1		C
C19	URC_SQ_VCCTX0	1		
A18	URC_SQ_HDOUTP0	1		T
A19	URC_SQ_VCCOB0	1		
B18	URC_SQ_HDOUTN0	1		C
C18	URC_SQ_VCCTX1	1		
B17	URC_SQ_HDOUTN1	1		C
C17	URC_SQ_VCCOB1	1		
A17	URC_SQ_HDOUTP1	1		T
C21	URC_SQ_VCCRX1	1		
B20	URC_SQ_HDINN1	1		C
C20	URC_SQ_VCCIB1	1		
A20	URC_SQ_HDINP1	1		T
B16	URC_SQ_VCCAUX33	1		
E17	URC_SQ_REFCLKN	1		C
D17	URC_SQ_REFCLKP	1		T
C16	URC_SQ_VCCP	1		
A12	URC_SQ_HDINP2	1		T
C12	URC_SQ_VCCIB2	1		
B12	URC_SQ_HDINN2	1		C
C11	URC_SQ_VCCRX2	1		
A15	URC_SQ_HDOUTP2	1		T
C15	URC_SQ_VCCOB2	1		
B15	URC_SQ_HDOUTN2	1		C
C14	URC_SQ_VCCTX2	1		
B14	URC_SQ_HDOUTN3	1		C
A13	URC_SQ_VCCOB3	1		
A14	URC_SQ_HDOUTP3	1		T
C13	URC_SQ_VCCTX3	1		
B11	URC_SQ_HDINN3	1		C
B10	URC_SQ_VCCIB3	1		

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
A11	URC_SQ_HDINP3	1		T
C10	URC_SQ_VCCRX3	1		
VCC	VCC			
VCC	VCC			
GND	GND	1		
E13	PT46B	1		C
GND	GND	1		
D12	PT46A	1		T
GND	GND	1		
A9	PT45B	1		C
A8	PT45A	1		T
A7	PT44B	1		C
A6	PT44A	1		T
E12	PT43B	1		C
F12	PT43A	1		T
A5	PT42B	1		C
A4	PT42A	1		T
GND	GND	1		
B7	PT41B	1		C
B8	PT41A	1		T
VCC	VCC			
G11	PT40B	1		C
E11	PT40A	1		T
D11	PT39B	1	VREF2_1	C
D10	PT39A	1	VREF1_1	T
G10	PT38B	1	PCLKC1_0	C
F11	PT38A	1	PCLKT1_0	T
GND	GND	1		
G9	PT37B	0	PCLKC0_0	C
GND	GND	0		
F9	PT37A	0	PCLKT0_0	T
C9	PT36B	0	VREF2_0	C
D9	PT36A	0	VREF1_0	T
A2	PT35B	0		C
A3	PT35A	0		T
B3	PT34B	0		C
C4	PT34A	0		T
E10	PT33B	0		C
VCC	VCC	0		
F10	PT33A	0		T
C7	PT32B	0		C
GND	GND	0		
B6	PT32A	0		T
C6	PT31B	0		C

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
C5	PT31A	0		T
C8	PT30B	0		C
D8	PT30A	0		T
E8	PT29B	0		C
VCCAUX	VCCAUX			
E9	PT29A	0		T
GND	GND	0		
VCC	VCC			
VCC	VCC			
VCCAUX	VCCAUX			
F8	PT10B	0		C
GND	GND	0		
G8	PT10A	0		T
GND	GND	0		
F7	PT9B	0		C
G7	PT9A	0		T
C3	PT8B	0		C
D4	PT8A	0		T
F6	PT7B	0		C
E6	PT7A	0		T
E5	PT6B	0		C
D6	PT6A	0		T
GND	GND	0		
D3	PT5B	0		C
E3	PT5A	0		T
VCC	VCC			
D5	PT4B	0		C
E4	PT4A	0		T
C2	PT3B	0		C
B2	PT3A	0		T
B1	PT2B	0		C
C1	PT2A	0		T
GND	GND	0		
R8	VCCPLL			
H15	VCCPLL			
H8	VCCPLL			
R15	VCCPLL			
J10	VCC			
J11	VCC			
J12	VCC			
J13	VCC			
K14	VCC			
K9	VCC			
L14	VCC			

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
L9	VCC			
M14	VCC			
M9	VCC			
N14	VCC			
N9	VCC			
P10	VCC			
P11	VCC			
P12	VCC			
P13	VCC			
VCC	VCC			
B5	VCCIO0	0		
B9	VCCIO0	0		
E7	VCCIO0	0		
H9	VCCIO0	0		
D13	VCCIO1	1		
E16	VCCIO1	1		
H14	VCCIO1	1		
E21	VCCIO2	2		
G18	VCCIO2	2		
J15	VCCIO2	2		
K19	VCCIO2	2		
N19	VCCIO3	3		
P15	VCCIO3	3		
T18	VCCIO3	3		
V21	VCCIO3	3		
AA18	VCCIO4	4		
R14	VCCIO4	4		
V16	VCCIO4	4		
W13	VCCIO4	4		
AA5	VCCIO5	5		
R9	VCCIO5	5		
V7	VCCIO5	5		

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
W10	VCCIO5	5		
N4	VCCIO6	6		
P8	VCCIO6	6		
T5	VCCIO6	6		
V2	VCCIO6	6		
E2	VCCIO7	7		
G5	VCCIO7	7		
J8	VCCIO7	7		
K4	VCCIO7	7		
AA22	VCCIO8	8		
U19	VCCIO8	8		
H11	VCCAUX			
H12	VCCAUX			
L15	VCCAUX			
L8	VCCAUX			
M15	VCCAUX			
M8	VCCAUX			
R11	VCCAUX			
R12	VCCAUX			
VCCAUX	VCCAUX			
A1	GND			
A10	GND			
A16	GND			
A22	GND			
AA19	GND			
AA4	GND			
AB1	GND			
AB22	GND			
B13	GND			
B19	GND			
B4	GND			
D16	GND			
D2	GND			
D21	GND			
D7	GND			
G19	GND			

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
G4	GND			
H10	GND			
H13	GND			
J14	GND			
J9	GND			
K10	GND			
K11	GND			
K12	GND			
K13	GND			
K15	GND			
K20	GND			
K3	GND			
K8	GND			
L10	GND			
L11	GND			
L12	GND			
L13	GND			
M10	GND			
M11	GND			
M12	GND			
M13	GND			
N10	GND			
N11	GND			
N12	GND			
N13	GND			
N15	GND			
N20	GND			
N3	GND			
N8	GND			
P14	GND			
P9	GND			
R10	GND			
R13	GND			
T19	GND			
T4	GND			
W16	GND			
W2	GND			
W21	GND			
W7	GND			
Y10	GND			
Y13	GND			
D14	NC			
D15	NC			
E14	NC			

**ECP2M-35 Logic Signal Connections: 484 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
E15	NC			
F13	NC			
F14	NC			
F15	NC			
G12	NC			
G13	NC			
G14	NC			
G15	NC			
U6	NC			

\*Supports dedicated LVDS outputs.

**ECP2-50 Logic Signal Connections: 672 fpBGA**

Ball Number	Ball Function	Bank	Dual Function	Differential
D2	PL2A	7	VREF2_7	T*
D1	PL2B	7	VREF1_7	C*
GND	GND	7		
F6	PL5A	7		T
F5	PL5B	7		C
VCCIO	VCCIO	7		
E4	PL6A	7		T*
E3	PL6B	7		C*
E2	PL7A	7		T
E1	PL7B	7		C
GND	GND	7		
H6	PL8A	7	LDQS8	T*
H5	PL8B	7		C*
F2	PL9A	7		T
VCCIO	VCCIO	7		
F1	PL9B	7		C
H8	PL10A	7		T*
J9	PL10B	7		C*
G4	PL11A	7		T
GND	GND	7		
G3	PL11B	7		C
H7	PL12A	7		T*
J8	PL12B	7		C*
G2	PL13A	7		T
G1	PL13B	7		C
H3	PL14A	7		T*
VCCIO	VCCIO	7		
H4	PL14B	7		C*
J5	PL15A	7		T
J4	PL15B	7		C
J3	PL16A	7	LDQS16	T*
GND	GND	7		
K4	PL16B	7		C*
H1	PL17A	7		T
H2	PL17B	7		C
VCCIO	VCCIO	7		
K6	PL18A	7		T*
K7	PL18B	7		C*
J1	PL19A	7		T
J2	PL19B	7		C
GND	GND	7		
VCCIO	VCCIO	7		
K3	PL23A	7		T

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
K2	PL23B	7		C
GND	GND	7		
K1	PL24A	7	LDQS24	T*
L2	PL24B	7		C*
L1	PL25A	7	LUM0_SPLL_IN_A	T
VCCIO	VCCIO	7		
M2	PL25B	7	LUM0_SPLL_IN_A	C
M1	PL26A	7	LUM0_SPLL_FB_A	T
N2	PL26B	7	LUM0_SPLL_FB_A	C
GND	GND	7		
M8	LUM0_VCCPLL	7		
VCCIO	VCCIO	7		
GND	GND	7		
N1	PL37A	7		
L8	PL38A	7		T
K8	PL38B	7		C
VCCIO	VCCIO	7		
L6	PL39A	7		T*
K5	PL39B	7		C*
L7	PL40A	7		T
L5	PL40B	7		C
GND	GND	7		
P1	PL41A	7	LDQS41	T*
P2	PL41B	7		C*
M6	PL42A	7		T
VCCIO	VCCIO	7		
N8	PL42B	7		C
R1	PL43A	7		T*
R2	PL43B	7		C*
M7	PL44A	7	PCLKT7_0	T
GND	GND	7		
N9	PL44B	7	PCLKC7_0	C
GND	GND	7		
M4	PL46A	6	PCLKT6_0	T*
M5	PL46B	6	PCLKC6_0	C*
N7	PL47A	6	VREF2_6	T
P9	PL47B	6	VREF1_6	C
N3	PL48A	6		T*
VCCIO	VCCIO	6		
N4	PL48B	6		C*
N5	PL49A	6		T
P7	PL49B	6		C
T1	PL50A	6	LDQS50	T*
GND	GND	6		

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
T2	PL50B	6		C*
P8	PL51A	6		T
P6	PL51B	6		C
VCCIO	VCCIO	6		
P5	PL52A	6		T*
P4	PL52B	6		C*
U1	PL53A	6		T
V1	PL53B	6		C
GND	GND	6		
P3	PL54A	6		T*
R3	PL54B	6		C*
R4	PL55A	6		T
U2	PL55B	6		C
VCCIO	VCCIO	6		
V2	PL56A	6		T*
W2	PL56B	6		C*
T6	PL57A	6		T
R5	PL57B	6		C
GND	GND	6		
R6	PL58A	6	LDQS58	T*
R7	PL58B	6		C*
W1	PL59A	6		T
VCCIO	VCCIO	6		
Y2	PL59B	6		C
Y1	PL60A	6	LLM0_GDLLT_IN_A	T*
AA2	PL60B	6	LLM0_GDLLC_IN_A	C*
T5	PL61A	6	LLM0_GDLLT_FB_A	T
GND	GND	6		
T7	PL61B	6	LLM0_GDLLC_FB_D	C
GND	GND	6		
R8	LLM0_VCCPLL	6		
T8	LLM0_PLLCAP	6		
U3	PL63A	6	LLM0_GPLLT_IN_A	T*
U4	PL63B	6	LLM0_GPLLC_IN_A	C*
V3	PL64A	6	LLM0_GPLLT_FB_A	T
U5	PL64B	6	LLM0_GPLLC_FB_A	C
V4	PL65A	6		T*
VCCIO	VCCIO	6		
V5	PL65B	6		C*
Y3	PL66A	6		T
Y4	PL66B	6		C
W3	PL67A	6	LDQS67	T*
GND	GND	6		
W4	PL67B	6		C*

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
AA1	PL68A	6		T
AB1	PL68B	6		C
VCCIO	VCCIO	6		
U8	PL69A	6		T*
U7	PL69B	6		C*
V8	PL70A	6		T
U6	PL70B	6		C
GND	GND	6		
W6	PL71A	6		T*
W5	PL71B	6		C*
AC1	PL72A	6		T
AD1	PL72B	6		C
VCCIO	VCCIO	6		
Y6	PL73A	6		T*
Y5	PL73B	6		C*
AE2	PL74A	6		T
AD2	PL74B	6		C
GND	GND	6		
AB3	PL75A	6	LDQS75	T*
AB2	PL75B	6		C*
W7	PL76A	6		T
VCCIO	VCCIO	6		
W8	PL76B	6		C
Y7	PL77A	6		T*
Y8	PL77B	6		C*
AC2	PL78A	6		T
GND	GND	6		
AD3	PL78B	6		C
AC3	TCK	9		
AA8	TDI	9		
AB4	TMS	9		
GND	GND	6		
AA5	TDO	9		
AB5	VCCJ	9		
AE3	PB2A	5	VREF2_5	T
AF3	PB2B	5	VREF1_5	C
AC4	PB3A	5		T
AD4	PB3B	5		C
AE4	PB4A	5		T
AF4	PB4B	5		C
VCCIO	VCCIO	5		
V9	PB5A	5		T
W9	PB5B	5		C
GND	GND	5		

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
AA6	PB6A	5	BDQS6	T
AB6	PB6B	5		C
AC5	PB7A	5		T
AD5	PB7B	5		C
AA7	PB8A	5		T
AB7	PB8B	5		C
VCCIO	VCCIO	5		
AE5	PB9A	5		T
AF5	PB9B	5		C
AC7	PB10A	5		T
AD7	PB10B	5		C
GND	GND	5		
VCCIO	VCCIO	5		
W10	PB20A	5		T
Y10	PB20B	5		C
W11	PB21A	5		T
AA10	PB21B	5		C
AC8	PB22A	5		T
AD8	PB22B	5		C
VCCIO	VCCIO	5		
AB8	PB23A	5		T
AB10	PB23B	5		C
GND	GND	5		
AE6	PB24A	5	BDQS24	T
AF6	PB24B	5		C
AA11	PB25A	5		T
AC9	PB25B	5		C
AB9	PB26A	5		T
AD9	PB26B	5		C
VCCIO	VCCIO	5		
Y11	PB27A	5		T
AB11	PB27B	5		C
AE7	PB28A	5		T
AF7	PB28B	5		C
GND	GND	5		
AC10	PB29A	5		T
GND	GND	5		
AD10	PB29B	5		C
AA12	PB30A	5		T
W12	PB30B	5		C
AB12	PB31A	5		T
VCCIO	VCCIO	5		
Y12	PB31B	5		C
AD12	PB32A	5		T

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
AC12	PB32B	5		C
AC13	PB33A	5	BDQS33	T
GND	GND	5		
AA13	PB33B	5		C
AD13	PB34A	5		T
AC14	PB34B	5		C
AE8	PB35A	5		T
VCCIO	VCCIO	5		
AF8	PB35B	5		C
AB15	PB36A	5		T
Y13	PB36B	5		C
AE9	PB37A	5		T
GND	GND	5		
AF9	PB37B	5		C
W13	PB38A	5		T
AA14	PB38B	5		C
AE10	PB39A	5		T
AF10	PB39B	5		C
W14	PB40A	5		T
AB13	PB40B	5		C
VCCIO	VCCIO	5		
Y14	PB41A	5		T
AB14	PB41B	5		C
GND	GND	5		
AE11	PB42A	5	BDQS42	T
AF11	PB42B	5		C
AD14	PB43A	5		T
AA15	PB43B	5		C
AE12	PB44A	5	PCLKT5_0	T
AF12	PB44B	5	PCLKC5_0	C
VCCIO	VCCIO	5		
GND	GND	5		
AD15	PB49A	4	PCLKT4_0	T
VCCIO	VCCIO	4		
AC15	PB49B	4	PCLKC4_0	C
AE13	PB50A	4		T
AF13	PB50B	4		C
AB17	PB51A	4	BDQS51	T
GND	GND	4		
Y15	PB51B	4		C
AE14	PB52A	4		T
AF14	PB52B	4		C
AA16	PB53A	4		T
VCCIO	VCCIO	4		

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
W15	PB53B	4		C
AC17	PB54A	4		T
AB16	PB54B	4		C
AE15	PB55A	4		T
GND	GND	4		
AF15	PB55B	4		C
AE16	PB56A	4		T
AF16	PB56B	4		C
Y16	PB57A	4		T
AB18	PB57B	4		C
AD17	PB58A	4		T
AD18	PB58B	4		C
VCCIO	VCCIO	4		
AC18	PB59A	4		T
AD19	PB59B	4		C
GND	GND	4		
AC19	PB60A	4	BDQS60	T
AE17	PB60B	4		C
AB19	PB61A	4		T
AE19	PB61B	4		C
AF17	PB62A	4		T
AE18	PB62B	4		C
VCCIO	VCCIO	4		
W16	PB63A	4		T
AA17	PB63B	4		C
AF18	PB64A	4		T
AF19	PB64B	4		C
GND	GND	4		
AA19	PB65A	4		T
GND	GND	4		
W17	PB65B	4		C
Y19	PB66A	4		T
Y17	PB66B	4		C
AF20	PB67A	4		T
VCCIO	VCCIO	4		
AE20	PB67B	4		C
AA20	PB68A	4		T
W18	PB68B	4		C
AD20	PB69A	4	BDQS69	T
GND	GND	4		
AE21	PB69B	4		C
AF21	PB70A	4		T
AF22	PB70B	4		C
VCCIO	VCCIO	4		

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
GND	GND	4		
AE22	PB74A	4		T
AD22	PB74B	4		C
AF23	PB75A	4		T
AE23	PB75B	4		C
AD23	PB76A	4		T
AC23	PB76B	4		C
VCCIO	VCCIO	4		
AB20	PB77A	4		T
AC20	PB77B	4		C
GND	GND	4		
AB21	PB78A	4	BDQS78	T
AC22	PB78B	4		C
W19	PB79A	4		T
AA21	PB79B	4		C
AF24	PB80A	4		T
AE24	PB80B	4		C
VCCIO	VCCIO	4		
Y20	PB81A	4		T
AB22	PB81B	4		C
Y21	PB82A	4	VREF2_4	T
AB23	PB82B	4	VREF1_4	C
GND	GND	4		
AD24	CFG2	8		
W20	CFG1	8		
AC24	CFG0	8		
GND	GND	8		
V19	PROGRAMN	8		
AA22	CCLK	8		
AB24	INITN	8		
AD25	DONE	8		
GND	GND	8		
W21	PR77B	8	WRITEN	C
Y22	PR77A	8	CS1N	T
AC25	PR76B	8	CSN	C
AB25	PR76A	8	D0	T
VCCIO	VCCIO	8		
AD26	PR75B	8	D1	C
AC26	PR75A	8	D2	T
GND	GND	8		
Y23	PR74B	8	D3	C
GND	GND	8		
W22	PR74A	8	D4	T
AA25	PR73B	8	D5	C

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
AB26	PR73A	8	D6	T
W23	PR72B	8	D7	C
VCCIO	VCCIO	8		
V22	PR72A	8	DI	T
Y24	PR71B	8	DOUT, CS0N	C
Y25	PR71A	8	BUSY	T
W24	PR70B	3		C
GND	GND	3		
V23	PR70A	3		T
AA26	PR69B	3		C*
Y26	PR69A	3		T*
U21	PR68B	3		C
VCCIO	VCCIO	3		
U19	PR68A	3		T
W25	PR67B	3		C*
GND	GND	3		
W26	PR67A	3	RDQS67	T*
GND	GND	3		
V24	PR66B	3		C
V25	PR66A	3		T
V26	PR65B	3		C*
U26	PR65A	3		T*
VCCIO	VCCIO	3		
U22	PR64B	3	RLM0_GPLL_C_FB_A	C
U23	PR64A	3	RLM0_GPLL_T_FB_A	T
U24	PR63B	3	RLM0_GPLL_C_IN_A	C*
U25	PR63A	3	RLM0_GPLL_T_IN_A	T*
R20	RLM0_PLLCAP	3		
P18	RLM0_VCCPLL	3		
GND	GND	3		
T19	PR61B	3	RLM0_GDLL_C_FB_A	C
U20	PR61A	3	RLM0_GDLL_T_FB_A	T
GND	GND	3		
T25	PR60B	3	RLM0_GDLL_C_IN_A	C*
T26	PR60A	3	RLM0_GDLL_T_IN_A	T*
T20	PR59B	3		C
T22	PR59A	3		T
VCCIO	VCCIO	3		
R26	PR58B	3		C*
R25	PR58A	3	RDQS58	T*
GND	GND	3		
R22	PR57B	3		C
GND	GND	3		
T21	PR57A	3		T

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
P26	PR56B	3		C*
P25	PR56A	3		T*
R24	PR55B	3		C
VCCIO	VCCIO	3		
R23	PR55A	3		T
P20	PR54B	3		C*
R19	PR54A	3		T*
P21	PR53B	3		C
GND	GND	3		
P19	PR53A	3		T
P23	PR52B	3		C*
P22	PR52A	3		T*
N22	PR51B	3		C
VCCIO	VCCIO	3		
R21	PR51A	3		T
N26	PR50B	3		C*
GND	GND	3		
N25	PR50A	3	RDQS50	T*
GND	GND	3		
N19	PR49B	3		C
N20	PR49A	3		T
M26	PR48B	3		C*
M25	PR48A	3		T*
VCCIO	VCCIO	3		
N18	PR47B	3	VREF2_3	C
N21	PR47A	3	VREF1_3	T
L26	PR46B	3	PCLKC3_0	C*
L25	PR46A	3	PCLKT3_0	T*
GND	GND	2		
N24	PR44B	2	PCLKC2_0	C
M23	PR44A	2	PCLKT2_0	T
GND	GND	2		
L21	PR43B	2		C*
K22	PR43A	2		T*
M24	PR42B	2		C
N23	PR42A	2		T
VCCIO	VCCIO	2		
K26	PR41B	2		C*
K25	PR41A	2	RDQS41	T*
GND	GND	2		
M20	PR40B	2		C
GND	GND	2		
M19	PR40A	2		T
L22	PR39B	2		C*

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
M22	PR39A	2		T*
K21	PR38B	2		C
VCCIO	VCCIO	2		
M21	PR38A	2		T
K24	PR37B	2		C*
J24	PR37A	2		T*
GND	GND	2		
VCCIO	VCCIO	2		
L20	RUM0_VCCPLL	2		
GND	GND	2		
J26	PR26B	2	RUM0_SPLLC_FB_A	C
J25	PR26A	2	RUM0_SPLLT_FB_A	T
J23	PR25B	2	RUM0_SPLLC_IN_A	C
K23	PR25A	2	RUM0_SPLLT_IN_A	T
VCCIO	VCCIO	2		
H26	PR24B	2		C*
H25	PR24A	2	RDQS24	T*
GND	GND	2		
H24	PR23B	2		C
GND	GND	2		
H23	PR23A	2		T
VCCIO	VCCIO	2		
G26	PR19B	2		C
GND	GND	2		
G25	PR19A	2		T
F26	PR18B	2		C*
F25	PR18A	2		T*
K20	PR17B	2		C
VCCIO	VCCIO	2		
L19	PR17A	2		T
E26	PR16B	2		C*
GND	GND	2		
E25	PR16A	2	RDQS16	T*
GND	GND	2		
J22	PR15B	2		C
H22	PR15A	2		T
G24	PR14B	2		C*
G23	PR14A	2		T*
VCCIO	VCCIO	2		
K19	PR13B	2		C
J19	PR13A	2		T
D26	PR12B	2		C*
C26	PR12A	2		T*
F22	PR11B	2		C

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
E24	PR11A	2		T
GND	GND	2		
D25	PR10B	2		C*
C25	PR10A	2		T*
D24	PR9B	2		C
B25	PR9A	2		T
VCCIO	VCCIO	2		
H21	PR8B	2		C*
G22	PR8A	2	RDQS8	T*
GND	GND	2		
B24	PR7B	2		C
GND	GND	2		
C24	PR7A	2		T
D23	PR6B	2		C*
C23	PR6A	2		T*
G21	PR5B	2		C
VCCIO	VCCIO	2		
H20	PR5A	2		T
GND	GND	2		
E22	PR2B	2	VREF2_2	C*
F21	PR2A	2	VREF1_2	T*
GND	GND	1		
E23	PT82B	1	VREF2_1	C
GND	GND	1		
D22	PT82A	1	VREF1_1	T
G20	PT81B	1		C
J18	PT81A	1		T
F20	PT80B	1		C
VCCIO	VCCIO	1		
H19	PT80A	1		T
A24	PT79B	1		C
A23	PT79A	1		T
E21	PT78B	1		C
F19	PT78A	1		T
C22	PT77B	1		C
GND	GND	1		
E20	PT77A	1		T
B22	PT76B	1		C
VCCIO	VCCIO	1		
B23	PT76A	1		T
C20	PT75B	1		C
D20	PT75A	1		T
A22	PT74B	1		C
A21	PT74A	1		T

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
GND	GND	1		
GND	GND	1		
E19	PT71B	1		C
C19	PT71A	1		T
VCCIO	VCCIO	1		
B21	PT70B	1		C
B20	PT70A	1		T
D19	PT69B	1		C
B19	PT69A	1		T
GND	GND	1		
G17	PT68B	1		C
E18	PT68A	1		T
G19	PT67B	1		C
F17	PT67A	1		T
VCCIO	VCCIO	1		
A20	PT66B	1		C
A19	PT66A	1		T
E17	PT65B	1		C
D18	PT65A	1		T
GND	GND	1		
B18	PT64B	1		C
GND	GND	1		
A18	PT64A	1		T
E16	PT63B	1		C
G16	PT63A	1		T
F16	PT62B	1		C
VCCIO	VCCIO	1		
H18	PT62A	1		T
A17	PT61B	1		C
B17	PT61A	1		T
C18	PT60B	1		C
B16	PT60A	1		T
C17	PT59B	1		C
GND	GND	1		
D17	PT59A	1		T
E15	PT58B	1		C
VCCIO	VCCIO	1		
G15	PT58A	1		T
A16	PT57B	1		C
B15	PT57A	1		T
D15	PT56B	1		C
F15	PT56A	1		T
A14	PT55B	1		C
GND	GND	1		

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
B14	PT55A	1		T
GND	GND	1		
C15	PT54B	1		C
A15	PT54A	1		T
A13	PT53B	1		C
B13	PT53A	1		T
VCCIO	VCCIO	1		
H17	PT52B	1		C
H15	PT52A	1		T
D13	PT51B	1		C
C14	PT51A	1		T
GND	GND	1		
G14	PT50B	1		C
E14	PT50A	1		T
A12	PT49B	1		C
B12	PT49A	1		T
VCCIO	VCCIO	1		
F14	PT48B	1	PCLKC1_0	C
D14	PT48A	1	PCLKT1_0	T
GND	GND	1		
H16	XRES	1		
GND	GND	0		
H14	PT46B	0	PCLKC0_0	C
GND	GND	0		
H13	PT46A	0	PCLKT0_0	T
A11	PT45B	0		C
B11	PT45A	0		T
C13	PT44B	0		C
VCCIO	VCCIO	0		
E13	PT44A	0		T
D12	PT43B	0		C
F13	PT43A	0		T
A10	PT42B	0		C
B10	PT42A	0		T
C12	PT41B	0		C
GND	GND	0		
C10	PT41A	0		T
G13	PT40B	0		C
VCCIO	VCCIO	0		
H12	PT40A	0		T
A9	PT39B	0		C
B9	PT39A	0		T
E12	PT38B	0		C
G12	PT38A	0		T

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
A8	PT37B	0		C
GND	GND	0		
B8	PT37A	0		T
GND	GND	0		
E11	PT36B	0		C
C9	PT36A	0		T
A7	PT35B	0		C
B7	PT35A	0		T
VCCIO	VCCIO	0		
F12	PT34B	0		C
D10	PT34A	0		T
H11	PT33B	0		C
G11	PT33A	0		T
GND	GND	0		
A6	PT32B	0		C
B6	PT32A	0		T
D8	PT31B	0		C
C8	PT31A	0		T
VCCIO	VCCIO	0		
F11	PT30B	0		C
E10	PT30A	0		T
E9	PT29B	0		C
D9	PT29A	0		T
GND	GND	0		
G10	PT28B	0		C
GND	GND	0		
H10	PT28A	0		T
A5	PT27B	0		C
B5	PT27A	0		T
C7	PT26B	0		C
VCCIO	VCCIO	0		
D7	PT26A	0		T
E8	PT25B	0		C
F10	PT25A	0		T
F8	PT24B	0		C
H9	PT24A	0		T
C5	PT23B	0		C
GND	GND	0		
D5	PT23A	0		T
B4	PT22B	0		
VCCIO	VCCIO	0		
GND	GND	0		
C4	PT10B	0		C
GND	GND	0		

**ECP2-50 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
C3	PT10A	0		T
A4	PT9B	0		C
A3	PT9A	0		T
B3	PT8B	0		C
VCCIO	VCCIO	0		
B2	PT8A	0		T
D4	PT7B	0		C
D3	PT7A	0		T
C2	PT6B	0		C
C1	PT6A	0		T
G8	PT5B	0		C
GND	GND	0		
G7	PT5A	0		T
E7	PT4B	0		C
VCCIO	VCCIO	0		
F7	PT4A	0		T
E6	PT3B	0		C
E5	PT3A	0		T
G6	PT2B	0	VREF2_0	C
G5	PT2A	0	VREF1_0	T

\*Supports dedicated LVDS outputs.

**ECP2M35 Logic Signal Connections: 672 fpBGA**

Ball Number	Ball Function	Bank	Dual Function	Differential
C2	PL2A	7		T*
C1	PL2B	7		C*
F6	PL3A	7		T
H9	PL3B	7		C
D3	PL4A	7		T*
VCCIO	VCCIO7	7		
D2	PL4B	7		C*
F5	PL5A	7		T
H8	PL5B	7		C
E3	PL6A	7	LDQS6	T*
GND	GND	7		
E2	PL6B	7		C*
J9	PL7A	7		T
E4	PL7B	7		C
VCCIO	VCCIO7	7		
E1	PL8A	7		T*
D1	PL8B	7		C*
J8	PL9A	7	VREF2_7	T
F4	PL9B	7	VREF1_7	C
GND	GND	7		
GND	GND	7		
F3	PL11A	7	LUM0_SPLL_IN_A	T*
F1	PL11B	7	LUM0_SPLL_IN_A	C*
G6	PL12A	7	LUM0_SPLL_FB_A	T
K9	PL12B	7	LUM0_SPLL_FB_A	C
G5	PL13A	7		T*
VCCIO	VCCIO7	7		
G4	PL13B	7		C*
H5	PL14A	7		T
H6	PL14B	7		C
GND	GND	7		
J7	PL16A	7		T
H4	PL16B	7		C
VCCIO	VCCIO7	7		
H3	PL17A	7		T*
G3	PL17B	7		C*
GND	GND	7		
G1	PL19A	7		T*
H1	PL19B	7		C*
J3	PL20A	7		T
J4	PL20B	7		C
VCCIO	VCCIO7	7		
H2	PL21A	7		T*

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
J2	PL21B	7		C*
K7	PL22A	7		T
J6	PL22B	7		C
GND	GND	7		
K5	PL23A	7	LDQS23	T*
L5	PL23B	7		C*
K4	PL24A	7		T
VCCIO	VCCIO7	7		
L4	PL24B	7		C
K3	PL25A	7		T*
L3	PL25B	7		C*
J1	PL26A	7		T
GND	GND	7		
K2	PL26B	7		C
GND	GND	7		
K1	PL28A	7	LUM1_SPLL_IN_A	T*
L1	PL28B	7	LUM1_SPLLC_IN_A	C*
K8	PL29A	7	LUM1_SPLLFB_A	T
M5	PL29B	7	LUM1_SPLLC_FB_A	C
VCCIO	VCCIO7	7		
M4	PL30A	7		T*
M3	PL30B	7		C*
L8	PL31A	7		T
M6	PL31B	7		C
GND	GND	7		
M1	PL32A	7	LDQS32	T*
N1	PL32B	7		C*
N3	PL33A	7		T
VCCIO	VCCIO7	7		
N2	PL33B	7		C
N5	PL34A	7		T*
N4	PL34B	7		C*
M7	PL35A	7	PCLKT7_0	T
GND	GND	7		
M8	PL35B	7	PCLKC7_0	C
GND	GND	7		
P3	PL37A	6	PCLKT6_0	T*
P2	PL37B	6	PCLKC6_0	C*
P5	PL38A	6	VREF2_6	T
N6	PL38B	6	VREF1_6	C
P4	PL39A	6		T*
VCCIO	VCCIO6	6		
R3	PL39B	6		C*
P6	PL40A	6		T

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
P1	PL41A	6	LLM2_SPLLTT_IN_A	T*
GND	GND	0		
R1	PL41B	6	LLM2_SPLLC_IN_A	C*
N8	PL42A	6	LLM2_SPLLTT_FB_A	T
R5	PL42B	6	LLM2_SPLLC_FB_A	C
VCCIO	VCCIO6	6		
T3	PL44A	6		T*
T4	PL44B	6		C*
P8	PL45A	6		T
R6	PL45B	6		C
VCCIO	VCCIO6	6		
T1	PL46A	6		T*
U1	PL46B	6		C*
R7	PL47A	6		T
T5	PL47B	6		C
U3	PL48A	6	LDQS48	T*
U4	PL48B	6		C*
U5	PL49A	6		T
VCCIO	VCCIO6	6		
U6	PL49B	6		C
U2	PL50A	6		T*
V1	PL50B	6		C*
W2	PL51A	6		T
GND	GND	6		
V2	PL51B	6		C
GND	GND	6		
V4	PL55A	6		T*
VCCIO	VCCIO6	6		
V3	PL55B	6		C*
W4	PL57A	6	LLM0_GPLLTT_IN_ALD QS57	T*
GND	GND	6		
W3	PL57B	6	LLM0_GPLLC_IN_A	C*
W1	PL58A	6	LLM0_GPLLTT_FB_A	T
Y1	PL58B	6	LLM0_GPLLC_FB_A	C
VCCIO	VCCIO6	6		
AA1	PL59A	6	LLM0_GDLLT_IN_A	T*
AB1	PL59B	6	LLM0_GDLLC_IN_A	C*
U7	PL60A	6	LLM0_GDLLT_FB_A	T
V6	PL60B	6	LLM0_GDLLC_FB_A	C
GND	GND	6		
T8	LLM0_PLLCAP	6		
W5	PL62A	6		T*
Y4	PL62B	6		C*

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
U8	PL63A	6		T
W6	PL63B	6		C
VCCIO	VCCIO6	6		
Y3	PL64A	6		T*
AA3	PL64B	6		C*
Y5	PL65B	6		C
GND	GND	6		
AB2	PL66A	6	LDQS66	T*
AA4	PL66B	6		C*
Y6	PL67A	6		T
VCCIO	VCCIO6	6		
U9	PL67B	6		C
AA5	PL68A	6		T*
AA6	PL68B	6		C*
Y7	PL69A	6		T
GND	GND	6		
V9	PL69B	6		C
AC3	TCK	6		
W8	TDI	6		
AC4	TMS	6		
GND	GND	6		
V8	TDO	6		
AA7	VCCJ	6		
AB6	PB2A	5		T
GND	GND	5		
Y8	PB2B	5		C
AD1	PB3A	5		T
AD2	PB3B	5		C
AC5	PB4A	5		T
VCCIO	VCCIO5	5		
AA8	PB4B	5		C
AC6	PB5A	5		T
W9	PB5B	5		C
AB7	PB6A	5	BDQS6	T
GND	GND	5		
Y9	PB6B	5		C
AD3	PB7A	5		T
AD4	PB7B	5		C
AA9	PB8A	5		T
VCCIO	VCCIO5	5		
W10	PB8B	5		C
AC7	PB9A	5		T
Y10	PB9B	5		C
AE2	PB10A	5		T

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
GND	GND	5		
AD5	PB10B	5		C
AE4	PB11A	5		T
AE3	PB11B	5		C
AB8	PB12B	5		C
AE5	PB13A	5		T
AD6	PB13B	5		C
VCCIO	VCCIO5	5		
AA10	PB14A	5		T
AC8	PB14B	5		C
GND	GND	5		
W12	PB15A	5	BDQS15	T
AC9	PB15B	5		C
W13	PB16A	5		T
AB10	PB16B	5		C
AF3	PB17A	5		T
AF4	PB17B	5		C
VCCIO	VCCIO5	5		
AF5	PB18A	5		T
AF6	PB18B	5		C
Y12	PB19A	5		T
AB11	PB19B	5		C
GND	GND	5		
AD7	PB20A	5		T
GND	GND	5		
AF7	PB20B	5		C
AD8	PB21A	5		T
AA12	PB21B	5		C
AE8	PB22A	5		T
VCCIO	VCCIO5	5		
AF8	PB22B	5		C
AD9	PB23A	5		T
AC10	PB23B	5		C
AC11	PB24A	5	BDQS24	T
GND	GND	5		
AB12	PB24B	5		C
AD10	PB25A	5		T
Y13	PB25B	5		C
AF9	PB26A	5		T
VCCIO	VCCIO5	5		
AE9	PB26B	5		C
AF10	PB27A	5		T
AE10	PB27B	5		C
AD11	PB28A	5		T

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
GND	GND	5		
AF11	PB28B	5		C
VCCIO	VCCIO5	5		
GND	GND	5		
AA13	PB33A	5	BDQS33	T
AB13	PB33B	5		C
W14	PB34A	5	VREF2_5	T
AC12	PB34B	5	VREF1_5	C
AF12	PB35A	5	PCLKT5_0	T
AD12	PB35B	5	PCLKC5_0	C
VCCIO	VCCIO5	5		
GND	GND	4		
AC13	PB40A	4	PCLKT4_0	T
VCCIO	VCCIO4	4		
Y14	PB40B	4	PCLKC4_0	C
AE14	PB41A	4	VREF2_4	T
AC14	PB41B	4	VREF1_4	C
AB14	PB42A	4	BDQS42	T
GND	GND	4		
AA14	PB42B	4		C
VCCIO	VCCIO4	4		
GND	GND	4		
AF14	PB47A	4		T
AF15	PB47B	4		C
AC15	PB48A	4		T
AE15	PB48B	4		C
AB15	PB49A	4		T
AC16	PB49B	4		C
VCCIO	VCCIO4	4		
AE17	PB50A	4		T
AB16	PB50B	4		C
GND	GND	4		
AA15	PB51A	4	BDQS51	T
AF17	PB51B	4		C
Y15	PB52A	4		T
AC17	PB52B	4		C
AF18	PB53A	4		T
AE18	PB53B	4		C
VCCIO	VCCIO4	4		
W15	PB54A	4		T
AB17	PB54B	4		C
AF20	PB55A	4		T
AE20	PB55B	4		C
GND	GND	4		

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
GND	GND	4		
AB20	PB57A	4		T
AC19	PB57B	4		C
AC20	PB58A	4		T
VCCIO	VCCIO4	4		
AB19	PB59A	4		T
W16	PB59B	4		C
AA17	PB60A	4	BDQS60	T
GND	GND	4		
AA18	PB61A	4		T
Y17	PB61B	4		C
AF21	PB62A	4		T
VCCIO	VCCIO4	4		
AC21	PB63A	4		T
AE21	PB63B	4		C
AF23	PB64A	4		T
GND	GND	4		
W17	PB65A	4		T
AA19	PB65B	4		C
AE23	PB66B	4		C
AF24	PB67A	4		T
AE24	PB67B	4		C
VCCIO	VCCIO4	4		
Y18	PB68B	4		C
GND	GND	4		
AC22	PB69A	4	BDQS69	T
AB21	PB69B	4		C
AD26	PB70B	4		C
AC23	PB71A	4		T
AC25	PB71B	4		C
VCCIO	VCCIO4	4		
W20	PB72B	4		C
V17	PB73A	4		T
AA20	PB73B	4		C
GND	GND	4		
AA21	CFG2	8		
AA22	CFG1	8		
AB23	CFG0	8		
GND	GND	8		
AC26	PROGRAMN	8		
AB24	CCLK	8		
AA23	INITN	8		
AB25	DONE	8		
GND	GND	8		

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
Y19	PR68B	8	WRITEN	C
Y21	PR68A	8	CS1N	T
AB26	PR67B	8	CSN	C
Y22	PR67A	8	D0	T
VCCIO	VCCIO8	8		
W19	PR66B	8	D1	C
Y20	PR66A	8	D2	T
GND	GND	8		
W22	PR65B	8	D3	C
GND	GND	8		
W18	PR65A	8	D4	T
Y23	PR64B	8	D5	C
AA24	PR64A	8	D6	T
W21	PR63B	8	D7	C
VCCIO	VCCIO8	8		
V20	PR63A	8	DI	T
W23	PR62B	8	DOUT_CSON	C
Y24	PR62A	8	BUSY	T
V19	RLM0_PLLCAP	3		
V21	PR60B	3	RLM0_GDLLC_FB_A	C
GND	GND	3		
U19	PR60A	3	RLM0_GDLLT_FB_A	T
AA26	PR59B	3	RLM0_GDLLC_IN_A	C*
Y26	PR59A	3	RLM0_GDLLT_IN_A	T*
V23	PR58B	3	RLM0_GPLLC_IN_A	C
VCCIO	VCCIO3	3		
U20	PR58A	3	RLM0_GPLLT_IN_A	T
W24	PR57B	3	RLM0_GPLLC_FB_A	C*
GND	GND	3		
V24	PR57A	3	RLM0_GPLLT_FB_AR DQS57	T*
GND	GND	3		
U21	PR56A	3		T
W25	PR55B	3		C*
W26	PR55A	3		T*
VCCIO	VCCIO3	3		
U18	PR54B	3		C
U22	PR54A	3		T
V25	PR53B	3		C*
V26	PR53A	3		T*
GND	GND	3		
U24	PR51B	3		C
T24	PR51A	3		T
GND	GND	3		

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
T22	PR50B	3		C*
T23	PR50A	3		T*
U25	PR49B	3		C
U26	PR49A	3		T
VCCIO	VCCIO3	3		
T19	PR48B	3		C*
R19	PR48A	3	RDQS48	T*
GND	GND	3		
R21	PR47B	3		C
GND	GND	3		
R20	PR47A	3		T
T26	PR46B	3		C*
R26	PR46A	3		T*
P21	PR45B	3		C
VCCIO	VCCIO3	3		
P19	PR45A	3		T
R23	PR44B	3		C*
R24	PR44A	3		T*
GND	GND	3		
R22	PR42B	3	RLM2_SPLL_C_FB_A	C
VCCIO	VCCIO3	3		
N19	PR42A	3	RLM2_SPLL_T_FB_A	T
P23	PR41B	3	RLM2_SPLL_C_IN_A	C*
GND	GND	3		
P24	PR41A	3	RLM2_SPLL_T_IN_A	T*
GND	GND	3		
N21	PR40B	3		C
P22	PR40A	3		T
N20	PR39B	3		C*
N22	PR39A	3		T*
VCCIO	VCCIO3	3		
P25	PR38B	3	VREF2_3	C
P26	PR38A	3	VREF1_3	T
M21	PR37B	3	PCLKC3_0	C*
N23	PR37A	3	PCLKT3_0	T*
GND	GND	3		
N24	PR35B	2	PCLKC2_0	C
N25	PR35A	2	PCLKT2_0	T
GND	GND	2		
M22	PR34B	2		C*
M24	PR34A	2		T*
M23	PR33B	2		C
N26	PR33A	2		T
VCCIO	VCCIO2	2		

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
L22	PR32B	2		C*
L24	PR32A	2	RDQS32	T*
GND	GND	2		
L23	PR31B	2		C
GND	GND	2		
M20	PR31A	2		T
M26	PR30B	2		C*
L26	PR30A	2		T*
K22	PR29B	2	RUM1_SPLLC_FB_A	C
VCCIO	VCCIO2	2		
M19	PR29A	2	RUM1_SPLLT_FB_A	T
K25	PR28B	2	RUM1_SPLLC_IN_A	C*
K26	PR28A	2	RUM1_SPLLT_IN_A	T*
GND	GND	2		
K24	PR26B	2		C
K23	PR26A	2		T
GND	GND	2		
L19	PR25B	2		C*
K21	PR25A	2		T*
J23	PR24B	2		C
J24	PR24A	2		T
VCCIO	VCCIO2	2		
K20	PR23B	2		C*
J21	PR23A	2	RDQS23	T*
GND	GND	2		
H21	PR22B	2		C
GND	GND	2		
K18	PR22A	2		T
H22	PR21B	2		C*
J20	PR21A	2		T*
J25	PR20B	2		C
VCCIO	VCCIO2	2		
J26	PR20A	2		T
G21	PR19B	2		C*
J19	PR19A	2		T*
H23	PR18B	2		C
GND	GND	2		
H24	PR18A	2		T
H25	PR17B	2		C*
H26	PR17A	2		T*
G22	PR16B	2		C
VCCIO	VCCIO2	2		
K19	PR16A	2		T
G24	PR15B	2		C*

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
GND	GND	2		
G23	PR15A	2	RDQS15	T*
GND	GND	2		
J18	PR14B	2		C
F22	PR14A	2		T
F23	PR13B	2		C*
F24	PR13A	2		T*
VCCIO	VCCIO2	2		
H20	PR12B	2	RUM0_SPLL_C_FB_A	C
F21	PR12A	2	RUM0_SPLL_T_FB_A	T
G26	PR11B	2	RUM0_SPLL_C_IN_A	C*
F26	PR11A	2	RUM0_SPLL_T_IN_A	T*
GND	GND	2		
E24	PR9B	2	VREF2_2	C
GND	GND	2		
E23	PR9A	2	VREF1_2	T
GND	GND	2		
H19	XRES			
GND	GND	1		
C25	URC_SQ_VCCR0	1		
A24	URC_SQ_HDINP0	1		T
B25	URC_SQ_VCCIB0	1		
B24	URC_SQ_HDINN0	1		C
C22	URC_SQ_VCCTX0	1		
A21	URC_SQ_HDOUTP0	1		T
A22	URC_SQ_VCCOB0	1		
B21	URC_SQ_HDOUTN0	1		C
C21	URC_SQ_VCCTX1	1		
B20	URC_SQ_HDOUTN1	1		C
C20	URC_SQ_VCCOB1	1		
A20	URC_SQ_HDOUTP1	1		T
C24	URC_SQ_VCCR1	1		
B23	URC_SQ_HDINN1	1		C
C23	URC_SQ_VCCIB1	1		
A23	URC_SQ_HDINP1	1		T
B19	URC_SQ_VCCAUX33	1		
E19	URC_SQ_REFCLKN	1		C
D19	URC_SQ_REFCLKP	1		T
C19	URC_SQ_VCCP	1		
A15	URC_SQ_HDINP2	1		T
C15	URC_SQ_VCCIB2	1		
B15	URC_SQ_HDINN2	1		C
C14	URC_SQ_VCCR2	1		
A18	URC_SQ_HDOUTP2	1		T

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
C18	URC_SQ_VCCOB2	1		
B18	URC_SQ_HDOUTN2	1		C
C17	URC_SQ_VCCTX2	1		
B17	URC_SQ_HDOUTN3	1		C
A16	URC_SQ_VCCOB3	1		
A17	URC_SQ_HDOUTP3	1		T
C16	URC_SQ_VCCTX3	1		
B14	URC_SQ_HDINN3	1		C
B13	URC_SQ_VCCIB3	1		
A14	URC_SQ_HDINP3	1		T
C13	URC_SQ_VCCRX3	1		
GND	GND	1		
E17	PT46B	1		C
GND	GND	1		
D17	PT46A	1		T
GND	GND	1		
F17	PT45B	1		C
D16	PT45A	1		T
F19	PT44B	1		C
F18	PT44A	1		T
VCCIO	VCCIO1	1		
E16	PT43B	1		C
D15	PT43A	1		T
G18	PT42B	1		C
E15	PT42A	1		T
GND	GND	1		
G17	PT41B	1		C
E14	PT41A	1		T
D14	PT40B	1		C
D13	PT40A	1		T
VCCIO	VCCIO1	1		
F15	PT39B	1	VREF2_1	C
E12	PT39A	1	VREF1_1	T
H17	PT38B	1	PCLKC1_0	C
E13	PT38A	1	PCLKT1_0	T
GND	GND	0		
C12	PT37B	0	PCLKC0_0	C
GND	GND	0		
G15	PT37A	0	PCLKT0_0	T
C11	PT36B	0	VREF2_0	C
F14	PT36A	0	VREF1_0	T
A12	PT35B	0		C
VCCIO	VCCIO0	0		
A11	PT35A	0		T

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

Ball Number	Ball Function	Bank	Dual Function	Differential
D12	PT34B	0		C
H16	PT34A	0		T
H18	PT33B	0		C
H15	PT33A	0		T
A10	PT32B	0		C
GND	GND	0		
B10	PT32A	0		T
D11	PT31B	0		C
VCCIO	VCCIO0	0		
G14	PT31A	0		T
E11	PT30B	0		C
F13	PT30A	0		T
D10	PT29B	0		C
H14	PT29A	0		T
GND	GND	0		
GND	GND	0		
VCCIO	VCCIO0	0		
A9	PT24B	0		C
GND	GND	0		
C10	PT23B	0		C
E8	PT23A	0		T
B9	PT22B	0		C
A8	PT22A	0		T
VCCIO	VCCIO0	0		
F12	PT21B	0		C
E10	PT21A	0		T
G13	PT20B	0		C
C9	PT20A	0		T
GND	GND	0		
GND	GND	0		
B8	PT19B	0		C
GND	GND	0		
A7	PT19A	0		T
D9	PT18B	0		C
H13	PT18A	0		T
D6	PT17B	0		C
VCCIO	VCCIO0	0		
C7	PT17A	0		T
C8	PT16B	0		C
G12	PT16A	0		T
D8	PT15B	0		C
H12	PT15A	0		T
A6	PT14B	0		C
GND	GND	0		

**ECP2M35 Logic Signal Connections: 672 fpBGA (Cont.)**

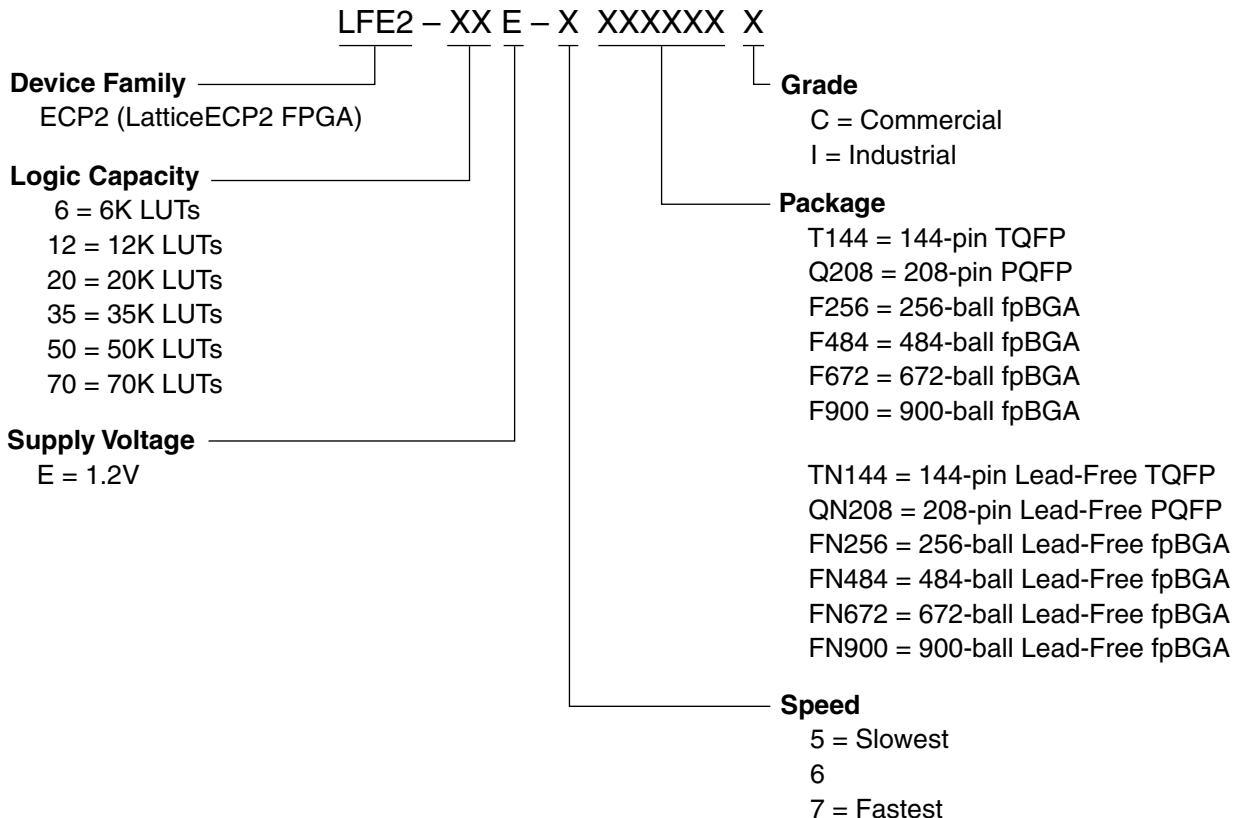
Ball Number	Ball Function	Bank	Dual Function	Differential
A5	PT14A	0		T
A4	PT13B	0		C
VCCIO	VCCIO0	0		
A3	PT13A	0		T
C6	PT12B	0		C
F10	PT12A	0		T
D7	PT11B	0		C
H11	PT11A	0		T
D5	PT10B	0		C
GND	GND	0		
E6	PT10A	0		T
GND	GND	0		
G10	PT9B	0		C
F9	PT9A	0		T
H10	PT8B	0		C
E7	PT8A	0		T
VCCIO	VCCIO0	0		
B3	PT7B	0		C
C5	PT7A	0		T
B2	PT6B	0		C
C4	PT6A	0		T
GND	GND	0		
G9	PT5B	0		C
F7	PT5A	0		T
C3	PT4B	0		C
D4	PT4A	0		T
VCCIO	VCCIO0	0		
J10	PT3B	0		C
F8	PT3A	0		T
G8	PT2B	0		C
G7	PT2A	0		T

\*Supports dedicated LVDS outputs.

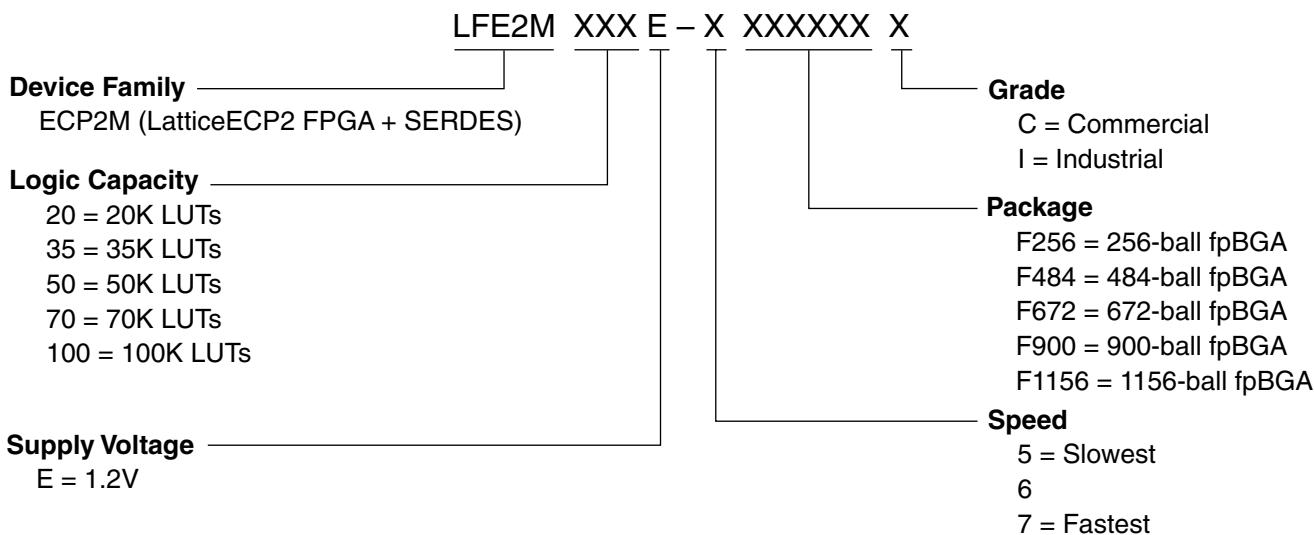
September 2006

Advance Data Sheet DS1007

### LatticeECP2 Part Number Description

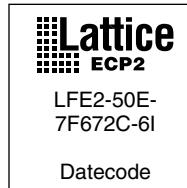


### LatticeECP2M Part Number Description



## Ordering Information

Note: LatticeECP2/M devices are dual marked. For example, the commercial speed grade LFE2-50E-7F672C is also marked with industrial grade -6I (LFE2-50E-6F672I). The commercial grade is one speed grade faster than the associated dual mark industrial grade. The slowest commercial speed grade does not have industrial markings. The markings appear as follows:



### Commercial

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs (K)
LFE2-12E-5T144C	93	1.2V	-5	TQFP	144	Com	12
LFE2-12E-6T144C	93	1.2V	-6	TQFP	144	Com	12
LFE2-12E-7T144C	93	1.2V	-7	TQFP	144	Com	12
LFE2-12E-5Q208C	131	1.2V	-5	PQFP	208	Com	12
LFE2-12E-6Q208C	131	1.2V	-6	PQFP	208	Com	12
LFE2-12E-7Q208C	131	1.2V	-7	PQFP	208	Com	12
LFE2-12E-5F256C	193	1.2V	-5	fpBGA	256	Com	12
LFE2-12E-6F256C	193	1.2V	-6	fpBGA	256	Com	12
LFE2-12E-7F256C	193	1.2V	-7	fpBGA	256	Com	12
LFE2-12E-5F484C	297	1.2V	-5	fpBGA	484	Com	12
LFE2-12E-6F484C	297	1.2V	-6	fpBGA	484	Com	12
LFE2-12E-7F484C	297	1.2V	-7	fpBGA	484	Com	12

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs (K)
LFE2-50E-5F484C	339	1.2V	-5	fpBGA	484	COM	50
LFE2-50E-6F484C	339	1.2V	-6	fpBGA	484	COM	50
LFE2-50E-7F484C	339	1.2V	-7	fpBGA	484	COM	50
LFE2-50E-5F672C	500	1.2V	-5	fpBGA	672	COM	50
LFE2-50E-6F672C	500	1.2V	-6	fpBGA	672	COM	50
LFE2-50E-7F672C	500	1.2V	-7	fpBGA	672	COM	50

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs (K)
LFE2M35E-5F484C	301	1.2V	-5	fpBGA	484	COM	35
LFE2M35E-6F484C	301	1.2V	-6	fpBGA	484	COM	35
LFE2M35E-7F484C	301	1.2V	-7	fpBGA	484	COM	35
LFE2M35E-5F672C	411	1.2V	-5	fpBGA	672	COM	35
LFE2M35E-6F672C	411	1.2V	-6	fpBGA	672	COM	35
LFE2M35E-7F672C	411	1.2V	-7	fpBGA	672	COM	35

**Industrial**

<b>Part Number</b>	<b>I/Os</b>	<b>Voltage</b>	<b>Grade</b>	<b>Package</b>	<b>Pins</b>	<b>Temp.</b>	<b>LUTs (K)</b>
LFE2-12E-5T144I	93	1.2V	-5	TQFP	144	Ind	12
LFE2-12E-6T144I	93	1.2V	-6	TQFP	144	Ind	12
LFE2-12E-5Q208I	131	1.2V	-5	PQFP	208	Ind	12
LFE2-12E-6Q208I	131	1.2V	-6	PQFP	208	Ind	12
LFE2-12E-5F256I	193	1.2V	-5	fpBGA	256	Ind	12
LFE2-12E-6F256I	193	1.2V	-6	fpBGA	256	Ind	12
LFE2-12E-5F484I	297	1.2V	-5	fpBGA	484	Ind	12
LFE2-12E-6F484I	297	1.2V	-6	fpBGA	484	Ind	12

<b>Part Number</b>	<b>I/Os</b>	<b>Voltage</b>	<b>Grade</b>	<b>Package</b>	<b>Pins</b>	<b>Temp.</b>	<b>LUTs (K)</b>
LFE2-50E-5F484I	339	1.2V	-5	fpBGA	484	IND	50
LFE2-50E-6F484I	339	1.2V	-6	fpBGA	484	IND	50
LFE2-50E-5F672I	500	1.2V	-5	fpBGA	672	IND	50
LFE2-50E-6F672I	500	1.2V	-6	fpBGA	672	IND	50

<b>Part Number</b>	<b>I/Os</b>	<b>Voltage</b>	<b>Grade</b>	<b>Package</b>	<b>Pins</b>	<b>Temp.</b>	<b>LUTs (K)</b>
LFE2M35E-5F484I	301	1.2V	-5	fpBGA	484	IND	35
LFE2M35E-6F484I	301	1.2V	-6	fpBGA	484	IND	35
LFE2M35E-5F672I	411	1.2V	-5	fpBGA	672	IND	35
LFE2M35E-6F672I	411	1.2V	-6	fpBGA	672	IND	35



# LatticeECP2/M Family Data Sheet

## Supplemental Information

---

September 2006

Advance Data Sheet

### For Further Information

A variety of technical notes for the LatticeECP2 family are available on the Lattice web site at [www.latticesemi.com](http://www.latticesemi.com).

- LatticeECP2M SERDES/PCS Usage Guide (TN1124)
- LatticeECP2/M sysIO Usage Guide (TN1102)
- LatticeECP2/M sysCLOCK PLL Design and Usage Guide (TN1103)
- LatticeECP2/M Memory Usage Guide (TN1104)
- LatticeECP2/M High-Speed I/O Interface (TN1105)
- Power Estimation and Management for LatticeECP2/M Devices (TN1106)
- LatticeECP2/M sysDSP Usage Guide (TN1107)
- LatticeECP2/M sysCONFIG Usage Guide (TN1108)
- LatticeECP2/M Configuration Encryption Usage Guide (TN1109)
- LatticeECP2/M Soft Error Detection (SED) Usage Guide (TN1113)

For further information on interface standards refer to the following web sites:

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



# LatticeECP2/M Family Data Sheet

## Revision History

September 2006

Advance Data Sheet DS1007

Date	Version	Section	Change Summary
February 2006	01.0	—	Initial release.
August 2006	01.1	Introduction	
			Updated Table 1-1 "LatticeECP2 Family Selection Guide"
		Architecture	Updated figure 2-2 "PFU Diagram"
			Updated figure 2-13 "Secondary Clock Regions ECP2-50"
			Updated figure 2-25 "PIC Diagram"
			Updated figure 2-26 " Input Register Block for Left , Right and Bottom Edges"
			Updated figure 2-28 " Output Register Block for Left , Right and Bottom Edges"
			Updated figure 2-30 " DQS Input Routing for Left and Right Edges"
			Updated figure 2-32 " Edge Clock, DLL Calibration and DQS Local Bus Distribution"
			Table 2-15 Selectable Master clock (CCLK) frequencies Removed frequencies 15,20,21,22,23,30,34,41,45,51,55,60
		DC and Switching Characteristics	Replaced "CLKINDEL" with "CLKO"
			Updated SED section
			Qualified device migration capability when using DQS banks for DDR interfaces
			Added VCCPLL to the Recommended Operating Conditions Table
			Remove Note 5 from "Hot Specifications" section
			Added note 7 & 8 to "Initialization Supply current Table"
			Change Note 6 - "...down to 95MHz" to "...down to 95MHz for DDR and 133MHz for DDR2"
			New "Typical Building Block Function Performance" numbers
			New External Switching Characteristics numbers
			New Internal Switching Characteristics numbers
			New Family Timing Adders numbers
		Pinout Information	Updated Timings for GPLLS, SPLLS and DLLs
			Added sysConfig waveforms.
			Remove HSTL15D_II from sysIO Recommended Operating Condition Table
			Updated Supply and initialization currents for ECP2-50
			Added VCCPLL to the Signal Descriptions Table
			Updated Logic signal Connections tables to include 484-fpBGA for the ECP2-50.
			Added Logic signal Connections tables for ECP2-12 devices.

© 2006 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.

Date	Version	Section	Change Summary
August 2006 (cont.)	01.1 (cont.)	Pinout Information (cont.)	Added Information on : Available Device Resources per Packaged Device table
		Ordering Information	Updated ordering part number table to include ECP2-12.
			Updated topside mark drawing
September 2006	02.0	Multiple	Added information regarding LatticeECP2M support throughout.
September 2006	02.1	DC and Switching Characteristics	Added Receiver Total Jitter Tolerance Specification table.
			Removed power-up requirements for proper configuration footnote in Recommended Operating Conditions table.



## **Section II. LatticeECP2/M Family Technical Notes**

---

## Features

- Up to 16 Channels of High-Speed SERDES
  - 270 Mbps to 3.125 Gbps per channel
  - 3.2Gbps operation with low 100mW power
  - Receive equalization and transmit pre-emphasis for small form factor backplane operation
  - Supports PCI Express, Ethernet (1GbE and SGMII) plus multiple other standards
  - Beacon support for PCI Express
  - Out-of-band signal interface for low speed (<270 Mbps) inputs
- 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- or 10-bit interface to FPGA logic
- Gigabit Ethernet Support
  - IEEE 1000BASE-X compliant
  - 8b10b encoding/decoding
  - Insertion of /I2/ symbols into the receive data stream for auto-negotiation support
  - Comma character word alignment
  - Clock Tolerance Compensation circuit
- PCI Express Support
  - x1 to x4 support in one PCS quad
  - Integrated Word aligner
  - 8b10b encoding/decoding
  - Clock Tolerance Compensation circuit
  - Electrical Idle and Receiver Detection support
  - Support for Beacon Transmission and Beacon Detection
- 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
  - Far End Loopback (RX to TX) provided for testing line connections to and from LatticeSC™ device
  - Parallel Far-end Loopback allows testing of PCS receive and transmit logic as part of loopback
  - Integrated 2^7 and 2^31 PRBS generator/checkers per channel.

## Introduction to PCS

The LatticeECP2M™ family of FPGAs combines a high-performance FPGA fabric, high-performance I/Os and large embedded RAM in a single industry leading architecture. All LatticeECP2M devices also feature 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 data transfer protocols.

Each channel of PCS logic contains dedicated transmit and receive SERDES for high-speed full-duplex serial data transfers 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 Ethernet (1GbE and SGMII), PCI Express, CPRI, and OBSAI. 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 also be independently DC coupled and can allow for both high-speed and low-speed operation on the same SERDES pin for such applications as Serial Digital Video.

## Architecture Overview

The PCS logic is arranged in quads containing logic for four independent full-duplex data channels. The LatticeECP2M Family Selection Guide in the LatticeECP2/M Family Data Sheet shows the number of PCS channels available for all devices and package types in the LatticeECP2M family. The table on the cover page of the LatticeECP2M Family Data Sheet contains the number of PCS channels present on the chip. Note that in some packages (particularly lower pin count packages), not all channels from all quads on a given device may be bonded to package pins.

### PCS Quad

Figure 8-1 is a layout of a LatticeECP2M device showing the arrangement of PCS quads on the device (the largest array containing 4 quads is shown. Other devices have fewer quads).

**Figure 8-1. LatticeECP2M Block Diagram**

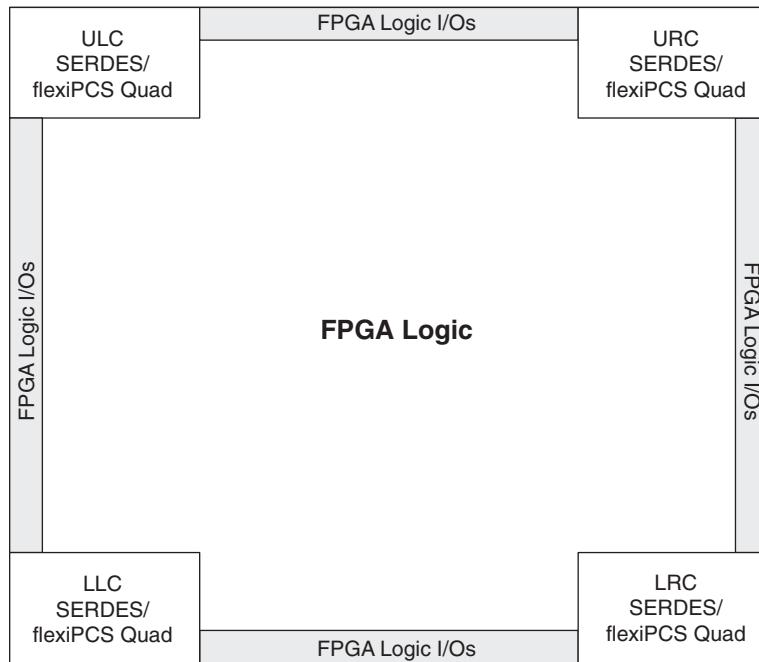


Table 8-1 shows the availability of SERDES/PCS quads for each device in the LatticeECP2M family.

**Table 8-1. SERDES/PCS Quads per LatticeECP2M Device**

Device	ECP2M20	ECP2M35	ECP2M50	ECP2M70	ECP2M100
Quad URC	Yes	Yes	Yes	Yes	Yes
Quad LRC	—	—	Yes	Yes	Yes
Quad ULC	—	—	—	Yes	Yes
Quad LLC	—	—	—	Yes	Yes

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.

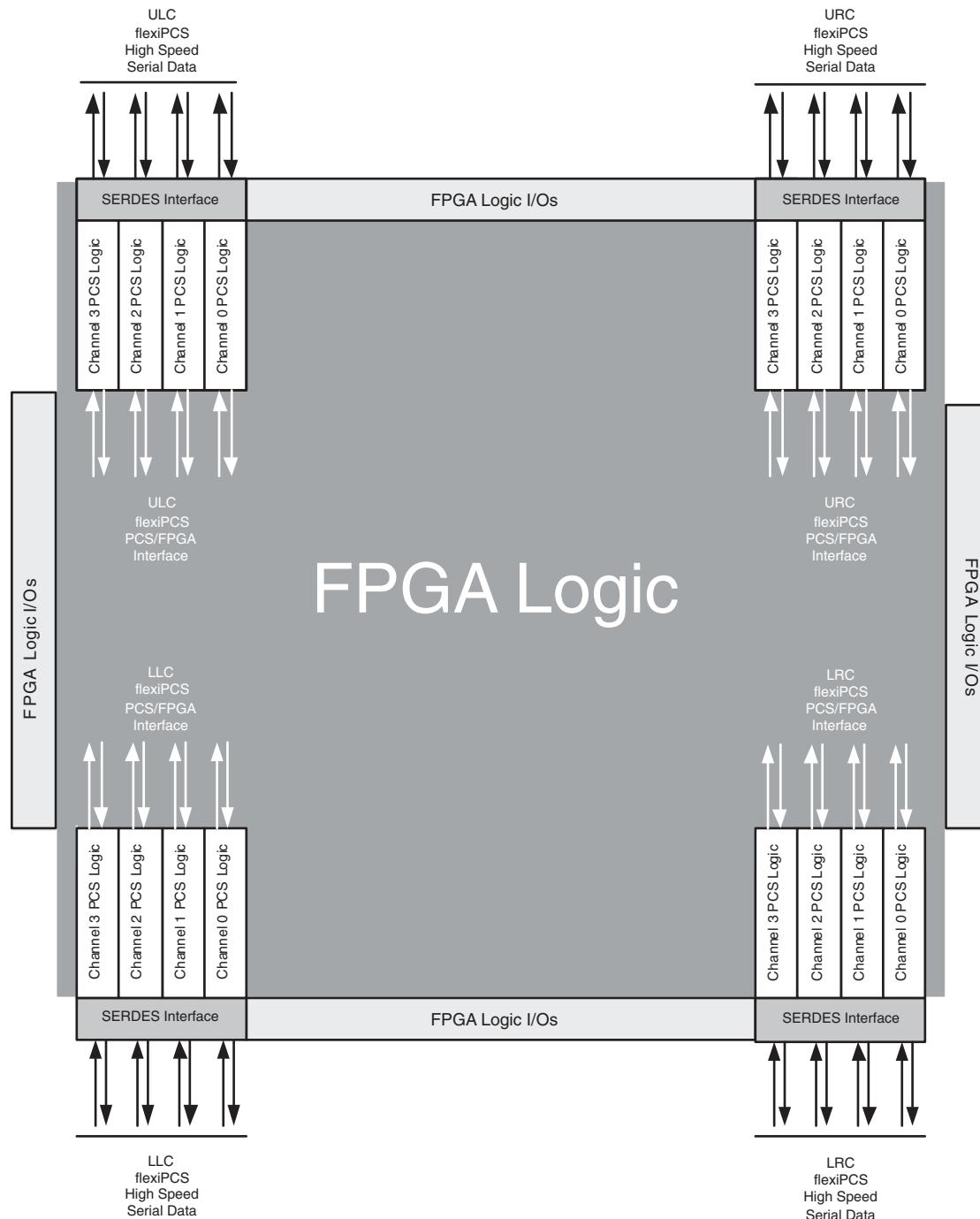
Since each quad has its own reference clock, different quads can support different standards on the same chip. This feature makes the LatticeECP2M family of devices ideal for bridging between different standards.

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, modes governing user-defined word alignment, and clock tolerance compensation can be programmed for non-protocol operation.

### PCS Channel

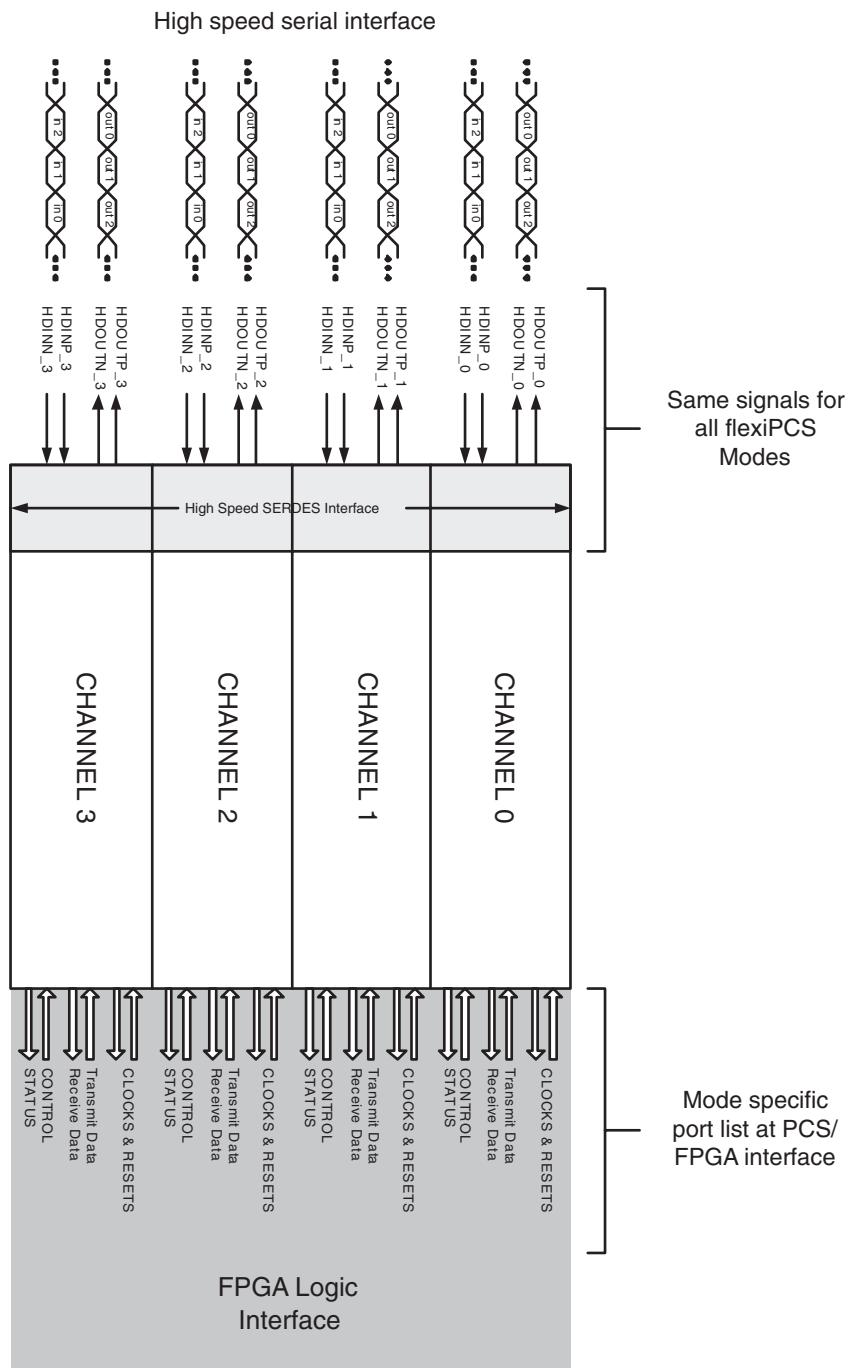
Each quad on a device supports up to four channels of full-duplex data. The user can utilize anywhere from one to four channels in a quad depending on the application. Many options can be set by the user for each channel independently within a given quad.

Figure 8-2 shows an example of a device with four PCS quads which contain a total of 16 PCS channels. Quads are named according to the location of the respective quad on the LatticeECP2M array: ULC (Upper left corner), LLC (Lower left corner), URC (Upper right corner), LRC (Lower right corner).

**Figure 8-2. Example of LatticeECP2M Device with Four PCS Quads**

### Per Channel PCS/FPGA Interface Ports

All PCS quads regardless of chosen mode 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. A simplified diagram showing the channels within a single quad is shown in Figure 8-3.

**Figure 8-3. PCS Quad External and Internal FPGA Interfaces**

A general description of the FPGA interface signals follows:

### Clocks & 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 these 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 the 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

For each channel in the quad, 8-bit or 10-bit (depending on mode) transmit data from the FPGA is supplied to the PCS, where it is serialized, and sent off chip by the SERDES. A gearing option per channel is provided for a 16 or 20-bit FPGA transmit data interface which runs at one half the nominal rate.

Data is synchronized to the quad reference clock supplied either from external pins or the FPGA logic.

### Receive data

For each channel in the quad, serial data and clock is supplied by an external source to the SERDES pins. The data is deserialized and manipulated by the PCS logic and passed as 8-bit or 10-bit (depending on mode) data to the FPGA logic. A gearing option per channel is provided for a 16 or 20-bit FPGA receive data interface which would run at half the nominal rate. Clocks are recovered for each channel and made available at the internal FPGA logic interface.

### 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 duplicate the effect of writing to a corresponding control register bit or bits. The ispLEVER® design tools give the user the option to bring these ports out to the FPGA interface.

### 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 correspond to a specific status register bit or bits. The ispLEVER design tools give the user the option to bring these port out to the PCS FPGA interface.

## SCI (SERDES Client Interface) Bus

The SERDES Client Interface (SCI) is a soft IP that allow the SERDES/PCS Quad block to be controlled by registers as oppose to the configuration memory cells. It is a simple register configuration interface.

## 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 modes which allow a user to define their own custom application settings. ispLEVER design tools allow the user to define the mode for each quad in their design. This document describes operation of the SERDES and PCS for all modes supported by ispLEVER.

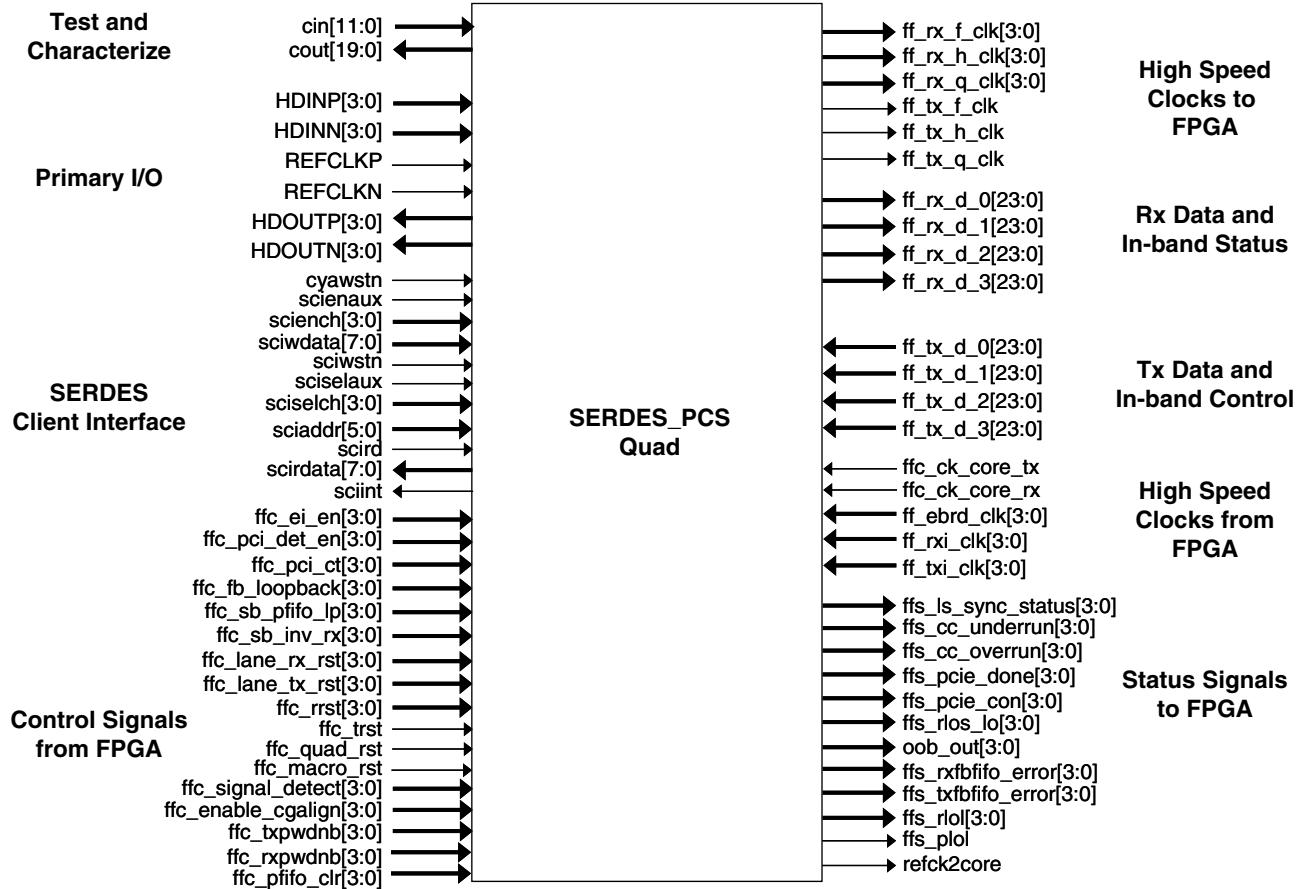
This document provides a thorough description of the complete functionality of the embedded SERDES and associated PCS logic. Electrical and Timing Characteristics of the embedded SERDES are provided in the LatticeECP2/M Family Data Sheet. Operation of the PCS logic is provided in the PCS section. A table 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 Memory Map section. Package pinout information is provided in the **Architecture** section of the LatticeECP2/M Family Data Sheet.

## SERDES

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 or by the FPGA core. The quad SERDES macro performs the serialization and de-serialization function for four lanes of data. In addition, the PLL within the SERDES 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 Level Diagram**



**I/O Definitions****Table 8-2. SERDES\_PCS IO Description**

Signal	I/O	Description
HDINP0	I	High-speed input, positive, channel 0.
HDINN0	I	High-speed input, negative, channel 0.
HDINP1	I	High-speed input, positive, channel 1.
HDINN1	I	High-speed input, negative, channel 1.
HDINP2	I	High-speed input, positive, channel 2.
HDINN2	I	High-speed input, negative, channel 2.
HDINP3	I	High-speed input, positive, channel 3.
HDINN3	I	High-speed input, negative, channel 3.
HDOUTP0	O	High-speed output, positive, channel 0.
HDOUTN0	O	High-speed output, negative, channel 0.
HDOUTP1	O	High-speed output, positive, channel 1.
HDOUTN1	O	High-speed output, negative, channel 1.
HDOUTP2	O	High-speed output, positive, channel 2.
HDOUTN2	O	High-speed output, negative, channel 2.
HDOUTP3	O	High-speed output, positive, channel 3.
HDOUTN3	O	High-speed output, negative, channel 3.
REFCLKP	I	Reference clock input, positive, aux channel
REFCLKN	I	Reference clock input, negative, aux channel
FFC_EI_EN[3:0]	I	Control transmission of electrical idle by SERDES transmitter. 1 = Force SERDES transmitter to output Electrical idle. 0 = Normal operation.
FFC_PCIE_DET_EN[3:0]		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.
FFC_PCIE_CT[3:0]	I	1 = Request transmitter to do far-end receiver detection. 0 = Normal data operation.
FFC_SB_INV_RX[3:0]	I	Control the inversion of received data. 1 = invert the data 0 = don't invert the data
FFC_ENABLE_CGALIGN[3:0]	I	Control comma aligner. 1 = Enable comma aligner, 0 = Lock comma aligner at current position.
FFC_SIGNAL_DETECT[3:0]	I	Signal detect 1 = Signal present, 0 = No signal present.
FFC_FB_LOOPBACK[3:0]	I	FPGA Bridge Loopback. 1 = Enable loopback from RX to TX, 0 = Normal data operation
FFC_SB_PFIFO_LP[3:0]	I	SERDES Bridge Parallel Loopback 1 = Enable loopback from RX to TX, 0 = Normal data operation
FFC_PFIFO_CLR[3:0]	I	
FFC_LANE_RX_RST[3:0]	I	Active high, asynchronous input. Resets individual TX channel logic only in PCS.
FFC_LANE_TX_RST[3:0]	I	Active high, asynchronous input. Resets individual RX channel logic only in PCS.
FFC_RRST[3:0]	I	Resets selected digital logic in the SERDES Receive channels.
FFC_TRST	I	Resets selected digital logic in the SERDES Transmit channels.

**Table 8-2. SERDES\_PCS IO Description (Continued)**

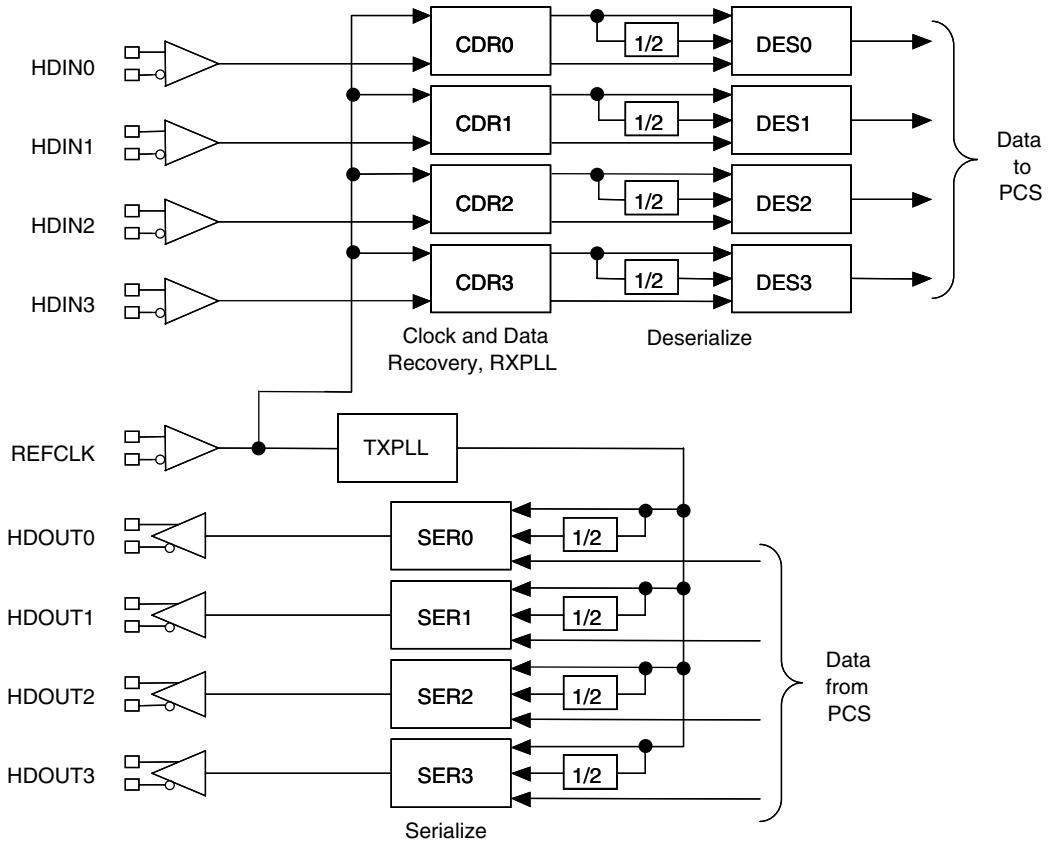
Signal	I/O	Description
FFC_QUAD_RST	I	Active high, asynchronous input. Resets all SERDES channels including the auxiliary channel and PCS.
FFC_MACRO_RST	I	Active high, asynchronous input to SERDES quad. Gated with software register bit fpga_reset_enable. By default fpga_reset_enable is '1'.
FFC_TXPWDNB[3:0]	I	Active low transmit channel power down. 0 = Transmit channel should be powered down.
FFC_RXPWDNB[3:0]	I	Active low receive channel power down. 0 = Receive channel should be powered down.
FF_RX_F_CLK[3:0]	O	Receive channel recovered clock. In user mode, the source is always the channel's recovered clock. For standards such as GbE, 10 GbE that support clock compensation, the source is the respective transmit channel's system clock. For PCS bypass modes, it is also the TX clock system clock, thus requiring raw mode to actually be done using either 8B/10B mode with the 8B/10B decoder disabled (10-bit or 20-bit data path).
FF_RX_H_CLK[3:0]	O	Receive channel recovered half clock. In 2:1 gearing mode, it is a divide-by-2 output.
FF_RX_Q_CLK[3:0]	O	Receive channel recovered quarter clock. Available for further 2:1 gearing.
FF_TX_F_CLK	O	TX PLL clock.
FF_TX_H_CLK	O	TX PLL half clock.
FF_TX_Q_CLK	O	TX PLL quarter clock.
FF_RX_D_0[23:0]	O	Data signals for the channel 0 transmit path.
FF_RX_D_1[23:0]	O	Data signals for the channel 1 transmit path.
FF_RX_D_2[23:0]	O	Data signals for the channel 2 transmit path.
FF_RX_D_3[23:0]	O	Data signals for the channel 3 transmit path.
FF_TX_D_0[23:0]	I	Data signals for the channel 0 receive path.
FF_TX_D_1[23:0]	I	Data signals for the channel 1 receive path.
FF_TX_D_2[23:0]	I	Data signals for the channel 2 receive path.
FF_TX_D_3[23:0]	I	Data signals for the channel 3 receive path.
FFC_CK_CORE_TX	I	Reference clock to transmit SERDES PLL
FFC_CK_CORE_RX	I	Reference clock to receive CDR in SERDES
FF_EBRD_CLK[3:0]	I	Receive channel clock input from FPGA for CTC FIFO (Elastic Buffer Read).
FF_RXI_CLK[3:0]	I	Receive channel clock input from FPGA for FPGA bridge FIFO.
FF_TXI_CLK[3:0]	I	Transmit channel clock input from FPGA.
FFS_PCIE_DONE[3:0]	O	1 = Far-end receiver detection complete, 0 = Far-end receiver detection incomplete.
FFS_PCIE_CON[3:0]	O	Result of far-end receiver detection. 1 = Far end receiver detected, 0 = Far end receiver not detected.
FFS_RLOS_LO[3:0]	O	Loss of signal detection for each channel. Includes a high value and a low value, but only the low value will be sent to the FPGA interface. Register bits rlos_sel[2:0] are used to set the low value and high value thresholds of these loss-of-signal.
FFS_LS_SYNC_STATUS[3:0]	O	1 = Lane is synchronous to commas. 0 = Lane has not found comma.
FFS_CC_UNDERRUN[3:0]	O	1 = Receive clock compensator FIFO underrun error, 0 = No FFIFO errors.
FFS_CC_OVERRUN[3:0]	O	1 = Receive clock compensator FIFO overrun error, 0 = No FIFO errors.
FFS_RXFBFIFO_ERROR[3:0]	O	1=Receive FPGA bridge FIFO error, 0 = No FIFO errors.

**Table 8-2. SERDES\_PCS IO Description (Continued)**

Signal	I/O	Description
FFS_TXFBFIFO_ERROR[3:0]	O	1=Transmit FPGA bridge FIFO error, 0 = No FIFO errors.
FFS_RLOL[3:0]	O	1=Receive CDR loss of lock, 0 = Lock maintained.
FFS_PLOL	O	1=Receive PLL loss of lock, 0=Lock maintained
OOB_OUT[3:0]	O	Single ended outputs to video SERDES (in FPGA).
REFCK2CORE	O	Reference clock to FPGA core.
CYAWSTN	I	Copy all memory cells to registers.
SCIENAUX	I	Selects between memory cells and sciw data for AUX registers.
SCIENCH[3:0]	I	Selects between memory cells and sciw data for channel registers.
SCIWDATA[7:0]	I	Write data input.
SCIWSTN	I	Write input strobe.
SCISELAUX	I	Selects AUX registers.
SCISELCH[3:0]	I	Selects channel registers.
SCIADDR[5:0]	I	Address bus input.
SCIRD	I	Read data select.
SCIRDATA[7:0]	O	Read data output.
SCIINT	O	Interrupt input.
CIN[11:0]	I	Characterization test bus logic data input.
COUT[19:0]	O	Characterization test bus logic data output.

## Reference Clock Usage

One reference clock (REFCLK) is supported in the LatticeECP2M family. The TX PLL and the four RX PLLs all run at the same frequency, which is multiple of the reference clock frequency. The TX serializer in each channel can be independently programmed to run at this rate (full-data-rate mode) or half of this rate (half-data-rate mode). Similarly, the RX deserializer in each channel can be independently programmed to run at this rate (full-data-rate mode) or half of this rate (half-data-rate mode). If all TX and RX are programmed in the same mode (normally this will be full-rate) then all four channels in the quad will run at the same frequency, both TX and RX.

**Figure 8-5. Block Diagram, Reference Clock Usage**

## Configuration GUIs

IPexpress™ is used to create and configure SERDES and PCS blocks. Designers use the graphical user interface (GUI) to select the SERDES Protocol Standard for a particular Quad. IPexpress takes the input from this GUI and generates a configuration 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 ispLEVER bitgen program. It is strongly recommended that designers make changes and updates in IPexpress and then regenerate the configuration file. In some exceptional occasions, users can modify the configuration file. Figure 8-6 shows the tools flow when using IPexpress to generate the SERDES/PCS block for the SERDES Protocol Standard.

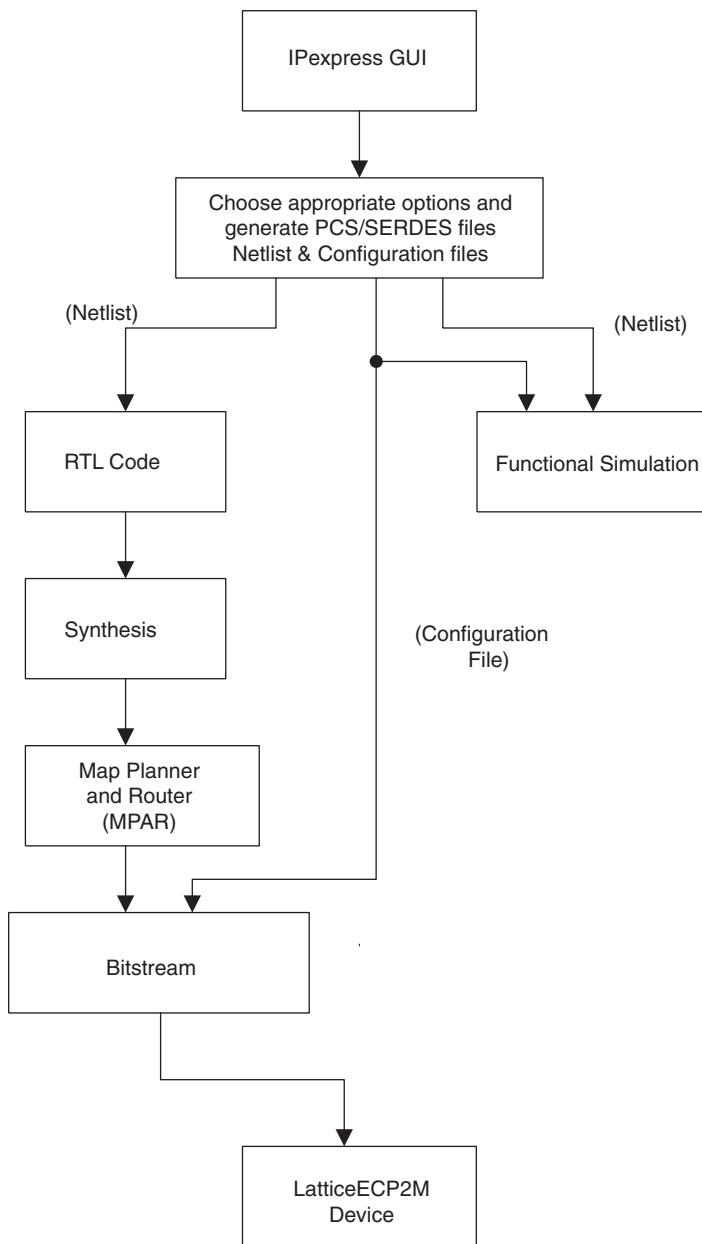
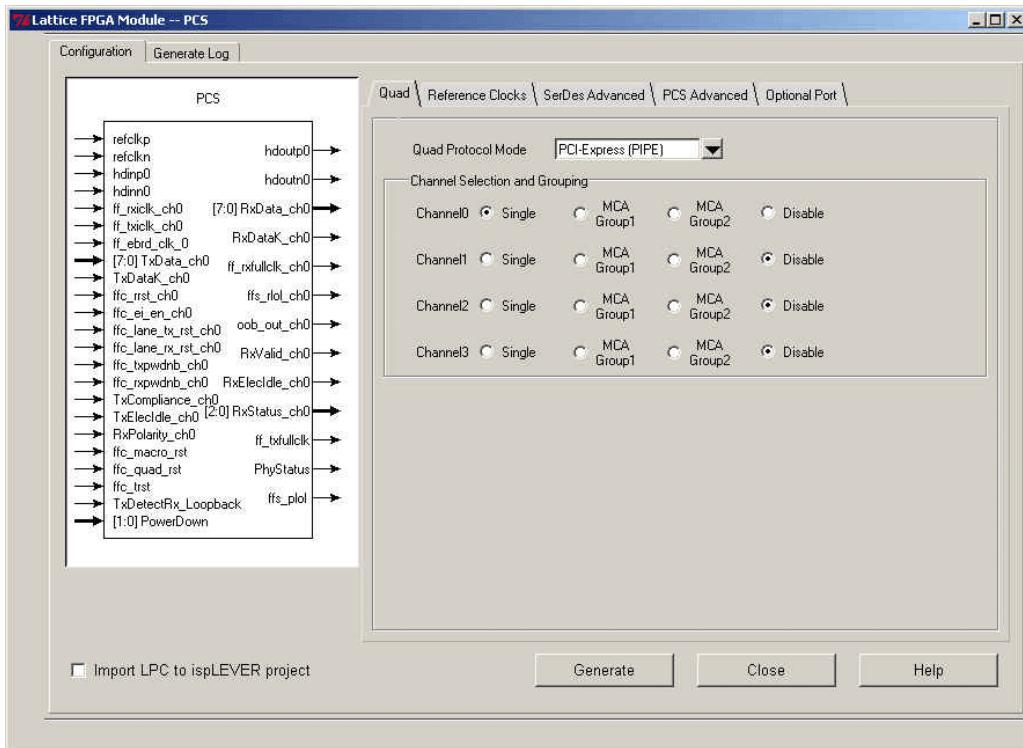
**Figure 8-6. SERDES\_PCS ispLEVER User Flow**

Figure 8-7 shows the main window when SERDES is selected. The first entry required in this window is Quad Protocol mode. Other entries on this screen are channel selection and group selections. There are four additional tabs shown in Figures 8-8 through 8-11, listing all the user-accessible attributes with default values settings:

**Figure 8-7. Configuration GUI - Quad Setup Tab**



**Table 8-3. SERDES\_PCS GUI Attributes (LatticeECP2M) - Quad Setup Tab**

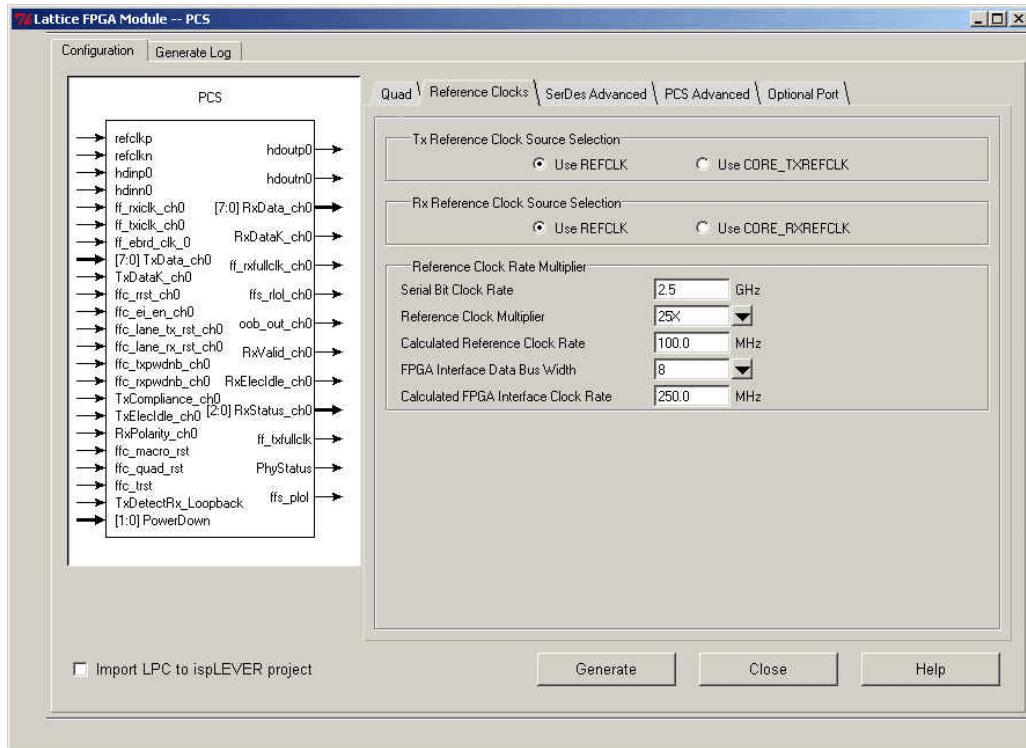
GUI Text	Attribute Names	Button/Box Type	Range	Default Value
Quad Protocol Mode	PROTOCOL	Drop Down	Gigabit Ethernet (GIGE), PCI Express (PCIE), PCI Express (PIPE) (PIPE), OBSAI (OBSAI), CPRI (CPRI), Generic 8b10b (G8B10B), 10-bit SERDES Only (10BSER), 8-bit SERDES Only (8BSER), User Defined (USERDEF)	PCI Express (PIPE)
Channel Selection and Grouping Channel0	CH0_MODE	Radio	Single (SINGLE), MCA Group1 (GROUP1), MCA Group2 (GROUP2), Disable (DISABLE)	None
Channel Selection and Grouping Channel1	CH1_MODE	Radio	Single (SINGLE), MCA Group1 (GROUP1), MCA Group2 (GROUP2), Disable (DISABLE)	None
Channel Selection and Grouping Channel2	CH2_MODE	Radio	Single (SINGLE), MCA Group1 (GROUP1), MCA Group2 (GROUP2), Disable (DISABLE)	None
Channel Selection and Grouping Channel3	CH3_MODE	Radio	Single (SINGLE), MCA Group1 (GROUP1), MCA Group2 (GROUP2), Disable (DISABLE)	None

Note: MCA: Multi-channel alignment in soft logic.

### Reference Clock Setup Tab

In this tab, the attributes of the TX and RX reference clock sources are selected. Users can select either a REFCLK or CORE\_TXREFCLK as a TX reference clock source for the quad. Similarly, in a quad all the channels use the same common TX and RX reference sources. 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 quad to the core.

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



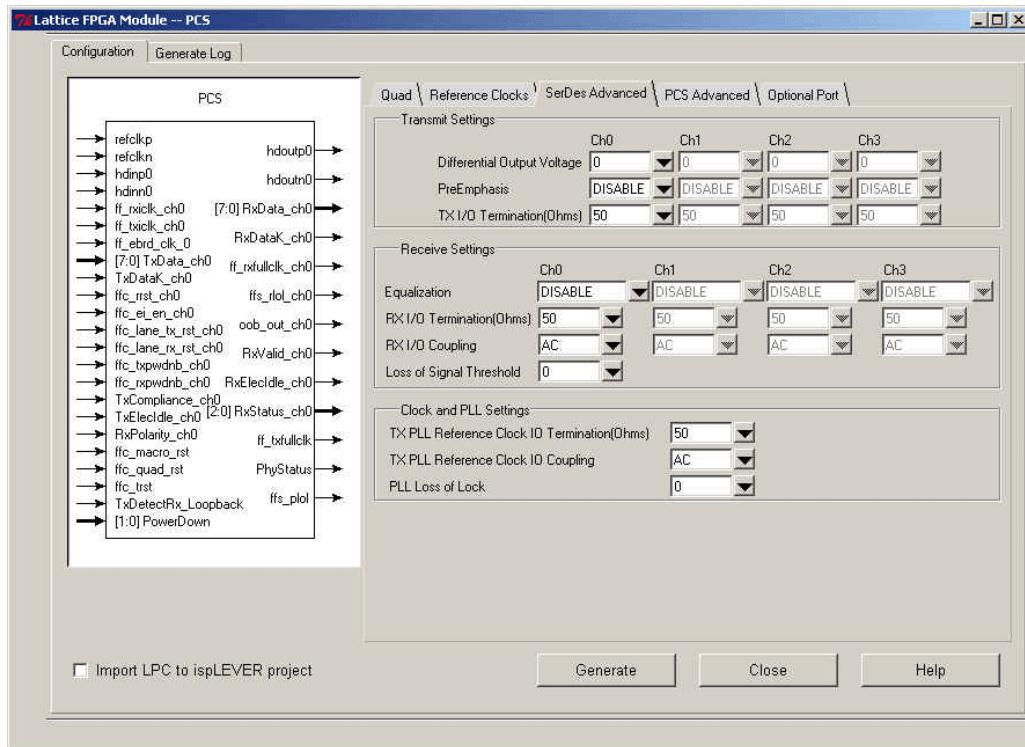
**Table 8-4. SERDES\_PCS GUI Attributes (LatticeECP2M) - Reference Clocks Setup Tab**

GUI Text	Attribute Names	Button/Box Type	Range	Default Value	Comment
TX Reference Clock Source Selection	PLL_SRC	Radio	REFCLK, CORE_TXREFCLK	REFCLK	
RX Reference Clock Source Selection	CH0_CDR_SRC CH1_CDR_SRC CH2_CDR_SRC CH3_CDR_SRC	Radio	REFCLK, CORE_RXREFCLK	REFCLK	
Serial Bit Clock Rate (GHz)	DATARANGE	Check Box	0.27 to 3.125	2.5	LOW: MEDLOW: MED: MEDHIGH: HIGH:
Reference Clock Multiplier	CH0_REFCK_MULT CH1_REFCK_MULT CH2_REFCK_MULT CH3_REFCK_MULT	Drop Down		25X	
Calculated Reference Clock Rate (MHz)	CP: REFCLK_RATE	Text Box			Not an editable field
FPGA Interface Data Bus Width	CH0_DATA_WIDTH CH1_DATA_WIDTH CH2_DATA_WIDTH CH3_DATA_WIDTH	Drop Down		8	
Calculated FPGA Interface Clock Rate (MHz)	CP: FPGAINCLK_RATE	Text Box			Not an editable field

### 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 8-9. Configuration GUI - SERDES Advanced Setup Tab**



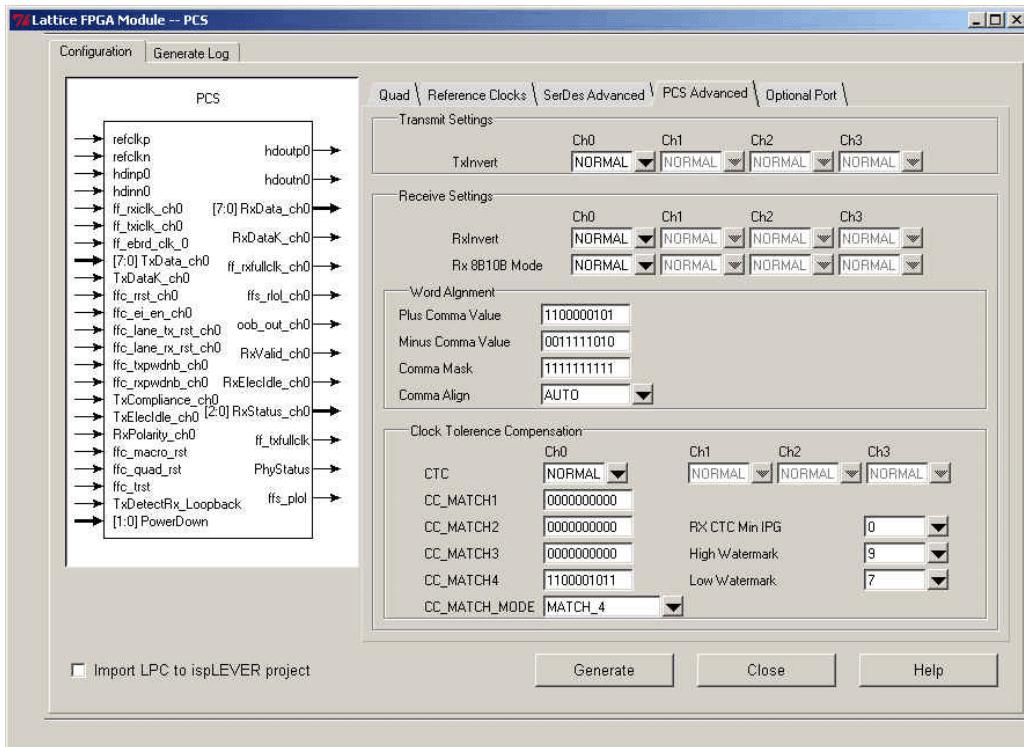
**Table 8-5. SERDES\_PCS GUI Attributes (LatticeECP2M) - SERDES Advanced Setup Tab**

GUI Text	Attribute Names	Button/ Box Type	Range	Default Value
Differential Output Voltage	CH0_TDRV_AMP CH1_TDRV_AMP CH2_TDRV_AMP CH3_TDRV_AMP	Drop Down	Based on the Protocol	0
PreEmphasis	CH0_TX_PRE CH1_TX_PRE CH2_TX_PRE CH3_TX_PRE	Drop Down	Based on the Protocol	DISABLE
TX I/O Termination (Ohms)	CH0_RTERM_TX CH1_RTERM_TX CH2_RTERM_TX CH3_RTERM_TX	Drop Down	Based on the Protocol	50
Equalization	CH0_RX_EQ CH1_RX_EQ CH2_RX_EQ CH3_RX_EQ	Drop Down	Based on the Protocol	DISABLE
RX I/O Termination (Ohms)	CH0_RTERM_RX CH1_RTERM_RX CH2_RTERM_RX CH3_RTERM_RX	Drop Down	Based on the Protocol	50
RX I/O Coupling	CH0_RX_DCC CH1_RX_DCC CH2_RX_DCC CH3_RX_DCC	Drop Down	Based on the Protocol	AC
Loss of Signal Threshold	LOS_THRESHOLD	Drop Down	Based on the Protocol	0
TX PLL Reference Clock I/O Termination (Ohms)	PLL_TERM	Drop Down	Based on the Protocol	50
TX PLL Reference Clock I/O Coupling	PLL_DCC	Drop Down	Based on the Protocol	AC
PLL Loss of Lock	PLL_LOL_SET	Drop Down	Based on the Protocol	0

### PCS Advance Setup

This tab is used to access the advanced attributes of the transmit and receive PCS for all four channels. Polarity of each individual TX and RX channel can be individually selected. The operating mode (e.g. 8b/10b) of the individual channels can be selected. In addition, word alignment values such as comma values, comma mask and comma align can be selected. This tab is also used for setting values for the Clock Tolerance Compensation block.

**Figure 8-10. Configuration GUI - PCS Advanced Setup Tab**



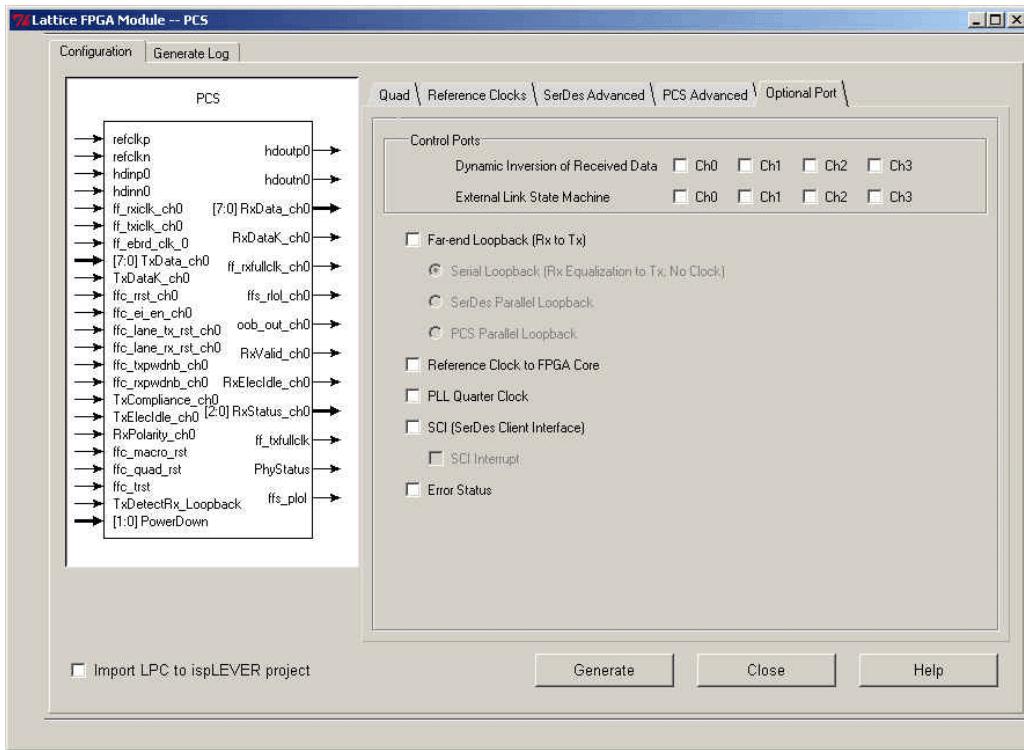
**Table 8-6. Tab 4. SERDES\_PCS GUI attributes (LatticeECP2M) - PCS Advanced Setup Tab**

GUI Text	Attribute Names	Button/ Box Type	Range	Default Value
TX Invert	CH0_TX_SB CH1_TX_SB CH2_TX_SB CH3_TX_SB	Drop Down	Based on the Protocol	NORMAL
RX Invert	CH0_RX_SB CH1_RX_SB CH2_RX_SB CH3_RX_SB	Drop Down	Based on the Protocol	NORMAL
RX_TX 8b10b Mode	CH0_8B10B CH1_8B10B CH2_8B10B CH3_8B10B	Drop Down	Based on the Protocol	NORMAL
Plus Comma Value	COMMA_A	Text Box	Based on the Protocol	1100000101
Minus Comma Value	COMMA_B	Text Box	Based on the Protocol	0011111010
Comma Mask	COMMA_M	Text Box	Based on the Protocol	1111111111
Comma Align	CH0_COMMAS_ALIGN CH1_COMMAS_ALIGN CH2_COMMAS_ALIGN CH3_COMMAS_ALIGN	Drop Down	Based on the Protocol	AUTO
CTC	CH0_CTC_BYP CH1_CTC_BYP CH2_CTC_BYP CH3_CTC_BYP	Drop Down	Based on the Protocol	NORMAL
CC_MATCH1	CC_MATCH1	Text Box	Based on the Protocol	0000000000
CC_MATCH2	CC_MATCH2	Text Box	Based on the Protocol	0000000000
CC_MATCH3	CC_MATCH3	Text Box	Based on the Protocol	0000000000
CC_MATCH4	CC_MATCH4	Text Box	Based on the Protocol	1100001011
CC_MATCH_MODE	CC_MATCH_MODE	Text Box	Based on the Protocol	MATCH_4
RX CTC Min IPG	CC_MIN_IPG	Drop Down	Based on the Protocol	0
CCHMARK	CCHMARK	Drop Down	Based on the Protocol	9
CCLMARK	CCLMARK	Drop Down	Based on the Protocol	7

### Optional Setup

This tab allows users to select the dynamic logic inversion and dynamic external link state machine capability per channel. In addition, users can enable the SCI, error reporting, PLL quarter clock, and far-end loop-back capability.

**Figure 8-11. Configuration GUI - Optional Setup Tab**



**Table 8-7. Tab5. SERDES\_PCS GUI Attributes (LatticeECP2M) - Optional Setup Tab**

GUI Text	Attribute Names	Button/Box Type	Range	Default Value
Dynamic Inversion of Receive Data		Check Box	TRUE, FALSE	FALSE
External Link State Machine		Check Box	TRUE, FALSE	FALSE
Far-end Loopback (Rx to TX) <sup>1</sup>		Check Box	TRUE, FALSE	FALSE
Far-end Loopback Type	OS_SSLB OS_SPLBPORTS OS_PCSLBPORTS	Radio Box	Serial Loopback, SERDES Parallel Loopback, PCS Parallel Loopback	Serial Loopback
Reference Clock to FPGA Core	OS_REFCK2CORE	Check Box	TRUE, FALSE	FALSE
PLL Quarter Clock	OS_PLLQCLKPORTS	Check Box	TRUE, FALSE	FALSE
SCI		Check Box	TRUE, FALSE	FALSE
SCI Interrupt	OS_INT_ALL	Check Box	TRUE, FALSE	FALSE
Error Status		Check Box	TRUE, FALSE	FALSE

## Configuration File Description

IPExpress generates this file which contains attribute level map information. This file is input for simulation, synthesis and the ispLEVER bitgen program. It is strongly recommended that designers make changes in the IPExpress and then regenerate the configuration file. In some exceptional occasions, users can modify the Configuration File.

Here is an example of the Configuration File:

```
PROTOCOL          "PIPE"
CH0_MODE         "SINGLE"
CH1_MODE         "SINGLE"
CH2_MODE         "DISABLE"
CH3_MODE         "DISABLE"
PLL_SRC          "REFCLK"
CH0_CDR_SRC     "REFCLK"
CH1_CDR_SRC     "REFCLK"
CH2_CDR_SRC     "REFCLK"
CH3_CDR_SRC     "REFCLK"
DATARANGE        "HIGH"
CH0_REFCK_MULT  "25X"
CH1_REFCK_MULT  "25X"
CH2_REFCK_MULT  "25X"
CH3_REFCK_MULT  "25X"
# REFCLK_RATE    <cval1>
CH0_DATA_WIDTH   "8"
CH1_DATA_WIDTH   "8"
CH2_DATA_WIDTH   "8"
CH3_DATA_WIDTH   "8"
# FPGAINTCLK_RATE <cval2>
CH0_TDRV_AMP    "0"
CH1_TDRV_AMP    "0"
CH2_TDRV_AMP    "0"
CH3_TDRV_AMP    "0"
CH0_TX_PRE       "DISABLE"
CH2_TX_PRE       "DISABLE"
CH3_TX_PRE       "DISABLE"
CH4_TX_PRE       "DISABLE"
CH0_RTERM_TX    "50"
CH1_RTERM_TX    "50"
CH2_RTERM_TX    "50"
CH3_RTERM_TX    "50"
CH0_RX_EQ        "DISABLE"
CH1_RX_EQ        "DISABLE"
CH2_RX_EQ        "DISABLE"
CH3_RX_EQ        "DISABLE"
CH0_RTERM_RX    "50"
CH1_RTERM_RX    "50"
CH2_RTERM_RX    "50"
```

CH3_RTERM_RX	"50"
CH0_RX_DCC	"AC"
CH1_RX_DCC	"AC"
CH2_RX_DCC	"AC"
CH3_RX_DCC	"AC"
LOS_THRESHOLD	"0"
PLL_TERM	"50"
PLL_DCC	"AC"
PLL_LOL_SET	"0"
CH0_TX_SB	"NORMAL"
CH1_TX_SB	"NORMAL"
CH2_TX_SB	"NORMAL"
CH3_TX_SB	"NORMAL"
CH0_RX_SB	"NORMAL"
CH1_RX_SB	"NORMAL"
CH2_RX_SB	"NORMAL"
CH3_RX_SB	"NORMAL"
CH0_8B10B	"NORMAL"
CH1_8B10B	"NORMAL"
CH2_8B10B	"NORMAL"
CH3_8B10B	"NORMAL"
COMMA_A	"1100000101"
COMMA_B	"0011111010"
COMMA_M	"1111111111"
CH0_COMMA_ALIGN	"AUTO"
CH1_COMMA_ALIGN	"AUTO"
CH2_COMMA_ALIGN	"AUTO"
CH3_COMMA_ALIGN	"AUTO"
CH0_CTC_BYP	"NORMAL"
CH1_CTC_BYP	"NORMAL"
CH2_CTC_BYP	"NORMAL"
CH3_CTC_BYP	"NORMAL"
CC_MATCH1	"0000000000"
CC_MATCH2	"0000000000"
CC_MATCH3	"0000000000"
CC_MATCH4	"1100001011"
CC_MATCH_MODE	"MATCH_4"
CC_MIN_IPG	"0"
CCHMARK	"9"
CCLMARK	"7"
OS_SSLB	"0"
OS_SPLBPORTS	"0"
OS_PCSLBPORTS	"0"
OS_REFCK2CORE	"0"
OS_PLLQCLKPORTS	"0"
OS_INT_ALL	"0"

---

## PCS

The LatticeECP2M PCS (Physical Coding Sublayer) provides support for many industry standard transmission protocols. The LatticeECP2M PCS can support applications up to 3.2 Gbps per channel.

The LatticeECP2M PCS supports the following operations:

### Transmit Path (From LatticeECP2M Device to Line):

- 8b10b Encoding

### Receive Path (From Line to LatticeECP2M Device):

- Word Alignment to a user defined word alignment character.
- 8b10b Decoding
- Optional Per Channel Clock Tolerance Compensation (CTC).

The LatticeECP2M PCS can be set to several 8-bit and 10-bit data modes:

#### 8-bit Modes:

- Gigabit Ethernet
- PCI Express
- PIPE
- Generic 8b10b
- 8-bit SERDES Only
- OBSAI
- CPRI

#### 10-bit Mode:

- 10-bit SERDES Only

## LatticeECP2M PCS Quad Module

Devices in the LatticeECP2M family have from 1 to 4 quads of embedded PCS logic. Each quad in turn supports 4 independent full-duplex data channels. A single channel can support a data link and each quad can support up to four such channels. Note that mode selection is done on a per quad basis. For example, a selection of Gigabit Ethernet mode for a quad dedicates all four channels in that quad to Gigabit Ethernet mode.

The embedded SERDES PLLs support data rates which cover a wide range of industry standard protocols.

Operation of the SERDES requires the user to provide a reference clock (or clocks) to each active quad to provide a synchronization reference for the SERDES PLLs. Reference clocks are shared among all four channels of a quad. If the transmit and receive frequencies are within the specified ppm tolerance for the LatticeECP2/M SERDES (see DC and Switching Characteristics section of the LatticeECP2/M Family Data Sheet), only one reference clock for both transmit and receive is needed for both transmit and receive directions. If the receive frequency is not within the specified ppm tolerance for the LatticeECP2/M SERDES (see the DC and Switching Characteristics section of the LatticeECP2/M Family Data Sheet) of the transmit frequency, then separate reference clocks for the transmit and receive directions must be provided.

Simultaneous support of different (non-synchronous) high-speed data rates is possible with the use of multiple quads. Multiple frequencies that are exact integer multiples can be supported on a per channel basis through use of the full-rate/half-rate mode options (see the **SERDES Functionality section** for more details).

---

## PCS Functional Setup

The LatticeECP2M PCS can be configured for use in various applications. Setup is chosen with the ispLEVER 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 ispLEVER bitstream generator to write the user selections into the bitstream. To change PCS option selections it is recommended that the user re-runs IPExpress to regenerate a PCS module and create a new auto-configuration file. Some options can be changed with manually editing the auto-configuration file before running the bitstream generator.

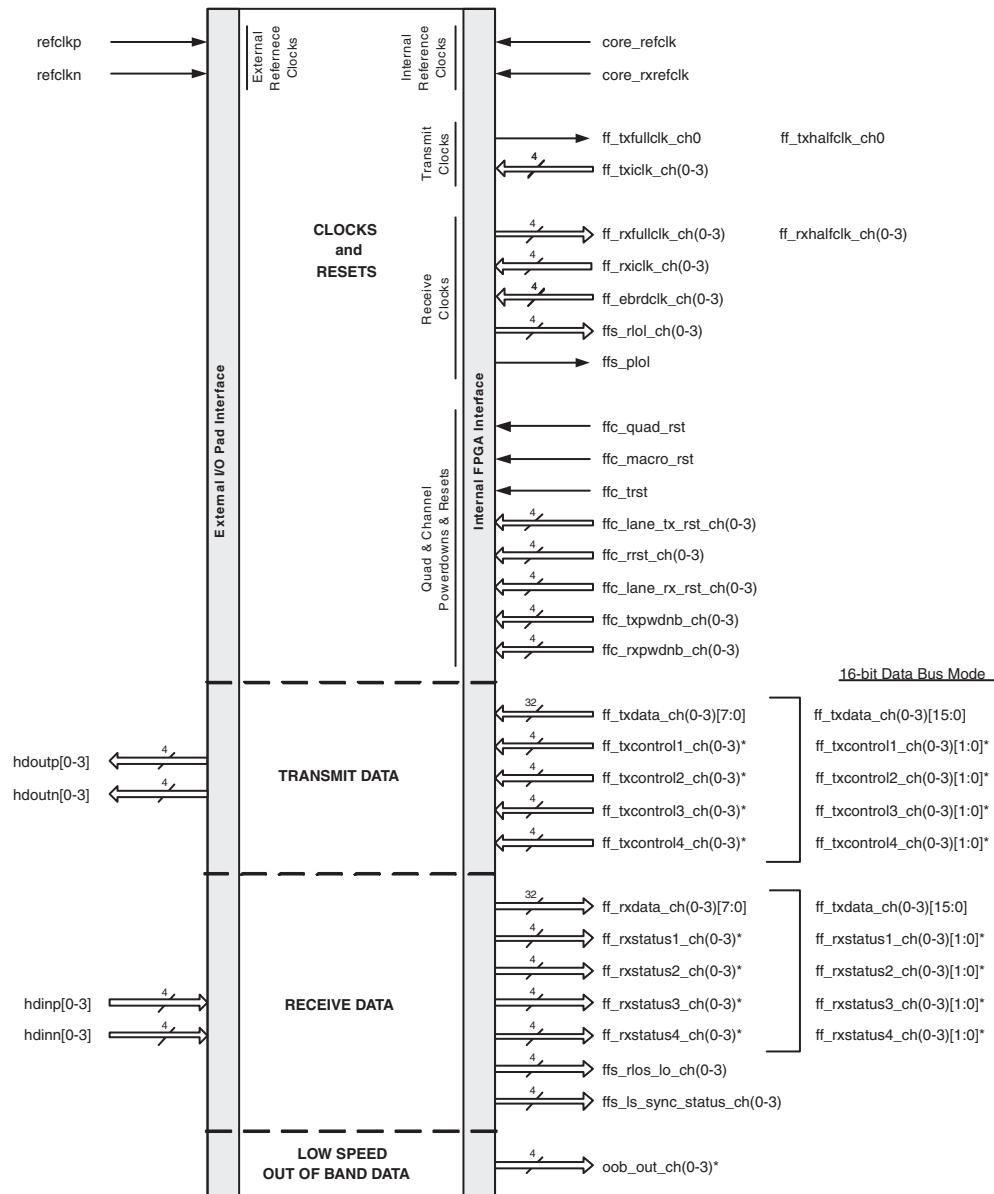
After configuration PCS options can be changed dynamically by writing to PCS registers via the optional SERDES Client Interface (SCI) bus. The SERDES Client Interface is soft IP that allows the SERDES/PCS quad to be controlled by registers as opposed to the configuration memory cells. A table of control and status registers accessible through the SCI is provided in the Memory Map section of this document.

### Auto-Configuration File

Initial register setup for each PCS mode can be performed without accessing the system bus by using the auto-configuration feature in ispLEVER. 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 (SERDES Client Interface) is therefore not needed if all quads are to be set via auto-configuration files. However, the SCI must be included in a design if the user needs to change control registers or monitor status registers during operation. Note that a quad reset (`ffc_quad_rst port` at the FPGA interface) will reset all registers back to their default (not auto-configuration) values. If a quad reset is issued, the PCS registers need to be rewritten via the SCI.

## LatticeECP2M PCS 8-bit Modes

The ispLEVER module generator tools allows the designer to choose the mode for each PCS quad used in a given design. Figure 8-12 displays the resulting default port interface to one PCS quad that has been set to one of the 8-bit data modes (except PCI Express and PIPE) with all four channels enabled.

**Figure 8-12. 8-bit Mode Default Port Diagram**

\* Port name is mode specific

## 8-bit Mode Pin Description

Table 8-8 lists all default inputs and outputs to/from a PCS quad in an 8-bit mode (except PCI Express and PIPE). A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **8-bit Mode Detailed Description**.

**Table 8-8. Default 8-bit Mode Pin Description**

Symbol	Direction/ Interface	Description
<b>Reference Clocks</b>		
refclkp	In from I/O pad	Reference clock input, positive. Dedicated CML input.
refclkn	In from I/O pad	Reference clock input, negative. Dedicated CML input
core_refclk	In from FPGA	Optional reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.  The ff_refclk inputs for all active quads on a device must be connected to the same clock
core_rxrefclk	In from FPGA	Optional receive reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.
<b>Transmit Clocks</b>		
ff_txfullclk_ch0	Out to FPGA	Full rate transmit PLL clock when in 8 bit data bus mode.
ff_txhalfclk_ch0*		*Half rate transmit PLL clock when in 16 bit data bus mode.
ff_txiclk_ch[0-3]	In 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>Receive Clocks</b>		
ff_rxfullclk_ch[0-3]	Out to FPGA	Full rate per channel Receive Channel Recovered Clock when in 8-bit data bus mode. For non-CTC modes, the source is always the channel recovered clock. For standards such as GbE, 10 GbE that support clock tolerance compensation, the source is the respective transmit channel's system clock.
ff_rxhalfclk_ch[0-3]*		*Half rate per channel Receive Channel Recovered Clock when in 16-bit data bus mode
ff_rxiclk_ch[0-3]	In from FPGA	Per channel receive clock inputs from FPGA. Used to clock the RX FPGA Interface FIFO with a clock synchronous to the reference and/or receive reference clock.
ff_ebrdclk_ch[0-3]	In from FPGA	Per channel receive clock inputs from FPGA. Used to clock the read side of the CTC FIFO.
ffs_rlol_ch[0-3]	Out to FPGA	Per channel receive CDR loss of lock.  1 = Receive CDR Loss of Lock 0 = Lock maintained
ffs_plol		Receive PLL loss of lock.  1 = Receive PLL Loss of Lock 0 = Lock maintained
<b>Resets</b>		
ffc_quad_RST	In from FPGA	Active high, asynchronous input. Resets all SERDES channels including the auxiliary channel and PCS.
ffc_macro_RST	In from FPGA	Active high, asynchronous input to SERDES quad. Resets all SERDES channels including the auxiliary channel but not PCS logic.
ffc_trst	In from FPGA	Active high reset. Resets selected digital logic in all transmit channels

**Table 8-8. Default 8-bit Mode Pin Description**

Symbol	Direction/ Interface	Description
ffc_lane_tx_rst_ch[0-3]	In from FPGA	Per channel, active high, asynchronous reset. Resets individual channel TX PCS logic.
ffc_rrst_ch[0-3]	In from FPGA	Per channel active high reset. Resets selected digital logic in corresponding receive channel.
ffc_lane_tx_rst_ch[0-3]	In from FPGA	Per channel, active high, asynchronous reset. Resets individual channel RX PCS logic.
ffc_txpwdnb_ch[0-3]	In from FPGA	Per channel active low Transmit Channel Power Down.
ffc_rxpwdnb_ch[0-3]	In from FPGA	Per channel active low Receive Channel Power Down.
<b>Transmit Data</b>		
hdoutp[0-3]	Out to I/O pad	High-speed CML serial output, positive.
hdoutn[0-3]	Out to I/O pad	High-speed CML serial output, negative.
ff_txdata_ch(0-3)[7:0]	In from FPGA	Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide in 8-bit data bus mode.  *Bus is 16-bits wide in 16-bit data bus mode.
ff_txdata_ch(0-3)[15:0]*		
ff_txcontrol1_ch(0-3)	In from FPGA	Per channel mode specific transmit control port.
ff_txcontrol1_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_txcontrol2_ch(0-3)	In from FPGA	Per channel mode specific transmit control port.
ff_txcontrol2_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_txcontrol3_ch(0-3)	In from FPGA	Per channel mode specific transmit control port.
ff_txcontrol3_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_txcontrol4_ch(0-3)	In from FPGA	Per channel mode specific transmit control port.
ff_txcontrol4_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
<b>Receive Data</b>		
hdinp[0-3]	In from I/O pad	High-speed CML serial input, positive.
hdinn[0-3]	In from I/O pad	High-speed CML serial input, negative.
ff_rxdata_ch(0-3)[7:0]	Out to FPGA	Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide in 8-bit data bus mode.  *Bus is 16-bits wide in 16-bit data bus mode.
ff_rxdata_ch(0-3)[15:0]*		
ff_rxstatus1_ch(0-3)	Out to FPGA	Per channel mode specific receive status port.
ff_rxstatus1_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_rxstatus2_ch(0-3)	Out to FPGA	Per channel mode specific receive status port.
ff_rxstatus2_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_rxstatus3_ch(0-3)	Out to FPGA	Per channel mode specific receive status port.
ff_rxstatus3_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_rxstatus4_ch(0-3)	Out to FPGA	Per channel mode specific receive status port.
ff_rxstatus4_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ffs_rlos_lo_ch(0-3)	Out to FPGA	Per channel Loss of Signal detection.

**Table 8-8. Default 8-bit Mode Pin Description**

Symbol	Direction/ Interface	Description
ffs_ls_sync_status_ch(0-3)	Out to FPGA	Per channel active link status. 1 = Receive channel is synchronized to commas 0 = Receive channel has not found comma
<b>Low Speed Out-of-Band Data</b>		
oob_out_ch(0-3)	Out to FPGA	Per channel serial low speed data to the FPGA.

The names for the ff\_txcontrol[1-4]\_ch(0-3) ports are determined by the PCS mode selected. Table 8-9 shows the port names for the ff\_txcontrol[1-4]\_ch(0-3) ports for all 8-bit modes selectable with the IPExpress tool.

**Table 8-9. Names of ff\_txcontrol[1-4]-ch(0-3) Buses in All 8-bit Modes**

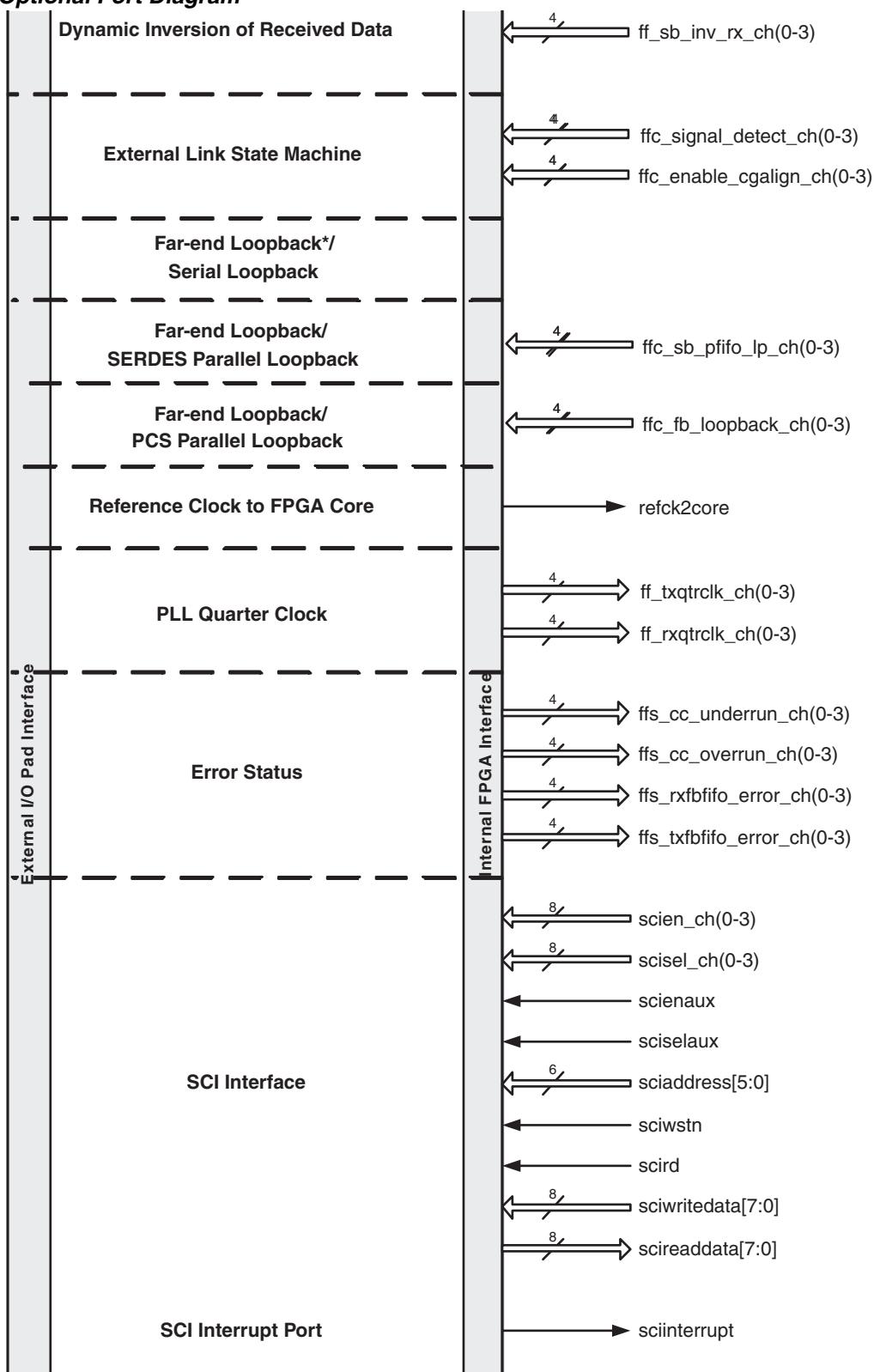
Mode(s)	ff_txcontrol1_ch(0-3)	ff_txcontrol2_ch(0-3)	ff_txcontrol3_ch(0-3)	ff_txcontrol4_ch(0-3)
Gigabit Ethernet OBSAI CPRI	ff_tx_k_cntrl_ch(0-3)[0]	NA	ff_xmit_ch(0-3)[0]	ff_correct_disp_ch(0-3)[0]
PCI Express	ff_tx_k_cntrl_ch(0-3)[0]	ff_force_disp_ch(0-3)[0]	ff_disp_sel_ch(0-3)[0]	ff_pci_ei_en_ch(0-3)[0]
PIPE	TxDataK_ch(0-3)[0]	NA	NA	NA
Generic 8b10b	ff_tx_k_cntrl_ch(0-3)[0]	ff_force_disp_ch(0-3)[0]	ff_disp_sel_ch(0-3)[0]	ff_correct_disp_ch(0-3)[0]

The names for the ff\_rxstatus[1-4]\_ch(0-3) ports are determined by the PCS mode selected. Table 8-10 shows the port names for the ff\_rxstatus[1-4]\_ch(0-3) ports for all 8-bit modes selectable with the IPExpress tool.

**Table 8-10. Names of ff\_rxstatus[1-4]-ch(0-3) Buses in All 8-bit Modes**

Mode(s)	ff_rxstatus1_ch(0-3)	ff_rxstatus2_ch(0-3)	ff_rxstatus3_ch(0-3)	ff_rxstatus4_ch(0-3)
Gigabit Ethernet OBSAI CPRI	ff_rx_k_cntrl_ch(0-3)[0]	ff_disp_err_ch(0-3)[0]	ff_cv_ch(0-3)[0]	ff_rx_even_ch(0-3)[0]
Generic 8b10b	ff_rx_k_cntrl_ch(0-3)[0]	ff_disp_err_ch(0-3)[0]	ff_cv_ch(0-3)[0]	NA
PCI Express	ff_rx_k_cntrl_ch(0-3)[0]	ff_rxstatus0_ch(0-3)[0]	ff_rxstatus1_ch(0-3)[0]	ff_rxstatus2_ch(0-3)[0]
PIPE	RxDataK_ch(0-3)[0]	NA	NA	NA

In addition to the default ports, the user can choose optional ports for a PCS quad using the IPExpress GUI. Figure 8-12 displays the optional ports available for one PCS quad through the IPExpress GUI.

**Figure 8-13. Optional Port Diagram**

\*Register bit setting via attribute OS\_SSLB.

## Optional Pin Description

Table 8-8 lists all optional inputs and outputs to/from a PCS quad. A brief description for each port is given in the table. A more detailed description of the function of each port is given in the **8-bit Mode Detailed Description**.

**Table 8-11. Optional Pin Description**

Symbol	Direction/ Interface	Description
<b>Dynamic Inversion of Received Data</b>		
ff_sb_inv_rx_ch(0-3)	In from FPGA	Per channel control to invert received data. 1 = Invert the data, 0 = Do not invert the data.
<b>External Link State Machine</b>		
ffc_signal_detect_ch(0-3)	In from FPGA	Per channel Signal detect. Used to enable embedded link state machines. 1 = LSM Enabled, 0 = LSM Disabled
ffc_enable_cgalign_ch(0-3)	In from FPGA	Per channel comma aligner unlock. Low pulse unlocks comma aligner to allow lock to next detected comma.
<b>Far-end Loopback/Serial Loopback</b>		
Signal name?	In from FPGA	Far-end Loopback, Serial Loopback select 1 = Enable Serial Far-end Loopback. 0 = Normal operation
<b>Far-end Loopback/SERDES Parallel Loopback</b>		
ffc_sb_pfifo_lp_ch(0-3)	In from FPGA	Per channel Far-end Loopback, SERDES Parallel Loopback select. 1 = Enable SERDES Parallel loopback. 0 = Normal operation
<b>Far-end Loopback/PCS Parallel Loopback</b>		
ffc_fb_loopback_ch(0-3)	In from FPGA	Per channel Far-end Loopback, PCS Parallel Loopback select. 1 = Enable PCS Parallel Loopback 0 = Normal operation
<b>Reference Clock to FPGA Core Port</b>		
refck2core	Out to FPGA	Locked Reference Clock to FPGA Core.
<b>PLL Quarter Clock Ports</b>		
ff_txqtrclk_ch(0-3)	Out to FPGA	Per channel Transmit PLL Quarter Rate Clock.
ff_rxqtrclk_ch(0-3)	Out to FPGA	Per channel Receive PLL Quarter Rate Clock.
<b>Error Status Ports</b>		
ffs_cc_underrun_ch(0-3)	Out to FPGA	Per channel CTC FIFO underrun status. 1 = Receive CTC FIFO underrun error 0 = No CTC FIFO underrun.
ffs_cc_overrun_ch(0-3)	Out to FPGA	Per channel CTC FIFO overrun status. 1 = Receive CTC FIFO overrun error 0 = No CTC FIFO overrun.
<b>SCI Ports</b>		
scien_ch(0-3)	In from FPGA	Per channel select between control via embedded memory cells or sciwdata. 1 = Control with memory cell. 0 = Control through sciwdata.
scisel_ch(0-3)	In from FPGA	Channel register select.
scinaux	In from FPGA	Select between control via embedded memory cells or sciwdata for SERDES aux channel functions. 1 = Control with memory cell. 0 = Control through sciwdata.
sciselaux	In from FPGA	SERDES aux register select.
sciaddress[5:0]	In from FPGA	SCI read/write address bus.

**Table 8-11. Optional Pin Description**

Symbol	Direction/ Interface	Description
sciwstn	In from FPGA	SCI write input strobe.
scird	In from FPGA	SCI read data select.
sciwdata[7:0]	In from FPGA	SCI write data bus.
scireaddata[7:0]	Out to FPGA	SCI read data bus.
sciinterrupt	Out to FPGA	SCI interrupt.

## 8-bit Mode Detailed Description

The following section provides a detailed description of the operation of a PCS quad in an 8-bit mode. The functional description is organized according to the port listing shown in Figure 8-12.

### Clocks & Resets

A detailed description of all SERDES clock, data, and reset ports and recommended reset sequencing is provided in the SERDES Functionality section of this document.

#### Quad & Channel Resets & Powerdowns

Resets are provided to reset an entire quad (either SERDES logic only or all SERDES/PCS logic) or each individual channel (all PCS logic per channel). These resets are driven from the FPGA logic. A summary of the control signals provided for reset are listed below. More detail on recommended initialization and reset sequences for the SERDES is provided in the **SERDES Functionality** section of this document.

**ffc\_quad\_rst** - Active high, asynchronous signal from FPGA resets all SERDES and PCS logic in the quad.

**ffc\_macro\_rst** - Active high, asynchronous signal from FPGA resets all SERDES (but not PCS) logic in the quad.

**ffc\_trst** - Active high, asynchronous signal from FPGA. Resets selected digital logic in all the SERDES Transmit channels.

**ffc\_rrst\_ch(0-3)** - Active high, asynchronous signal from FPGA. Resets selected digital logic in all the SERDES Transmit channels. Resets selected digital logic in selected SERDES Receive channels.

**ffc\_lane\_tx\_rst\_ch(0-3)** - Active high, asynchronous signals from FPGA. Resets individual TX channel logic only in PCS.

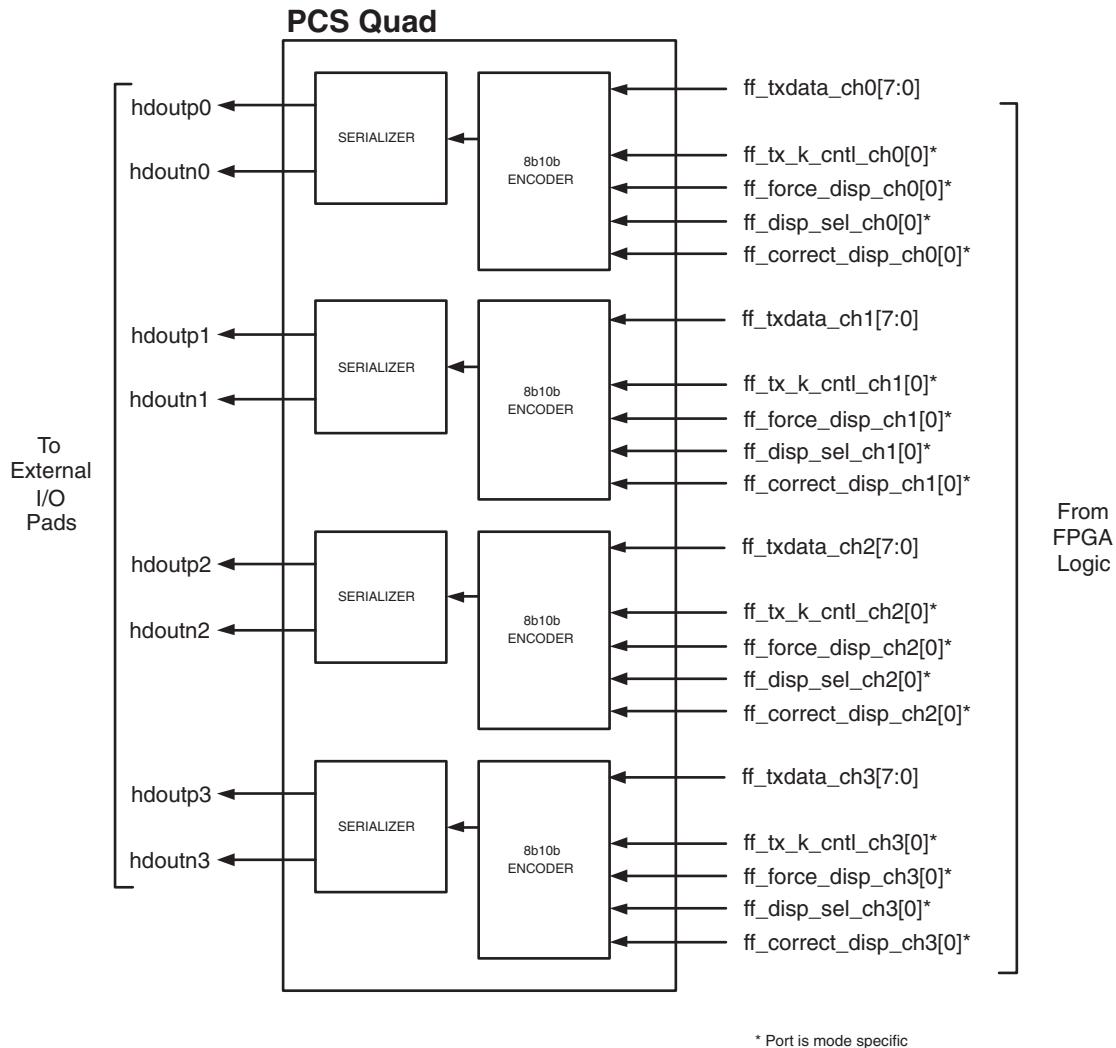
**ffc\_txpwdnb\_ch(0-3)** - Active low Transmit Channel Power Down.

**ffc\_rxpwdnb\_ch(0-3)** - Active low Receive Channel Power Down.

### Transmit Data

The PCS quad transmit data path consists of the following sub-blocks per channel: 8b10b Encoder and Serializer. Figure 8-14 shows the four channels of transmit data paths in a PCS quad.

Figure 8-14. 8-bit Mode Transmit Path (1 Quad)



\* Port is mode specific

**ff\_txdata\_ch\_(0-3)[7:0]** - Per channel transmit data from the FPGA logic interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data by default. Each **ff\_txdata\_ch\_(0-3)[7:0]** is an eight bit data bus that is driven synchronously with respect to the corresponding ff\_tx\_i\_ch(0-3).

In 16 Bit Data Bus Mode, this bus is 16-bits wide (**ff\_txdata\_ch\_(0-3)[15:0]**).

**ff\_tx\_k\_cntl\_ch(0-3)[0]** - Per channel, active high control character indicator.

In 16 Bit Data Bus Mode, tx\_k is 2 bits wide (**ff\_tx\_k\_cntl\_ch(0-3)[1:0]**) with ff\_tx\_k\_cntl\_ch(0-3)[1] associated with ff\_txdata\_ch\_(0-3)[15:8] and ff\_tx\_k\_cntl\_ch(0-3)[0] associated with ff\_txdata\_ch\_(0-3)[7:0].

**ff\_force\_disp\_ch(0-3)[0]** - Per channel, active high signal which instructs the PCS to accept disparity value from the ff\_disp\_sel\_ch(0-3)[0] FPGA interface input.

In 16 Bit Data Bus Mode, ff\_force\_disp\_ch is 2 bits wide (**ff\_force\_disp\_ch(0-3)[1:0]**) with ff\_force\_disp\_ch(0-3)[1] associated with ff\_txdata\_ch\_(0-3)[15:8] and ff\_force\_disp\_ch(0-3)[0] associated with ff\_txdata\_ch\_(0-3)[7:0].

**ff\_disp\_sel\_ch(0-3)[0]** - Per channel, disparity value supplied from FPGA logic. It is valid when ff\_force\_disp\_ch(0-3)[0] is high.

In 16 Bit Data Bus Mode, ff\_disp\_select\_ch is 2 bits wide (**ff\_disp\_sel\_ch(0-3)[1:0]**) with ff\_disp\_sel\_ch(0-3)[1] associated with ff\_txdata\_ch\_(0-3)[15:8] and ff\_disp\_sel\_ch(0-3)[0] associated with ff\_txdata\_ch\_(0-3)[7:0].

**ff\_correct\_disp\_ch(0-3)[0]** - Per channel, disparity value supplied from FPGA logic. It is valid when ff\_force\_disp\_ch(0-3)[0] is high.

In 16 Bit Data Bus Mode, ff\_correct\_disp\_ch is 2 bits wide (**ff\_correct\_disp\_ch(0-3)[1:0]**) with ff\_correct\_disp\_ch(0-3)[1] associated with ff\_txdata\_ch\_(0-3)[15:8] and ff\_correct\_disp\_ch(0-3)[0] associated with ff\_txdata\_ch\_(0-3)[7:0].

### 8b10b Encoding

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 the specification, along with maintaining the running disparity rules as specified. The 8b10b encoder can be bypassed on a per channel basis by setting attribute CHx\_8B10B to "BYPASS" where x is the channel number.

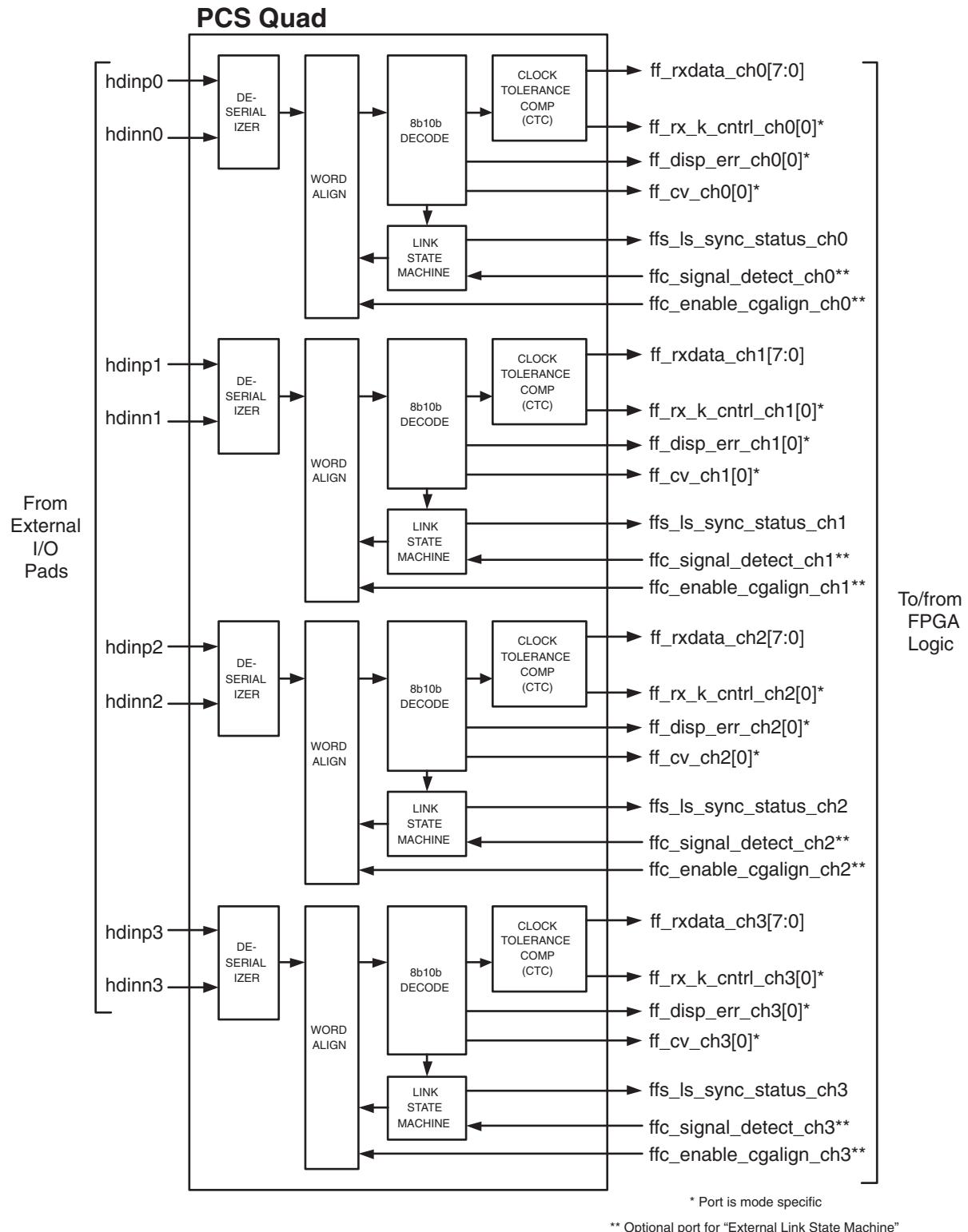
### Serializer

The 8b10b encoded data undergoes parallel to serial conversion and is transmitted off chip via the embedded SERDES. For detailed information on the operation of the LatticeECP2M SERDES, refer to the SERDES Functionality section of this document.

### 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. Table 8-15 shows the four channels of receive data paths in a PCS quad.

Figure 8-15. 8-bit Mode Receive Data Path (1 Quad)



**rxd\_[0-3](7:0)** - Per channel receive data to the FPGA interface. Each quad supports up to 4 independent channels of 8-bit wide parallel data. The rxd\_[0-3] signal transitions synchronously with respect to rclk\_[0-3].

In 16 Bit Data Bus Mode, this bus is 16-bits wide (**rxd\_[0-3](15:0)**) and changes on every other edge of the corresponding rclk\_[0-3].

**rx\_k\_[0-3]** - Per channel, active high control character indicator.

In 16 Bit Data Bus Mode, rx\_k is 2 bits wide (**rx\_k\_[0-3](1:0)**) with rx\_k\_[0-3](1) associated with rxd\_[0-3](15:8) and rx\_k\_[0-3](0) associated with rxd\_[0-3](7:0).

**rx\_disp\_err\_detect\_[0-3]** - Per channel, active high signal driven by the PCS to indicate a disparity error was detected with the associated data.

In 16 Bit Data Bus Mode, rx\_disp\_err\_detect is 2 bits wide (**rx\_disp\_err\_detect\_[0-3](1:0)**) with rx\_disp\_err\_detect\_[0-3](1) associated with rxd\_[0-3](15:8) and rx\_disp\_err\_detect\_[0-3](0) associated with rxd\_[0-3](7:0).

**rx\_cv\_detect\_[0-3]** - Per channel, active high signal driven by the PCS to indicate a code violation was detected with the associated data.

In 16 Bit Data Bus Mode, rx\_cv\_detect is 2 bits wide (**rx\_cv\_detect\_[0-3](1:0)**) with rx\_cv\_detect\_[0-3](1) associated with rxd\_[0-3](15:8) and rx\_cv\_detect\_[0-3](0) associated with rxd\_[0-3](7:0).

**word\_align\_en\_[0-3]** - Signal which unlocks the word aligner on any minimum one clock cycle wide low pulse for the appropriate receive channel.

### Deserializer

Data is brought on chip to the embedded SERDES where it undergoes serial to parallel. For detailed information on the operation of the LatticeECP2M SERDES, refer to the SERDES Functionality section of this document.

### Word Alignment

The word aligner module performs the alignment code word detection and alignment operation. The word alignment character is used by the receive logic to perform 10-bit word alignment upon the incoming data stream. The word aligner can be bypassed on a per channel basis by setting attribute CHx\_COMMA\_ALIGN to "BYPASS" where x is the channel number.

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

- 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 compare. Alignment characters and the mask register is set on a per quad basis. For many protocols, the word alignment characters can be set to "XXX0000011" (jhgfiedcba bits for positive running disparity comma character matching code groups K28.1, K28.5, and K28.7) and "XXX1111100" (jhgfiedcba bits for negative running disparity comma character matching code groups K28.1, K28.5, and K28.7). However the user can define any bit pattern up to 10 bits long.
- 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 attribute CHx\_COMMA\_ALIGN is set to  $\infty$ AUTO $\pm$ , 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 description 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. The 8b10b decoder can be bypassed on a per channel basis by setting attribute CHx\_8B10B to "BYPASS" where x is the channel number.

When a code violation, disparity error, or data error is detected, the ff\_rxdata receive data is set to 0xEE with ff\_rx\_k\_cntl\_ch set to '1'.

### Protocol Specific Link State Machine

The PCS implements link state machines for various protocols that are used in various quad modes.

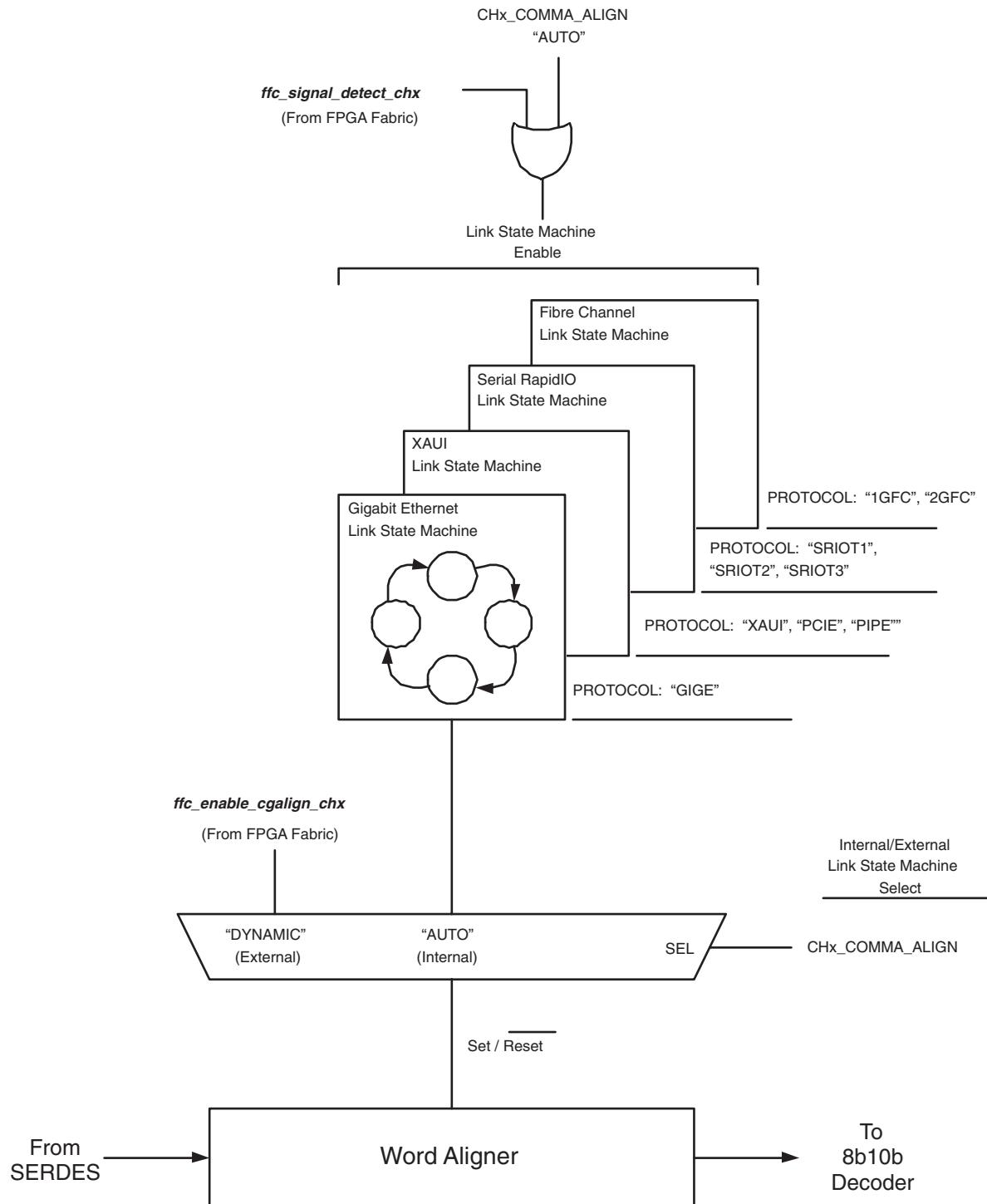
When a protocol specific Link State Machine is selected, that channel's Link State Machine must be enabled by setting the protocol CH(0-3)\_COMMA\_ALIGN to "AUTO". Selection of the specific Link State Machine that is enabled in each mode is described below and summarized in Figure 8-16.

The Link State Machine for Gigabit Ethernet is selected when attribute PROTOCOL is "GIGE". Link synchronization is achieved after the successful detection and alignment of the required number of consecutive aligned code words. The Gigabit Ethernet link synchronization state machine implements the Synchronization State Diagram shown in Figure 36-9 of the 802.3- 2002 1000BASE-X specification.

### External Link State Machine Option

When attribute CHx\_COMMA\_ALIGN is set to "DYNAMIC", the protocol specific Link State Machines are bypassed. In that case, the word aligner will lock alignment (it will stop comparing the incoming data to the user-defined word alignment characters and will maintain current alignment) on the first successful compare to either the first or second user-defined word alignment character after ffc\_enable\_cgalign\_ch(0-3) is pulsed low for at least one clock cycle. Any subsequent low pulse on the ffc\_enable\_cgalign\_ch(0-3) port at the FPGA interface will unlock the word aligner. The word aligner will then re-lock on the next match to one of the user-defined word alignment characters. If desired, ffc\_enable\_cgalign\_ch(0-3) 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 8-20 illustrates the link state machine options.

**Figure 8-16. PCS Word Aligner and Link State Machine Options**

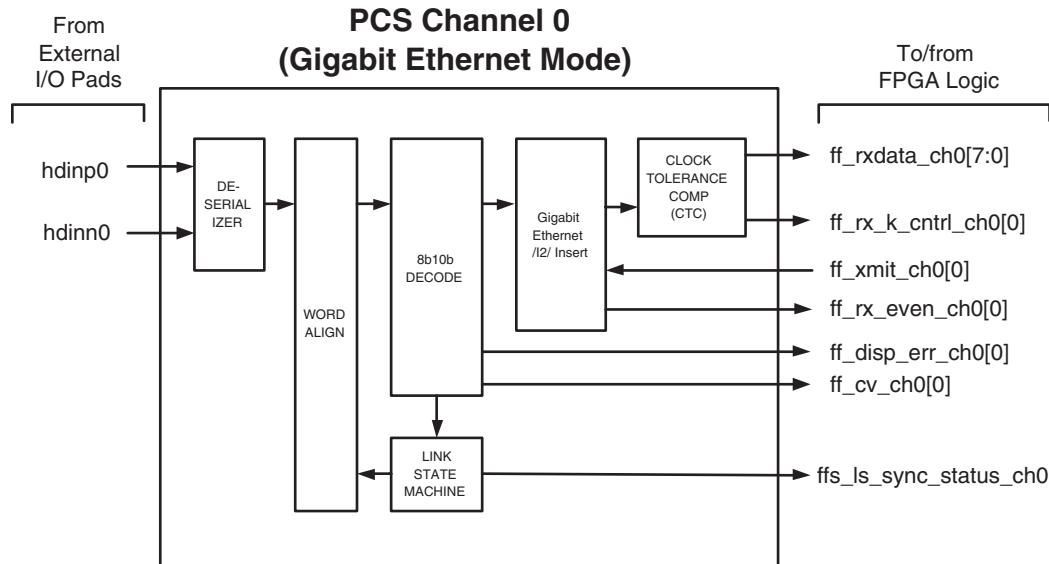
**Table 8-12. Link State Machine Selection for Generic 8b10b Mode**

Attribute Set-up		Word Alignment Control
CHx_COMMA_ALIGN "DYNAMIC"		External Link State Machines selected Word alignment automatically locked on first alignment character match after low pulse on <b>ffc_enable_cgalign_chx</b> . Lock/unlock controlled by a change on the per channel <b>ffc_enable_cgalign_chx</b> signal at FPGA interface
CHx_COMMA_ALIGN "AUTO"	PROTOCOL "GIGE"	Gigabit Ethernet Link State Machine selected and enabled for appropriate channel Word alignment follows Gigabit Ethernet word alignment protocol

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

#### 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 **ff\_xmit\_ch(0-3)** input to the PCS which is driven from the auto-negotiation soft logic. The signal **ff\_rx\_even\_ch(0-3)[0]** from the PCS to the auto-negotiation soft logic is also provided. Figure 8-17 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.

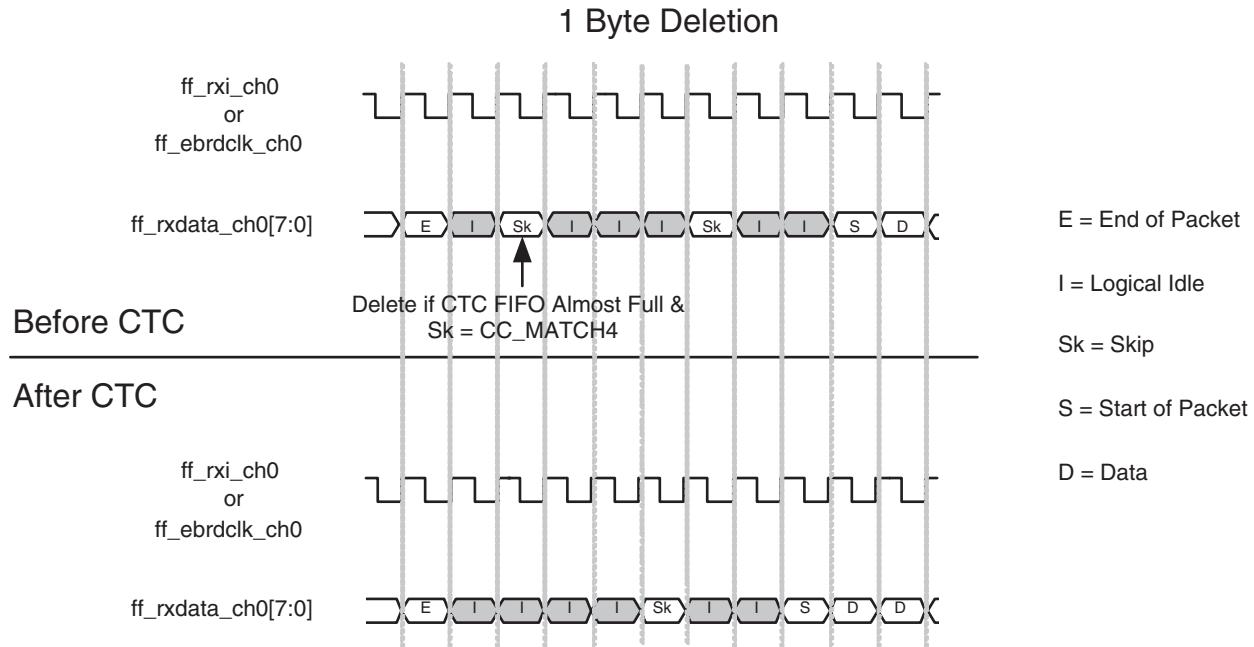
**Figure 8-17. PCS Receive path for Gigabit Ethernet Mode (Channel 0 Example)**

#### Clock Tolerance Compensation

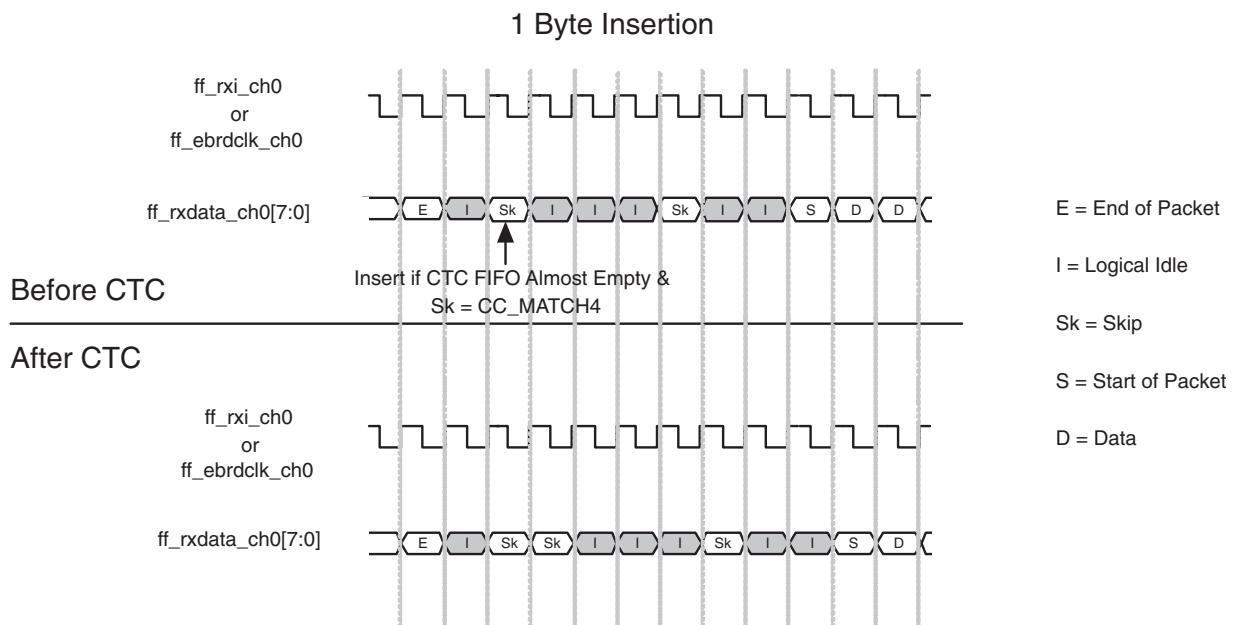
The Clock Tolerance Compensation module performs clock rate adjustment between the recovered receive clocks and the locked reference clock. Clock compensation is performed by inserting or deleting bytes at pre-defined positions, without causing loss of packet data. A 16-Byte elasticity FIFO is used to transfer data between the two clock domains and will accommodate clock differences of up to the specified ppm tolerance for the LatticeECP2M SERDES (See DC and Switching Characteristics section of the LatticeECP2/M Family Data Sheet).

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

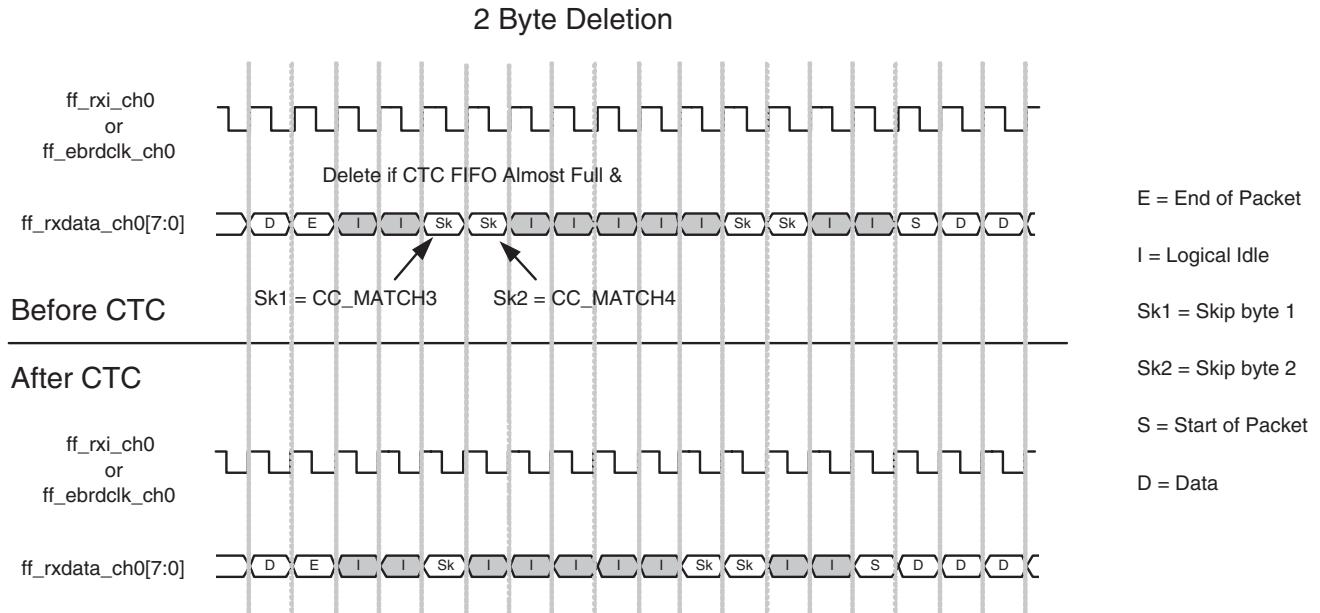
A diagram illustrating 1 byte deletion is shown in Figure 8-18:

**Figure 8-18. Clock Tolerance Compensation 1 Byte Deletion Example**

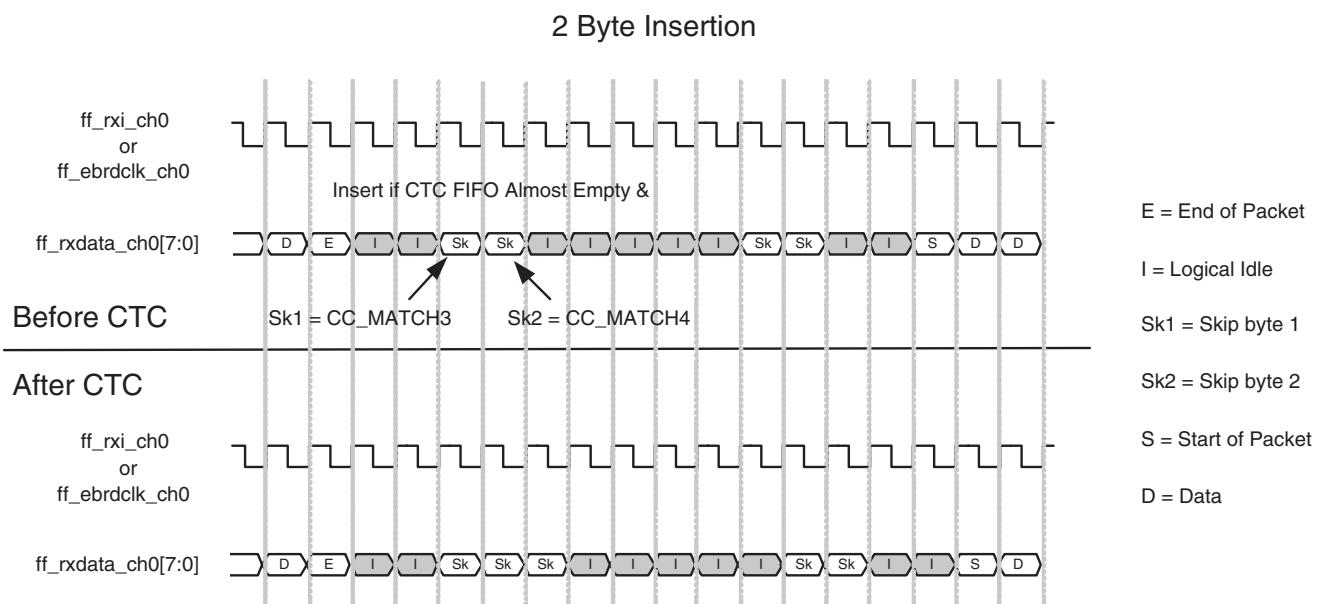
A diagram illustrating 1 byte insertion is shown in Figure 8-19:

**Figure 8-19. Clock Tolerance Compensation 1 Byte Insertion Example**

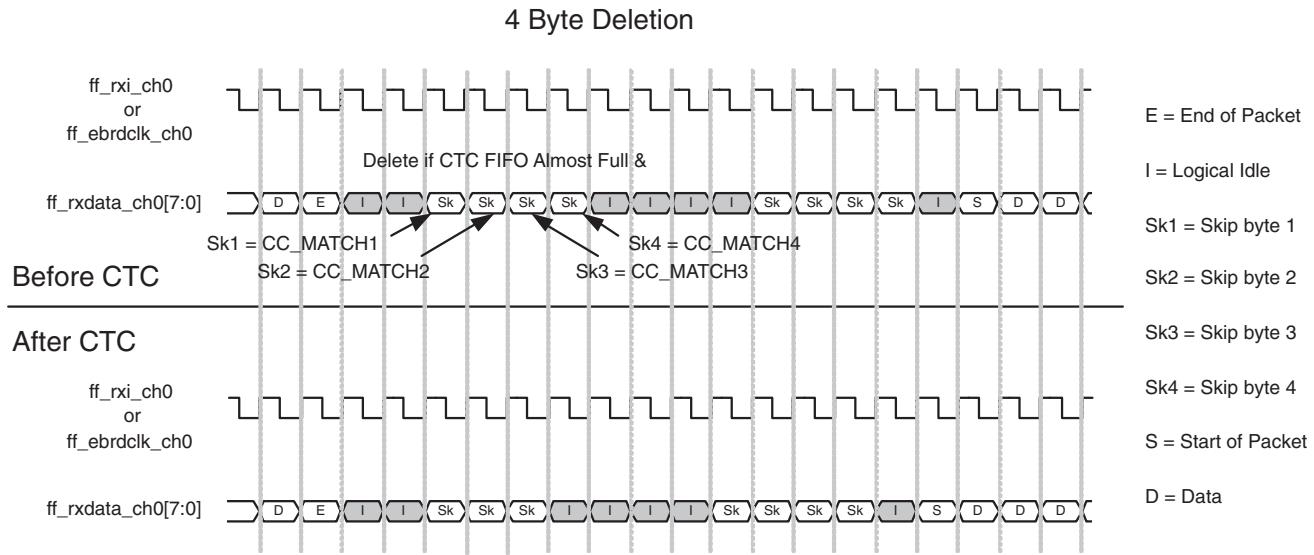
A diagram illustrating 2 byte deletion is shown in Figure 8-20:

**Figure 8-20. Clock Tolerance Compensation 2 Byte Deletion Example**

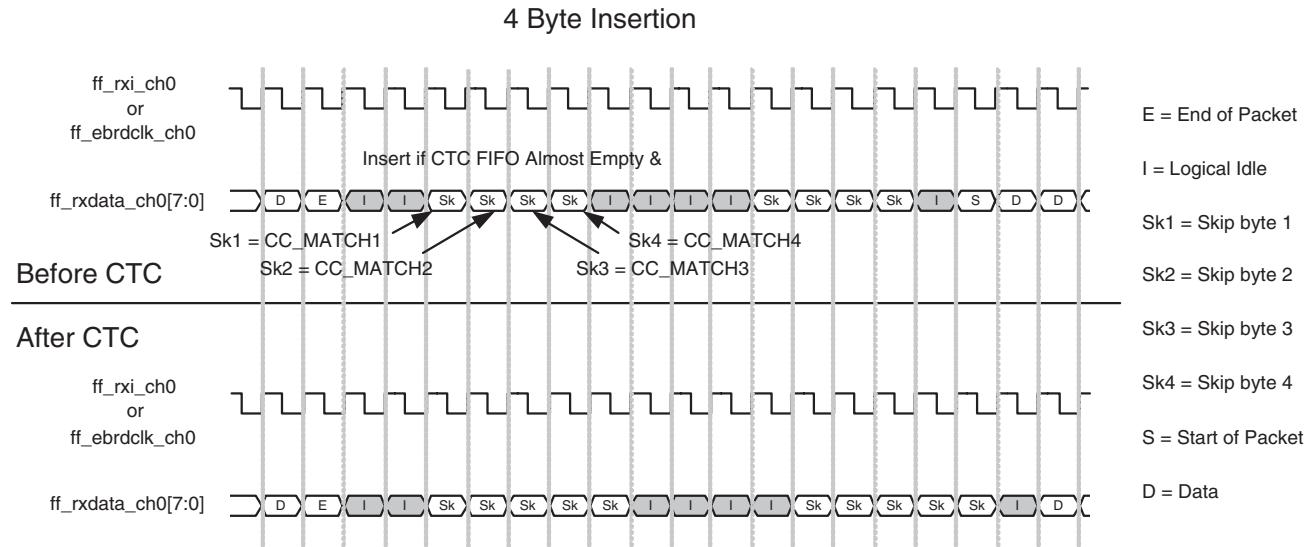
A diagram illustrating 2 byte insertion is shown in Figure 8-21:

**Figure 8-21. Clock Tolerance Compensation 2 Byte Insertion Example**

A diagram illustrating 4 byte deletion is shown in Figure 8-22:

**Figure 8-22. Clock Tolerance Compensation 4 Byte Deletion Example**

A diagram illustrating 4 byte insertion is shown in Figure 8-23:

**Figure 8-23. Clock Tolerance Compensation 4 Byte Insertion Example**

Clock compensation values are set on a quad basis. The CTC can be bypassed on a per channel basis by setting attribute `CHx_CTC_BYP` to “BYPASS” where  $x$  is the channel number. Setting `CHx_CTC_BYP` to “NORMAL” means the CTC is active. In the isplever module generator, defining a channel as “Single” creates an auto-configuration file which enables CTC. Defining a channel as “MAC Group1” or “MAC Group 2” creates an auto-configuration file which bypasses CTC. 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_MATCHMODE` attribute. This sets the number of skip bytes the CTC compares to before performing an insertion or deletion. Values for `CC_MATCHMODE` are “MATCH\_4” (1 byte insertion/deletion), “MATCH\_3\_4” (2 bytes insertion/deletion), and “MATCH\_1\_2\_3\_4” (4 bytes insertion/deletion) to 1. The minimum inter-packet gap must also be set as appropriate for the targeted application. The inter-packet 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 inter-packet gap after skip character deletion is performed based on these attribute settings is described in Table 8-13 below.

- The Skip byte or ordered set must be set corresponding to the CC\_MATCHMODE chosen. For 4 byte insertion/deletion (CC\_MATCHMODE = “MATCH\_1\_2\_3\_4”), the first byte must be assigned to attribute MATCH\_1, the second byte must be assigned to attribute MATCH\_2, the third byte must be assigned to attribute MATCH\_3, and the fourth byte must be assigned to attribute MATCH\_4. 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 “MATCH\_1” should be “0110111100”, “MATCH\_2”, = “0010010101”, and “MATCH\_3” = “MATCH\_4” = “0010110101”. For 2 byte insertion/deletion (CC\_MATCHMODE = “MATCH\_3\_4”), the first byte must be assigned to attribute MATCH\_3, and the second byte must be assigned to attribute MATCH\_4. For 1 byte insertion/deletion (CC\_MATCHMODE = “MATCH\_4”), the skip byte must be assigned to attribute MATCH\_4.
- 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 although 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 ffs\_cc\_overrun\_ch(0-3) if “Error Status Ports” is selected when generating the PCS block with the ispLEVER module generator.
- Clock compensation FIFO underrun can be monitored on a per channel basis on the PCS/FPGA interface port labeled ffs\_cc\_underrun\_ch(0-3) if “Error Status Ports” is selected when generating the PCS block with the ispLEVER module generator.

### Calculating Minimum Inter-packet Gap

Table 8-13 shows the relationship between the user-defined values for inter-packet 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 inter-packet 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\_MATCHMODE is set to “MATCH\_1\_2\_3\_4”), and the minimum inter-packet gap attribute CC\_MIN\_IPG is set to “2”, then the minimum inter-packet gap is equal to 4 (CC\_MATCHMODE = “MATCH\_1\_2\_3\_4”) times 3 (Table 8-13 with CC\_MIN\_IPG = “2”) or 12 bytes. The PCS will not perform a skip character deletion until the minimum number of inter-packet bytes have passed through the CTC.

**Table 8-13. Minimum Inter-packet Gap Multiplier**

CC_MIN_IPG	Insertion/Deletion Multiplier Factor
“0”	1 X
“1”	2 X
“2”	3 X
“3”	4 X

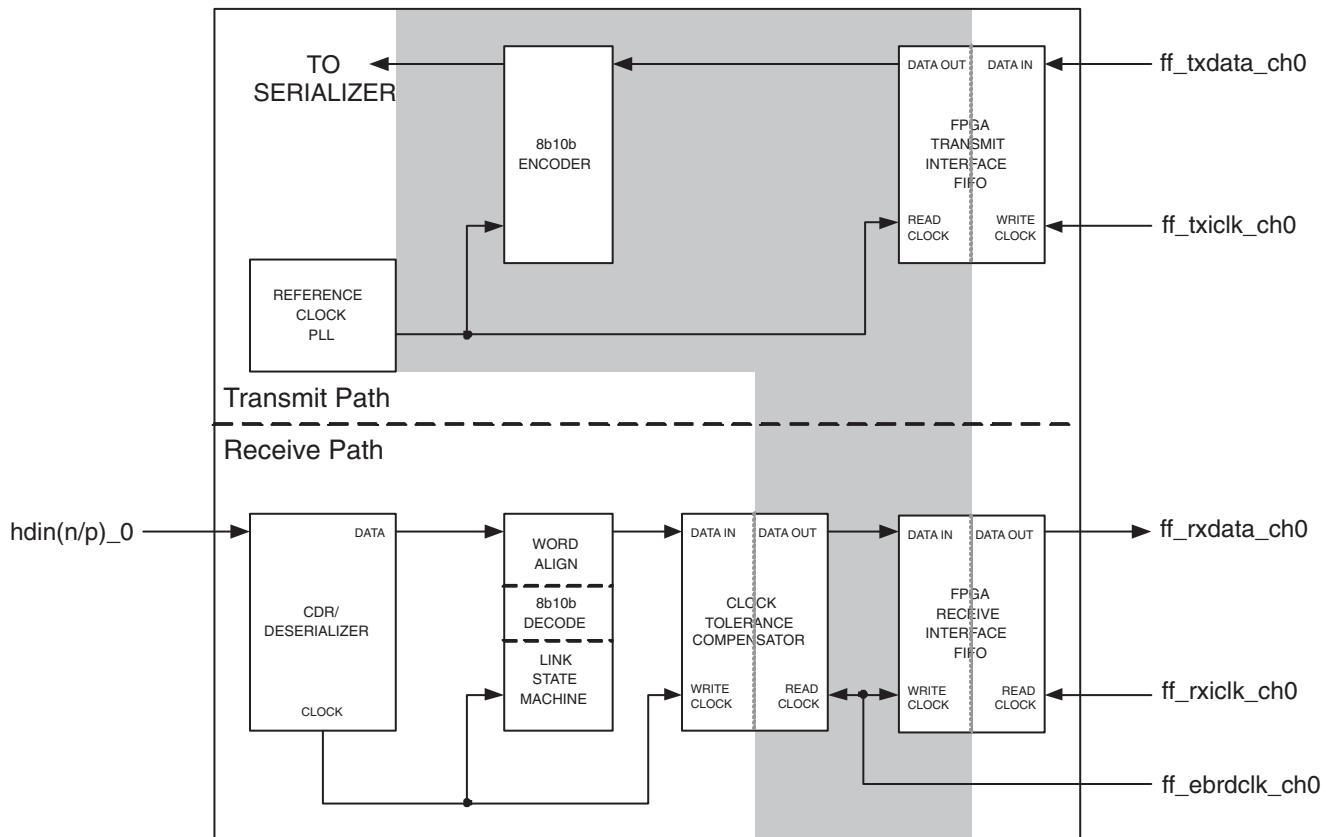
### Clock Domains

Figure 8-24 shows the clock domains for both transmit and receive directions for a single channel inside the PCS for modes which utilize the Clock Tolerance Compensation (CTC) block.

On the transmit side, a clock domain transfer from the ff\_txclk\_ch input at the FPGA interface to the locked reference clock occurs in the FPGA Transmit Interface FIFO. The FPGA Transmit Interface FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These FIFOs (one per channel) cannot compensate for frequency variations.

On the receive side, a clock domain transfer occurs from the channel recovered clocks to the locked reference clock at the Clock Tolerance Compensator (CTC) block. The CTC can adjust for frequency differences between the recovered receive clock and the reference clock up to the maximum phase difference specification for the LatticeECP2M. Downstream from the CTC, another clock interface occurs between the locked reference clock and the ff\_rxclk\_ch at the FPGA Receive Interface FIFO. The FPGA Receive Interface FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. The FPGA Receive Interface FIFOs (one per channel) cannot compensate for frequency variations.

**Figure 8-24. PCS Clock Domain Transfers for CTC Modes**



To guarantee a synchronous interface, both the input transmit clocks and input receive clock should be driven from one of the output reference clocks for a PCS quad set to Generic 8b10b mode. Figure 8-27 illustrates the possible connections that would result in a synchronous interface.

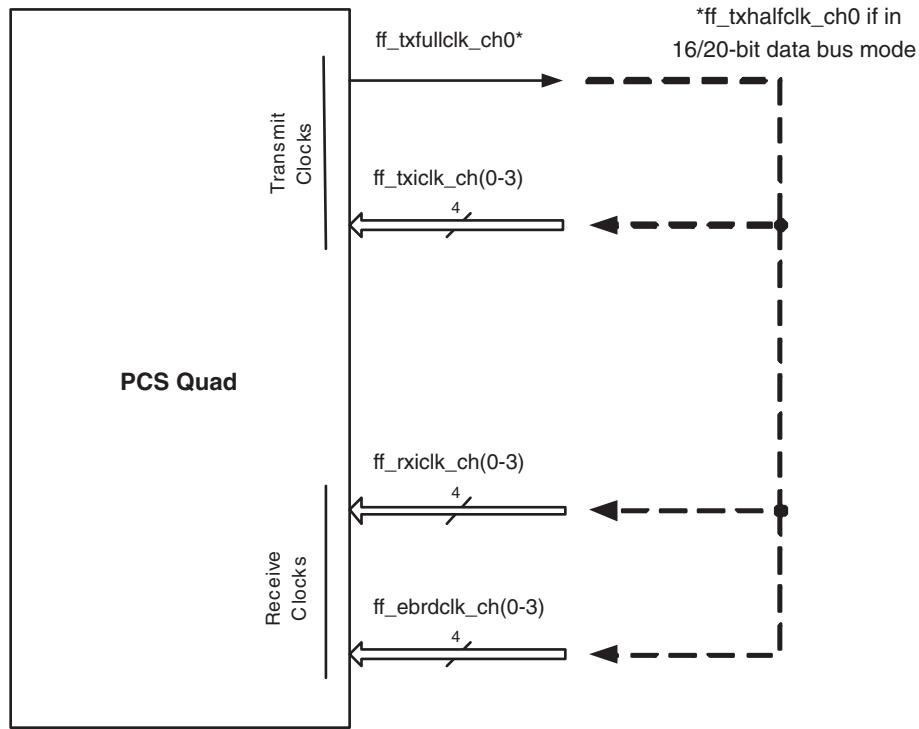
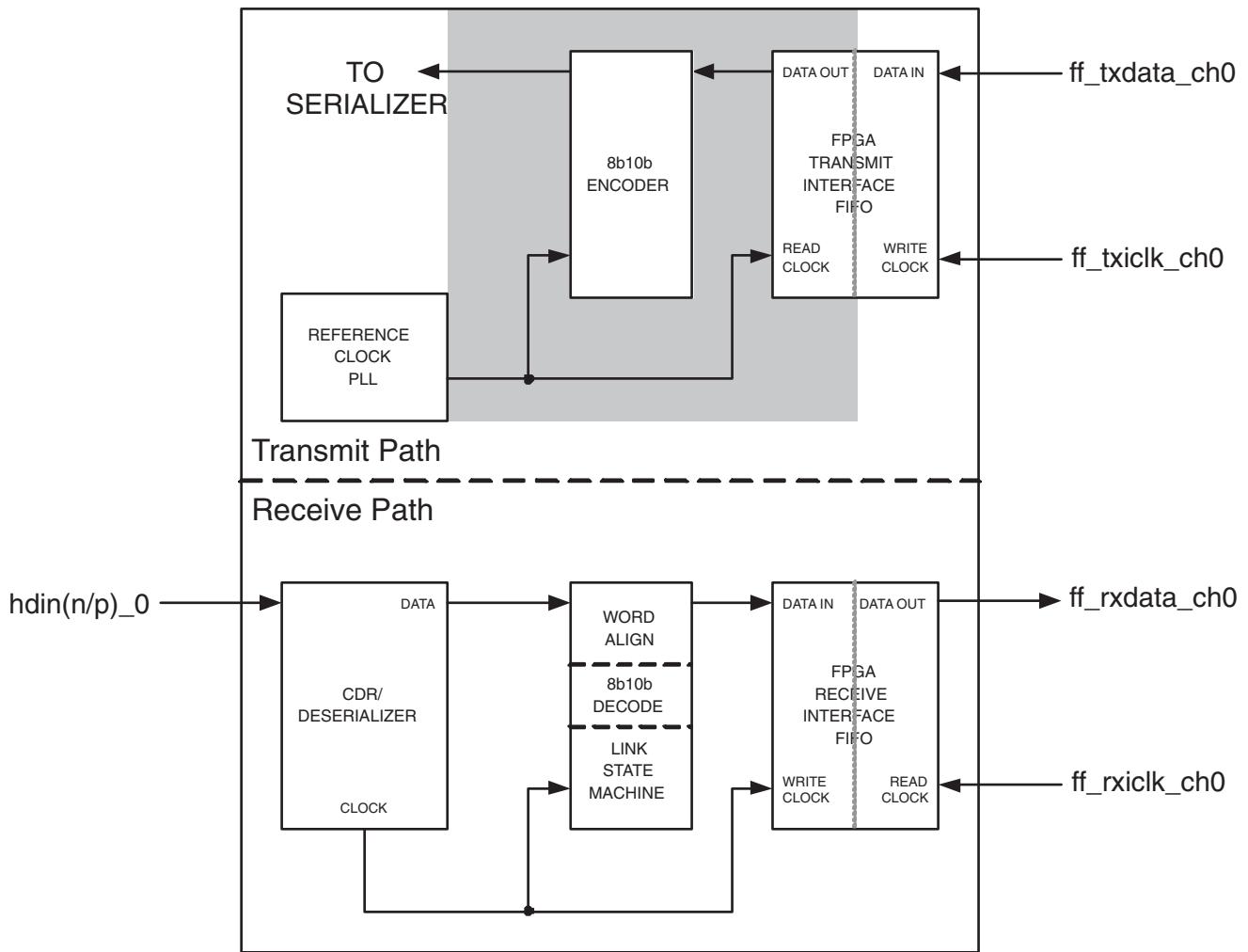
**Figure 8-25. Synchronous input clocks to PCS quad with CTC Used**

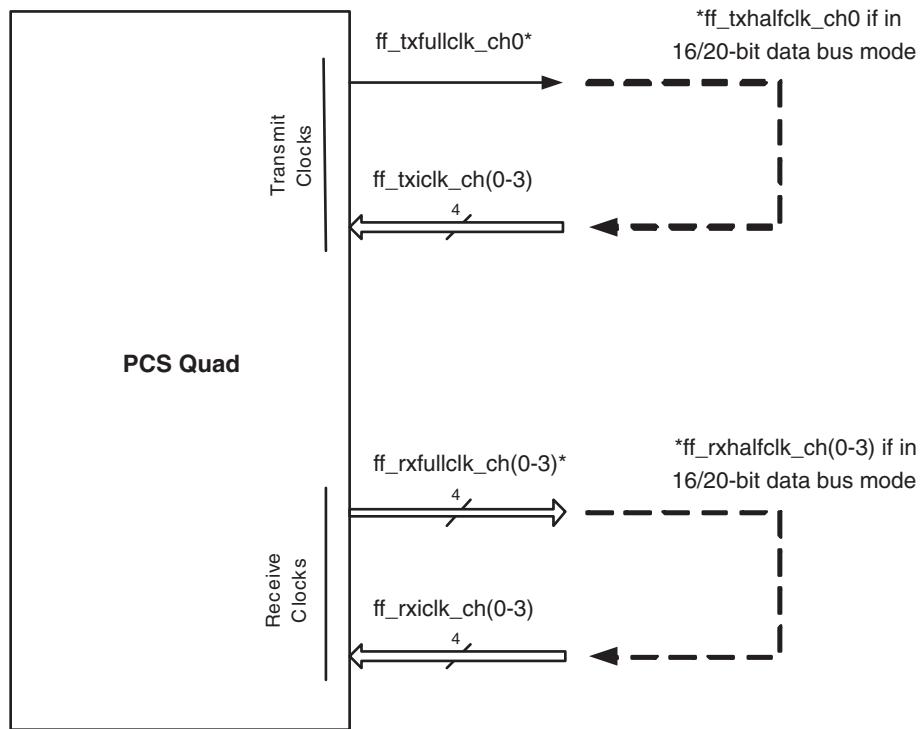
Figure 8-24 shows the clock domains for both transmit and receive directions for a single channel inside the PCS which do not utilize the Clock Tolerance Compensation (CTC) block.

On the transmit side, a clock domain transfer from the `ff_txiclk_ch` input at the FPGA interface to the locked reference clock occurs in the FPGA Transmit Interface FIFO. The FPGA Transmit Interface FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. These FIFOs (one per channel) cannot compensate for frequency variations.

On the receive side, a clock domain transfer occurs between the recovered receive clock and the `ff_rxiclk_ch` at the FPGA Receive Interface FIFO. The FPGA Receive Interface FIFO is intended to adjust for phase differences between two clocks which are of the same frequency only. The FPGA Receive Interface FIFOs (one per channel) cannot compensate for frequency variations.

**Figure 8-26. PCS Clock Domain Transfers for Non-CTC Modes**

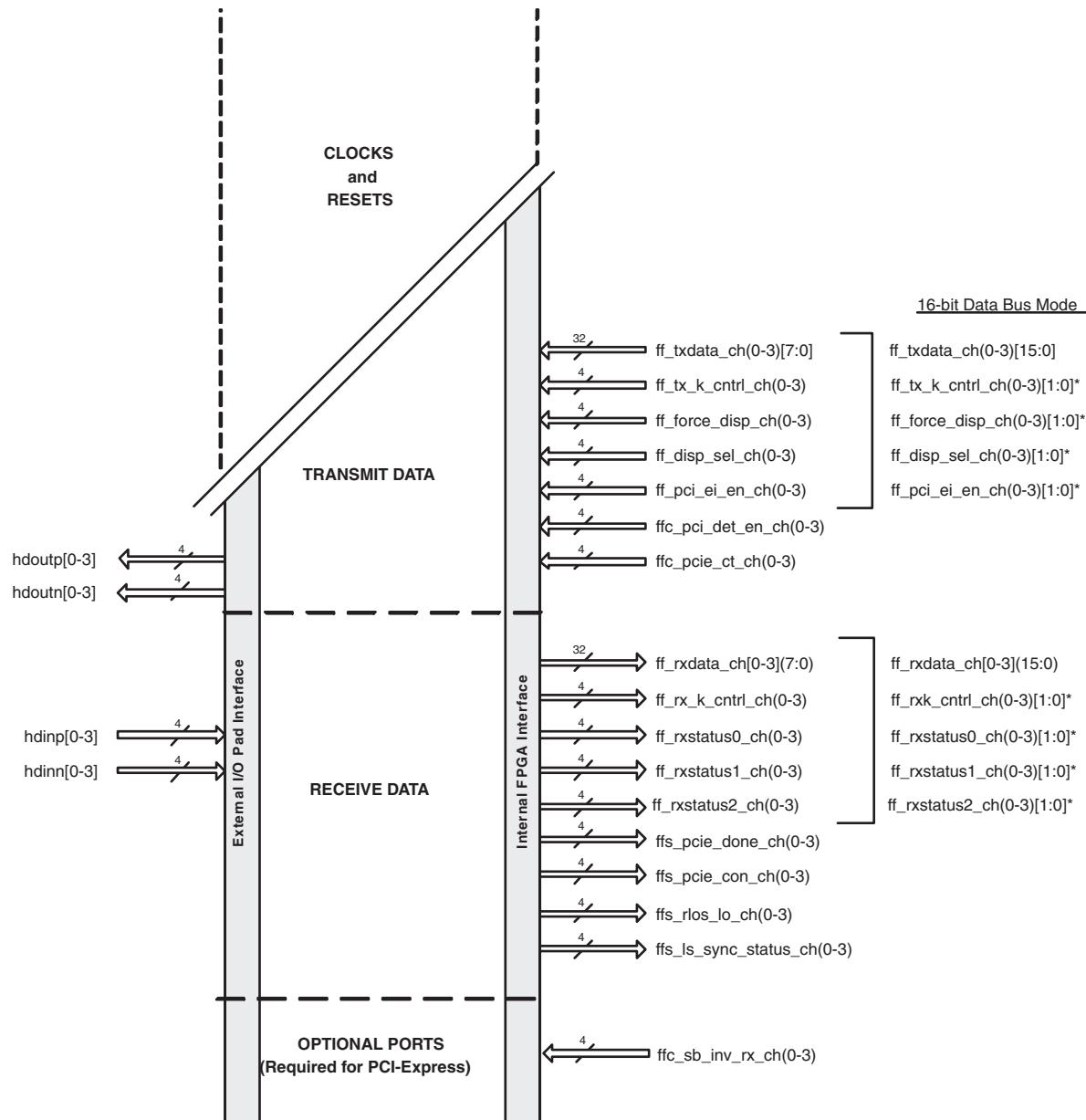
To guarantee a synchronous interface, both the input transmit clocks and input receive clock should be driven from one of the output reference clocks for a PCS quad set to Generic 8b10b mode. Figure 8-27 illustrates the possible connections that would result in a synchronous interface.

**Figure 8-27. Synchronous input clocks to PCS quad with CTC**

## LatticeECP2M PCS in PCI Express Mode

An LatticeECP2M quad set to PCI Express Mode in IPexpress has additional ports at the FPGA interface to enable electrical functions required by the PCI Express specification such as receiver detection. Figure 8-12 displays the resulting port interface to one PCS quad that has been set to PCI Express mode with all four channels enabled. Note that the clocks and resets are not shown as they are the same as other 8-bit modes.

**Figure 8-28. PCI Express Mode Port Diagram**



## PCI Express Mode Pin Description

Table 8-14 lists all default inputs and outputs to/from a PCS quad in PCI Express Mode. A brief description for each port is given in the table. A more detailed description of the function of unique ports are given in the **PCI Express Mode Detailed Description**.

**Table 8-14. PCI Express Mode Pin Description**

Symbol	Direction/ Interface	Description
<b>Reference Clocks</b>		
refclkp	In from I/O pad	Reference clock input, positive. Dedicated CML input.
refclkn	In from I/O pad	Reference clock input, negative. Dedicated CML input
core_refclk	In from FPGA	Optional reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.  The ff_refclk inputs for all active quads on a device must be connected to the same clock
core_rxrefclk	In from FPGA	Optional receive reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.
<b>Transmit Clocks</b>		
ff_txfullclk_ch0	Out to FPGA	Full rate transmit PLL clock when in 8 bit data bus mode.
ff_txhalfclk_ch0*		*Half rate transmit PLL clock when in 16 bit data bus mode.
ff_txiclk_ch[0-3]	In 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>Receive Clocks</b>		
ff_rxfullclk_ch[0-3]	Out to FPGA	Full rate per channel Receive Channel Recovered Clock when in 8-bit data bus mode. For non-CTC modes, the source is always the channel's recovered clock. For standards such as GbE, 10 GbE that support clock tolerance compensation, the source is the respective transmit channel's system clock.
ff_rxhalfclk_ch[0-3]*		*Half rate per channel Receive Channel Recovered Clock when in 16-bit data bus mode
ff_rxiclk_ch[0-3]	In from FPGA	Per channel receive clock inputs from FPGA. Used to clock the RX FPGA Interface FIFO with a clock synchronous to the reference and/or receive reference clock.
ff_ebrdclk_ch[0-3]	In from FPGA	Per channel receive clock inputs from FPGA. Used to clock the read side of the CTC FIFO.
ffs_rlol_ch[0-3]	Out to FPGA	Per channel receive CDR loss of lock.  1 = Receive CDR Loss of Lock 0 = Lock maintained
ffs_plol		Receive PLL loss of lock.  1 = Receive PLL Loss of Lock 0 = Lock maintained

**Table 8-14. PCI Express Mode Pin Description (Continued)**

Symbol	Direction/ Interface	Description
<b>Resets</b>		
ffc_quad_rst	In from FPGA	Active high, asynchronous input. Resets all SERDES channels including the auxiliary channel and PCS.
ffc_macro_rst	In from FPGA	Active high, asynchronous input to SERDES quad. Resets all SERDES channels including the auxiliary channel but not PCS logic.
ffc_trst	In from FPGA	Active high reset. Resets selected digital logic in all transmit channels
ffc_lane_tx_rst_ch[0-3]	In from FPGA	Per channel, active high, asynchronous reset. Resets individual channel TX PCS logic.
ffc_rrst_ch[0-3]	In from FPGA	Per channel active high reset. Resets selected digital logic in corresponding receive channel.
ffc_lane_tx_rst_ch[0-3]	In from FPGA	Per channel, active high, asynchronous reset. Resets individual channel RX PCS logic.
ffc_txpwdnb_ch[0-3]	In from FPGA	Per channel active low Transmit Channel Power Down.
ffc_rxpwdnb_ch[0-3]	In from FPGA	Per channel active low Receive Channel Power Down.
<b>Transmit Data</b>		
hdoutp[0-3]	Out to I/O pad	High-speed CML serial output, positive.
hdoutn[0-3]	Out to I/O pad	High-speed CML serial output, negative.
ff_txdata_ch(0-3)[7:0]	In from FPGA	Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide in 8-bit data bus mode.
ff_txdata_ch(0-3)[15:0]*		*Bus is 16-bits wide in 16-bit data bus mode.
ff_tx_k_cntl_ch(0-3)	In from FPGA	Per channel, active high control character indicator.
ff_tx_k_cntl_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_force_disp_ch(0-3)	In from FPGA	Per channel, active high signal which instructs the PCS to accept disparity value from the ff_disp_sel_ch(0-3) FPGA interface input.
ff_force_disp_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_disp_sel_ch(0-3)	In from FPGA	Per channel, disparity value supplied from FPGA logic. It is valid when ff_force_disp_ch(0-3) is high.
ff_disp_sel_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_pci_ei_en_ch(0-3)	In from FPGA	Per channel active high electrical idle enable. Puts SERDES transmit buffers into electrical idle state.
ff_pci_ei_en_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ffc_pci_det_en_ch(0-3)	In from FPGA	Per channel active high Receiver Detect enable.
ffc_pci_det_en_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ffc_PCIE_ct_ch(0-3)	In from FPGA	Per channel active high Receiver Detect test start.
ffc_PCIE_ct_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
<b>Receive Data</b>		
hdinp[0-3]	In from I/O pad	High-speed CML serial input, positive.
hdinn[0-3]	In from I/O pad	High-speed CML serial input, negative.
ff_rxdata_ch(0-3)[7:0]	Out to FPGA	Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide in 8-bit data bus mode.
ff_rxdata_ch(0-3)[15:0]*		*Bus is 16-bits wide in 16-bit data bus mode.

**Table 8-14. PCI Express Mode Pin Description (Continued)**

Symbol	Direction/ Interface	Description
ff_rx_k_cntl_ch(0-3)	In from FPGA	Per channel, active high control character indicator.
ff_rx_k_cntl_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_rxstatus0_ch(0-3)	Out to FPGA	Per channel PCI Express receive status port. <b>ff_rxstatus_ch is an encoded status of the receive data path for compliance with PIPE.</b>
ff_rxstatus0_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_rxstatus1_ch(0-3)	Out to FPGA	Per channel PCI Express receive status port. <b>ff_rxstatus_ch is an encoded status of the receive data path for compliance with PIPE.</b>
ff_rxstatus1_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ff_rxstatus2_ch(0-3)	Out to FPGA	Per channel PCI Express receive status port. <b>ff_rxstatus_ch is an encoded status of the receive data path for compliance with PIPE.</b>
ff_rxstatus2_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
ffs_PCIE_done_ch(0-3)	Out to FPGA	Per channel Receiver Detect test done status port.
ffs_PCIE_con_ch(0-3)	Out to FPGA	Per channel Receiver Detection status port.
ffs_RLOS_lo_ch(0-3)	Out to FPGA	Per channel Loss of Signal detection.
ffs_ls_sync_status_ch(0-3)	Out to FPGA	Per channel active link status.  1 = Receive channel is synchronized to commas 0 = Receive channel has not found comma
<b>Optional Ports (Required for PCI Express)</b>		
ffc_sb_inv_rx_ch(0-3)	In from FPGA	Per channel control to invert received data. 1 = Invert the received data bits. 0 = Do not invert the data.

ff\_rxstatus\_ch is an encoded status of the receive data path for compliance with PIPE. The encoding of this status bus is shown in Table 8-15 below:

**Table 8-15. ff\_rxstatus Port Decoding**

ff_rxstatus[2:0]			Description
0	0	0	Received data OK
0	0	1	1 SKP inserted by CTC
0	1	0	1 SKP deleted by CTC
0	1	1	Receiver detected
1	0	0	8b10b decoder error (code violation)
1	0	1	CTC FIFO overflow
1	1	0	CTC FIFO underflow
1	1	1	Receive data disparity error

## PCI Express Mode Detailed Description

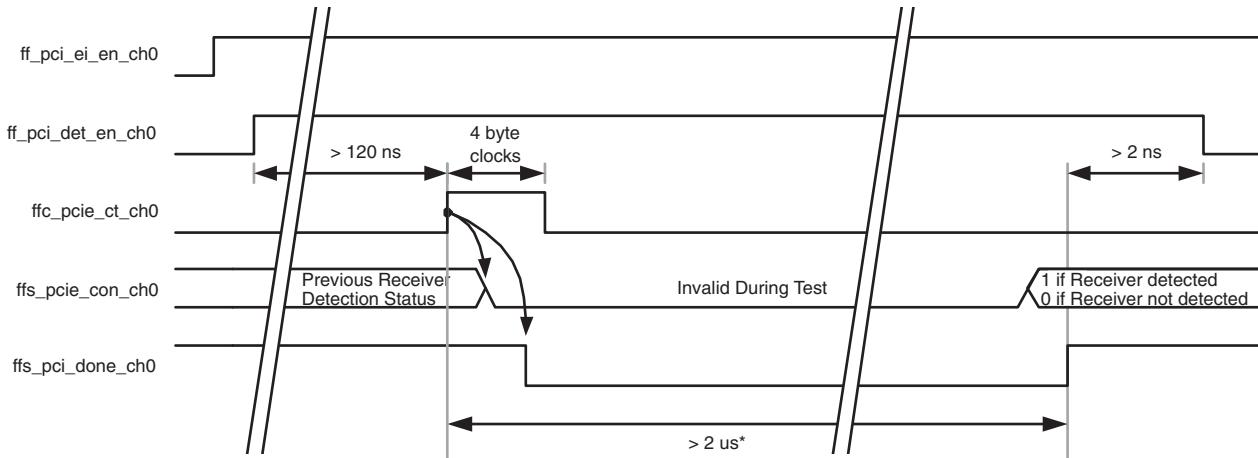
The following section provides a detailed description of the operation of a PCS quad in PCI Express mode. The functional description is organized according to the port listing shown in Figure 8-29.

### Receiver Detection

Figure 8-29 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 **ff\_pci\_ei\_en\_ch** input high. The Receiver Detection test can begin 120 ns after **tx\_elec\_idle** is set high by driving the appropriate **ffc\_pcie\_det\_en\_ch** 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 120ns, the receiver detect test can be initiated by driving the channel's **ffc\_pcie\_ct\_ch** input high for four byte (word) clock cycles. The corresponding channel's **ffc\_pcie\_done\_ch** 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 **ffs\_pcie\_done\_ch** receiver detect status port will go high and the Receiver Detect status can be monitored at the **ffs\_pcie\_con\_ch** port. If at that time the **ffs\_pcie\_con\_ch** port is high, then a receiver has been detected on that channel. If, however, the **ffs\_pcie\_con\_ch** port is low, then no receiver has been detected for that channel. Once the Receiver Detect test is complete, **ff\_pci\_ei\_en\_ch** can be deasserted.

**Figure 8-29. PCI Express Mode Receiver Detection Sequence (Example for Channel 0)**



### PCI Express Beacon Support

This section highlights how the LatticeECP2M PCS can support Beacon Detection and Transmission. The PCI Express requirements for Beacon Detection are presented with the PCS support for Beacon Transmission and Beacon Detection.

- **Beacon Detection Requirements** (PCI Express Base Specification, Rev 1.0a, Chapter 4, Page 209-210)
  - 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  $\geq$  2ns (500MHz) and  $<$  16us (30Khz).
  - Maximum time between pulses should be  $<$  16us.
  - DC balance must be restored within  $<$  32us.
  - For pulse widths  $>$  500ns, output beacon voltage level must be 6db down from VTX-DIFFp-p (800mV to 1200mV).
  - For pulse widths  $<$  500ns, output beacon voltage level must be  $\leq$  VTX-DIFFp-p and  $\geq$  3.5db down from VTX-DIFFp-p.

- **PCS Beacon Detection Support**

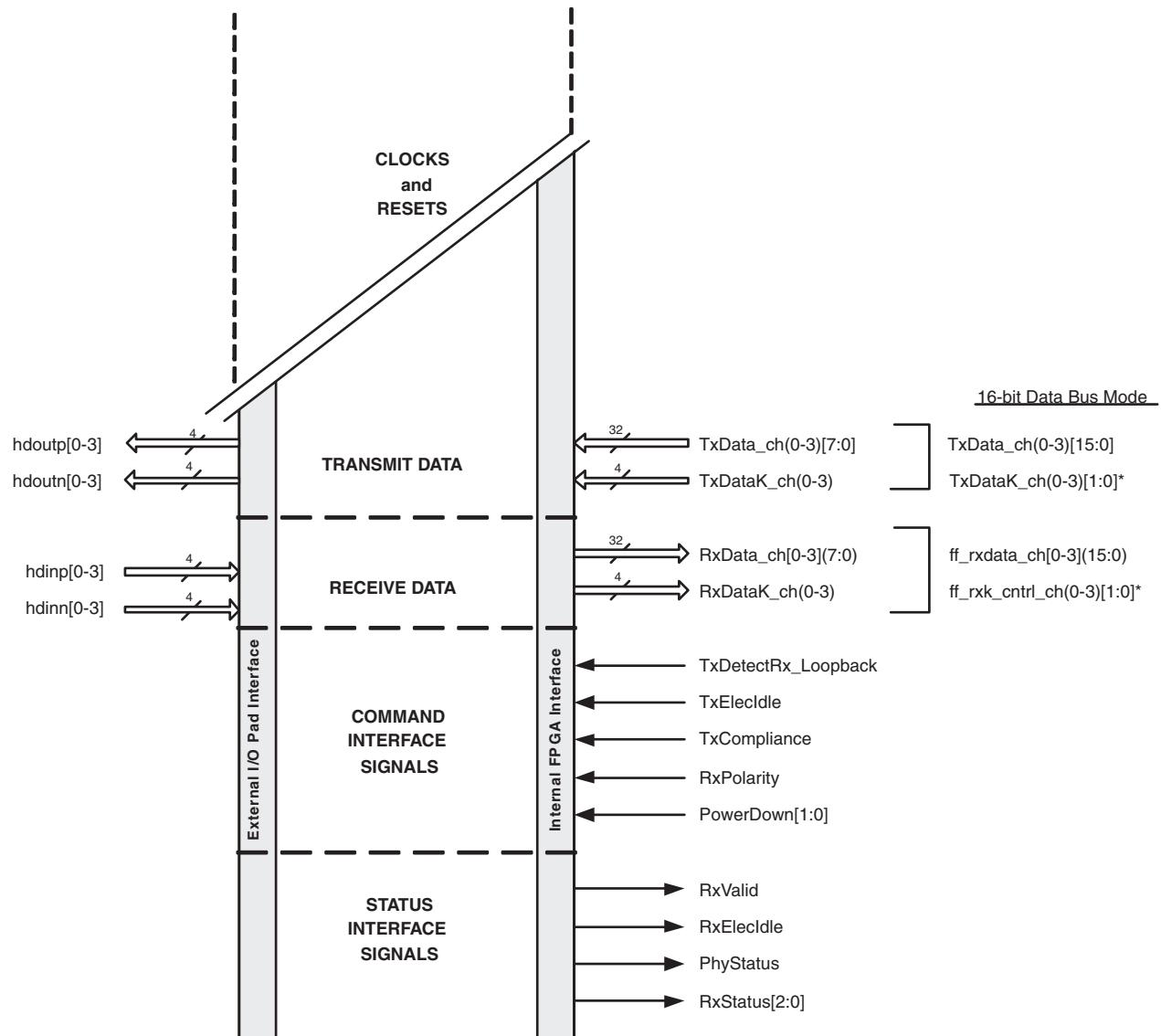
- The signal loss threshold detection circuit senses if the specified voltage level exists at the receiver buffer. This is indicated by ffs\_rlos\_lo\_ch(0-3) signal.
- This setting can be used both for PCI Express Electrical Idle Detection and PCI Express Beacon Detection (when in power state P2).
- The remote transmitting device can have a Beacon Output voltage of 6db down from V<sub>TX-DIFFpp</sub> (i.e 201mV). If this signal can be detected, it can be stated that Beacon is detected.
- Setting rlos\_lset[2:0] register bits to default of 75mV min and 100mV max. This ensures that a voltage below 100mV will be recognized as Electrical Idle and voltage above 100mV (<201mV) will be recognized as Beacon Detect in power state P2.

- PCS Beacon Transmission Support

- Sending the K28.5 character (IDLE) (5 1's followed by 5 0's) provides a periodic pulse with of 2ns occurring every 2ns (1.0UI = 400ps, multiplied by 5 = 2ns). This meets the lower requirement. The output beacon voltage level can then be V<sub>TX-DIFFp-p</sub>. This would be valid Beacon Transmission.

## LatticeECP2M PCS in PIPE Mode

A LatticeECP2M quad set to PIPE Mode in IPexpress has additional ports at the FPGA interface to enable electrical functions required by the PCI Express PIPE specification. Figure 4-20 displays the resulting port interface to one PCS quad that has been set to PCI Express mode with all four channels enabled. Note that the clocks and resets are not shown as they are the same as other 8-bit modes.

**Figure 8-30. PIPE Mode Port Diagram**

## PIPE Mode Pin Description

Table 8-9 lists all default inputs and outputs to/from a PCS quad in PIPE Mode. A brief description for each port is given in the table. A more detailed description of the function of unique ports are given in the PIPE Mode Detailed Description.

**Table 8-16. PCI Express Mode Pin Description**

Symbol	Direction/ Interface	Description
<b>Reference Clocks</b>		
refclkp	In from I/O pad	Reference clock input, positive. Dedicated CML input.
refclkn	In from I/O pad	Reference clock input, negative. Dedicated CML input
core_refclk	In from FPGA	Optional reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.  The ff_refclk inputs for all active quads on a device must be connected to the same clock
core_rxrefclk	In from FPGA	Optional receive reference clock input from FPGA logic. Can be used instead of I/O pin reference clock.
<b>Transmit Clocks</b>		
ff_txfullclk_ch0	Out to FPGA	Full rate transmit PLL clock when in 8 bit data bus mode.
ff_txhalfclk_ch0*		*Half rate transmit PLL clock when in 16 bit data bus mode.
ff_txiclk_ch[0-3]	In 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>Receive Clocks</b>		
ff_rxfullclk_ch[0-3]	Out to FPGA	Full rate per channel Receive Channel Recovered Clock when in 8-bit data bus mode. In Generic 8b10b mode, the source is always the channel's recovered clock. For standards such as GbE, 10 GbE that support clock tolerance compensation, the source is the respective transmit channel's system clock.
ff_rxhalfclk_ch[0-3]*		*Half rate per channel Receive Channel Recovered Clock when in 16-bit data bus mode
ff_rxiclk_ch[0-3]	In from FPGA	Per channel receive clock inputs from FPGA. Used to clock the RX FPGA Interface FIFO with a clock synchronous to the reference and/or receive reference clock.
ff_ebrdclk_ch[0-3]	In from FPGA	Per channel receive clock inputs from FPGA. Used to clock the read side of the CTC FIFO.
ffs_rlol_ch[0-3]	Out to FPGA	Per channel receive CDR loss of lock.  1 = Receive CDR Loss of Lock 0 = Lock maintained
ffs_plol	Out to FPGA	Receive PLL loss of lock.  1 = Receive PLL Loss of Lock 0 = Lock maintained
<b>Resets</b>		
ffc_quad_RST	In from FPGA	Active high, asynchronous input. Resets all SERDES channels including the auxiliary channel and PCS.
ffc_macro_RST	In from FPGA	Active high, asynchronous input to SERDES quad. Resets all SERDES channels including the auxiliary channel but not PCS logic.
ffc_trst	In from FPGA	Active high reset. Resets selected digital logic in all transmit channels

**Table 8-16. PCI Express Mode Pin Description**

Symbol	Direction/ Interface	Description
ffc_lane_tx_rst_ch[0-3]	In from FPGA	Per channel, active high, asynchronous reset. Resets individual channel TX PCS logic.
ffc_rrst_ch[0-3]	In from FPGA	Per channel active high reset. Resets selected digital logic in corresponding receive channel.
ffc_lane_tx_rst_ch[0-3]	In from FPGA	Per channel, active high, asynchronous reset. Resets individual channel RX PCS logic.
ffc_txpwdnb_ch[0-3]	In from FPGA	Per channel active low Transmit Channel Power Down.
ffc_rxpwdnb_ch[0-3]	In from FPGA	Per channel active low Receive Channel Power Down.
<b>Transmit Data</b>		
hdoutp[0-3]	Out to I/O pad	High-speed CML serial output, positive.
hdoutn[0-3]	Out to I/O pad	High-speed CML serial output, negative.
TxData_ch(0-3)[7:0]	In from FPGA	Per channel parallel transmit data bus from the FPGA. Bus is 8-bits wide in 8-bit data bus mode.
TxData_ch(0-3)[15:0]*		*Bus is 16-bits wide in 16-bit data bus mode.
TxDataK_ch(0-3)	In from FPGA	Per channel, active high control character indicator.
TxDataK_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
<b>Receive Data</b>		
hdinp[0-3]	In from I/O pad	High-speed CML serial input, positive.
hdinn[0-3]	In from I/O pad	High-speed CML serial input, negative.
RxData_ch(0-3)[7:0]	Out to FPGA	Per channel parallel receive data bus to the FPGA. Bus is 8-bits wide in 8-bit data bus mode.
RxData_ch(0-3)[15:0]*		*Bus is 16-bits wide in 16-bit data bus mode.
RxDataK_ch(0-3)	In from FPGA	Per channel, active high control character indicator.
RxDataK_ch(0-3)[1:0]*		*2 bits wide if in 16-bit data bus mode.
<b>Command Interface Signals</b>		
TxDetectRx_Loopback	In from FPGA	Used to tell the PHY to begin a receiver detection operation or to begin loopback 1 = Request transmitter to do far-end receiver detection 0 = Normal data operation
TxEleclidle	In from FPGA	Forces TX Output to electrical idle when asserted in all power states. 1 = Force SERDES transmitter to output Electrical Idle 0 = Normal operation  <ul style="list-style-type: none"> <li>• When de-asserted while in P0 (operational) (as indicated by powerdown signals), indicates that there is valid data present on the TxData and TxDataK pins and that the data should be transmitted.</li> <li>• When de-asserted in P2 (lowest power state) (as indicated by the Power-Down signals), indicates that the PHY should begin transmitting beacon signalling. In this case the signal is asynchronous. TxEleclidle must always be asserted while in power states P0s and P1 (as indicated by the Power-Down signals).</li> </ul>
TxCompliance	In from FPGA	Sets the running disparity to negative. Used when transmitting the compliance pattern.
RxPolarity	In from FPGA	Tells PHY to do a polarity inversion on the received data. 0 = PHY does no polarity inversion 1 = PHY does polarity inversion

**Table 8-16. PCI Express Mode Pin Description**

Symbol	Direction/ Interface	Description
PowerDown[1:0]	In from FPGA	<p>00: P0, Normal Operation</p> <p>01: P0s, Power Saving, low recovery latency (Normal Operation)</p> <p>10: P1, Lower Power, Longer recovery latency (64us) (Normal Operation)</p> <p>11: P2, Lowest Power (Power Down)</p> <p>In the SERDES/PCS quad, two states are supported – Normal Operation and Power Down. Power Down occurs only when PowerDown[1:0] = 11.</p> <p>When connecting, a PIPE compliant IP to the block it is required that the connection be as following:</p> $\text{ffc_rxpwdnb} \leq \neg (\text{PowerDown}[1] \& \text{PowerDown}[0])$ $\text{ffc_txpwdnb} \leq \neg (\text{PowerDown}[1] \& \text{PowerDown}[0])$
<b>Status Interface Signals</b>		
RxValid	Out to FPGA	<p>Indicates symbol lock and valid data on RxData and RxDataK.</p> <p>1 = Lane is synchronized to commas</p> <p>0 = Lane has not found comma</p>
RxEleidle	Out to FPGA	<p>Indicates receiver detection of an electrical idle.</p> <p>Loss of Signal detection for each channel. Includes a high value and a low value, but only the low value will be sent to the FPGA interface. Register bits are used to set the low value and high value thresholds for these loss-of-signal pins.</p>
PhyStatus	Out to FPGA	<p>Used to indicate completion of several PHY functions including power management state transitions, and receiver detection. When this signal transitions during entry and exit from P2 and PCLK is not running, this signal is asynchronous.</p>
RxPolarity	In from FPGA	<p>Encodes receiver status &amp; error codes for the received data stream &amp; Receiver detection</p> <p>NOTE: A priority is required. This is shown in Table 8-17.</p>

ff\_rxstatus\_ch is an encoded status of the receive data path for compliance with PIPE. The encoding of this status bus is shown in Table 8-17.

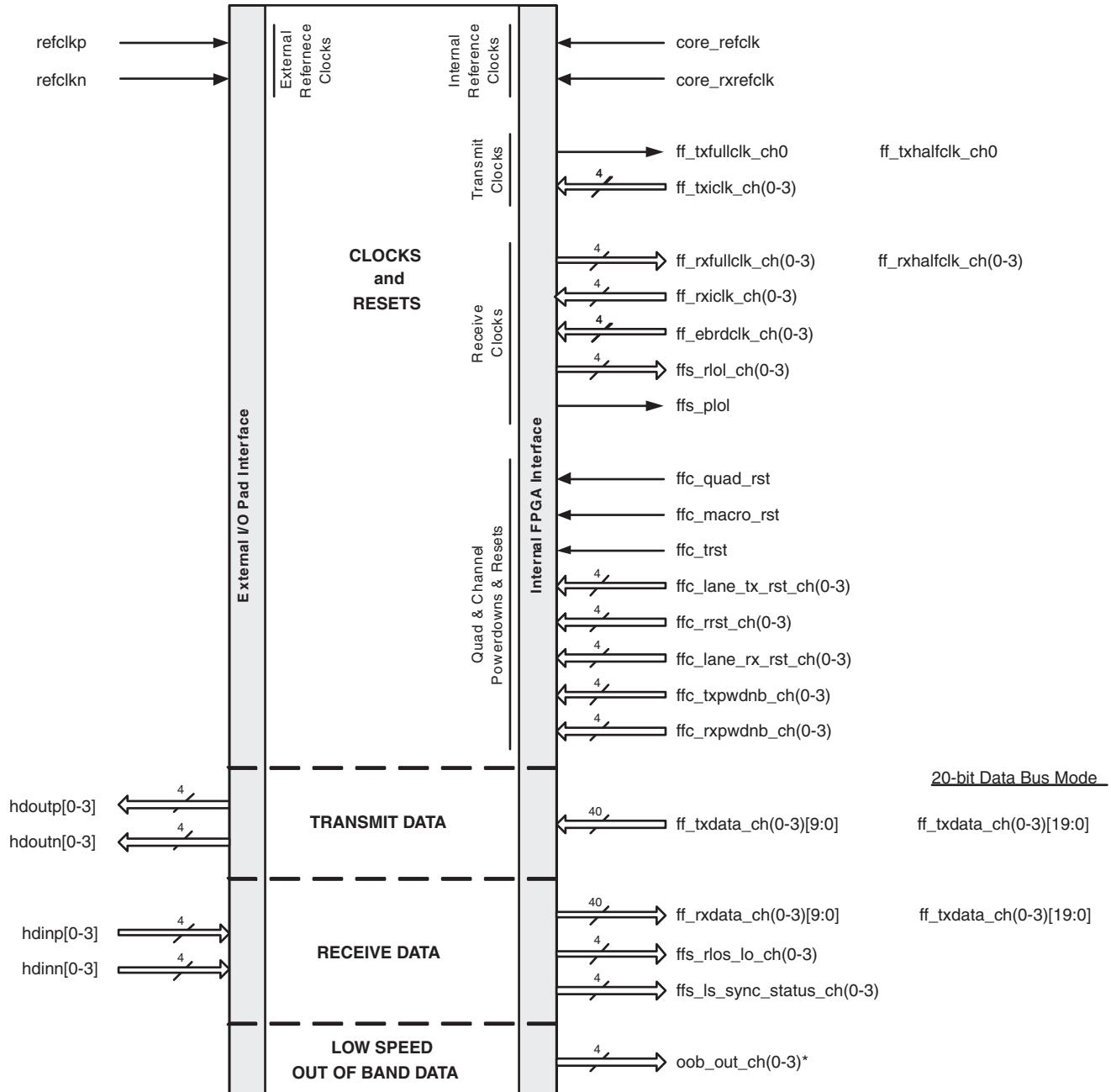
**Table 8-17. Rx Polarity Decoding and Priority**

ff_rxstatus[2:0]			Description	Priority
0	0	0	Received data OK	8
0	0	1	1 SKP inserted by CTC	7
0	1	0	1 SKP deleted by CTC	6
0	1	1	Receiver detected	1
1	0	0	8b10b decoder error (code violation)	2
1	0	1	CTC FIFO overflow	3
1	1	0	CTC FIFO underflow	4
1	1	1	Receive data disparity error	5

## LatticeECP2M PCS 10-bit Modes

The ispLEVER module generator tools allows the designer to choose the mode for each PCS quad used in a given design. Figure 8-31 displays the resulting default port interface to one PCS quad that has been set to one of the 10-bit data modes (10-bit SERDES Only mode) with all four channels enabled.

**Figure 8-31. 10-bit Mode Default Port Diagram**



## 10-bit Mode Pin Description

The pin description for the 10-bit modes are the same as for 8-bit modes except for a 10-bit transmit data bus, ff\_txdata\_ch(0-3)[9:0], and a 10-bit receive data bus, ff\_rxdata\_ch(0-3)[9:0].

## PCS Loopback Testing

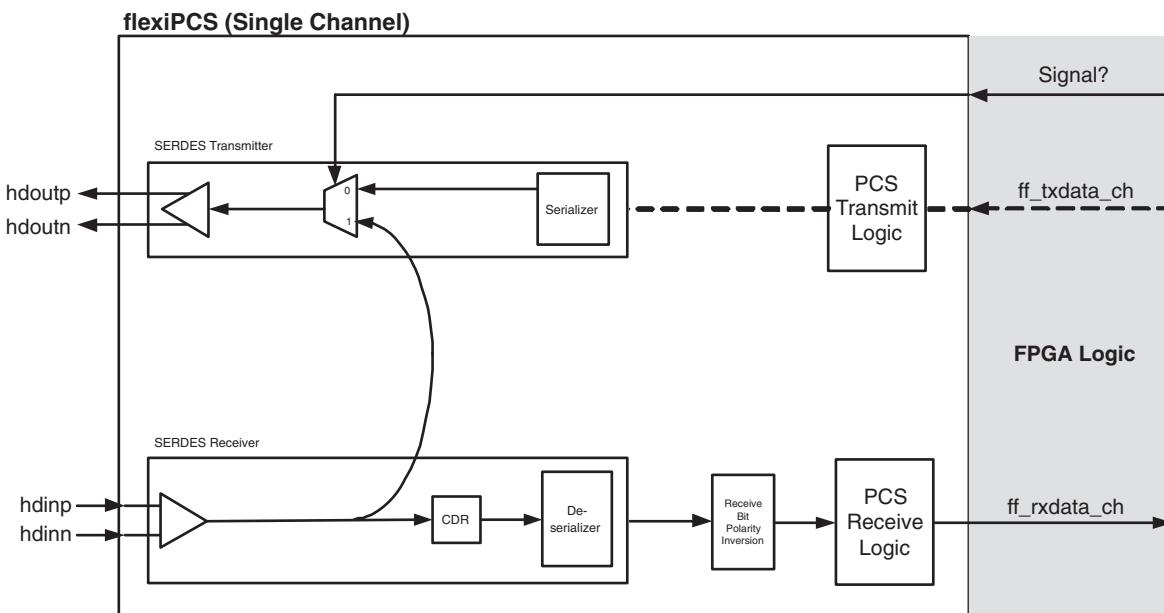
The LatticeECP2M family of devices provides loopback modes controlled by control signals at the PCS/FPGA interface for convenient testing of the external SERDES/board interface and the internal PCS/FPGA logic interface. Three far-end loopback modes are provided to loop received data back onto the transmit data path. The far-end loopback modes are useful for checking the high speed serial SERDES package pin connections as well as the embedded SERDES and/or PCS logic.

### Serial Far-end Loopback Mode

Loops serial receive clock/data back onto the transmit buffer without passing through the CDR or de-serializer. Serial far-end loopback can be selected by setting ?Signal Name? to a “1”.

Figure 8-32 illustrates the Serial far-end loopback mode for a single channel.

**Figure 8-32. Serial Far-End Loopback Mode (Single Channel)**

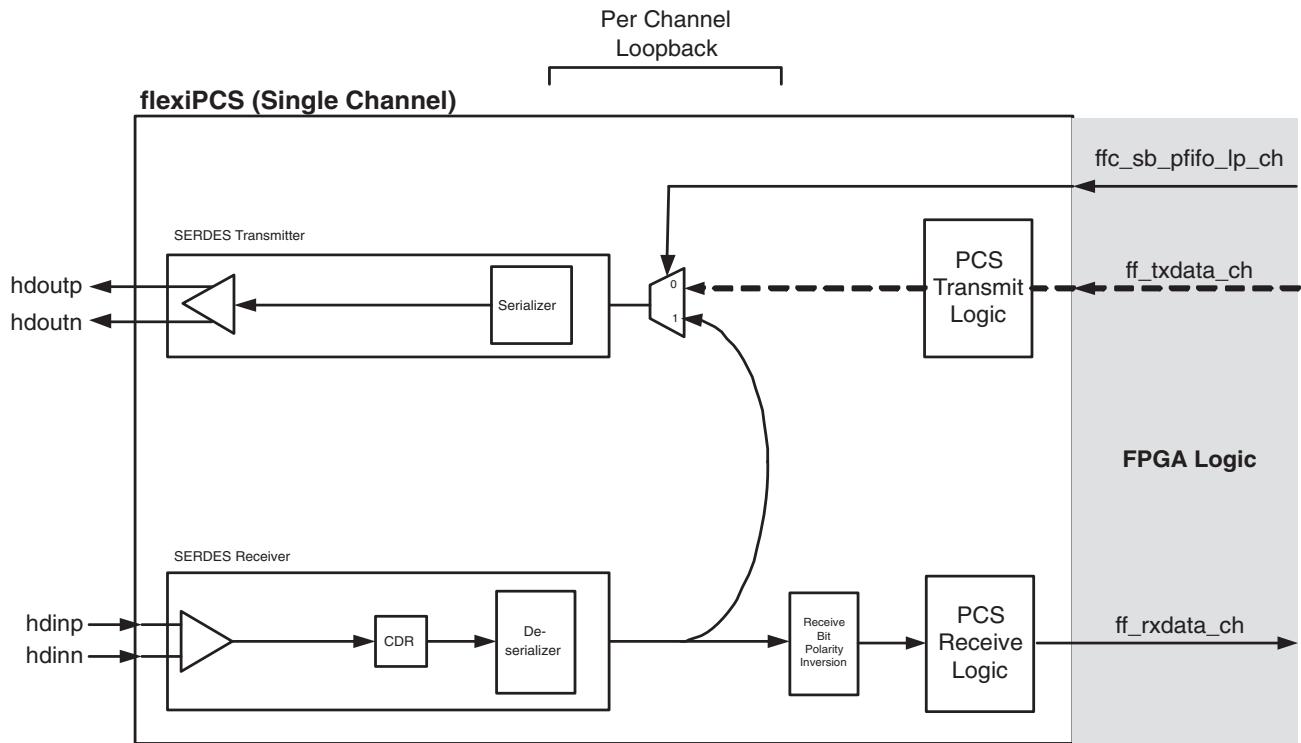


## PCS Parallel Far-end Loopback Mode

Loops parallel receive data back onto the transmit data path without passing through the PCS logic. SERDES parallel far-end loopback can be selected for each channel individually by setting the appropriate `ffc_sb_pfifo_lp_ch(0-3)` to a “1”.

Figure 4-23 illustrates the SERDES parallel far-end loopback mode for a single channel.

**Figure 8-33. SERDES Parallel Far-End Loopback Mode (Single Channel)**

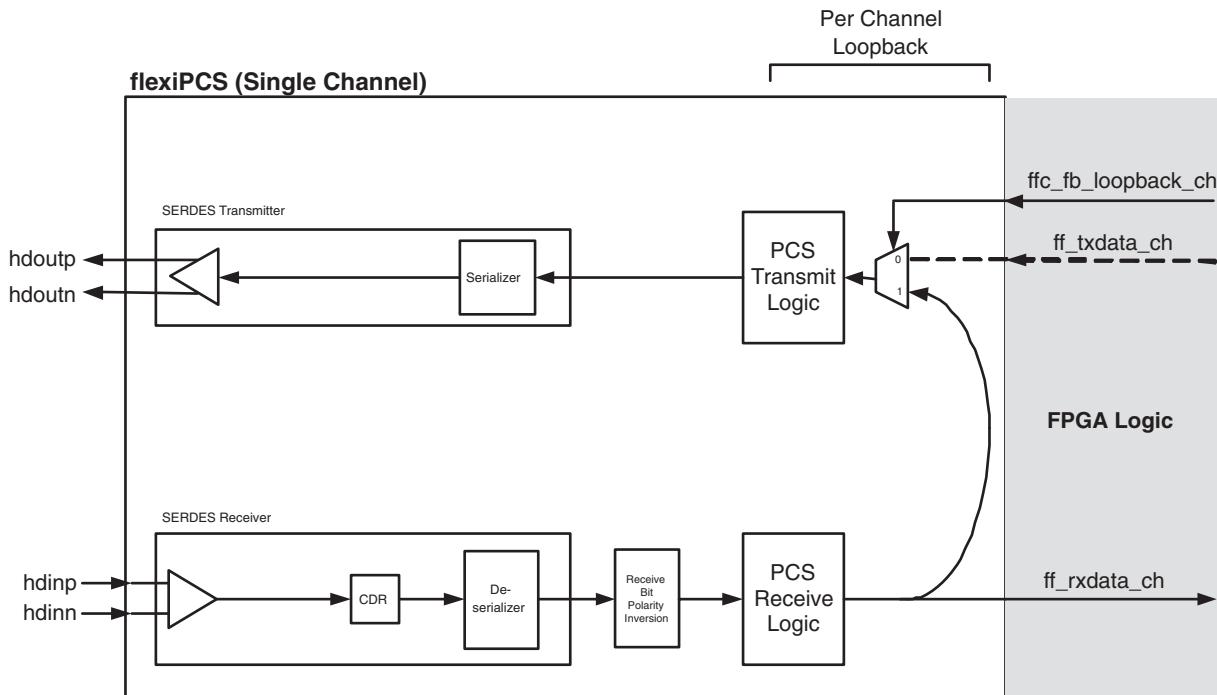


## PCS Parallel Far-end Loopback Mode

Loops parallel receive data back onto the transmit data path without passing across the PCS/FPGA interface. Input serial data at the SERDES hdin package pins passes through the SERDES receive logic where it is converted to parallel data, passes through the entire PCS receive logic path, is looped back through the entire PCS transmit logic path, is reconverted back to serial data by the SERDES transmitter and sent out onto the hdout SERDES package pins. PCS Parallel far-end loopback can be selected for each channel individually by setting the appropriate ffc\_fb\_loopback\_ch(0-3) port to a “1”.

Figure 8-34 illustrates the far-end loopback mode for a single channel.

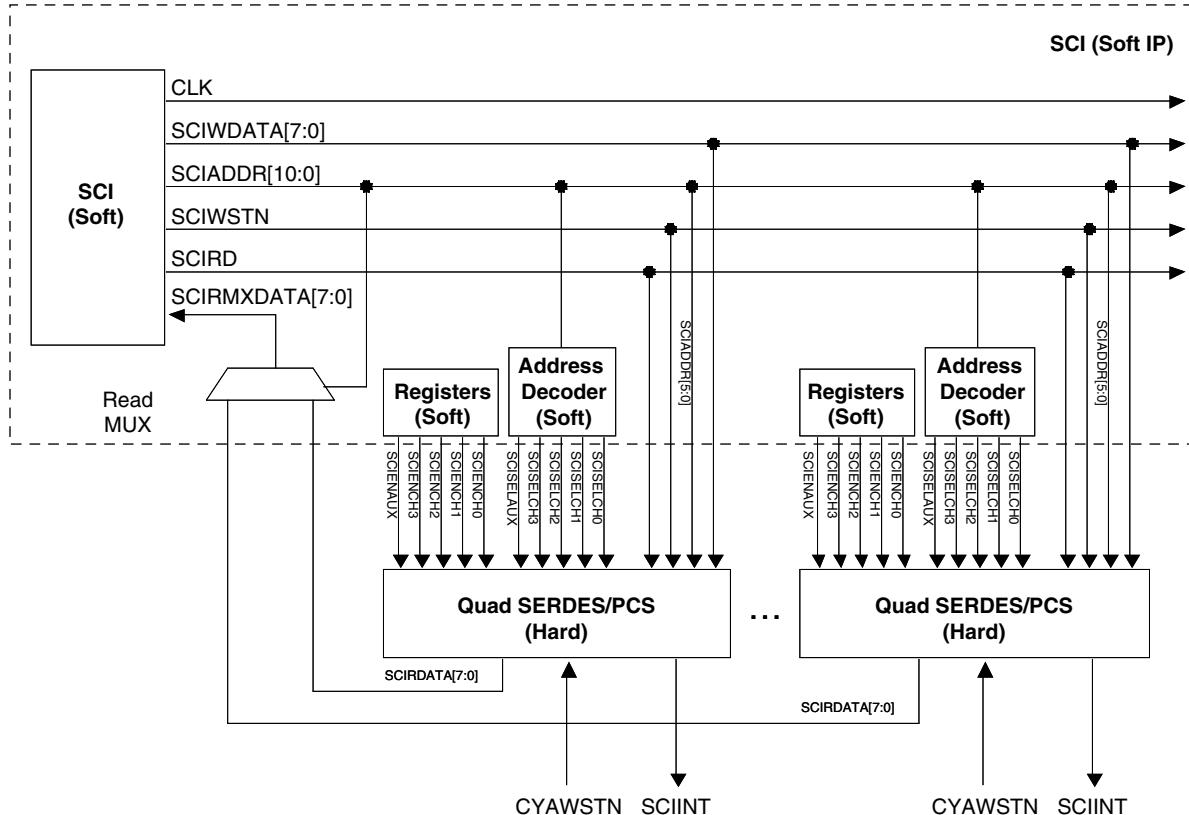
**Figure 8-34. Far-End Loopback Mode (Single Channel)**



## SERDES Client Interface (SCI)

The SERDES Client Interface (SCI) consists of both soft IP that resides in the FPGA core and permanent logic that is included in the SERDES/PCS quad. 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. It shows all the major signals required. The block diagram of the soft IP part of the SCI that resides in the FPGA core is shown in Figure 8-35.

**Figure 8-35. SCI Interface Block Diagram**

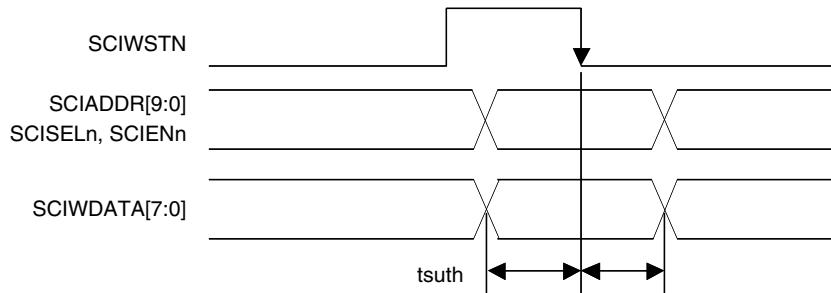


The SCIADDR bus is six bits wide within the block. The bus width at the block boundary is 11 bits. The upper five bits are used for quad block selection and channel selection. Table 8-18 shows the SCI address map for the SERDES quad.

**Table 8-18. SCI Address Map for Up to Four SERDES/PCS Quads**

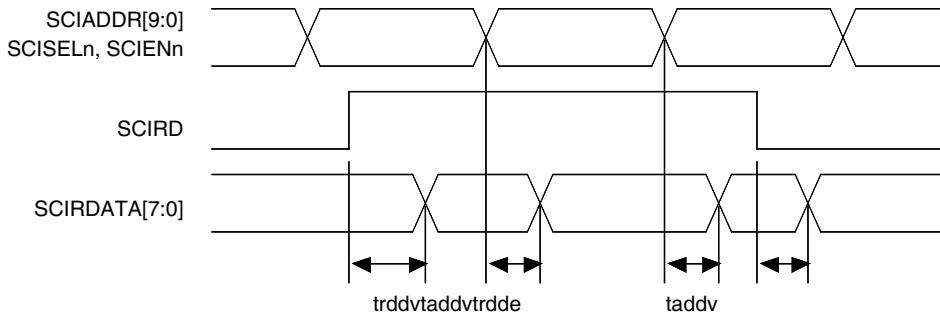
Address Bits	Description
SCIADDR[5:0]	Register address bits 000000 = select register 0 000001 = select register 1 ... 111110 = select register 62 111111 = select register 63
SCIADDR[8:6]	Channel address bits 000 = select channel 0 001 = select channel 1 010 = select channel 2 011 = select channel 3 100 = select aux channel 101 = unused 110 = unused 111 = unused
SCIADDR[10:9]	Quad address bits 00 = select quad 0 01 = select quad 1 10 = select quad 2 11 = select quad 3

Read and write operations through this interface are asynchronous. In the WRITE cycle the Write data and Write address need to be setup and held in relation to the falling edge of the SCIWSTN. In the READ cycle the timing has to be in relation with the SCIRD pulse. Figures 8-36 and 8-37 shows the WRITE and READ cycles respectively.

**Figure 8-36. SCI WRITE Cycle, Critical Timing**

## Notes :

- 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.

**Figure 8-37. SCI READ Cycle, Critical Timing****Notes:**

- 1) trddv is the time from assertion of the read pulse until read data is valid.
- 2) taddv is the time from address change until data is valid while read pulse is asserted.
- 3) trdde is the hold time of the read data after the deassertion of the read pulse.

The SCI interface is as simple as memory read/write. Here is an example of the pseudo code:

**Write**

```
Cycle 1: Set sciaddr[5:0], sciwdata[7:0], scien* = 1'b1 scisel* = 1'b1
Cycle 2 : Set sciwstn from 1 => 0
Cycle 3 : Set sciwstn from 0 => 1, scien* = 1'b0, scisel* = 1'b0
```

**Read**

```
Cycle 1 : Set sciaddr[5:0], scisel* = 1'b1
Cycle 2 : Set scird from 0 => 1
Cycle 3 : Obtain reading data from scirdata[7:0]
Cycle 4 : Set scierd from 1 => 0
```

Note: When loading the fuses, all five bits of scien\* must be set to zero.

**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 SCIINT 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 is cleared.

Once the aggregated source of interrupt is determined, the user can read the registers in the associated quad or channel to determine the actual source of the interrupt. Tables 8-19 and 8-20 list all the sources of interrupt.

**Table 8-19. Quad Interrupt Sources**

Quad SCI_INT Source	Description	Register Name
int_quad_out	Quad Interrupt. If there is an interrupt event anywhere in the quad this register bit will be active. This register bit gets cleared when all interrupt events have been cleared.	PCS Quad Status Register 1
int_cha_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 1
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 Status Register 3
~PLOL, PLOL	Interrupt generated on ~PLOL and PLOL - PLL Loss of Lock	SERDES Quad Status Register 2

**Table 8-20. Channel Interrupt Sources**

Quad SCI_INT Source	Description	Register Name
fb_tx_fifo_error_int fb_rx_fifo_error_int cc_overrun_int cc_underrun_int	FPGA Bridge Up Sample TX FIFO Error Interrupt FPGA Bridge Down Sample RX FIFO Error Interrupt CTC (Elastic Buffer) Overrun and Underrun Interrupts	PCS Channel General Interrupt Status Register 4
pci_det_done_int rlos_lo_int ~rlos_lo_int rlos_hi_int ~rlos_hi_int rlol_int ~rlol_int	Interrupt generated for pci_det_done Interrupt generated for rlos_lo Interrupt generated for ~rlos_lo Interrupt generated for rlos_hi Interrupt generated for ~rlos_hi Interrupt generated for rlol Interrupt generated for ~rlol	SERDES Channel Interrupt Status Register 5

## Dynamic Configuration of SERDES/PCS Quad

The SERDES/PCS quad can be controlled either by the configuration memory cells or by registers that are accessed through the optional “SERDES Client Interface” (SCI). The SCI consists of both a soft IP that resides in the FPGA core and permanent logic that is included in the SERDES/PCS quad. The SCI is only available if the soft IP is present in the FPGA core.

After configuration is complete, those configuration memory cells that have associated registers are automatically copied into the registers. Subsequent changes to the contents of the registers will not affect the value stored in the configuration memory cells but will of course change the operation of the SERDES/PCS quad.

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 must NOT assume that the soft IP is present in the FPGA core.

## SERDES Debug Capabilities

Lattice has tools to help in the debug of the SERDES/PCS operation in LatticeECP2M devices.

Lattice ORCAstra software is a PC-based graphical user interface for configuring the operational mode of a LatticeECP2/M device by programming control bits in the on-chip registers. This 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, SCI IP must be instantiated in the user logic.

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.

## Memory Map

### Configuration Register Definition

There are specific quad-level registers and channel-level registers. In each category, there are SERDES specific registers and PCS specific registers. Within these sub-categories there are:

- Control registers
- Status registers
- Status registers with clear-on-read
- Interrupt Control registers
- Interrupt Status registers
- Interrupt Source registers (clear-on-read)

All register bits shown below are with D0 as the LEAST significant bit on the left.

The following indications are the nomenclature for what types of register each is: (R/W = Read/Write, RO = Read Only, CR = Clear on a read).

Each of these register bits is “shadowed” with a memory cell. These memory cells are only programmable through bitstream control. After configuration is complete, the configuration memory cells that have associated registers are automatically copied into the registers. Subsequent changes to the contents of the registers will not affect the value stored in the configuration memory cells but will change the operation of the SERDES/PCS quad.

All “reserved” bits are written to zero.

## Per Quad Register Overview

**Table 8-21. Quad Interface Register Map**

BA <sup>1</sup>	Register Name	D0	D1	D2	D3	D4	D5	D6	D7
<b>1. PER QUAD CONTROL REGISTERS (28)</b>									
<b>Per Quad PCS Control Registers (17)</b>									
00	pcs_ctl_1_qd_00	uc_mode	fc_mode	pcie_mode	rio_mode	xge_mode	char_mode	force_int	
01	pcs_ctl_2_qd_01	bist_head_sel[0]	bist_head_sel[1]	bist_time_sel[0]	bist_time_sel[1]	bist_res_sel[0]	bist_res_sel[1]	bist_rpt_ch_sel[0]	bist_rpt_ch_sel[1]
02	pcs_ctl_3_qd_02	low_mark[0]	low_mark[1]	low_mark[2]	low_mark[3]	high_mark[0]	high_mark[1]	high_mark[2]	high_mark[3]
03	pcs_ctl_4_qd_03	sel_test_clk	asyn_mode	pfifo_clr_sel		match_2_enable	match_4_enable	min_ipg_cnt[0]	min_ipg_cnt[1]
04	pcs_ctl_5_qd_04	cc_match_1[0]	cc_match_1[1]	cc_match_1[2]	cc_match_1[3]	cc_match_1[4]	cc_match_1[5]	cc_match_1[6]	cc_match_1[7]
05	pcs_ctl_6_qd_05	cc_match_2[0]	cc_match_2[1]	cc_match_2[2]	cc_match_2[3]	cc_match_2[4]	cc_match_2[5]	cc_match_2[6]	cc_match_2[7]
06	pcs_ctl_7_qd_06	cc_match_3[0]	cc_match_3[1]	cc_match_3[2]	cc_match_3[3]	cc_match_3[4]	cc_match_3[5]	cc_match_3[6]	cc_match_3[7]
07	pcs_ctl_8_qd_07	cc_match_4[0]	cc_match_4[1]	cc_match_4[2]	cc_match_4[3]	cc_match_4[4]	cc_match_4[5]	cc_match_4[6]	cc_match_4[7]
08	pcs_ctl_9_qd_08	cc_match_1[8]	cc_match_1[9]	cc_match_2[8]	cc_match_2[9]	cc_match_3[8]	cc_match_3[9]	cc_match_4[8]	cc_match_4[9]
09	pcs_ctl_10_qd_09	udf_comma_mask[0]	udf_comma_mask[1]	udf_comma_mask[2]	udf_comma_mask[3]	udf_comma_mask[4]	udf_comma_mask[5]	udf_comma_mask[6]	udf_comma_mask[7]
0A	pcs_ctl_11_qd_0a	udf_comma_a[0]	udf_comma_a[1]	udf_comma_a[2]	udf_comma_a[3]	udf_comma_a[4]	udf_comma_a[5]	udf_comma_a[6]	udf_comma_a[7]
0B	pcs_ctl_12_qd_0b	udf_comma_b[0]	udf_comma_b[1]	udf_comma_b[2]	udf_comma_b[3]	udf_comma_b[4]	udf_comma_b[5]	udf_comma_b[6]	udf_comma_b[7]
0C	pcs_ctl_13_qd_0c	bist_en	bist_mode	udf_comma_mask[8]	udf_comma_mask[9]	udf_comma_b[8]	udf_comma_b[9]	udf_comma_a[8]	udf_comma_a[9]
0D	pcs_ctl_14_qd_0d	bist_udf_def_head[er[0]]	bist_udf_def_head[er[1]]	bist_udf_def_head[er[2]]	bist_udf_def_head[er[3]]	bist_udf_def_head[er[4]]	bist_udf_def_head[er[5]]	bist_udf_def_head[er[6]]	bist_udf_def_head[er[7]]
0E	pcs_ctl_15_qd_0e	bist_udf_def_head[er[8]]	bist_udf_def_head[er[9]]	bist_udf_def_head[er[10]]	bist_udf_def_head[er[11]]	bist_udf_def_head[er[12]]	bist_udf_def_head[er[13]]	bist_udf_def_head[er[14]]	bist_udf_def_head[er[15]]
0F	pcs_ctl_16_qd_0f	bist_udf_def_head[er[16]]	bist_udf_def_head[er[17]]	bist_udf_def_head[er[18]]	bist_udf_def_head[er[19]]	bist_ptn_sel[0]	bist_ptn_sel[1]	bist_ptn_sel[2]	bist_bus8bit_sel
10	pcs_ctl_17_qd_int_10 (Note 2)	ls_sync_statusn_0_int_ctl	ls_sync_statusn_1_int_ctl	ls_sync_statusn_2_int_ctl	ls_sync_statusn_3_int_ctl	ls_sync_status_0_int_ctl	ls_sync_status_1_int_ctl	ls_sync_status_2_int_ctl	ls_sync_status_3_int_ctl
<b>Per Quad SERDES Control Registers (6)</b>									
11	ser_ctl_1_qd_11	refclk_out_sel[0]	refclk_out_sel[1]	refclk_out_sel[2]	refclk_rterm	refclk_dcc_en	tx_refclk_sel	reserved	reserved
12	ser_ctl_2_qd_12	rlos_lset[0]	rlos_lset[1]	rlos_lset[2]	rlos_hset[0]	rlos_hset[1]	rlos_hset[2]	bus8bit_sel	refck25x
13	ser_ctl_3_qd_13	cdr_lo_lset[0]	cdr_lo_lset[1]	reserved	reserved	reserved	reserved	refck_mode[0]	refck_mode[1]
14	ser_ctl_4_qd_14	tx_vco_ck_div[0]	tx_vco_ck_div[1]	tx_vco_ck_div[2]	pll_lo_lset[0]	pll_lo_lset[1]	reserved	reserved	reserved
15	ser_ctl_5_qd_15	reserved							
16	ser_ctl_6_qd_int_16 (Note 2)	reserved	reserved	reserved	reserved	reserved	reserved	~plol_int_ctl	plol_int_ctl
<b>Per Quad Clock Reset Registers (5)</b>									
17	rst_ctl_1_qd_17	lane_tx_rst0	lane_tx_rst1	lane_tx_rst2	lane_tx_rst3	lane_rx_rst0	lane_rx_rst1	lane_rx_rst2	lane_rx_rst3
18	rst_ctl_2_qd_18	rrst[0]	rrst[1]	rrst[2]	rrst[3]	trst	quad_rst	macro_rst	macropdb
19	rst_ctl_3_qd_19	sel_sd_rx_clk0	sel_sd_rx_clk1	sel_sd_rx_clk2	sel_sd_rx_clk3	bist_sync_head_r_eq[0]	bist_sync_head_r_eq[1]	bist_bypass_tx_ga_te	bist_rx_data_sel
1a	rst_ctl_4_qd_1a	ff_rx_clk_sel[1 0]	ff_rx_clk_sel[1 1]	ff_rx_clk_sel[1 2]	ff_rx_clk_sel[1 3]	ff_rx_clk_sel[2 0]	ff_rx_clk_sel[2 1]	ff_rx_clk_sel[2 2]	ff_rx_clk_sel[2 3]
1b	rst_ctl_5_qd_1b	ff_rx_clk_sel[0 0]	ff_rx_clk_sel[0 1]	ff_rx_clk_sel[0 2]	ff_rx_clk_sel[0 3]	reserved	ff_tx_clk_sel[0]	ff_tx_clk_sel[1]	ff_tx_clk_sel[2]
<b>2. PER QUAD STATUS REGISTERS (9)</b>									
<b>Per Quad PCS Status Registers (5)</b>									
20	pcs_sts_1_qd_20	int_cha[0]	int_cha[1]	int_cha[2]	int_cha[3]	int_qua_out			
21	pcs_sts_2_qd_21 (Note 3)	ls_sync_statusn_0	ls_sync_statusn_1	ls_sync_statusn_2	ls_sync_statusn_3	ls_sync_status_0	ls_sync_status_1	ls_sync_status_2	ls_sync_status_3
22	pcs_sts_3_qd_int_22 (Note 4)	ls_sync_statusn_0_int	ls_sync_statusn_1_int	ls_sync_statusn_2_int	ls_sync_statusn_3_int	ls_sync_status_0_int	ls_sync_status_1_int	ls_sync_status_2_int	ls_sync_status_3_int
23	pcs_sts_4_qd_23	bist_report[0]	bist_report[1]	bist_report[2]	bist_report[3]	bist_report[4]	bist_report[5]	bist_report[6]	bist_report[7]
24	pcs_sts_5_qd_24	bist_report[8]	bist_report[9]	bist_report[10]	bist_report[11]	bist_report[12]	bist_report[13]	bist_report[14]	bist_report[15]
<b>Per Quad SERDES Status Registers (4)</b>									
25	ser_sts_1_qd_25 (Note 3)	reserved	reserved	reserved	reserved	reserved	reserved	~plol	plol
26	ser_sts_2_qd_int_26 (Note 4)	reserved	reserved	reserved	reserved	reserved	reserved	~plol_int	plol_int
27	ser_sts_3_qd_27	pll_calib_status[0]	pll_calib_status[1]	pll_calib_status[2]	pll_calib_status[3]	pll_calib_status[4]	pll_calib_status[5]	reserved	reserved
28	ser_sts_4_qd_28	reserved							

1. BA = Base Address (Hex)

2. Interrupt control register related to an interruptible status (int\_sts\_x) register.

3. Status register which has an associated interruptible status (int\_sts\_x) register.

4. Interruptible status register; clear on read (has associated control register and status register)

Note: Default value is "0", unless otherwise specified.

## Per Quad PCS Control Register Details

**Table 8-22. PCS Control Register 1 (pcs\_ctl\_1\_qd\_00)**

Bit	Name	Description	Type	Default
0	uc_mode	1 = Selects User Configured mode 0 = Selects other mode (PCI Express, RapidIO, 10GbE, 1GbE)		
1	fc_mode	0 = Selects other mode (PCI Express, RapidIO, 10GbE, 1GbE)	R/W	0
2	pcie_mode	1 = PCI Express mode of operation 0 = Selects other mode (RapidIO, 10GbE, 1GbE)	R/W	0
3	rio_mode	1 = Selects Rapid-IO mode 0 = Selects other mode (10GbE, 1GbE)	R/W	0
4	xge_mode	1 = Selects 10Gb Ethernet 0 = Selects 1Gb Ethernet mode	R/W	0
5	char_mode	1 = Enable SERDES characterization mode 0 = Disable SERDES characterization mode	R/W	0
6	force_int	1 = Force to generate interrupt signal 0 = Normal operation	R/W	0
7	Reserved			

Note: Bits 0 to 3 are mutually exclusive. Only one of the bits can be set.

**Table 8-23. PCS Control Register 2 (pcs\_ctl\_2\_qd\_01)**

Bit	Name	Description	Type	Default
1:0	bist_head_sel [1:0]	BIST header selection 00 = K28_5 (K28_5=10'h305 K28_5_=10'h0FA) 01 = A1A2 (MA1=10'h1F6; MA2=10'h128) 10 = 10'h1BC 11 = user defined	R/W	0
3:2	bist_time_sel [1:0]	BIST time selection: 00 = 5e+8 cycles 01 = 5e+9 cycles 10 = 5e+6 cycles 11 = 100K cycles	R/W	00
5:4	bist_res_sel [1:0]	BIST resolution selection 00 = no error 01 < 2 errors 10 < 16 errors 11 < 128 errors	R/W	00
7:6	bist_rpt_ch_sel [1:0]	00 = BIST report from channel 0 01 = BIST report from channel 1 10 = BIST report from channel 2 11 = BIST report from channel 3	R/W	00

**Table 8-24. PCS Control Register 3 (pcs\_ctl\_3\_qd\_02)**

Bit	Name	Description	Type	Default
3:0	low_mark [3:0]	Clock compensation FIFO low water mark. Mean is 4'b1000	R/W	4'b1001
7:4	high_mark [3:0]	Clock compensation FIFO high water mark. Mean is 4'b1000	R/W	4'b0111

**Table 8-25. PCS Control Register 4 (pcs\_ctl\_4\_qd\_03)**

Bit	Name	Description	Type	Default
0	sel_test_clk	For test only, select test clock	R/W	0
1	asyn_mode	For test only, select asynchronous reset	R/W	0
2	pfifo_clr_sel	1 = pfifo_clr signal or channel register bit clears the FIFO 0 = pfifo_error internal signal self clears the FIFO	R/W	0
3	Reserved			
4	match_2_enable	1 = enable two character skip matching (using match 4,3)	R/W	1
5	match_4_enable	1 = enable four character skip matching (using match 4, 3, 2, 1)	R/W	0
7:6	min_ipg_cnt [1:0]	Minimum IPG to enforce	R/W	2'b11

**Table 8-26. PCS Control Register 5 - CC match 1 LO (pcs\_ctl\_5\_qd\_04)**

Bit	Name	Description	Type	Default
7:0	cc_match_1 [7:0]	Lower bits of user defined clock compensator skip pattern 1	R/W	8'h00

**Table 8-27. PCS Control Register 6 - CC match 2 LO (pcs\_ctl\_6\_qd\_05)**

Bit	Name	Description	Type	Default
7:0	cc_match_2 [7:0]	Lower bits of user defined clock compensator skip pattern 2	R/W	8'h00

**Table 8-28. PCS Control Register 7 - CC match 3 LO (pcs\_ctl\_7\_qd\_06)**

Bit	Name	Description	Type	Default
7:0	cc_match_3 [7:0]	Lower bits of user defined clock compensator skip pattern 3	R/W	8'hBC

**Table 8-29. PCS Control Register 8 - CC match 4 LO (pcs\_ctl\_8\_qd\_07)**

Bit	Name	Description	Type	Default
7:0	cc_match_4 [7:0]	Lower bits of user defined clock compensator skip pattern 4	R/W	8'h50

**Table 8-30. PCS Control Register 9 - CC match HI (pcs\_ctl\_9\_qd\_08)**

Bit	Name	Description	Type	Default
1:0	cc_match_1 [9:8]	Upper bits of user defined clock compensator skip pattern 1 [9] = Disparity error [8] = K control	R/W	2'b00
2:3	cc_match_2 [9:8]	Upper bits of user defined clock compensator skip pattern 2 [9] = Disparity error [8] = K control	R/W	2'b00
4:5	cc_match_3 [9:8]	Upper bits of user defined clock compensator skip pattern 3 [9] = Disparity error [8] = K control	R/W	2'b01
7:6	cc_match_4 [9:8]	Upper bits of user defined clock compensator skip pattern 4 [9] = Disparity error [8] = K control	R/W	2'b01

**Table 8-31. PCS Control Register 10 - UDF comma mask LO (pcs\_ctl\_10\_qd\_09)**

Bit	Name	Description	Type	Default
7:0	udf_comma_mask [7:0]	Lower bits of user defined comma mask	R/W	8'hFF

**Table 8-32. PCS Control Register 11 - UDF comma a LO (pcs\_ctl\_11\_qd\_0a)**

Bit	Name	Description	Type	Default
7:0	udf_comma_a [7:0]	Lower bits of user defined comma character 'a'	R/W	8'h83

**Table 8-33. PCS Control Register 12 - UDF comma b LO (pcs\_ctl\_12\_qd\_0b)**

Bit	Name	Description	Type	Default
7:0	udf_comma_b [7:0]	Lower bits of user defined comma character 'b'	R/W	8'h7C

**Table 8-34. PCS Control Register 13 - UDF comma HI (pcs\_ctl\_13\_qd\_0c)**

Bit	Name	Description	Type	Default
0	bist_en	1 = Enable PCS BIST 0 = Normal operation.	R/W	0
1	bist_mode	1 = Continuous Bist Mode 0 = Timed BIST mode	R/W	0
3:2	udf_comma_mask [9:8]	Upper bits of user defined comma mask	R/W	2'b11
5:4	udf_comma_b [9:8]	Upper bits of user defined comma character 'b'	R/W	2'b01
7:6	udf_comma_a [9:8]	Upper bits of user defined comma character 'a'	R/W	2'b10

**Table 8-35. PCS Control Register 14 - UDF BIST header LO (pcs\_ctl\_14\_qd\_0d)**

Bit	Name	Description	Type	Default
7:0	bist_udf_def_header [7:0]	Lower bits of user defined header for BIST	R/W	8'h00

**Table 8-36. PCS Control Register 15 - UDF BIST header MD (pcs\_ctl\_15\_qd\_0e)**

Bit	Name	Description	Type	Default
7:0	bist_udf_def_header [15:8]	Middle bits of user defined header for BIST	R/W	8'h00

**Table 8-37. PCS Control Register 16 - UDF BIST header HI (pcs\_ctl\_16\_qd\_0f)**

Bit	Name	Description	Type	Default
3:0	bist_udf_def_header [19:16]	High bits of user defined header for BIST	R/W	8'h00
6:4	bist_ptn_sel[2:0]	BIST pattern selection: 000 = PRBS11 001 = max data rate 010 = PRBS31 011 = PRBS21100 = K28_5 101 = A1A2110 = 5150 (repeat) 111 = 2120 (repeat)	R/W	0
7	bist_bus8bit_sel	1 = 8 bit data BIST 0 = 10 bit data BIST	R/W	0

**Table 8-38. PCS Interrupt Control Register 17 (pcs\_ctl\_17\_qd\_int\_10)**

Bit	Name	Description	Type	Default
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)	R/W	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)	R/W	0
3	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)	R/W	0
4	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)	R/W	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)	R/W	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)	R/W	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)	R/W	0
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)	R/W	0

## Per Quad SERDES Control Register Details

Note: Except indicated, all channels must be reset after writing any SERDES control register.

**Table 8-39. SERDES Control Register 1 (ser\_ctl\_1\_qd\_11)**

Bit	Name	Description	Type	Default
2:0	REFCK_OUT_SEL	[2:0]	Refck outputs enable/disable control [0]: 0 = rx_refck_local output enable 1 = rx_refck_local output disable [1]: 0 = refck2core output disable 1 = refck2core output enable [2]: reserved	R/W
3'b000	3	REFCK_RTERM	Termination at reference clock input buffer 0 = 50 ohm (default) 1 = high impedance	R/W
0	4	REFCK_DCC_EN	1 = Reference clock DC coupling enable 0 = Reference clock DC coupling disable (default)	R/W
0	5	TX_REFCK_SEL	TxPLL reference clock select 0 = rx_refck_local1 = ck_core_tx	R/W
0	7:6	Reserved		

**Table 8-40. SERDES Control Register 2 (ser\_ctl\_2\_qd\_12)**

Bit	Name	Description	Type	Default
2:0	RLOS_LSET [2:0]	LOS detector reference current adjustment for smaller swing 000 = default 001 = +10% 010 = +20% 011 = +30% 100 = -10% 101 = -20% 110 = -30% 111 = -40%	R/W	3'b000
5:3	RLOS_HSET [2:0]	LOS detector reference current adjustment for larger swing 000 = default 001 = +10% 010 = +20% 011 = +30% 100 = -10% 101 = -20% 110 = -30% 111 = -40%	R/W	3'b000
6	BUS8BIT_SEL	1 = Select 8-bit bus width 0 = Select 10-bit bus width (default)	R/W	0
7	REFCK25X	1 = Internal high speed bit clock is 25x (for reference clock = 100MHz only) 0 = see REFCK_MODE	R/W	0

**Table 8-41. SERDES Control Register 3 - (ser\_ctl\_3\_qd\_13)**

Bit	Name	Description	R/W	Default
1:0	CDR_LOL_SET [1:0]	CDR loss of lock setting: Lock 00 = +/-1000ppm x2 01 = +/-2000ppm x2 10 = +/-4000ppm 11 = +/-300ppm Unlock +/-1500ppm x2 +/-2500ppm x2 +/-7000ppm +/-450ppm	R/W	2'b00
5:2	Reserved			
7:6	REFCK_MODE [1:0]	00 = Internal high speed bit clock is 20x or 16x 01 = Internal high speed bit clock is 10x or 8x 1X = Reserved	R/W	2'b00

**Table 8-42. SERDES Control Register 4 (ser\_ctl\_4\_qd\_14)**

Bit	Name	Description	R/W	Default
2:0	TX_VCO_CK_DIV[2:0]	VCO output frequency select: 000 = divided by 1      001 = divided by 2 010 = divided by 4      011 = divided by 8 100 = reserved      101 = reserved 110 = divided by 16      111 = divided by 32	R/W	3'b000
4:3	PLL_LOL_SET [1:0]	TxPLL loss of lock setting Lock 00 = +/-300ppm x2 01 = +/-300ppm 10 = +/-1500ppm 11 = +/-4000ppm Unlock +/-600ppm x2 +/-2000ppm +/-2200ppm +/-6000ppm	R/W	2'b00
7:5	Reserved			

**Table 8-43. SERDES Control Register 5 (ser\_ctl\_5\_qd\_15)**

Bit	Name	Description	R/W	Default
7:0	Reserved			

**Table 8-44. SERDES Interrupt Control Register 6 (ser\_ctl\_6\_qd\_int\_16)**

Bit	Name	Description	R/W	Default
5:0	Reserved			
6	~PLOL_INT_CTL	1 = Interrupt enabled for obtaining lock on PLOL. 0 = Interrupt not enabled for obtaining lock PLOL.	RO CR	0
7	PLOL_INT_CTL	1 = Interrupt enabled for loss of lock on PLOL. 0 = Interrupt not enabled for loss of lock PLOL.	RO CR	0

## Per Quad Reset and Clock Control Register Details

**Table 8-45. Reset and Clock Control Register 1 (rst\_ctl\_1\_qd\_17)**

Bit	Name	Description	R/W	Default
0	lane_tx_rst0	1 = Assert reset signal to channel 0 transmit logic	R/W	0
1	lane_tx_rst1	1 = Assert reset signal to channel 1 transmit logic	R/W	0
2	lane_tx_rst2	1 = Assert reset signal to channel 2 transmit logic	R/W	0
3	lane_tx_rst3	1 = Assert reset signal to channel 3 transmit logic	R/W	0
4	lane_rx_rst0	1 = Assert reset signal to channel 0 receive logic	R/W	0
5	lane_rx_rst1	1 = Assert reset signal to channel 1 receive logic	R/W	0
6	lane_rx_rst2	1 = Assert reset signal to channel 2 receive logic	R/W	0
7	lane_rx_rst3	1 = Assert reset signal to channel 3 receive logic	R/W	0

**Table 8-46. Reset and Clock Control Register 2 (rst\_ctl\_2\_qd\_18)**

Bit	Name	Description	R/W	Default
3:0	rrst [3:0]	1 = Rx channel-based reset	R/W	0
4	trst	1 = Tx Reset	R/W	0
5	quad_rst	1 = Assert quad reset	R/W	0
6	macro_rst	1 = Assert macro reset	R/W	0
7	macropdb	0 = Assert power down	R/W	1

**Table 8-47. Reset and Clock Control Register 3 (rst\_ctl\_3\_qd\_19)**

Bit	Name	Description	R/W	Default
0	sel_sd_rx_clk0	1 = Select sd_rx_clk for channel 0	R/W	0
1	sel_sd_rx_clk1	1 = Select sd_rx_clk for channel 1	R/W	0
2	sel_sd_rx_clk2	1 = Select sd_rx_clk for channel 2	R/W	0
3	sel_sd_rx_clk3	1 = Select sd_rx_clk for channel 3	R/W	0
5:4	bist_sync_head_req [1:0]	BIST sync header counter selection 00 = 5'd5 01 = 5'd8 10 = 5'd14 11 = 5'd24	R/W	00
6	bist_bypass_tx_gate	0 = start to send BIST data after finding the head 1 = force to send BIST data whether the head is found or not	R/W	0
7	bist_rx_data_sel	0 = data from SERDES 1 = data from after 8b10b decoder	R/W	0

**Table 8-48. Reset and Clock Control Register 4 (rst\_ctl\_4\_qd\_1a)**

Bit	Name	Description	R/W	Default
0	ff_rx_clk_sel0_1	1 = Enable ff_rx_h_clk for channel 0	R/W	0
1	ff_rx_clk_sel1_1	1 = Enable ff_rx_h_clk for channel 1	R/W	0
2	ff_rx_clk_sel2_1	1 = Enable ff_rx_h_clk for channel 2	R/W	0
3	ff_rx_clk_sel3_1	1 = Enable ff_rx_h_clk for channel 3	R/W	0
4	ff_rx_clk_sel0_2	1 = Disable ff_rx_f_clk for channel 0	R/W	0
5	ff_rx_clk_sel1_2	1 = Disable ff_rx_f_clk for channel 1	R/W	0
6	ff_rx_clk_sel2_2	1 = Disable ff_rx_f_clk for channel 2	R/W	0
7	ff_rx_clk_sel3_2	1 = Disable ff_rx_f_clk for channel 3	R/W	0

**Table 8-49. Reset and Clock Control Register 5 (rst\_ctl\_5\_qd\_1b)**

Bit	Name	Description	R/W	Default
0	ff_rx_clk_sel0_0	1 = Enable ff_rx_q_clk for channel 0	R/W	0
1	ff_rx_clk_sel1_0	1 = Enable ff_rx_q_clk for channel 1	R/W	0
2	ff_rx_clk_sel2_0	1 = Enable ff_rx_q_clk for channel 2	R/W	0
3	ff_rx_clk_sel3_0	1 = Enable ff_rx_q_clk for channel 3	R/W	0
4	Reserved			
5	ff_tx_clk_sel0	1 = Enable ff_tx_q_clk for quad	R/W	0
6	ff_tx_clk_sel1	1 = Enable ff_tx_h_clk for quad	R/W	0
7	ff_tx_clk_sel2	1 = Disable ff_tx_f_clk for quad	R/W	0

## Per Quad PCS Status Register Details

**Table 8-50. PCS Status Register 1 (pcs\_sts\_1\_qd\_20)**

Bit	Name	Description	R/W	Int?
3:0	int_cha_out [3:0]	Per Channel Interrupt status	RO	N
4	int_qua_out	Per Quad Interrupt status	RO	N
5	ion_delay	Delayed global resetn from tri_ion	RO	N
7:6	Reserved			

**Table 8-51. PCS Status Register 2 (pcs\_sts\_2\_qd\_21)**

Bit	Name	Description	R/W	Int?
0	ls_sync_statusn_0	1 = Alarm generated on sync_statusn_0 when it goes low (out of sync) 0 = Alarm not generated on sync_statusn_0 when it goes low (out of sync)	RO	Y
1	ls_sync_statusn_1	1 = Alarm generated on sync_statusn_1 when it goes low (out of sync) 0 = Alarm not generated on sync_statusn_1 when it goes low (out of sync)	RO	Y
2	ls_sync_statusn_2	1 = Alarm generated on sync_statusn_2 when it goes low (out of sync) 0 = Alarm not generated on sync_statusn_2 when it goes low (out of sync)	RO	Y
3	ls_sync_statusn_3	1 = Alarm generated on sync_statusn_3 when it goes low (out of sync) 0 = Alarm not generated on sync_statusn_3 when it goes low (out of sync)	RO	Y
4	ls_sync_status_0	1 = Alarm generated on sync_status_0. 0 = Alarm not generated on sync_status_0.	RO	Y
5	ls_sync_status_1	1 = Alarm generated on sync_status_1. 0 = Alarm not generated on sync_status_1.	RO	Y
6	ls_sync_status_2	1 = Alarm generated on sync_status_2. 0 = Alarm not generated on sync_status_2.	RO	Y
7	ls_sync_status_3	1 = Alarm generated on sync_status_3. 0 = Alarm not generated on sync_status_3.	RO	Y

**Table 8-52. Packet Interrupt Status Register 3 (pcs\_sts\_3\_qd\_int\_22)**

Bit	Name	Description	R/W	Int?
0	ls_sync_statusn_0_int	1 = Interrupt generated on sync_statusn_0 when it goes low (out of sync) 0 = Interrupt not generated on sync_statusn_0 when it goes low (out of sync)	RO CR	Y
1	ls_sync_statusn_1_int	1 = Interrupt generated on sync_statusn_1 when it goes low (out of sync) 0 = Interrupt not generated on sync_statusn_1 when it goes low (out of sync)	RO CR	Y
2	ls_sync_statusn_2_int	1 = Interrupt generated on sync_statusn_2 when it goes low (out of sync) 0 = Interrupt not generated on sync_statusn_2 when it goes low (out of sync)	RO CR	Y
3	ls_sync_statusn_3_int	1 = Interrupt generated on sync_statusn_3 when it goes low (out of sync) 0 = Interrupt not generated on sync_statusn_3 when it goes low (out of sync)	RO CR	Y
4	ls_sync_status_0_int	1 = Interrupt generated on sync_status_0 (in sync) 0 = Interrupt not generated on sync_status_0 (in sync)	RO CR	Y
5	ls_sync_status_1_int	1 = Interrupt generated on sync_status_1 (in sync) 0 = Interrupt not generated on sync_status_1 (in sync)	RO CR	Y
6	ls_sync_status_2_int	1 = Interrupt generated on sync_status_2 (in sync) 0 = Interrupt not generated on sync_status_2 (in sync)	RO CR	Y
7	ls_sync_status_3_int	1 = Interrupt generated on sync_status_3 (in sync) 0 = Interrupt not generated on sync_status_3 (in sync)	RO CR	Y

**Table 8-53. PCS BIST Status Register 4 (pcs\_sts\_4\_qd\_23)**

Bit	Name	Description	R/W	Int?
7:0	bist_report [7:0]	Lower bits of BIST report	RO	N

**Table 8-54. PCS BIST Status Register 5 (pcs\_sts\_5\_qd\_24)**

Bit	Name	Description	R/W	Int?
7:0	bist_report [15:8]	Higher bits of BIST report	RO	N

**Per Quad SERDES Status Register Details****Table 8-55. SERDES Status Register 1 (ser\_sts\_1\_qd\_25)**

Bit	Name	Description	R/W	Int?
5:0	Reserved			
6	~PLOL	1 = PLL lock obtained	RO	Y
7	PLOL	1 = PLL Loss of lock	RO	Y

**Table 8-56. SERDES Interrupt Status Register 2 (ser\_sts\_2\_qd\_int\_26)**

Bit	Name	Description	R/W	Int?
5:0	Reserved			
6	~PLOL_INT	1 = Interrupt generated on ~PLOL. 0 = Interrupt not generated ~PLOL.	RO CR	Y
7	PLOL_INT	1 = Interrupt generated on PLOL. 0 = Interrupt not generated PLOL.	RO CR	Y

**Table 8-57. SERDES Status Register 3 (ser\_sts\_3\_qd\_27)**

Bit	Name	Description	R/W	Int?
5:0	PLL_CALIB_STATUS	[5:0]	TxPLL VCO calibration status output	RO
N	7:6	Reserved		

**Table 8-58. SERDES Status Register 4 (ser\_sts\_4\_qd\_28)**

Bit	Name	Description	R/W	Int?
7:0	Reserved			

## Per Channel Register Overview

**Table 8-59. Channel Interface Register Map**

BA <sup>1</sup>	Register name	D0	D1	D2	D3	D4	D5	D6	D7
<b>1. CONTROL REGISTERS (13)</b>									
<b>Per Channel General Registers (7)</b>									
00	pcs_ctl_1_ch_00	invert_rx	invert_tx			ge_an_enable	prbs_lock	prbs_enable	enable_cg_align
01	pcs_ctl_2_ch_01	tx_ch	rx_ch	tx_gear_mode	rx_gear_mode	pcs_det_time_sel[0]	pcs_det_time_sel[1]	pcie_ei_en	pfifo_clr
02	pcs_ctl_3_ch_02	fb_loopback	tx_gear_bypass	sel_bist_txd4enc	enc_bypass	sb_bist_sel	sb_pfifo_lp	sb_bypass	bus_width8
03	pcs_ctl_4_ch_03	sb_loopback	rx_sb_bypass	wa_bypass	dec_bypass	ctc_bypass	rx_gear_bypass	signal_detect	lsm_disable
04	pcs_ctl_5_ch_int_04 (Note 2)	fb_tx_fifo_error_int_ctl	fb_rx_fifo_error_int_ctl	cc_overrun_int_ctl	cc_underrun_int_ctl				
05	pcs_ctl_6_ch_05								
06	pcs_ctl_7_ch_06								
<b>Per Channel SERDES Registers (6)</b>									
07	ser_ctl_1_ch_07	rx_dco_ck_div[0]	rx_dco_ck_div[1]	rx_dco_ck_div[2]	rate_sel[0]	rate_sel[1]	rcv_dcc_en	req_lvl_set	req_en
08	ser_ctl_2_ch_08	rterm_rx[0]	rterm_rx[1]	rterm_rxadj[0]	rterm_rxadj[1]	lb_ctl[0]	lb_ctl[1]	lb_ctl[2]	lb_ctl[3]
09	ser_ctl_3_ch_09	tdrv_dat_sel[0]	tdrv_dat_sel[1]	tdrv_pre_set[0]	tdrv_pre_set[1]	tdrv_pre_set[2]	tdrv_amp[0]	tdrv_amp[1]	tdrv_amp[2]
0a	ser_ctl_4_ch_0a	tpwdnb	rate_mode_tx	rterm_tx[0]	rterm_tx[1]	tdrv_pre_en			
0b	ser_ctl_5_ch_0b	rpwdnb	rate_mode_rx	rx_refck_sel[0]	rx_refck_sel[1]	oob_en			
0c	ser_ctl_6_ch_int_0c (Note 2)		pci_det_done_int_ctl	rlos_lo_int_ctl	~rlos_lo_int_ctl	rlos_hi_int_ctl	~rlos_hi_int_ctl	rlol_int_ctl	~rlol_int_ctl
<b>2. STATUS REGISTERS (13)</b>									
<b>Per Channel General Registers (6)</b>									
20	pcs_sts_1_ch_20 (Note 3)	fb_tx_fifo_error	fb_rx_fifo_error	cc_overrun	cc_underrun				
21	pcs_sts_2_ch_21 (Note 5)	prbs_error_cnt[0]	prbs_error_cnt[1]	prbs_error_cnt[2]	prbs_error_cnt[3]	prbs_error_cnt[4]	prbs_error_count[5]	prbs_error_cnt[6]	prbs_error_cnt[7]
22	pcs_sts_3_ch_22								
23	pcs_sts_4_ch_int_23	fb_tx_fifo_error_int	fb_rx_fifo_error_int	cc_overrun_int	cc_underrun_int			fb_rx_fifo_error_int	fb_tx_fifo_error_int
24	pcs_sts_5_ch_24	cc_we_o	cc_re_o			fb_txrst_o	fb_rxrst_o	ffs_ls_sync_status	
25	pcs_sts_6_ch_25								
<b>Per Channel SERDES Registers (7)</b>									
26	ser_sts_1_ch_26 (Note 3)		pci_det_done	rlos_lo	~rlos_lo	rlos_hi	~rlos_hi	rlol	~rlol
27	ser_sts_2_ch_27	pci_connect				dco_facq_done	dco_facq_err	dco_calib_done	dco_calib_err
28	ser_sts_3_ch_28	dco_status[0]	dco_status[1]	dco_status[2]	dco_status[3]	dco_status[4]	dco_status[5]	dco_status[6]	dco_status[7]
29	ser_sts_4_ch_29	dco_status[8]	dco_status[9]	dco_status[10]	dco_status[11]	dco_status[12]	dco_status[13]	dco_status[14]	dco_status[15]
2a	ser_sts_5_ch_int_2a (Note 4)		pci_det_done_int	rlos_lo_int	~rlos_lo_int	rlos_hi_int	~rlos_hi_int	rlol_int	~rlol_int
2b	ser_sts_6_ch_2b								
2c	ser_sts_7_ch_2c								

1. BA = Base Address (Hex)

2. Interrupt control register related to an interruptible status (int\_sts\_x) register.

3. Status register which has an associated interruptible status (int\_sts\_x) register.

4. Interruptible status register; clear on read (has associated control register and status register).

5. Status register with clear on read.

Note: Default value is "0", unless otherwise specified.

**Table 8-60. PCS Control Register 1 (pcs\_ctl\_1\_ch\_00)**

Bit	Name	Description	R/W	Default
0	invert_rx	1 = Invert received data 0 = Don't invert received data.	R/W	0
1	invert_tx	1 = Invert transmitted data 0 = Don't invert transmitted data.	R/W	0
3:2	Reserved			
4	ge_an_enable	1 = Enable GigE Auto Negotiation 0 = Disable GigE Auto Negotiation	R/W	0
5	prbs_lock	1 = Lock receive PRBS checker 0 = Unlock receive PRBS checker	R/W	0
6	prbs_enable	1 = Enable PRBS generator & checker 0 = Normal operational mode.	R/W	0
7	enable_cg_align	Only valid when operating in uc_mode 1 = Enable continuous comma alignment 0 = Disable continuous comma alignment	R/W	0

**Table 8-61. PCS Control Register 2 (pcs\_ctl\_2\_ch\_01)**

Bit	Name	Description	R/W	Default
0	tx_ch	1 = Transmit PCS inputs are sourced from test characterization ports. The test characterization mode should be enabled.	R/W	0
1	rx_ch	1 = Receive outputs can be monitored on the test characterization pins. The test characterization mode (bit 6 in pcs_ctl_4_qd_03) should be set to '1'.	R/W	0
2	tx_gear_mode	1 = Enable 2:1 gearing for transmit path on all channels 0 = Disable 2:1 gearing for transmit path on all channels (no gearing)	R/W	0
3	rx_gear_mode	1 = Enable 2:1 gearing for receive path on all channels 0 = Disable 2:1 gearing for receive path on all channels (no gearing)	R/W	0
5:4	pcs_det_time_sel[1:0]	PCS connection detection time 11 = 16us 10 = 4us 01 = 2us 00 = 8us	R/W	0
6	pcie_ei_en	1 = PCI Express Electrical Idle 0 = Normal operation	R/W	0
7	pfifo_clr	1 = Clears PFIFO if quad register bit pfifo_clr_sel is set to 1. This signal is ORed with interface signal pfifo_clr. 0 = Normal operation	R/W	0

**Table 8-62. PCS Control Register 3 (pcs\_ctl\_3\_ch\_02)**

Bit	Name	Description	R/W	Default
0	fb_loopback	1 = Enable loopback in the PCS just before FPGA bridge from RX to TX. 0 = Normal data operation.	R/W	0
1	tx_gear_bypass	1 = Bypass PCS Tx gear box 0 = Normal operation	R/W	0
2	sel_bist_txd4enc	1 = Enable BIST data to Tx before 8b10b ENC 0 = Normal operation	R/W	0
3	enc_bypass	1 = Bypass 8b10b encoder 0 = Normal operation	R/W	0
4	sb_bist_sel	1 = Select BIST data 0 = Select Normal data	R/W	0
5	sb_pfifo_lp	1 = Enable Parallel Loopback from Rx to Tx via parallel FIFO 0 = Normal data operation	R/W	0
6	sb_bypass	1 = Bypass Tx SERDES Bridge 0 = Normal operation	R/W	0
7	bus_width8	1 = 8-bit bus between PCS and SER 0 = 10-bit bus between PCS and SER.	R/W	0

**Table 8-63. PCS Control Register 4 (pcs\_ctl\_4\_ch\_03)**

Bit	Name	Description	R/W	Default
0	sb_loopback	1 = Enable loopback in the PCS from Tx to Rx in SERDES bridge. 0 = Normal data operation.	R/W	0
1	rx_sb_bypass	1 = Bypass Rx SERDES Bridge 0 = Normal operation	R/W	0
2	wa_bypass	1 = Bypass word alignment 0 = Normal operation	R/W	0
3	dec_bypass	1 = Bypass 8b10b decoder 0 = Normal operation	R/W	0
4	ctc_bypass	1 = Bypass clock toleration compensation 0 = Normal operation	R/W	0
5	rx_gear_bypass	1 = Bypass PCS Rx gear box 0 = Normal operation	R/W	0
6	signal_detect	1 = force enabling the Rx link state machine 0 = dependent of ffc_signal_detect to enable the Rx link state machine	R/W	0
7	lsm_disable	1 = disable Rx link state machine 0 = enable Rx link state machine	R/W	0

**Table 8-64. PCS Interrupt Control Register 5 (pcs\_ctl\_5\_ch\_int\_04)**

Bit	Name	Description	R/W	Default
0	fb_tx_fifo_error_int_ctl	1 = Enable interrupt on empty/full condition in the transmit FPGA bridge FIFO.	R/W	0
1	fb_rx_fifo_error_int_ctl	1 = Enable interrupt on empty/full condition in the receive FPGA bridge FIFO.	R/W	0
2	cc_overrun_int_ctl	1 = Enable interrupt for cc_overrun 0 = Disable interrupt for cc_overrun.	RW	0
3	cc_underrun_int_ctl	1 = Enable interrupt for cc_underrun 0 = Disable interrupt for cc_underrun.	RW	0
7:4	Reserved			

**Table 8-65. PCS Control Register 6 (pcs\_ctl\_6\_ch\_05)**

Bit	Name	Description	R/W	Default
7:0	Reserved			

**Table 8-66. PCS Control Register 7 (pcs\_ctl\_7\_ch\_06)**

Bit	Name	Description	R/W	Default
7:0	Reserved			

### Per Channel SERDES Control Register Details

Note: Except indicated, all channels must be reset after writing any SERDES control register.

**Table 8-67. SERDES Control Register 1 (ser\_ctl\_1\_ch\_07)**

Bit	Name	Description	R/W	Default
2:0	RX_DCO_CK_DIV[2:0]	VCO output frequency select: 000 = divided by 1001 = divided by 2 010 = divided by 4011 = divided by 8 100 = reserved 110 = divided by 16111 = divided by 32	R/W	3'b000
4:3	RATE_SEL [1:0]	Equalizer pole position select: 00 = pole position for high frequency range 01 = pole position for medium frequency range 10 = pole position for low frequency range 11 = not used	R/W	2'b00
5	RCV_DCC_EN	1 = Receiver DC coupling enable. 0 = AC coupling (default)	R/W	0
6	REQ_LVL_SET	Level setting for equalization 1 = long-reach equalization 0 = mid-length route equalization	R/W	0
7	REQ_EN	1 = Receiver equalization enable 0 = Receiver equalization disable	R/W	0

**Table 8-68. SERDES Control Register 2 (ser\_ctl\_2\_ch\_08)**

Bit	Name	Description	R/W	Default
1:0	RTERM_RX [1:0]	00 = 50 ohm 01 = 75 ohm 10 = 2K ohm 11 = 60 ohm	R/W	2'b00
3:2	RTERM_RXADJ [1:0]	Termination resistor compensation 00 = default 01= -7% 10= -14% 11= -20%	R/W	2'b00
7:4	LB_CTL [3:0]	Loop back control: [3] = slb_r2t_dat_en, serial rx to tx LB enable(CDR data) [2] = slb_r2t_ck_en, serial rx to tx LB enable(CDR clock) [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

**Table 8-69. SERDES Control Register 3 (ser\_ctl\_3\_ch\_09)**

Bit	Name	Description	R/W	Default
1:0	TDRV_DAT_SEL [1:0]	Driver output select: 00 = data from Serializer muxed to driver (normal operation) 01 = data rate clock from Serializer muxed to driver 10 = serial Rx to Tx LB (data) if slb_r2t_dat_en='1' 10 = serial Rx to Tx LB (clock) if slb_r2t_ck_en='1' 11 = serial LB from equalizer to driver if slb_eq2t_en='1'	R/W	0
4:2	TDRV_PRE_SET [2:0]	TX driver pre-emphasis level setting	R/W	0
7:5	TDRV_AMP [2:0]	CML driver amplitude setting. 000 = default swing amplitude(50uA) 001 = +10% default swing 010 = +26% default swing 011 = +44% default swing 100 = -30% default swing 101 = -24% default swing 110 = -16% default swing 111 = -10% default swing	R/W	0

**Table 8-70. SERDES Control Register 4 (ser\_ctl\_4\_ch\_0a)**

Bit	Name	Description	R/W	Default
0	tpwdnb	0 = Power down transmit channel 1= Power up transmit channel	R/W	0
1	RATE_MODE_TX	0 = Full rate selection for transmit 1 = Half rate selection for transmit	R/W	0
3:2	RTERM_TX [1:0]	TX resistor termination select. Disabled when PCI Express feature is enabled 00 = 50 ohm (default) 01 = 75 ohm 10 = 5K ohm	R/W	2'b00
4	TDRV_PRE_EN	1 = TX driver pre-emphasis enable 0 = TX driver pre-emphasis disable.	R/W	0
7:5	Reserved			

**Table 8-71. SERDES Control Register 4 (ser\_ctl\_5\_ch\_0b)**

Bit	Name	Description	R/W	Default
0	rpwdnb	0 = Power down receive channel. 1= Power up receive channel	R/W	0
1	RATE_MODE_RX	0 = Full rate selection for receive 1 = Half rate selection for receive	R/W	0
3:2	rx_refck_sel [1:0]	Rx CDR Reference Clock Select 00 = rx_refck_local 01 = ck_core_rx 1x = reserved	R/W	2'b00
4	oob_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 application).	R/W	0
7:5	Reserved			

**Table 8-72. SERDES Interrupt Control Register 1 (ser\_ctl\_6\_ch\_int\_0c)**

Bit	Name	Description	R/W	Default
0	Reserved			
1	pci_det_done_int_ctl	1 = Enable interrupt for detection of far-end receiver for PCI Express	R/W	0
2	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
3	~rlos_lo_int_ctl	1 = Enable interrupt for receiver Rx Loss of Signal when input level meets or is greater than programmed LOW threshold	RW	0
4	rlos_hi_int_ctl	1 = Enable interrupt for Rx Loss of Signal when input levels fall below the programmed HIGH threshold (using rlos_set)	RW	0
5	~rlos_hi_int_ctl	1 = Enable interrupt for Rx Loss of Signal when input level meets or is greater than programmed HIGH threshold	RW	0
6	rlol_int_ctl	1= Enable interrupt for receiver loss of lock	R/W	0
7	~rlol_int_ctl	1= Enable interrupt when receiver recovers from loss of lock	R/W	0

**Per Channel PCS Status Register Details****Table 8-73. PCS Status Register 1 (pcs\_sts\_1\_ch\_20)**

Bit	Name	Description	R/W	Int?
0	fb_tx_fifo_error	1 = FPGA bridge (FB) TX FIFO overrun 0 = FB TX FIFO not overrun	RO	Y
1	fb_rx_fifo_error	1 = FPGA bridge (FB) RX FIFO overrun 0 = FB RX FIFO not overrun	RO	Y
2	cc_overrun	1 = CC FIFO overrun 0 = CC FIFO not overrun	RO	Y
3	cc_underrun	1 = CC FIFO underrun 0 = CC FIFO not underrun	RO	Y
4	pfifo_error	1 = Parallel FIFO error 0 = No Parallel FIFO error	RO	Y
7:5	Reserved			

**Table 8-74. PCS Status Register 2 (pcs\_sts\_2\_ch\_21)**

Bit	Name	Description	R/W	Int?
7:0	prbs_errors	Count of the number of PRBS errors. Clears to zero on read. Sticks at FF.	RO CR	N

**Table 8-75. PCS Status Register 3 (pcs\_sts\_3\_ch\_22)**

Bit	Name	Description	R/W	Int?
7:0	Reserved			

**Table 8-76. PCS General Interrupt Status Register 4 (pcs\_sts\_4\_ch\_int\_23)**

Bit	Name	Description	R/W	Int?
0	fb_tx_fifo_error_int	1 = Interrupt generated on fb_tx_fifo_error 0 = Interrupt not generated fb_tx_fifo_error.	RO CR	Y
1	fb_rx_fifo_error_int	1 = Interrupt generated on fb_rx_fifo_error. 0 = Interrupt not generated fb_rx_fifo_error.	RO CR	Y
2	cc_overrun_int	1 = Interrupt generated on cc_overrun 0 = Interrupt not generated on cc_overrun	RO CR	Y
3	cc_underrun_int	1 = Interrupt generated on cc_underrun 0 = Interrupt not generated on cc_underrun	RO CR	Y
7:4	Reserved			

**Table 8-77. PCS Status Register 5 (pcs\_sts\_5\_ch\_24)**

Bit	Name	Description	R/W	Int?
0	cc_we_o	1 = elastic FIFO write enable 0 = elastic FIFO write disable	RO	N
1	cc_re_o	1 = elastic FIFO read enable 0 = elastic FIFO read disable	RO	N
2	Reserved			
3	Reserved			
4	fb_txrst_o	1 = FPGA bridge Tx reset 0 = FPGA bridge Tx reset	RO	N
5	fb_rxrst_o	1 = FPGA bridge Rx reset 0 = FPGA bridge Rx reset	RO	N
6	ffs_ls_sync_status	1 = sync in the link state machine. 0 = not sync in the LSM.	RO	N
7	Reserved			

**Table 8-78. PCS Status Register 6 (pcs\_sts\_6\_ch\_25)**

Bit	Name	Description	R/W	Int?
7:0	Reserved			

## Per Channel SERDES Status Register Details

**Table 8-79. SERDES Status Register 1 (ser\_sts\_1\_ch\_26)**

Bit	Name	Description	R/W	Int?
0	Reserved			
1	pci_det_done	1 = Receiver detection process completed by SERDES transmitter. 0 = Receiver detection process not completed by SERDES transmitter.	RO CR	Y
2	rlos_lo	1= Indicates that the input signal detected by receiver is below the programmed LOW threshold	RO CR	Y
3	~rlos_lo	1= Indicates that the input signal detected by receiver is greater than or equal to the programmed LOW threshold	RO CR	Y
4	rlos_hi	1= Indicates that the input signal detected by receiver is below the programmed HIGH threshold	RO CR	Y
5	~rlos_hi	1= Indicates that the input signal detected by receiver is greater than or equal to the programmed HIGH threshold	RO CR	Y
6	rlol	1=Indicates CDR loss of lock to data. CDR is locked to reference clock.	RO	Y
7	~rlol	1 = Indicates that CDR has locked to data.	RO	Y

**Table 8-80. SERDES Status Register 2 (ser\_sts\_2\_ch\_27)**

Bit	Name	Description	R/W	Int?
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	N
3:1	Reserved			
4	DCO_FACQ_DONE	1 = indicates DCO frequency acquisition done	RO	N
5	DCO_FACQ_ERR	1 = indicates DCO frequency acquisition error (>300ppm)	RO	N
6	DCO_CALIB_DONE	1 = indicates DCO calibration done	RO	N
7	DCO_CALIB_ERR	1 = indicates DCO calibration might be wrong (L/H boundary band selected)	RO	N

**Table 8-81. SERDES Status Register 3 (ser\_sts\_3\_ch\_28)**

Bit	Name	Description	R/W	Int?
7:0	DCO_STATUS[7:0]	setdcoidac[7:0]	RO	N

**Table 8-82. SERDES Status Register 4 (ser\_sts\_4\_ch\_29)**

Bit	Name	Description	R/W	Int?
7:0	DCO_STATUS[15:8]	[1:0] = setdcoidac[9:8] [7:2] = setdcoband[5:0]	RO	N

**Table 8-83. SERDES Interrupt Status Register 5 (ser\_sts\_5\_ch\_int\_2a)**

Bit	Name	Description	R/W	Int?
0	Reserved			
1	pci_det_done_int	1 = Interrupt generated for pci_det_done	RO CR	Y
2	rlos_lo_int	1 = Interrupt generated for rlos_lo	RO CR	Y
3	~rlos_lo_int	1 = Interrupt generated for ~rlos_lo	RO CR	Y
4	rlos_hi_int	1 = Interrupt generated for rlos_hi	CO CR	Y
5	~rlos_hi_int	1 = Interrupt generated for ~rlos_hi	CO CR	Y
6	rlol_int	1 = Interrupt generated for rlol	CO CR	Y
7	~rlol_int	1 = Interrupt generated for ~rlol	CO CR	Y

**Table 8-84. SERDES Status Register 6 (ser\_sts\_6\_ch\_2b)**

Bit	Name	Description	R/W	Default
7:0	Reserved			

**Table 8-85. SERDES Status Register 7 (ser\_sts\_7\_ch\_2c)**

Bit	Name	Description	R/W	Default
7:0	Reserved			

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)  
+1-503-268-8001 (Outside North America)  
e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
September 2006	01.0	Initial release.

## Appendix A. 8b/10b Symbol Codes

**Table 8-86. 8b/10b Symbol Codes**

Symbol Name (Mode Combination Table)	Symbol Code (8b/10b Code)	10-Bit GUI Representation
K28.0	8`b000_11100	0100011100
K28.5	8`b101_11100	0110111100
K29.7	8`b111_11101	0011111101
D16.2	8`b010_10000	0001010000
D21.4	8`b100_10101	0010010101
D21.5	8`b101_10101	0010110101
K28.5+	10`b110000_0101	1100000101
K28.5-	10`b001111_1010	0011111010

**Table 8-87. Lattice Mask for Symbol Code**

Symbol Name (Mode Combination Table)	Symbol Code (8b/10b Code)	10-Bit GUI Representation
28.5 (Mask)	10`b111111_1111	1111111111

## Appendix B. Attribute Cross-Reference Table

**Table 8-88. Attribute Cross-Reference Table**

Independent Attribute Name	Register Map Name
PROTOCOL	{pcs_ctl_1_qd_00[4:0], pcs_ctl_2_ch_01[5:4], ser_ctl_5_ch_0b[4], pcs_ctl_1_ch_00[4]}
CH[0,1,2,3]_MODE	{ser_ctl_4_ch_0a[0], ser_ctl_5_ch_0b[0]}
DATARANGE	{ ser_ctl_4_qd_14[2:0], ser_ctl_1_ch_07[2:0]}
CH[0,1,2,3]_REFCK_MULT	{ser_ctl_2_qd_12[6], {ser_ctl_2_qd_12[7], ser_ctl_3_qd_13[6], ser_ctl_5_ch_0b[1], s er_ctl_4_ch_0a[1]}
CH[0,1,2,3]_DATA_WIDTH	{ rst_ctl_5_qd_1b[7], rst_ctl_5_qd_1b[6], rst_ctl_4_qd_1a[7:4], rst_ctl_4_qd_1a[3:0], pcs_ctl_2_ch_01[2], pcs_ctl_2_ch_01[3], pcs_ctl_3_ch_02[7]}
PLL_SRC	{ ser_ctl_1_qd_11[5]}
CH[0,1,2,3]_CDR_SRC	{ser_ctl_5_ch_0b[3:2]}
CH[0,1,2,3]_TDRV_AMP	ser_ctl_3_ch_09[7:5]
CH[0,1,2,3]_TX_PRE	{ser_ctl_4_ch_0a[4], ser_ctl_3_ch_09[4:2]}
CH[0,1,2,3]_RTERM_TX	ser_ctl_4_ch_0a[3:2]
CH[0,1,2,3]_RX_EQ	{ ser_ctl_1_ch_07[7], ser_ctl_1_ch_07[6], ser_ctl_1_ch_07[4:3]}
CH[0,1,2,3]_RTERM_RX	{ser_ctl_2_ch_08[1:0]}
CH[0,1,2,3]_RX_DCC	{ ser_ctl_1_ch_07[5]}
LOS_THRESHOLD	{ ser_ctl_2_qd_12[2:0], ser_ctl_2_qd_12[5:3]}
PLL_TERM	{ser_ctl_1_qd_11[3]}
PLL_DCC	{ ser_ctl_1_qd_11[4]}
PLL_LOL_SET	{ ser_ctl_4_qd_14[4:3]}
CH[0,1,2,3]_TX_SB	{pcs_ctl_1_ch_00[1], pcs_ctl_3_ch_02[6]}
CH[0,1,2,3]_RX_SB	{pcs_ctl_1_ch_00[0], pcs_ctl_4_ch_03[1]}
CH[0,1,2,3]_8B10B	{pcs_ctl_3_ch_02[3], pcs_ctl_4_ch_03[3]}
COMMA_A	{pcs_ctl_11_qd_0a[0:7], pcs_ctl_13_qd_0c[6:7]}
COMMA_B	{pcs_ctl_12_qd_0b[0:7], pcs_ctl_13_qd_0c[4:5]}
COMMA_M	{pcs_ctl_10_qd_09[0:7], pcs_ctl_13_qd_0c[2:3], }

Independent Attribute Name	Register Map Name
CH[0,1,2,3]_COMMA_ALIGN	{pcs_ctl_4_ch_03[2], pcs_ctl_4_ch_03[7], pcs_ctl_1_ch_00[7], pcs_ctl_4_ch_03[6]}
CH[0,1,2,3]_CTC_BYP	{rst_ctl_3_qd_19[3:0]}
CC_MATCH1	{pcs_ctl_5_qd_04[7:0], pcs_ctl_9_qd_08[1:0]}
CC_MATCH2	{pcs_ctl_6_qd_05[7:0], pcs_ctl_9_qd_08[3:2]}
CC_MATCH3	{pcs_ctl_7_qd_06[7:0], pcs_ctl_9_qd_08[5:4]}
CC_MATCH4	{pcs_ctl_8_qd_07[7:0], pcs_ctl_9_qd_08[7:6]}
CC_MATCH_MODE	{pcs_ctl_4_qd_03[4], pcs_ctl_4_qd_03[5]}
CC_MIN_IPG	{pcs_ctl_4_qd_03[7:6]}
CCHMARK	{pcs_ctl_3_qd_02[7:4]}
CCLMARK	{pcs_ctl_3_qd_02[3:0]}
{PLL_SRC, CHn_CDR_SRC}	{ser_ctl_1_qd_11[0]}
OS_SSLB	{ser_ctl_2_ch_08[5]}
OS_SPLBPORTS	{pcs_ctl_4_qd_03[2], pcs_ctl_3_ch_02[5]}
OS_PCSLBPORTS	{pcs_ctl_4_ch_02[0]}
OS_REFCK2CORE	{ser_ctl_1_qd_11[1]}
OS_PLLQCLKPORTS	{rst_ctl_5_qd_1b[5], rst_ctl_5_qd_1b[3:0]}
OS_INT_ALL	{ser_ctl_6_qd_int_16[7], ser_ctl_6_qd_int_16[6], ser_ctl_6_ch_int_0c[1], ser_ctl_6_ch_int_0c[2], ser_ctl_6_ch_int_0c[3], ser_ctl_6_ch_int_0c[4], ser_ctl_6_ch_int_0c[5], ser_ctl_6_ch_int_0c[6], ser_ctl_6_ch_int_0c[7], pcs_ctl_17_qd_int_10[7:4], pcs_ctl_17_qd_int_10[3:0], pcs_ctl_5_ch_int_04[0], pcs_ctl_5_ch_int_04[1], pcs_ctl_5_ch_int_04[2], pcs_ctl_5_ch_int_04[3]}

September 2006

Technical Note TN1102

## Introduction

The LatticeECP2™ and LatticeECP2M™ 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 they can be implemented using Lattice's ispLEVER® design software.

## sysIO Buffer Overview

LatticeECP2/M sysIO interface contains multiple Programmable I/O Cells (PIC) blocks. Each PIC contains two Programmable I/Os (PIO), PIOA and PIOB, 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 Programmable I/O (PIO) includes a sysIO Buffer and I/O Logic (IOLevel). The LatticeECP2/M sysIO buffers supports 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 16/18 PIOs in the LatticeECP2/M 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 please refer to the LatticeECP2/M Family Data Sheet.

The IOLevel 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 IOLevel 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 IOLevel and the DDR memory interface support are discussed in more details in Lattice technical note number TN1105, *LatticeECP2/M DDR Usage Guide*.

## Supported sysIO Standards

The LatticeECP2/M sysIO buffer supports both single-ended and differential standards. Single-ended standards can be further subdivided into internally ratioed standard such as LVCMOS, LVTTL and PCI; and externally referenced standards such as HSTL and SSTL. The buffers support the LVTTL, LVCMOS 1.2, 1.5, 1.8, 2.5 and 3.3V standards. In the LVCMOS and LVTTL 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. Tables 1 and 2 list the sysIO standards supported in LatticeECP2/M devices.

**Table 9-1. Supported Input Standards**

Input Standard	V <sub>REF</sub> (Nom.)	V <sub>CCIO</sub> <sup>1</sup> (Nom.)
<b>Single Ended Interfaces</b>		
LVTTL	—	—
LVCMOS33	—	—
LVCMOS25	—	—
LVCMOS18	—	1.8
LVCMOS15	—	1.5
LVCMOS12	—	—
PCI 33	—	3.3
HSTL18 Class I, II	0.9	—
HSTL15 Class I	0.75	—

**Table 9-1. Supported Input Standards (Continued)**

Input Standard	$V_{REF}$ (Nom.)	$V_{CCIO}^1$ (Nom.)
SSTL3 Class I, II	1.5	—
SSTL2 Class I, II	1.25	—
SSTL18 Class I, II	0.9	—
<b>Differential Interfaces</b>		
Differential SSTL18 Class I, II	—	—
Differential SSTL2 Class I, II	—	—
Differential SSTL3 Class I, II	—	—
Differential HSTL15 Class I	—	—
Differential HSTL18 Class I, II	—	—
LVDS, MLVDS, LVPECL, BLVDS, RSDS	—	—

1 When not specified,  $V_{CCIO}$  can be set anywhere in the valid operating range.

**Table 9-2. Supported Output Standards**

Output Standard	Drive	$V_{CCIO}$ (Nom.)
<b>Single-ended Interfaces</b>		
LVTTL	4mA, 8mA, 12mA, 16mA, 20mA	3.3
LVCMOS33	4mA, 8mA, 12mA 16mA, 20mA	3.3
LVCMOS25	4mA, 8mA, 12mA, 16mA, 20mA	2.5
LVCMOS18	4mA, 8mA, 12mA, 16mA	1.8
LVCMOS15	4mA, 8mA	1.5
LVCMOS12	2mA, 6mA	1.2
LVCMOS33, Open Drain	4mA, 8mA, 12mA 16mA, 20mA	—
LVCMOS25, Open Drain	4mA, 8mA, 12mA 16mA, 20mA	—
LVCMOS18, Open Drain	4mA, 8mA, 12mA 16mA	—
LVCMOS15, Open Drain	4mA, 8mA	—
LVCMOS12, Open Drain	2mA, 6mA	—
PCI33/PCIX	N/A	3.3
HSTL18 Class I, II	N/A	1.8
HSTL15 Class I	N/A	1.5
SSTL3 Class I, II	N/A	3.3
SSTL2 Class I, II	N/A	2.5
SSTL18 Class I, II	N/A	1.8
<b>Differential Interfaces</b>		
Differential SSTL3, Class I, II	N/A	3.3
Differential SSTL2, Class I, II	N/A	2.5
Differential SSTL18, Class I, II	N/A	1.8
Differential HSTL18, Class I, II	N/A	1.8
Differential HSTL15, Class I	N/A	1.5
LVDS	N/A	2.5
MLVDS <sup>1</sup>	N/A	2.5
BLVDS <sup>1</sup>	N/A	2.5
LVPECL <sup>1</sup>	N/A	3.3
RSDS <sup>1</sup>	N/A	2.5

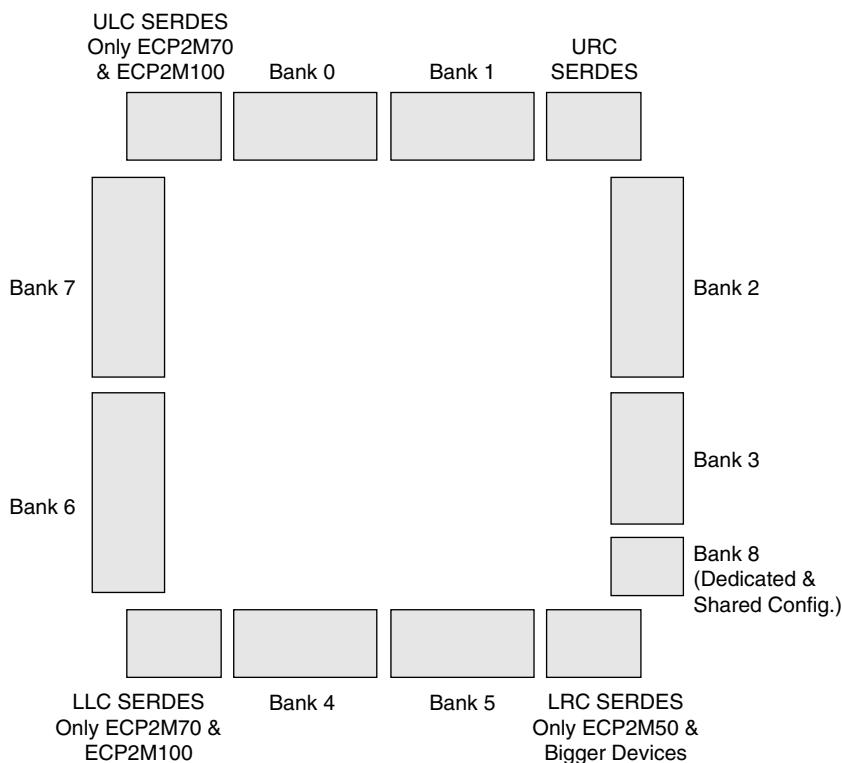
1. Emulated with external resistors.

## sysIO Banking Scheme

LatticeECP2/M devices have eight general purpose programmable sysIO banks and a ninth configuration bank. Each of the eight general purpose sysIO banks has a  $V_{CCIO}$  supply voltage, and two reference voltages,  $V_{REF1}$  and  $V_{REF2}$ . Figure 9-1 shows the eight 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 does have the power supply pads ( $V_{CCIO}$  and  $V_{CCAUX}$ ) but does not have any independent  $V_{REF}$  pads. The I/Os in Bank 8 are connected to  $V_{REF}$  from Bank 3.

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. LatticeECP2M sysIO Banking**



Note: URC = upper right corner, LRC = lower right corner,  
ULC = upper left corner and LLC = lower left corner.

## $V_{CCIO}$ (1.2V/1.5V/1.8V/2.5V/3.3V)

There are a total of eight  $V_{CCIO}$  supplies,  $V_{CCIO0}$  -  $V_{CCIO7}$ . Each bank has a separate  $V_{CCIO}$  supply that powers the single-ended output drivers and the ratioed input buffers such as LVTTI, LVCMS, and PCI. LVTTI, LVCMS3.3, LVCMS2.5 and LVCMS1.2 also have fixed threshold options allowing them to be placed in any bank. The  $V_{CCIO}$  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,  $V_{CCIO8}$  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, 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 LVC MOS JTAG pins, both the output high level and the input threshold.

Table 9-3 shows a summary of all 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	1.8V/2.5V/3.3V

1. Refer to LatticeECP2/M 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 locations of these V<sub>REF</sub> pins are 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.

**V<sub>REF1</sub> 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 more information on the DQS transition detect logic and its implementation, please refer to Lattice technical note number TN1105, *LatticeECP2/M DDR Usage Guide*. For DDR1 memory interfaces, the V<sub>REF1</sub> should be connected to 1.25V. Therefore, only SSTL25\_II signaling is allowed. For DDR2 memory interfaces this should be connected to 0.9V, and only SSTL18\_II signaling is allowed.

**Mixed Voltage Support in a Bank**

The LatticeECP2/M sysIO buffer is connected to three parallel ratioed input buffers. These three parallel buffers are connected to V<sub>CCIO</sub>, V<sub>CCAUX</sub> and V<sub>CC</sub>, giving support for thresholds that track with V<sub>CCIO</sub> as well as fixed thresholds for 3.3V (V<sub>CCAUX</sub>) and 1.2V (V<sub>CC</sub>) inputs. This allows the input threshold for ratioed buffers to be assigned on a pin-by-pin basis rather than tracking with V<sub>CCIO</sub>. This option is available for all 1.2V, 2.5V and 3.3V ratioed inputs and is independent of the bank V<sub>CCIO</sub> voltage. For example, if the bank V<sub>CCIO</sub> is 1.8V, 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.

Prior to device configuration, the ratioed input thresholds always tracks the bank V<sub>CCIO</sub>. This option only takes effect after configuration. Output standards within a bank are always set by V<sub>CCIO</sub>. Table 9-4 shows the sysIO standards that can be mixed in the same bank.

**Table 9-4. Mixed Voltage Support**

V <sub>CCIO</sub>	Input sysIO Standards					Output sysIO Standards				
	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

**sysIO Standards Supported by Bank****Table 9-5. I/O Standards Supported by Bank**

Description	Top Side Banks 0-1	Right Side Banks 2-3	Bottom Side Banks 4-5	Left Side Banks 6-7
I/O Buffers	Single-ended	Single-ended and Differential	Single-ended	Single-ended and Differential
Output Standards Supported	LVTTL LVCMOS33 LVCMOS25 LVCMOS18 LVCMOS15 LVCMOS12  SSTL18 Class I, II SSTL25 Class I, II SSTL33 Class I, II  HSTL15 Class I HSTL18_I, II  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> RSDS <sup>1</sup>	LVTTL LVCMOS33 LVCMOS25 LVCMOS18 LVCMOS15 LVCMOS12  SSTL18 Class I, II SSTL25 Class I, II SSTL33 Class I, II  HSTL15 Class I HSTL18 Class I, II  SSTL18D Class I, II SSTL25D Class I, II SSTL33D Class I, II  HSTL15D Class I HSTL18D Class I, II  LVDS LVDS25E <sup>1</sup> LVPECL <sup>1</sup> BLVDS <sup>1</sup> RSDS <sup>1</sup>	LVTTL LVCMOS33 LVCMOS25 LVCMOS18 LVCMOS15 LVCMOS12  SSTL18 Class I SSTL2 Class I, II SSTL3 Class I, II  HSTL15 Class I HSTL18 Class I, II  SSTL18D Class I, II SSTL25D Class I, II SSTL33D Class I, II  HSTL15D Class I HSTL18D Class I, II  PCI33 LVDS25E <sup>1</sup> LVPECL <sup>1</sup> BLVDS <sup>1</sup> RSDS <sup>1</sup>	LVTTL LVCMOS33 LVCMOS25 LVCMOS18 LVCMOS15 LVCMOS12  SSTL18 Class I SSTL2 Class I, II SSTL3 Class I, II  HSTL15 Class I, III HSTL18 Class I, II, III  SSTL18D Class I, SSTL25D Class I, II, SSTL33D_I, II  HSTL15D Class I HSTL18D Class I, II  PCI33 LVDS LVDS25E <sup>1</sup> LVPECL <sup>1</sup> BLVDS <sup>1</sup> RSDS <sup>1</sup>
Inputs	All Single-ended, Differential	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
PCI Support	PCI33 without clamp	PCI33 without clamp	PCI33 with clamp	PCI33 without clamp PCI33 with clamp (ECP2M)
LVDS Output Buffers		LVDS (3.5mA) Buffers <sup>2</sup>		LVDS (3.5mA) Buffers <sup>2</sup>

1. These differential standards are implemented by using a complementary LVCMOS driver with external resistor pack.

2. Available only on 50% of the I/Os in the bank.

## LVC MOS Buffer Configurations

All LVC MOS buffer have programmable pull, programmable drive and programmable slew configurations that can be set in the software.

### Bus Maintenance Circuit

Each pad has a weak pull-up, weak pull-down and weak buskeeper 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. Users 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. The weak bus keeper is not available when  $V_{CCIO}$  of the bank is assigned to 3.3V.

### Programmable Drive

Each LVC MOS or LV TTL, 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 the 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 9-6 shows the available drive settings for each out the output standards.

**Table 9-6. Programmable Drive Values for Single-ended Buffers**

Single Ended I/O Standards	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
LVC MOS12	2, 6
LVC MOS15	4, 8
LVC MOS18	4, 8, 12, 16
LVC MOS25	4, 8, 12, 16, 20
LVC MOS33	4, 8, 12, 16, 20
LV TTL	4, 8, 12, 16, 20

### Programmable Slew Rate

Each LVC MOS or LV TTL 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 and falling edges.

### Open-Drain Control

All LVC MOS and LV TTL 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 syn-

chronous 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 bottom of the device (for LatticeECP2) or on the left and bottom sides of the device (for LatticeECP2M) have an optional PCI clamp diode that may optionally be specified in the ispLEVER design tools.

Programmable PCICLAMP can be turned ON or OFF. This option is available on each I/O independently on the bottom side banks (for LatticeECP2) or on the left and bottom side banks (for LatticeECP2M).

## Programmable Input Delay

Each input can optionally be delayed before it is passed to the core logic or input registers. The primary use for the input delay is to achieve zero hold time for the input registers when using a direct drive primary clock. To arrive at zero hold time, the input delay will delay the data by at least as much as the primary clock injection delay. This option can be turned ON or OFF for each I/O independently in the software using the FIXEDDELAY attribute. This attribute is described in more detail in the software sysIO attribute section. Appendix A shows how this feature can be enabled in the software using HDL attributes.

## Software sysIO Attributes

sysIO attributes can be specified in the HDL, using the Preference Editor GUI or in the ASCII preference file (.prf) file directly. Appendices A, B and C list examples of how these can be assigned using each of these methods. This section describes each of these attributes in detail.

### IO\_TYPE

This 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 names itself. There is no separate attribute to set the  $V_{CCIO}$  requirements. Table 9-7 lists the available I/O types.

**Table 9-7. IO\_TYPE Attribute Values**

sysIO Signaling Standard	IO_TYPE
DEFAULT	LVCMOS25
LVDS 2.5V	LVDS25
RSDS	RSDS
Emulated LVDS 2.5V	LVDS25E <sup>1</sup>
Bus LVDS 2.5V	BLVDS25 <sup>1</sup>
LVPECL 3.3V	LVPECL33 <sup>1</sup>
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	SSTL18D_I, SSTL18D_II
LVTTL	LVTTL33
3.3V LVCMOS	LVCMOS33
2.5V LVCMOS	LVCMOS25
1.8V LVCMOS	LVCMOS18
1.5V LVCMOS	LVCMOS15
1.2V LVCMOS	LVCMOS12
3.3V PCI	PCI33

1. These differential standards are implemented by using a complementary LVCMOS driver with external resistor pack.

## OPENDRAIN

LVCMOS and LVTTL I/O standards can be set to open drain configuration by using the OPENDRAIN attribute.

Values: ON, OFF

Default: OFF

## DRIVE

The DRIVE attribute will set the programmable drive strength for the output standards that have programmable drive capability

**Table 9-8. 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
LVCMOS12	2, 6	6
LVCMOS15	4, 8	8
LVCMOS18	4, 8, 12, 16	12
LVCMOS25	4, 8, 12, 16, 20	12
LVCMOS33	4, 8, 12, 16, 20	12
LVTTL	4, 8, 12, 16, 20	12

## PULLMODE

The PULLMODE attribute is available for all the LVTTL and LVCMOS inputs and outputs. This attribute can be enabled for each I/O independently.

Values: UP, DOWN, NONE, KEEPER

Default: UP

**Table 9-9. PULLMODE Values**

PULL Options	PULLMODE Value
Pull-up (Default)	UP
Pull-down	DOWN
Bus Keeper	KEEPER
Pull Off	NONE

## PCICLAMP

PCI33 inputs on the bottom of the device (for LatticeECP2) or on the left and bottom sides of the device (for LatticeECP2M) have an optional PCI clamp that is enabled via the PCICLAMP attribute. The PCICLAMP is also available for all LVCMOS33 and LVTTL inputs.

Values: ON, OFF

Default: OFF

**Table 9-10. PCICLAMP Values**

Input Type	PCICLAMP Value
PCI33	ON
LVCMOS33	OFF (default), ON
LVTTL	OFF (default), ON

## SLEWRATE

The SLEWRATE attribute is available for all LVTTL and LVCMOS output drivers. Each I/O pin has an individual slew rate control. This allows a designer to specify slew rate control on a pin-by-pin basis.

Values: FAST, SLOW

Default: FAST

## FIXEDDELAY

The FIXEDDELAY attribute is available to each input pin. This attribute, when enabled, is used to achieve zero hold time for the input registers when using global clock. This attribute can only be assigned in the HDL source.

Values: TRUE, FALSE

Default: FALSE

## 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 or OFF.

Values: ON, OFF

Default: OFF

## DIN/DOUT

This attribute can be used to assign I/O registers. Using DIN will assert an input register and using the DOUT attribute will assert an output register. By default, the software will try to assign the I/O registers, if applicable. The user can turn this OFF by using the synthesis attribute or by using the Preference Editor of the ispLEVER software. These attributes can only be applied to 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. Designers can also assign pins directly using the GUI in the Preference Editor of the ispLEVER software. The appendices explain this in further detail.

## Design Considerations and Usage

This section discusses some of the design rules and considerations that must be taken into account when designing with the LatticeECP2/M sysIO buffer

### Banking Rules

- If  $V_{CCIO}$  or  $V_{CCJ}$  for any bank is set to 3.3 V, 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 bottom banks for LatticeECP2 (Banks 4 and 5) or left and bottom banks for LatticeECP2M (Banks 4, 5 and 6) will support PCI clamps.
- 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.

### 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 external resistor pack and complementary LVCMSO drivers.
  - Only 50% of the I/Os on the left and right sides can provide LVDS output buffer capability. LVDS can only be assigned to the TRUE pad. The ispLEVER design tool will automatically assign the other I/Os of the differential pair to the complementary pad. Refer to the device data sheet to see the pin listings for all LVDS pairs.
-

## Assigning V<sub>REF1</sub>/V<sub>REF2</sub> Groups for Referenced Inputs

Each bank has two dedicated V<sub>REF</sub> input pins, V<sub>REF1</sub> and V<sub>REF2</sub>. Designers can group buffers to a particular V<sub>REF</sub> rail, V<sub>REF1</sub> or V<sub>REF2</sub>. This grouping is done by assigning a PGROUP VREF preference along with the LOCATE PGROUP preference.

### Preference Syntax

```
PGROUP <pgrp_name> [(VREF <vref_name>)+] (COMP <comp_name>)+;  
LOCATE PGROUP <pgrp_name> BANK <bank_num>;  
LOCATE VREF <vref_name> SITE <site_name>;
```

### Example Showing VREF Groups

```
PGROUP "vref_pg1" VREF "ref1" COMP "ah(0)" COMP "ah(1)" COMP "ah(2)" COMP "ah(3)"  
COMP "ah(4)" COMP "ah(5)" COMP "ah(6)" COMP "ah(7)";  
  
PGROUP "vref_pg2" VREF "ref2" COMP "al(0)" COMP "al(1)" COMP "al(2)" COMP "al(3)"  
COMP "al(4)" COMP "al(5)" COMP "al(6)" COMP "al(7)";  
  
LOCATE VREF "ref1" SITE PR29C;  
LOCATE VREF "ref2" SITE PR48B;
```

or

```
LOCATE PGROUP " vref_pg1" BANK 2;  
LOCATE PGROUP " vref_pg2" BANK 2;
```

The example shows two V<sub>REF</sub> groups, "vref\_pg1" assigned to VREF "ref1" and "vref\_pg2" assigned to "ref2". The user must lock these V<sub>REF</sub> to either V<sub>REF1</sub> or V<sub>REF2</sub> using the LOCATE preference. Alternatively, users can designate to which bank the V<sub>REF</sub> group should be located. The software will then assign these to either the V<sub>REF1</sub> or V<sub>REF2</sub> of the bank.

If the PGROUP VREF is not used, the software will automatically group all pins that need the same V<sub>REF</sub> reference voltage. This preference is most useful when there is more than one bus that uses the same reference voltage and the user wishes to associate each of these busses to different V<sub>REF</sub> resources.

## Differential I/O Implementation

The LatticeECP2/M devices support a variety of differential standards as detailed in the following sections.

### LVDS

True LVDS (LVDS25) drivers are available on 50% of the I/Os on the left and right side of the devices. LVDS input support is provided on all sides of the device. All four sides of the device support LVDS using complementary LVC-MOS drivers with external resistors (LVDS25E). Refer to the LatticeECP2/M Family Data Sheet for a detailed explanation of these LVDS implementations.

### BLVDS

All single-ended sysIO buffers pairs support the Bus-LVDS standard using complementary LVCMOS drivers with external resistors. Please refer to the LatticeECP2/M Family Data Sheet for a detailed explanation of BLVDS implementation.

### RSDS

All single-ended sysIO buffers pairs support RSDS standard using complementary LVCMOS drivers with external resistors. Please refer to the LatticeECP2/M Family Data Sheet for a detailed explanation of RSDS implementation.

## LVPECL

All the sysIO buffers will support LVPECL inputs. LVPECL outputs are supported using complementary LVCMSO driver with external resistors. Please refer to the LatticeECP2/M Family Data Sheet for a detailed explanation of LVPECL implementation.

## Differential SSTL and HSTL

All single-ended sysIO buffers pairs support differential SSTL and HSTL. Please refer to the LatticeECP2/M Family Data Sheet for a detailed explanation of Differential HSTL and SSTL implementation.

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)  
+1-503-268-8001 (Outside North America)  
e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2006	01.0	Initial release.
September 2006	01.1	Updated to include LatticeECP2M support.

## Appendix A. HDL Attributes for Synplicity® and Precision® RTL Synthesis

Using these HDL attributes, designers can assign the sysIO attributes directly in their source. The attribute definition and syntax for the appropriate synthesis vendor must be used. Below are a list of all the sysIO attributes, syntax and examples for Precision RTL Synthesis and Synplicity. This section only lists the sysIO buffer attributes for these devices. You can refer to the Precision RTL Synthesis and Synplicity user manuals for a complete list of synthesis attributes. These manuals are available through the ispLEVER software Help system.

### VHDL Synplicity/Precision RTL Synthesis

This section lists syntax and examples for all the sysIO Attributes in VHDL when using the Precision RTL Synthesis or Synplicity synthesis tools.

#### Syntax

**Table 9-11. VHDL Attribute Syntax for Synplicity and Precision RTL Synthesis**

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";
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 "Slewrate Value";
FIXEDDELAY	attribute FIXEDDELAY: string; attribute FIXEDDELAY of Pinname: signal is "Fixeddelay 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";

#### 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 "LVC莫斯33";
ATTRIBUTE IO_TYPE OF portC: SIGNAL IS "LVDS25";
```

##### OPENDRAIN

```
-----Attribute Declaration-----
ATTRIBUTE OPENDRAIN: string;
-----DRIVE 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";
```

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 "ON";
```

SLEWRATE

```
----Attribute Declaration***  
ATTRIBUTE SLEWRATE : string;  
---- SLEWRATE assignment for I/O Pin***  
ATTRIBUTE SLEWRATE OF portB: SIGNAL IS "FAST";
```

FIXEDDELAY

```
----Attribute Declaration***  
ATTRIBUTE FIXEDDELAY: string;  
---- SLEWRATE assignment for I/O Pin***  
ATTRIBUTE FIXEDDELAY OF portB: SIGNAL IS "TRUE";
```

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 ";
```

## Verilog Synplicity

This section lists syntax and examples for all the sysIO Attributes in Verilog using the Synplicity synthesis tool.

### Syntax

**Table 9-12. Verilog Synplicity Attribute Syntax**

Attribute	Syntax
IO_TYPE	<i>PinType PinName</i> /* synthesis IO_TYPE="IO_Type Value"*/;
OPENDRAIN	<i>PinType PinName</i> /* synthesis OPENDRAIN ="OpenDrain Value"*/;
DRIVE	<i>PinType PinName</i> /* synthesis DRIVE="Drive Value"*/;
PULLMODE	<i>PinType PinName</i> /* synthesis PULLMODE="Pullmode Value"*/;
PCICLAMP	<i>PinType PinName</i> /* synthesis PCICLAMP =" PCIclamp Value"*/;
SLEWRATE	<i>PinType PinName</i> /* synthesis SLEWRATE="Slewrate Value"*/;
FIXEDDELAY	<i>PinType PinName</i> /* synthesis FIXEDDELAY="Fixeddelay Value"*/;
DIN	<i>PinType PinName</i> /* synthesis DIN=""*/;
DOUT	<i>PinType PinName</i> /* synthesis DOUT=""*/;
LOC	<i>PinType PinName</i> /* synthesis LOC="pin_locations"*/;

### 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" */;

//OPENDRAIN
output portA /*synthesis OPENDRAIN ="ON"*/;

//PCICLAMP
output portA /*synthesis IO_TYPE="PCI33" PULLMODE ="PCICLAMP"*/;

// Fixeddelay
input load /* synthesis FIXEDDELAY="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="" */;

//I/O 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" */;
```

## Verilog Precision

This section lists syntax and examples for all the sysIO Attributes in Verilog using the Precision RTL Synthesis tool.

### Syntax

**Table 9-13. Verilog Precision Attribute Syntax**

Attribute	Syntax
IO_TYPE	//pragma attribute <i>PinName</i> IO_TYPE IO_TYPE Value
OPENDRAIN	// pragma attribute <i>PinName</i> OPENDRAIN OpenDrain Value
DRIVE	// pragma attribute <i>PinName</i> DRIVE Drive Value
PULLMODE	// pragma attribute <i>PinName</i> IO_TYPE Pullmode Value
PCICLAMP	// pragma attribute <i>PinName</i> PCICLAMP PCIClamp Value
SLEWRATE	// pragma attribute <i>PinName</i> IO_TYPE Slewrate Value
FIXEDDELAY	// pragma attribute <i>PinName</i> IO_TYPE Fixeddelay Value
LOC	// pragma attribute <i>PinName</i> LOC pin_location

### Examples

```

//****IO_TYPE ***
//pragma attribute portA IO_TYPE PCI33
//pragma attribute portB IO_TYPE LVCMOS33
//pragma attribute portC IO_TYPE SSTL25_II

//*** Opendrain ***
//pragma attribute portB OPENDRAIN ON
//pragma attribute portD OPENDRAIN OFF

//*** Drive ***
//pragma attribute portB DRIVE 20
//pragma attribute portD DRIVE 8

//*** Pullmode ***
//pragma attribute portB PULLMODE UP

//*** PCIClamp ***
//pragma attribute portB PCICLAMP ON

//*** Slewrate ***
//pragma attribute portB SLEWRATE FAST
//pragma attribute portD SLEWRATE SLOW

// ***Fixeddelay***
// pragma attribute load FIXEDDELAY TRUE

//***LOC***
//pragma attribute portB loc E3

```

## Appendix B. sysIO Attributes Using the Preference Editor User Interface

Designers can assign sysIO buffer attributes using the Pre-Map Preference Editor GUI available in the ispLEVER design tool. The Pin Attribute Sheet list all the ports in a design and all the available sysIO attributes as preferences. By clicking on each of these cells, a list of all the valid I/O preference for that port is displayed. Each column takes precedence over the next. Therefore, when a particular IO\_TYPE is chosen, the DRIVE, PULLMODE and SLEWRATE columns will only list the valid combinations for that IO\_TYPE. The pin locations can be locked using the pin location column of the Pin Attribute sheet. Right-clicking on a cell will list the available pin locations. The Preference Editor will also conduct a DRC check to search for any incorrect pin assignments.

Designers can enter DIN/DOUT preferences using the Cell Attributes sheet of the Preference Editor. All the preferences assigned using the Preference Editor are written into the preference file (.prf).

Figures 2 and 3 show the Pin Attribute sheet and the Cell Attribute sheet views of the preference editor. For further information on how to use the Preference Editor, refer to the ispLEVER Help documentation in the Help menu option of the software.

**Figure 9-2. Pin Attributes Tab**

	Type	Signal/Gr...	Group...	Pin Location	IO Type	Drive	Slewrate	Pullmode	Output Load
2	Output Port	portD(3)	N/A			N/A	N/A	N/A	
3	Output Port	portD(2)	N/A			N/A	N/A	N/A	
4	Output Port	portD(1)	N/A			N/A	N/A	N/A	
5	Output Port	portD(0)	N/A			N/A	N/A	N/A	
6	Output Port	portC(4)	N/A	A17	LVCMS33			NONE	
7	Output Port	portC(3)	N/A	A18	BLVDS25			NONE	N/A
8	Output Port	portC(2)	N/A	A19	LVCMS25_OD			NONE	
9	Output Port	portC(1)	N/A	A20	LVCMS15			NONE	
10	Output Port	portC(0)	N/A	A15	LVPECL33			NONE	N/A
11	Output Port	portB(4)	N/A			N/A	N/A	N/A	
12	Output Port	portB(3)	N/A			N/A	N/A	N/A	
13	Output Port	portB(2)	N/A			N/A	N/A	N/A	
14	Output Port	portB(1)	N/A			N/A	N/A	N/A	

**Figure 9-3. Cell Attributes Tab**

	Type	Cell Name	Din / Dout
1	FFs	ix266	Din
2	FFs	ix205	Din
3	FFs	ix212	Din
4	FFs	ix215	Din
5	FFs	ix218	Din
6	FFs	ix221	Din
7	FFs	ix224	Dout
8	FFs	ix227	Dout
9	FFs	ix230	Dout
10	FFs	ix233	Din
11	FFs	ix236	Dout
12	FFs	ix239	Din
13	FFs	ix242	Din

## Appendix C. sysIO Attributes Using Preference File (ASCII File)

Designers can enter sysIO attributes directly in the preference (.prf) file as sysIO buffer preferences. The PRF file is an ASCII file containing two separate sections: a schematic section for those preferences created by the mapper or translator, and a user section for preferences entered by the user. User preferences can be written directly into this file. The synthesis attributes appear between the schematic start and schematic end of the file. The sysIO buffer preferences can be entered after the schematic end line using the preference file syntax. Below are a list of sysIO buffer preference syntax and examples.

### IOBUF

This preference is used to assign the attribute IO\_TYPE, PULLMODE, SLEWRATE and DRIVE.

#### Syntax

```
IOBUF [ALLPORTS | PORT <port_name> | GROUP <group_name>] (keyword=<value>)+;
```

where:

<port\_name> = These are not the actual top-level port names, but should be the signal name attached to the port. PIOs in the physical design (.ncd) file are named using this convention. Any multiple listings or wildcarding should be done using GROUPs

Keyword = IO\_TYPE, OPENDRAIN, DRIVE, PULLMODE, PCICLAMP, SLEWRATE.

#### Example

```
IOBUF PORT "port1" IO_TYPE=LVTTL33 OPENDRAIN=ON DRIVE=8 PULLMODE=UP
PCICLAMP =OFF SLEWRATE=FAST;
DEFINE GROUP "bank1" "in*" "out_[0-31]";
IOBUF GROUP "bank1" IO_TYPE=SSTL18_II;
```

### LOCATE

When this preference is applied to a specified component, it places the component at a specified site and locks the component to the site. If applied to a specified macro instance, it places the macro's reference component at a specified site, places all of the macro's pre-placed components (that is, all components that were placed in the macro's library file) in sites relative to the reference component, and locks all of these placed components at their sites. This can also be applied to a specified PGROUP.

#### Syntax

```
LOCATE [COMP <comp_name> | MACRO <macro_name>] SITE <site_name>;
LOCATE PGROUP <pgroup_name> [SITE <site_name>; | REGION <region_name>;]
LOCATE PGROUP <pgroup_name> RANGE <site_1> [<site_2> | <count>] [<direction>] |
RANGE <chip_side> [<direction>];
LOCATE BUS < bus_name> ROW|COL <number>;
<bus_name> := string
<number> := integer
```

Note: If the comp\_name, macro\_name, or site\_name begins with anything other than an alpha character (for example, "11C7"), you must enclose the name in quotes. Wildcard expressions are allowed in <comp\_name>.

#### Examples

This command places the port Clk0 on the site A4:

```
LOCATE COMP "Clk0" SITE "A4";
```

This command places the component PFU1 on the site named R1C7:

```
LOCATE COMP "PFU1" SITE "R1C7";
```

This command places bus1 on ROW 3 and bus2 on COL4

```
LOCATE BUS "bus1" ROW 3;
LOCATE BUS "bus2" COL 4;
```

## USE DIN CELL

This preference specifies the given register to be used as an input flip-flop.

### Syntax

```
USE DIN CELL <cell_name>;
```

where:

```
<cell_name> := string
```

### Example

```
USE DIN CELL "din0";
```

## USE DOUT CELL

Specifies the given register to be used as an output flip-flop.

### Syntax

```
USE DOUT CELL <cell_name>;
```

where:

```
<cell_name> := string
```

### Example

```
USE DOUT CELL "dout1";
```

## PGROUP VREF

This preference is used to group all the components that need to be associated to one V<sub>REF</sub> pin within a bank.

### Syntax

```
PGROUP <pgrp_name> [(VREF <vref_name>)+] (COMP <comp_name>)+;
LOCATE PGROUP <pgrp_name> BANK <bank_num>;
LOCATE VREF <vref_name> SITE <ssite_name>;
```

### Example

```
PGROUP "vref_pg1" VREF "ref1" COMP "ah(0)" COMP "ah(1)" COMP "ah(2)" COMP "ah(3)"
COMP "ah(4)" COMP "ah(5)" COMP "ah(6)" COMP "ah(7)";
PGROUP "vref_pg2" VREF "ref2" COMP "al(0)" COMP "al(1)" COMP "al(2)" COMP "al(3)"
COMP "al(4)" COMP "al(5)" COMP "al(6)" COMP "al(7)";
LOCATE VREF "ref1" SITE PR29C;
LOCATE VREF "ref2" SITE PR48B;
```

or:

```
LOCATE PGROUP " vref_pg1" BANK 2;  
LOCATE PGROUP " vref_pg2" BANK 2;
```

## Introduction

This user's guide describes the clock resources available in the LatticeECP2™ and LatticeECP2M™ device architectures. Details are provided for primary clocks, secondary clocks and edge clocks, as well as clock elements such as PLLs, DLLs, Clock Dividers and more.

The number of PLLs and DLLs for each package can be found in Tables 10-1 and 10-2.

**Table 10-1. Number of PLLs and DLLs: LatticeECP2 Family**

Device	Description	ECP2-6	ECP2-12	ECP2-20	ECP2-35	ECP2-50	ECP2-70
Number of SPLLs	Standard PLL (Subset of GPLL)	0	0	0	0	2	4
Number of GPLLs	General Purpose PLL	2	2	2	2	2	2
Number of DLLs	General Purpose DLL	2	2	2	2	2	2
Number of DQSDLLs	DLL for DDR Applications	2	2	2	2	2	2

**Table 10-2. Number of PLLs, DLLs and SERDES: LatticeECP2M Family**

Device	Description	ECP2M-20	ECP2M-35	ECP2M-50	ECP2M-70	ECP2M-100
Number of SPLLs	Standard PLL (Subset of GPLL)	6	6	6	6	6
Number of GPLLs	General Purpose PLL	2	2	2	2	2
Number of DLLs	General Purpose DLL	2	2	2	2	2
Number of DQSDLLs	DLL for DDR Applications	2	2	2	2	2
SERDES	4-Channel Quad SERDES	1	1	2	4	4

## Clock/Control Distribution Network

The LatticeECP2/M family provides global clock distribution in the form of eight quadrant-based primary clocks and flexible secondary clocks. The devices also provide two edge clocks on each edge of the device. Other clock sources include clock input pins, internal nodes, PLLs, DLLs, Slave Delay Lines and Clock Dividers.

## LatticeECP2/M Top Level View

Figure 10-1 shows the primary clocking structure of the LatticeECP2-50 device.

**Figure 10-1. LatticeECP2-50 Clocking Structure**

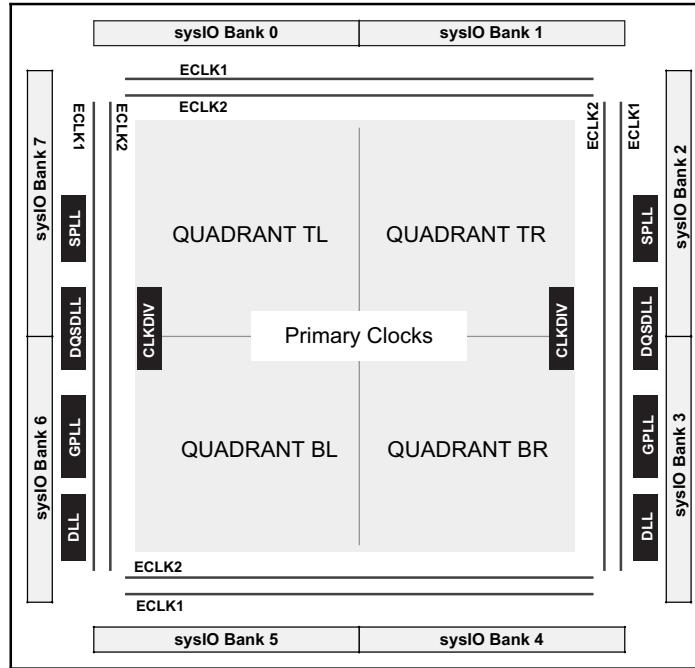
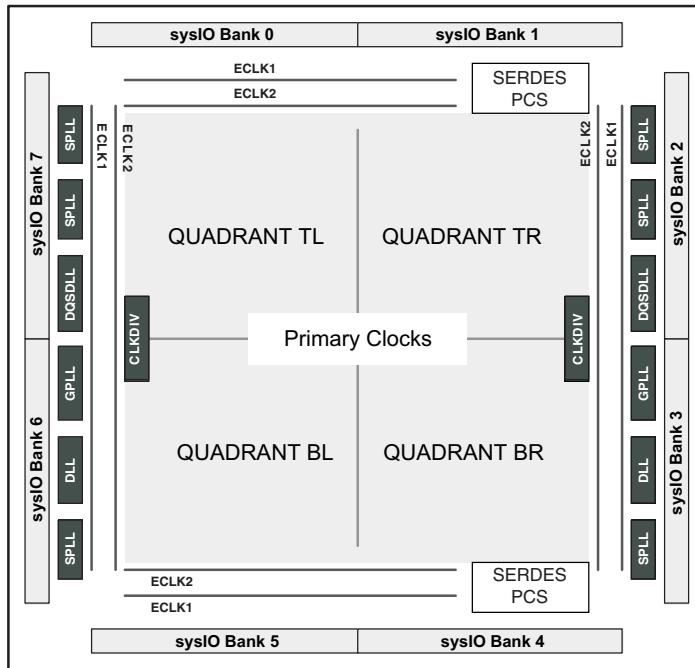


Figure 8-2 illustrates the primary clocking structure of the LatticeECP2M-50 device. The figure shows two SERDES blocks. The edge clocks on the top and bottom sides stop when they reach the SERDES block boundary. Other members of the LatticeECP2M family have a similar structure, except for the number of SERDES blocks.

**Figure 10-2. LatticeECP2M-50 Clocking Structure**



## Primary Clocks

Each quadrant receives up to eight primary clocks. 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 the primary clocks are:

- PLL outputs
- DLL outputs
- CLKDIV outputs
- Dedicated clock pins
- Internal nodes
- SERDES TX\_H\_CLK (LatticeECP2M only)

## Secondary Clocks

The LatticeECP2/M secondary clocks are a flexible region-based clocking resource. Each region can have four independent clock inputs. As a regional resource, it can cross the primary clock quadrant boundaries.

There are eight secondary clock muxes per quadrant. Each mux has inputs from four different sources. Three of these are from internal nodes. The fourth input comes from a primary clock pin. The input sources are not necessarily located in the same quadrant as the Mux. This structure enables global usage of secondary clocks.

The sources of secondary clocks are:

- Dedicated clock pins
- Internal nodes

## Edge Clocks

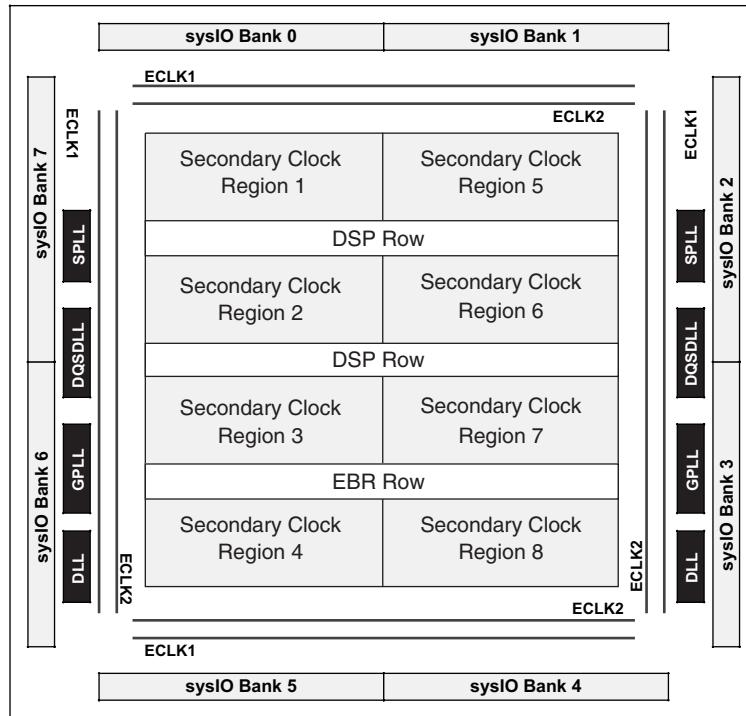
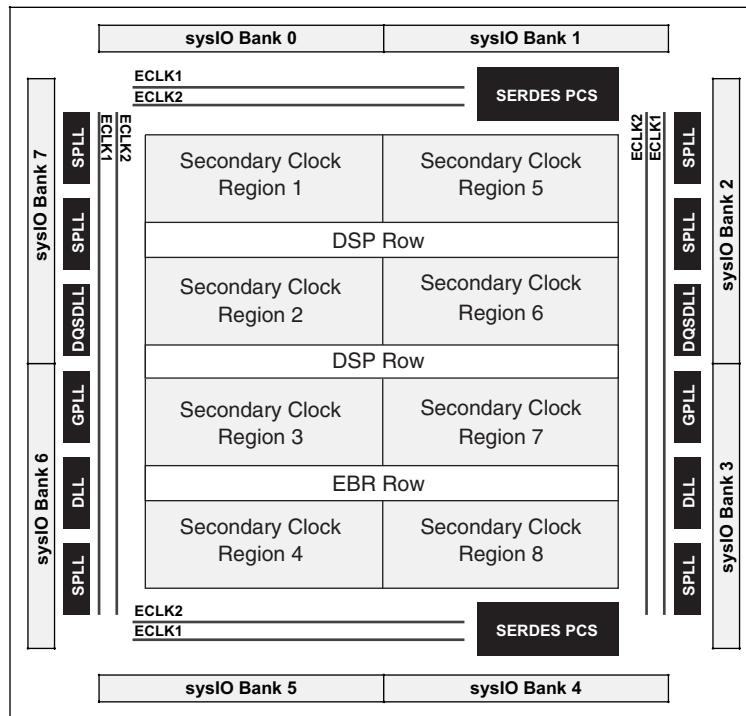
The LatticeECP2/M has two edge clocks per side. These clocks, which have low injection times and skew, are used to clock I/O registers. The edge clock (ECLK) resources are designed for high speed I/O interfaces with high fanout capability. Refer to Appendix B for detailed connectivity information.

The sources of the edge clocks are:

- Left and Right Edge Clocks
  - Dedicated clock pins
  - PLL outputs
  - DLL outputs
  - Internal nodes
- Top and Bottom Edge Clocks
  - Dedicated clock pins
  - Internal nodes

ECLK can directly drive the secondary clock resources and general routing resources. This means that an ECLK source clock can also route to the Primary Clock Net through general routing at the same time.

Figure 10-3 describes the secondary clock and edge clock structure.

**Figure 10-3. LatticeECP2-50 Secondary Clocks and Edge Clocks****Figure 10-4. LatticeECP2M-50 Secondary Clocks and Edge Clocks**

### Note on Primary Clocks

The CLKOP must be used as the feedback source to optimize the PLL performance.

Most designers use PLL for clock tree injection removal mode and the CLKOP should be assigned to the Primary Clock. This is done automatically by the software unless the user specifies otherwise.

CLKOP can route to CLK0 to CLK5 only and CLKOS/CLKOK can route to all Primary Clocks (CLK0 to CLK7).

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. See the DCS section of this document for further information.

## 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 10-4 illustrates how this can be done in the Pre-Map Preference Editor. Alternatively the Preference file can be used, as discussed in Appendix C.

### Primary-Pure and Primary-DCS

Primary Clock Net can be assigned to either Primary-Pure (CLK0 to CLK5) or Primary-DCS (CLK6 and CLK7).

### 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.

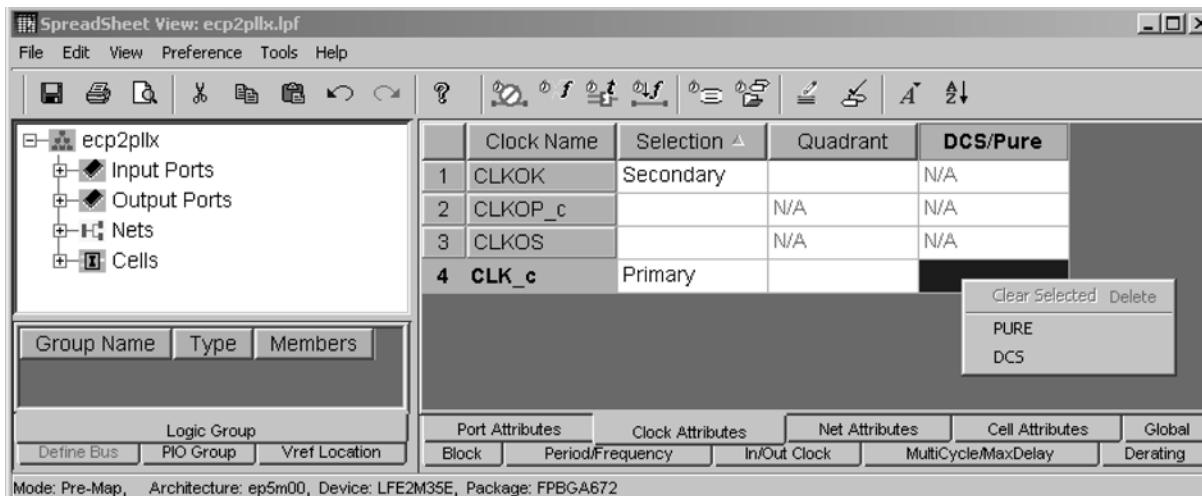
#### Quadrant Primary Clock

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, as long as all the Primary Clock sources are available.

**Figure 10-5. Clock Preferences in the Pre-Map Preference Editor**



### Note on Edge Clocks

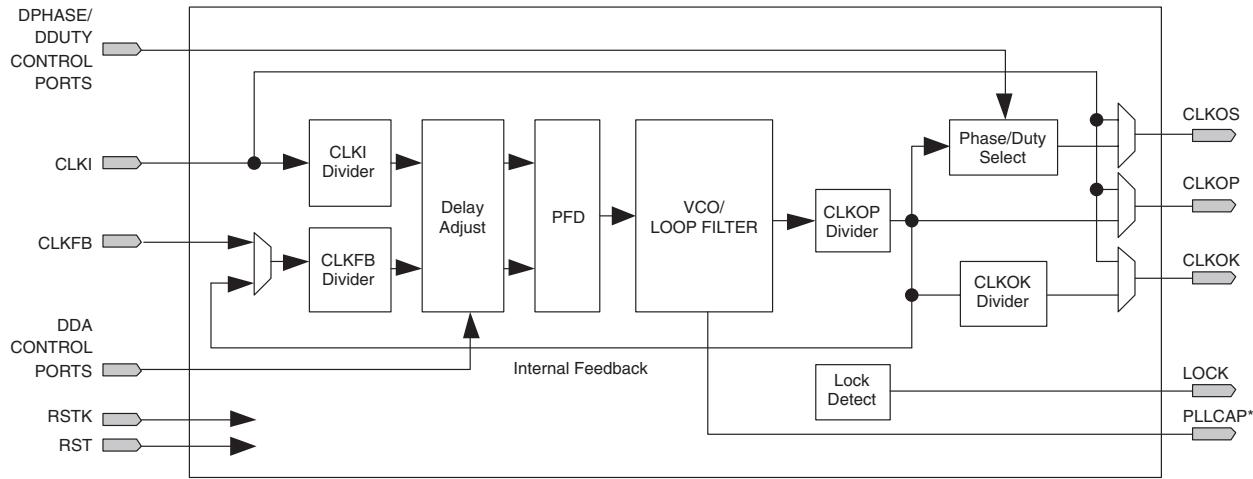
Edge Clock selection in the Pre-map Preference Editor will be available in ispLEVER® version 6.1.

Refer to Appendix A for detailed clock network diagrams.

## sysCLOCK PLL

The LatticeECP2/M PLL provides features such as clock injection delay removal, frequency synthesis, phase/duty cycle adjustment, and dynamic delay adjustment. Figure 10-6 shows the block diagram of the PLL.

**Figure 10-6. 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. 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 device data sheet.

#### Feedback Loop (CLKFB) Divider

The CLKFB divider is used to divide the feedback signal. 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 device data sheet.

#### Delay Adjustment

The delay adjust circuit provides programmable clock delay. The programmable clock delay allows for step delays in increments of 130ps (nominal) for a total of 1.04ns, lagging or leading. The time delay setting has a tolerance. See the device data sheet for details. Under this mode, CLKOP, CLKOS and CLKOK are identically affected. The delay adjustment has two modes of operation:

- **Static Delay Adjustment:** In this mode, the user-selected delay is configured at power-up. This feature will be supported in ipsLEVER version 6.1. Currently the delay is fixed to 0.
- **Dynamic Delay Adjustment (DDA):** In this mode, a simple bus is used to configure the delay. The bus signals are available to the general purpose FPGA.

#### 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 420MHz to 840MHz range to minimize jitter. The CLKOP Divider values are the same whether or not CLKOS is used.

#### CLKOK Divider

The CLKOK divider acts as a source for the global clock nets. It divides the CLKOP signal of the PLL by the value of the divider to produce a lower frequency clock.

### Phase Adjustment and Duty Cycle Select

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. However, 1/16th and 15/16th duty cycle options are not supported to avoid minimum pulse violation.

### Dynamic Phase Adjustment (DPHASE) and Dynamic Duty Cycle (DDUTY) Select

With LatticeECP2/M device families, users can control the Phase Adjustment and Duty Cycle Select 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, the other control inputs can be set with the fixed logic levels desired.

### External Capacitor

An optional external capacitor can be used with PLLs to accommodate low frequency input clocks. See the Optional External Capacitor section of this document for further information.

## PLL Inputs and Outputs

### CLKI Input

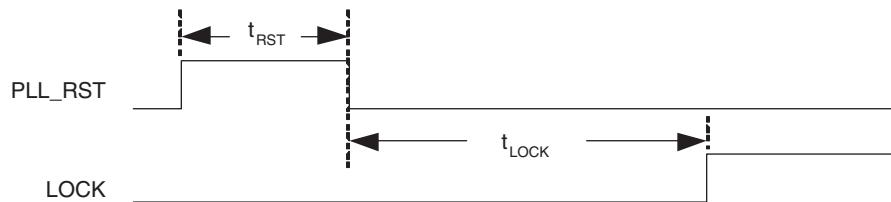
The CLKI signal is the reference clock for the PLL. It must conform to the specifications in the LatticeECP2/M Family Data Sheet for the PLL to operate correctly. The CLKI can be derived from a dedicated dual-purpose pin or from routing.

### RST Input

The PLL reset occurs under two conditions. At power-up an internal power-up reset signal from the configuration block resets the PLL. The user-controlled PLL reset signal RST is provided as part of the PLL module that can be driven by an internally generated reset function or a pin. This RST signal resets all internal PLL counters, flip-flops (including M-Dividers), and the charge pump. The M-Divider reset synchronizes the M-Divider output to the input clock. When RST goes inactive, the PLL will start the lock-in process, and will take the  $t_{LOCK}$  time to complete the PLL lock. Figure 10-7 shows the timing diagram of the RST Input. RST is active high.

The RESET signal is optional.

**Figure 10-7. RST Input Timing Diagram**



### RSTK Input

RSTK is the reset input for the K-Divider. The K-Divider reset is used to synchronize the K-Divider output clock to the input clock. The LatticeECP2/M 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.

### 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 (internal feedback) or from general routing. External feedback allows the designer to compensate for board-level clock alignment.

### CLKOP Output

The sysCLOCK PLL main clock output, CLKOP, is a signal available for selection as a primary clock and an edge clock. This clock signal is available at the CLK\_OUT pin.

**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 and an edge clock. The CLKOS is used when phase shift and/or duty cycle adjustment is desired. The programmable phase shift allows for different phases in increments of 22.5°. The duty select feature provides duty selection 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. This signal is available for selection as a primary clock.

**Dynamic Delay Control/Dynamic Phase Adjustment/Dynamic Duty Cycle**

Detailed information about these features are described later in this document. The I/O ports for these features are illustrated in Table 10-3.

**Table 10-3. Dynamic Delay Adjust and Dynamic Phase and Duty Cycle Adjust Ports**

Parameter	I/O	Description
DDAMODE	I	DDA (Dynamic Delay Adjust) Mode. "1": Pin control (dynamic), "0": Fuse Control (static)
DDAIZR	I	DDA Delay Zero. "1": delay = 0, "0": delay = on
DDAILAG	I	DDA Lag/Lead. "1": Lead, "0": Lag
DDAIDEL[2:0]	I	DDA Delay Step value
DPAMODE	I	DPA (Dynamic Phase Adjust/Duty Cycle Select) mode
DPHASE[3:0]	I	DPA Phase Adjust inputs
DDUTY[3:0]	I	DPA Duty Cycle Select inputs

**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. 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.

**PLLCAP**

This port is not included in the software module. Instead, it is hard-wired to the PLLCAP pin of the device. See the Optional External Capacitor section of this document for further information.

**PLL Attributes**

The PLL utilizes several attributes that allow the configuration of the PLL through source constraints and preference files. The following section details these attributes and their usage.

**FIN**

The input frequency can be any value within the specified frequency range based on the divider settings.

**CLKI\_DIV, CLKFB\_DIV, CLKOP\_DIV, CLKOK\_DIV**

These dividers determine the output frequencies of each output clock. The user is not allowed to input an invalid combination. This is determined by the input frequency, the dividers and the PLL specifications.

Note: Unlike PLLs in the LatticeECP™, LatticeEC™, LatticeXP™ and MacroXO™ devices, the CLKOP Divider values are the same whether or not CLKOS is used. The CLKOP\_DIV value is calculated to maximize the  $f_{VCO}$  within the specified range based on FIN and CLKOP\_FREQ in conjunction with CLKI\_DIV and CLKFB\_DIV values. These value settings are designed so that the output clock duty cycle is as close to 50% as possible.

**FREQUENCY\_PIN\_CLKI, FREQUENCY\_PIN\_CLKOP, FREQUENCY\_PIN\_CLKOK**

These input and output clock frequencies determine the divider values.

**CLKOP Frequency Tolerance**

When the desired output frequency is not achievable, the frequency tolerance of the clock output may be entered.

**FDEL (Fine Delay Adjust: EHXPPLL only)**

This feature will be supported in isplLEVER 6.1. Currently, the static delay is fixed to 0.

**PHASEADJ (Phase Shift Adjust)**

The PHASEADJ attribute is used to select Phase Shift for the CLKOS output. The phase adjustment is programmable in 22.5° increments.

**DUTY (Duty Cycle)**

The DUTY attribute is used to select the Duty Cycle for CLKOS output. The Duty Cycle is programmable at 1/16th of the period increment. Steps 2 to 14 are supported. 1/16th and 15/16th duty cycles are not supported to avoid the minimum pulse width violation.

**FB\_MODE**

There are three sources of feedback signals that can drive the CLKFB Divider: Internal, CLKOP (Clock Tree) and User Clock. CLKOP (Clock Tree) feedback is used by default. Internal feedback takes the CLKOP output at the CLKOP Divider output (CLKINTFB) before the Clock Tree to minimize the feedback path delay. User Clock feedback is driven from the dedicated pin, clock pin or user specified internal logic.

**DELAY\_CNTL**

This attribute is designed to select the Delay Adjustment mode. If the attribute is set to "DYNAMIC" the delay control switches between dynamic and static, depending upon the input logic of the DDAMODE pin. If the attribute is set to "STATIC", Dynamic Delay inputs are ignored in this mode.

**PHASE/DUTY\_CNTL**

This attribute is designed to select the Phase Adjustment/Duty Cycle Select mode. If the attribute is set to "DYNAMIC" the Phase Adjustment/Duty Cycle Select control switches between dynamic and static, depending upon the input logic of the DPAMODE pin. If the attribute is set to "STATIC", Dynamic Phase Adjustment/Duty Cycle Select inputs are ignored in this mode.

**CLKOS/CLKOK Select**

Users select these output clocks only when they are used in the design.

**CLKOP/CLKOS/CLKOK BYPASS**

These bypasses are enabled if set. The CLKI is routed directly to the corresponding output clock.

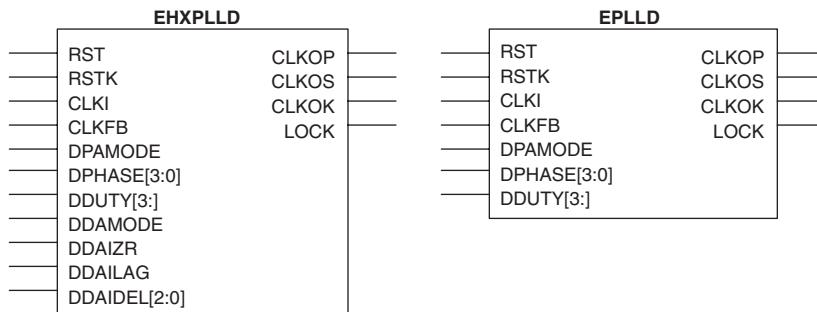
**RESET/RSTK Select**

Users select these reset signals only when they are used in the design.

**LatticeECP2/M PLL Primitive Definitions**

All LatticeECP2/M devices support two General Purpose PLLs (GPLLs) which are full-featured PLLs. In addition, some of the larger devices have two to four Standard PLLs (SPLLs) that have a subset of the GPLL functionalities.

Two PLL primitives are defined for LatticeECP2/M PLL implementation. Figure 10-8 shows the LatticeECP2/M PLL primitive library symbols. The GPLL may be configured as either EPLL or EHXPPLL. The SPLL can be configured as EPLL only.

**Figure 10-8. LatticeECP2/M PLL Primitive Symbols**

### **Dynamic Delay Adjustment (EHPLL Only)**

The Dynamic Delay Adjustment is controlled by the DDAMODE input. When the DDAMODE input is set to “1”, the delay control is done through the inputs, DDAIZR, DDAILAG and DDAIDEL(2:0). For this mode, the attribute “DELAY\_CNTL” must be set to “DYNAMIC”. Table 10-4 shows the delay adjustment values based on the attribute/input settings.

In this mode, the PLL may come out of lock due to the abrupt change of phase. RST must be asserted to re-lock the PLL. Upon de-assertion of RST, the PLL will start the lock-in process and will take the  $t_{LOCK}$  time to complete the PLL lock.

**Table 10-4. Delay Adjustment**

DDAMODE = 1: Dynamic Delay Adjustment			Delay 1 Tdly = 130 ps (nominal)	DDAMODE = 0
DDAIZR	DDAILAG	DDAIDEL[2:0]		Equivalent FDEL Value
0	1	111	Lead 8 Tdly	-8
0	1	110	Lead 7 Tdly	-7
0	1	101	Lead 6 Tdly	-6
0	1	100	Lead 5 Tdly	-5
0	1	011	Lead 4 Tdly	-4
0	1	010	Lead 3 Tdly	-3
0	1	001	Lead 2 Tdly	-2
0	1	000	Lead 1 Tdly	-1
1	Don't Care	Don't Care	no delay	0
0	0	000	Lag 1 Tdly	1
0	0	001	Lag 2 Tdly	2
0	0	010	Lag 3 Tdly	3
0	0	011	Lag 4 Tdly	4
0	0	100	Lag 5 Tdly	5
0	0	101	Lag 6 Tdly	6
0	0	110	Lag 7 Tdly	7
0	0	111	Lag 8 Tdly	8

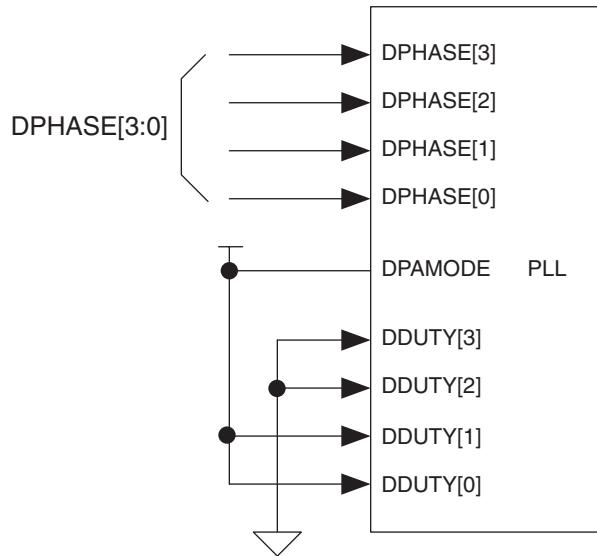
## Dynamic Phase/Duty Mode

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”.

To use Dynamic Phase Adjustment with a fixed Duty Cycle, simply set the DDUTY[3:0] inputs to the desired Duty Cycle value. Figure 10-9 illustrates an example circuit. This example assumes the user-desired Duty Cycle is 3/16.

**Figure 10-9. Example Dynamic Phase Adjustment Set-up with Duty Cycle Fixed to 3/16**



## Dynamic Phase Adjustment/Duty Cycle Select

Phase Adjustment settings are described in Table 10-5.

**Table 10-5. Dynamic Phase Adjustment Settings**

DPHASE[3:0]	Equivalent to PHASEADJ in Static Mode
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

Duty Cycle Select settings are described in Table 10-6.

**Table 10-6. Dynamic Duty Cycle Select Settings**

DDUTY[3:0]	Equivalent to DUTY in Static Mode (1/16 of 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

### Optional External Capacitor

An optional external capacitor can be used with both the EXHPLLD and the EPLL to change the frequency response of the on-chip loop filter. When an external capacitor is used, the frequency at the phase detector inputs (Fpd) can be as low as 1MHz, allowing the PLLs to extend the low-end of their operating ranges. The external capacitor has no effect on the high end of their operating ranges. IPexpress™ checks the phase detector frequency to determine if an external capacitor is required.

The allowable ranges for the PLL parameters with and without the external capacitor are described in the LatticeECP2/M Family Data Sheet.

#### Recommended Optional External Capacitor Specifications

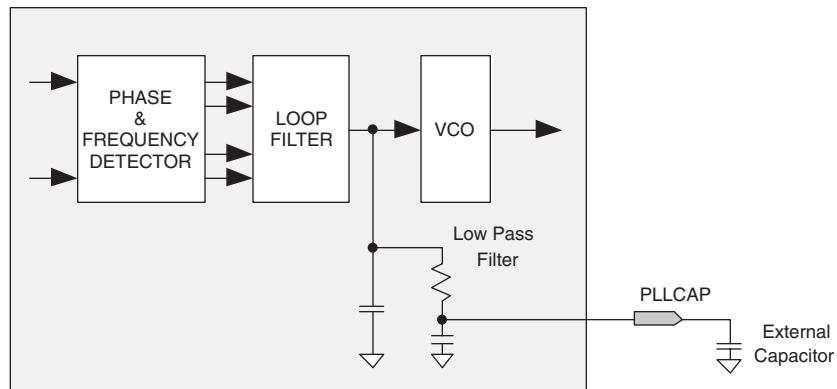
Value: 5.6 nF, +/- 20%

Type: Ceramic chip capacitor, NPO dielectric

Package: 1206 or smaller

Each device has two external capacitor pins, one for the left side PLLs and one for the right side PLLs. These pins are in fixed locations. They are dedicated function pins that are NOT shared with user I/Os.

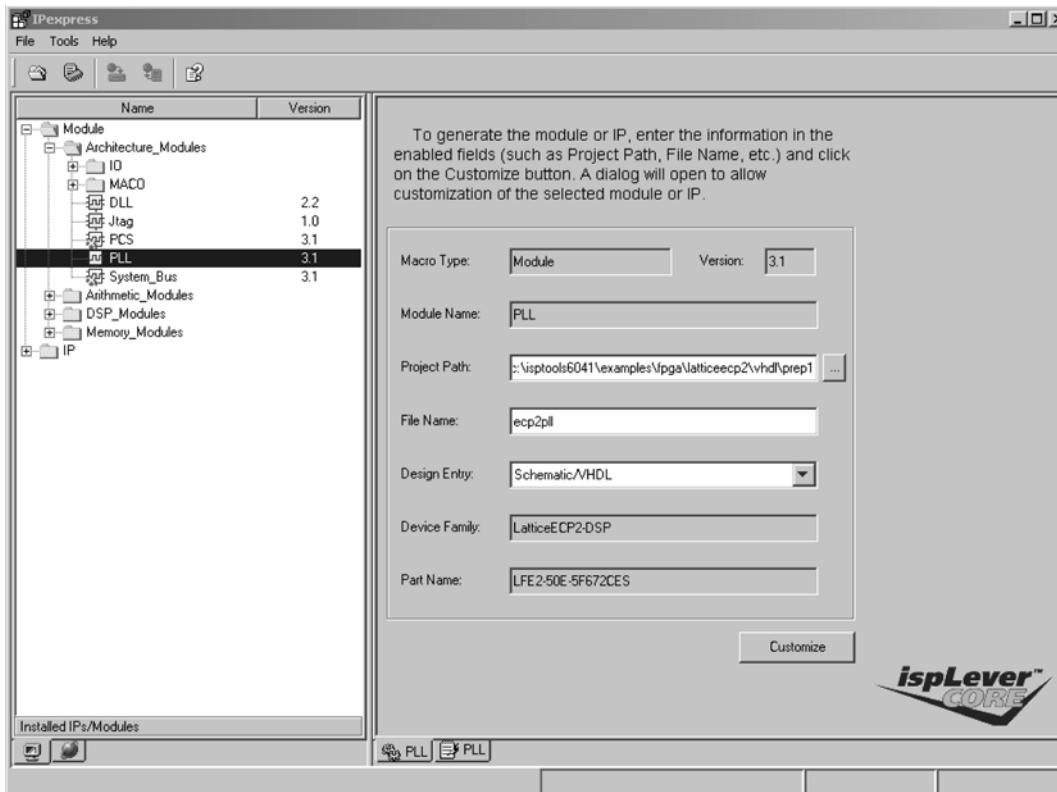
When an external capacitor pin is used by a PLL on one side of the device, it cannot be used by any other PLLs on the same side of the device. This means that a maximum of two PLLs per device, one on the left side and one on the right side, can have external capacitors attached.

**Figure 10-10. External Capacitor Usage**

## PLL Usage in IPexpress

IPexpress is used to create and configure a PLL. Designers use the graphical user interface to select parameters for the PLL. The result is an HDL model to be used in the simulation and synthesis flow.

Figure 10-11 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. These entries may be changed if desired. After entering the module name, click on **Customize** to open the **Configuration Tab** window as shown in Figure 10-12.

**Figure 10-11. IPexpress Main Window**

## 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.

## Modes

There are two modes for configuring the PLL in the Configuration Tab: Frequency Mode and Divider Mode.

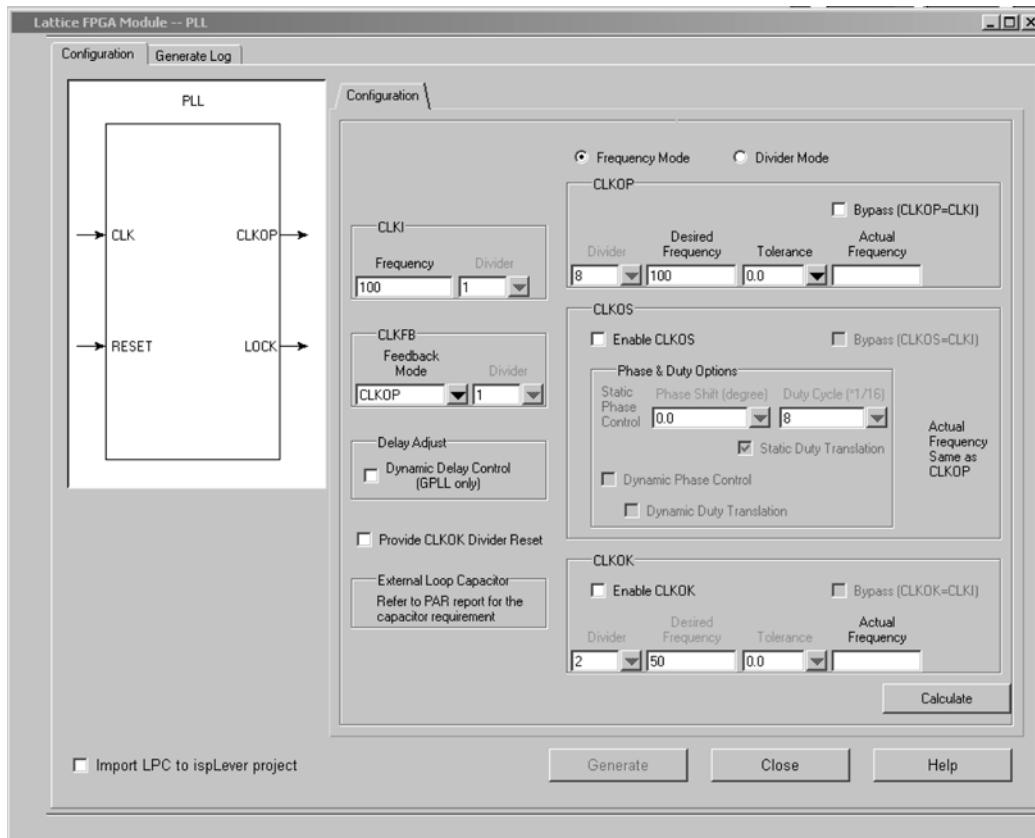
### Frequency Mode

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 the input frequency. The user must choose the CLKOP Divider value in order to maximize the  $f_{VCO}$  and achieve optimum PLL performance. After setting the input frequency and divider settings, click **Calculate** to display the frequencies. Figure 10-12 shows the Configuration Tab.

**Figure 10-12. LatticeECP2/M PLL Configuration Tab**



Note: In the External Loop Capacitor, the grayed out text turns on if the CLKI frequency is less than 25MHz to alert the user about the required external loop capacitor. Other grayed-out sections turn on as their sections are enabled.

Table 10-8 describes all user parameters in the IPexpress GUI.

**Table 10-7. User Parameters in the Configuration GUI**

User Parameters		Description	Range	Default
Frequency Mode		Desired input/output frequency	ON/OFF	ON
Divider Mode		Desired input frequency and divider settings	ON/OFF	OFF
CLKI	Frequency	Without external capacitor	25 (33 <sup>1</sup> ) MHz to 420 MHz	100 MHz
		With external capacitor	2 MHz to 420 MHz	—
	Divider	Without external capacitor	1 to 16 (12 <sup>1</sup> )	1
		With external capacitor	1 to 64	—
CLKFB	Feedback Mode	Feedback Mode	Internal, CLKOP, User Clock	CLKOP
	Divider	Without external capacitor	1 to 16 (12 <sup>1</sup> )	1
		With external capacitor	1 to 10	—
Dynamic Delay Control		Dynamic/Static Delay Select	ON/OFF	OFF
Provide CLKOK Divide Reset		Provide CLKOK Reset Port	ON/OFF	OFF
CLKOP	Bypass	Bypass PLL: CLKOP = CLKI	ON/OFF	OFF
	Desired Frequency	Without external capacitor	25 (33 <sup>1</sup> ) MHz to 420 MHz	100 MHz
		With external capacitor	5 MHz to 50 MHz	—
	Divider	CLKOP Divider Setting (Divider Mode)	2, 4, 8, 16, 32, 48, 64, 80, 96, 112, 128	8
	Tolerance	CLKOP tolerance users can tolerate	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0
CLKOS	Actual Frequency	Actual frequency achievable. Read only.	—	—
	Enable	Enable CLKOS output clock	ON/OFF	OFF
	Bypass	Bypass PLL: CLKOS = CLKI	ON/OFF	OFF
	Phase Shift	CLKOS Static Phase Shift	0°, 22.5°, 45°..337.5°	0°
	Duty Cycle	Duty Cycle Select	2 to 14	8
	Dynamic Phase Control	Dynamic/Static Phase and Dynamic/Static Duty Select	ON/OFF	OFF
CLKOK	Dynamic Duty Translation	This feature is not used		
	Enable	Enable CLKOK output clock	ON/OFF	OFF
	Bypass	Bypass PLL: CLKOK = CLKI	ON/OFF	OFF
	Desired Frequency	Without external capacitor	0.195 MHz to 210 MHz	50 MHz
		With external capacitor	0.016 MHz to 25 MHz	—
	Divider	CLKOK Divider Setting	2 to 128	2
CLKOK2	Tolerance	CLKOK tolerance users can tolerate	0.0, 0.1, 0.2, 0.5, 0.1, 0.2, 0.5, 1.0	0.00
	Actual Frequency	Actual frequency achievable. Read only.	—	—
CLKOK2	Enable	Enable CLKOK2 output clock	ON/OFF	OFF
Import LPC to ispLEVER project		Import .lpc file to ispLEVER project.	ON/OFF	OFF

1. These values apply to SPLL. All other values apply to both GPLL and SPLL.

## Frequency Calculation

Table 10-8 illustrates the Frequency limits at Phase Detector inputs. Users must select CLKI Divider and CLKFB Divider values so that the Phase Detector Frequency falls within the range.

Let    M = CLKI divider value  
       N = CLKFB divider value  
       V = CLKOP divider value

The basic equations are:

$$\begin{aligned} \text{CLKOP Frequency} &= \text{CLKI Frequency} * N/M \\ f_{\text{VCO}} (\text{VCO Frequency}) &= \text{CLKOP Frequency} * V \\ f_{\text{PFD}} (\text{PFD Frequency}) &= \text{CLKI Frequency} / M = \text{CLKFB Frequency} (= \text{CLKOP Frequency}) / N \end{aligned}$$

Example: If CLKI frequency is 25 MHz without external capacitor, the CLKI divider value can be only 1.

## DLL Primitive I/Os

**Table 10-8. Phase Detector Frequency ( $f_{\text{PFD}}$ ) Range**

PLL Type	External Capacitor	Frequency Range
GPLL	Without external capacitor	25 MHz to 420 MHz
	With external capacitor	2 MHz to 50 MHz
SPLL	Without external capacitor	33 MHz to 420 MHz
	With external capacitor	2 MHz to 50 MHz

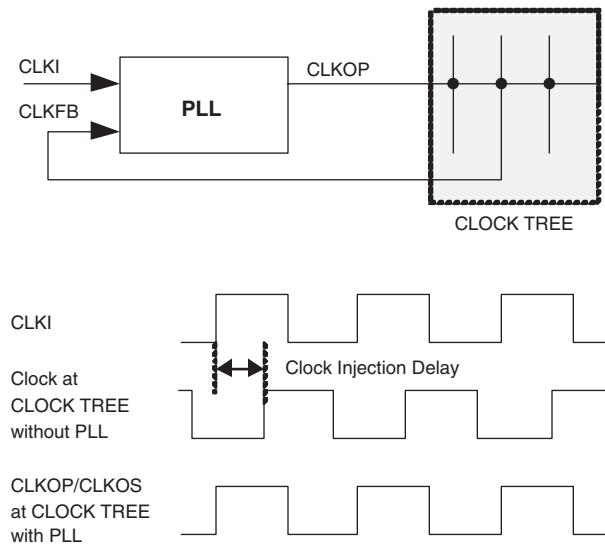
## PLL Modes of Operation

PLLs have many uses within 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 PLLs are 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 10-13 illustrates an example block diagram and waveform.

**Figure 10-13. Clock Injection Delay Removal Application**

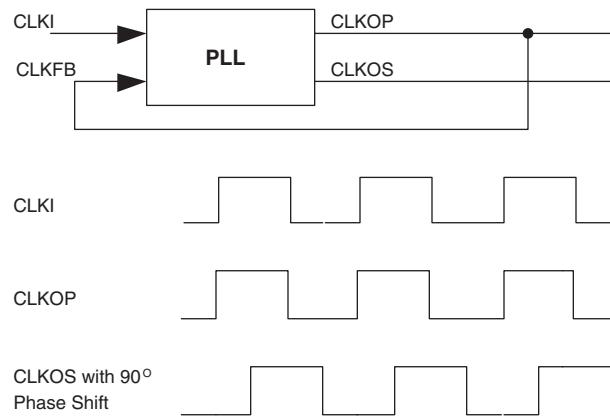


### PLL Clock Phase Adjustment

In this mode, the PLLs are used to create fixed phase relationships in 22.5° increments. Creating fixed phase relationships are useful for forward clock interfaces where a specific relationship between the clock and data is required.

The fixed phase relationship can be used between CLKI and CLKOS or between CLKOP and CLKOS.

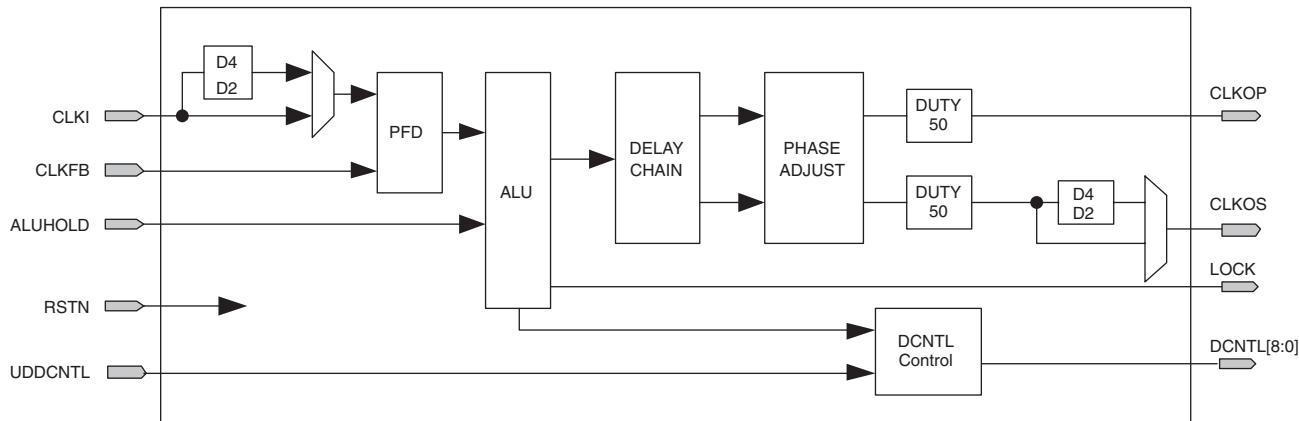
**Figure 10-14. CLKOS Phase Adjustment from CLKOP**



## sysCLOCK DLL

The LatticeECP2/M DLL provides features such as clock injection delay removal, delay match, 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[8:0]). The delay control vector is connected to a Slave Delay Line (DLLDEL) element located in the I/O logic which matches the delay cells in the DLL. This delay vector allows the DLL to dynamically delay an input signal by a specific amount. Figure 10-15 provides a block diagram of the LatticeECP2/M DLL.

**Figure 10-15. LatticeECP2/M DLL Block Diagram**



Both clock injection delay removal and output phase adjustment use only the clock outputs of the DLL. Time reference delay and delay match modes use the delay control vector output. Specific examples of these features are discussed later in this document.

## DLL Overview

The LatticeECP2/M DLL is created and configured by IPexpress. The following is a list of port names and descriptions for the DLL. There are three library elements used to implement the DLL: CIDDLA (Clock Injection Delay), CIMDLLA (Clock Injection delay Match), and TRDDLLA (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 or in clock delay match mode. 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.

In Clock Injection Delay Match mode, the DLL will calculate the delta between the CLKI and CLKFB signals. This delay value is then output on the DCNTL vector. The DLL CLKFB input has a preferred pin per DLL which is discussed later in this document. The preferred pin provides the lowest latency and best case performance.

### 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[8:0] Output

This output of the DLL is used to delay a signal by a specific amount. The DCNTL[8:0] vector connects to a Slave Delay Line element. The DLL can then control multiple input delays from a single DLL.

### UDDCNTL Input

This input is used to enable or disable updating of the DCNTL[8:0]. 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[8:0] outputs are continuously updated.

### ALUHOLD Input

This active high input stops the DLL from adding and subtracting delays to the CLKI signal. The DCNTL[8:0], 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. The clock is stopped so there is no clock to de-assert 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.

### 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.

## DLL Attributes

The LatticeECP2/M 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

Usually, the DLL is a ‘times one’ device, allowing neither frequency multiplication or division. But the LatticeECP2/M 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.

#### Optional Clock Fine Phase Shift

The optional fine phase shift in the CLKOS output path is built from a delay block that matches the other four blocks in the main delay chain. This delay block allows the CLKOS output to phase shifted a further 45, 22.5 or 11.25 degrees relative to its normal position.

#### Duty Cycle Selection

TRDLLA has an optional 50% Duty Cycle Selection feature. The inverted output of the DUTY50 block provides another 180-degree phase shift from the source.

CIDDLA also has optional 50% Duty Cycle Selection feature.

#### GSR

The PLL and DLL can be reset by the GSR, if enabled. The GSR keyword can be set to ENABLED/DISABLED. This option is provided in the IPexpress GUI. Below is an example of the use of this preference.

```
ASIC "d11/d11_0_0" TYPE "EHXPLLA" GSR=DISABLED;
```

#### 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 100 $\mu$ s (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  $\mu$ s until the lock output goes high. Below is an example of how to set this value for front-end simulation. This option is also available in the IPexpress GUI.

#### Verilog:

```
defparam mydll.mypll_0_0.LOCK_DELAY=500;
mydll dll_inst(.CLKI(clkin), .CLKOP(clk1), .CLKOS(clk2),
```

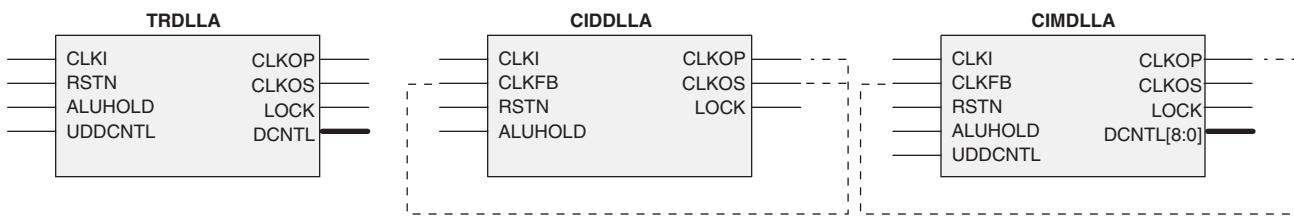
#### VHDL:

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 "EHXPLLA" LOCK_DELAY=200;
```

#### DLL Primitive Symbols

**Figure 10-16. DLL Primitive Symbols**



#### DLL Primitives Definitions

The Lattice library contains primitives to allow designers to utilize the DLL. These primitives use the DLL attributes defined in the “DLL Attributes” section.

The three modes of operation are presented as primitives (or library elements) as listed below.

**Table 10-9. DLL Primitives**

Primitive Name	Mode of Operation	Description
TRDLLA	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.
CIDDLLA	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.
CIMDLA	Clock Injection Match DLL	This mode matches the delay between the input clock and the feedback clock to one delay cell. This allows other inputs to take the registered ALU outputs and adjust their delay to match the clock-to-clock delay.

## DLL Primitive I/Os

**Table 10-10. DLL Primitive I/O Descriptions**

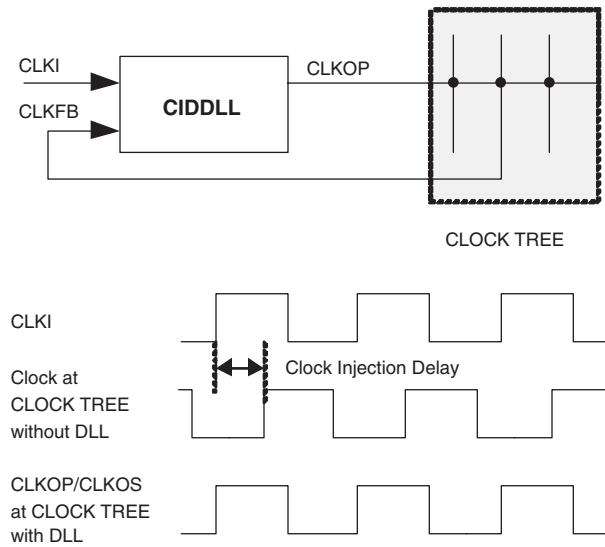
Signal	I/O	Description
CLKI	I	Clock input pin from dedicated clock input pin, other I/O or logic block.
CLKFB	I	Clock feedback input pin from dedicated feedback input pin, internal feedback, other I/O or logic block. This signal is not user selectable.
RSTN	I	Active low synchronous reset. From dedicated pin or internal node.
ALUHOLD	I	“1” freezes the ALU. For TRDLLA, CIDDLLA and CIMDLA.
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[8:0]	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 device data sheet for frequency specifications.

## DLL Modes of Operation

### Clock Injection Removal Mode (CIDDLLA)

The DLL can be used to reduce clock injection delay (CIDDLLA). 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 10-17 shows a circuit example and waveform.

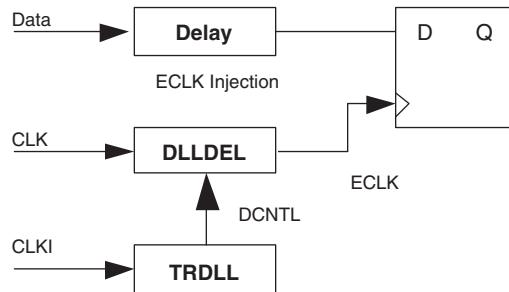
**Figure 10-17. Clock Injection Delay Removal via DLL**

Clock injection removal mode can also provide a DCNTL port. In this mode, the delay added to the CLKI signal is output on the DCNTL port so that other input signals can be delayed by the same amount. This is very useful if several clocks are used in the same circuit to minimize the number of DLLs required. 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 700MHz.

#### **Time Reference Delay Mode (TRDLLA: 90-Degree Phase Delay)**

The Time Reference Delay (TRDDLLA) 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 10-18 provides a circuit example of this mode.

**Figure 10-18. 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-DELA) to delay the signal by 90 degrees of the full period of CLKI.

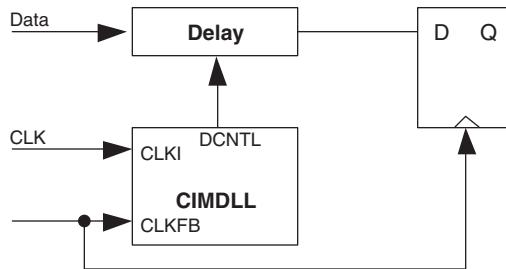
#### **Clock Delay Match Mode (CIMDLLA)**

CIMDLLA provides two delay match modes, Clock-to-Clock Delay Match mode and Clock Injection Delay Match mode.

The Clock-to-Clock Delay Match (CIMDLLA) mode of the DLL allows two synchronous clock inputs to be used to create a DCNTL vector that is the delta of the delay between the two clocks. In this circuit, the DLL will compare the phase difference between the CLKI and CLKFB inputs to create a delay vector. This is a useful mode when data is

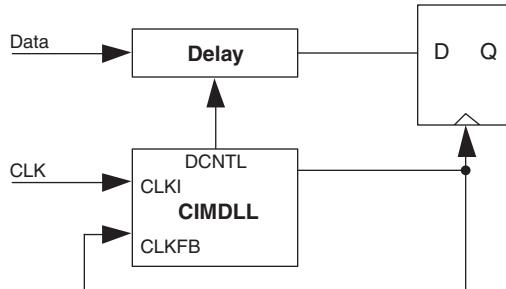
transmitted off one clock (CLKI) and captured on a different clock (CLKFB) that are synchronous to each other. By delaying the data inside the I/O logic before being latched by CLKFB, the data signal will be delayed the same amount as the difference in the clock delay. Figure 10-19 provides a circuit example of this mode.

**Figure 10-19. Clock-to-Clock Delay Match Circuit Example**



The Clock Injection Delay Match mode of the DLL allows the user to match the clock injection delay by routing the CLKOP to CLKFB input. Figure 10-20 shows a circuit example of this mode.

**Figure 10-20. Clock Injection Delay Match Circuit Example**



In Clock Delay Match mode the maximum delay is limited to 3.9ns.

## DLL Usage in IPexpress

IPexpress is used to create and configure a DLL. The IPexpress graphical user interface allows users to select parameters for the DLL. The result is an HDL model to be used in the simulation and synthesis flow.

### Configuration Tab

- **Usage Mode** – Select the mode of the DLL (Time Reference Delay [TRDLLA], Clock Injection Delay Removal [CIDDLLA], or Clock Delay Match [CIMDLLA]). This selection will enable or disable further options in the GUI.
- **CLKI Frequency** – The rate of the CLKI input in MHz.
- **CLKOS Divider** – Set the divider for the CLKOS output to be either no divide, divide by 2, or divide by 4.
- **CLKOS Phase Shift** – Set the phase offset of the CLKOS to the CLKOP output. CLKOP will lead CLKOS by the amount of phase shift selected. The phase increment is 11.25 degrees. The pull-down list numbers are abbreviated to a decimal point.
- **CLKFB Feedback Mode** – Sets the feedback mode of the DLL to either CLKOP, CLKOS or User Clock. If CLKOP/CLKOS is selected, the clock tree injection delay for the specific output clock will be removed. If User Clock is selected, the user will be provided with the CLKFB port on the DLL.
- **CLKFB Frequency** – This is used only with User Clock Feedback Mode.
- **Provide RSTN Port** – The RSTN port allows the user to reset the DLL through a user signal.
- **Enable GSR to Reset DLL** – If selected, the DLL will be reset via the GSR. No user signal is required. RSTN port can still be used.

- **Provide DCNTL Port** – In Clock Injection Delay Mode it is possible to obtain the delay added to the CLKI port on the DCNTL port. This can then be used to delay other clock inputs by the same amount by connecting the DCNTL vector to DELAY elements.

### 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.

### 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.

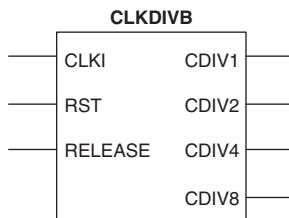
## Clock Dividers (CLKDIV)

The clock divider divides the high-speed clock by 2, 4 and 8. The divided output can then be used as a primary clock or secondary clock input. 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. There are two clock dividers, one on each side.

### CLKDIV Primitive Definition

Users can instantiate CLKDIV in the source code as defined in this section. Figure 10-21, Table 10-11 and Table 10-12 describe the definition of CLKDIVB.

**Figure 10-21. CLKDIV Primitive Symbol**



**Table 10-11. CLKDIVB Port Definitions**

Name	Description
CLKI	Clock input
RST <sup>1</sup>	Reset input, asynchronously forces all outputs low.
RELEASE <sup>1</sup>	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

1. Note: Unused RST must be tied to ground. Unused RELEASE must be tied to V<sub>CC</sub>.

**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
```

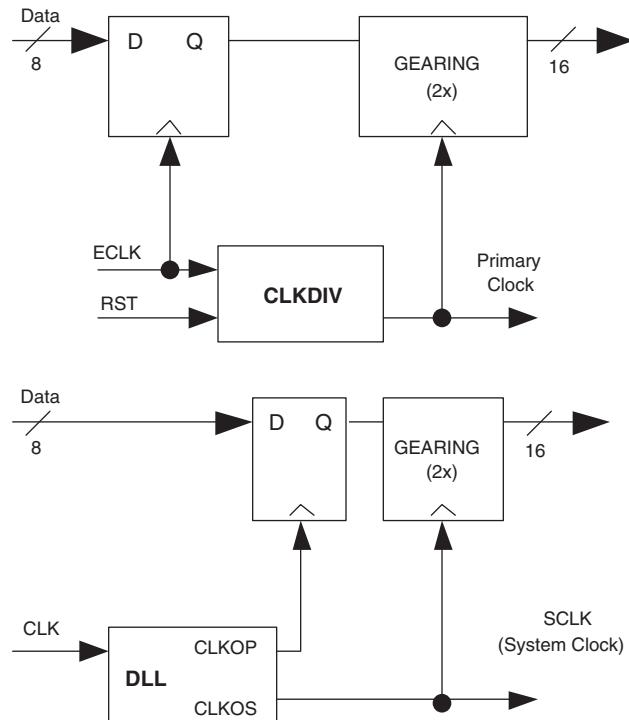
## CLKDIV Example Circuits

The clock divider (CLKDIV) can divide a clock by 2 or 4 and drives a primary clock network. The 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 come from an edge clock and the output drives a primary clock. In this case, they are phase matched. This is especially useful for synchronously resetting the I/O logic when Mux/DeMux gearing is used in order to synchronize the entire data bus as shown in Figure 10-22. 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.

The second circuit shows that a DLL can replace CLKDIV for x2 and x4 applications.

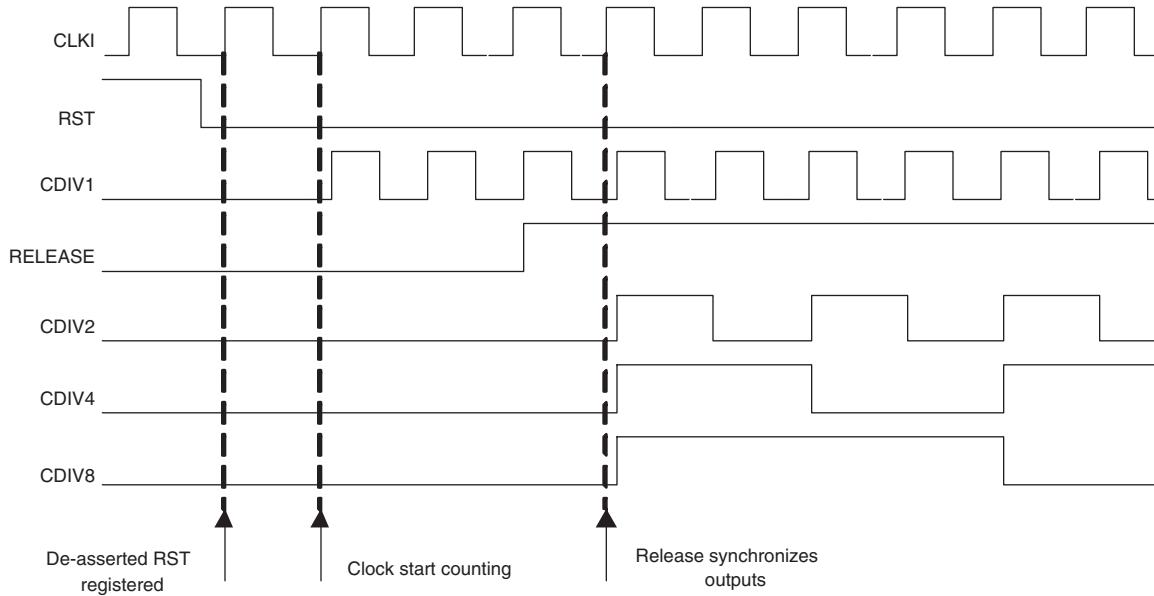
**Figure 10-22. CLKDIV Application Examples**



## Release Behavior

The port “Release” is used to synchronize all outputs after RST is de-asserted. Figure 10-23 illustrates the Release behavior.

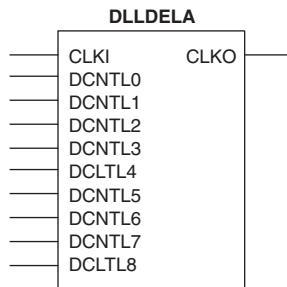
**Figure 10-23. CLKDIV Release Behavior**



## DLLDEL (Slave Delay Line)

The Slave Delay line is designed to generate the desired delay in DDR/SPI4 applications. The delay control inputs (DCNTL[8:0]) are fed from the general purpose DLL outputs. The primitive definitions are described in Figure 10-24 and Table 10-13.

**Figure 10-24. DLLDELA Primitive Symbol**



**Table 10-13. DLLDELA I/O**

Name	I/O	Description
CLKI	I	Clock Input
DCNTL[8:0]	I	Delay Control Bits
CLKO	O	Clock Output

**DLLDELA Declaration in VHDL Source Code**

```

COMPONENT DLLDELA
    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;
        DCNTL6 :IN std_logic;
        DCNTL7 :IN std_logic;
        DCNTL8 :IN std_logic;
        CLKO :OUT std_logic
    );
END COMPONENT;

begin
    DLLDELAinst0: DLLDELA1
        PORT MAP (
            CLKI => clkisig,
            DCNTL0 => dcntl0sig,
            DCNTL1 => dcntl1sig,
            DCNTL2 => dcntl2sig,
            DCNTL3 => dcntl3sig,
            DCNTL4 => dcntl4sig,
            DCNTL5 => dcntl5sig,
            DCNTL6 => dcntl6sig,
            DCNTL7 => dcntl7sig,
            DCNTL8 => dcntl8sig,
            CLKO => clkosig
        );

```

**DLLDELA Usage with TRDLLA - Verilog - Example**

Note: DLL0(TRDLLA) must be generated by IPexpress as a sub-module

```

module ddldel_top (rst,d,clkin,clkin2,clkout,aluhold,uddcntl,q);

    input rst,d,clkin,clkin2,aluhold,uddcntl;
    output clkout,q;

    wire [8:0]DCntl_int;
    reg qint;

    DLL0    dllinst0  (.clk(clkin2),   .aluhold(aluhold),   .uddcntl(uddcntl),   .clkop(),
    .clkos(),
    .dcntl(DCndl_int),.lock());
    DLLDELA delinst0 (.CLKI(clkin),.DCNTL0(DCndl_int[0]),.DCNTL1(DCndl_int[1]),
    .DCNTL2(DCndl_int[2]), .DCNTL3(DCndl_int[3]), .DCNTL4(DCndl_int[4]),
    .DCNTL5(DCndl_int[5]), .DCNTL6(DCndl_int[6]), .DCNTL7(DCndl_int[7]),
    .DCNTL8(DCndl_int[8]), .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;

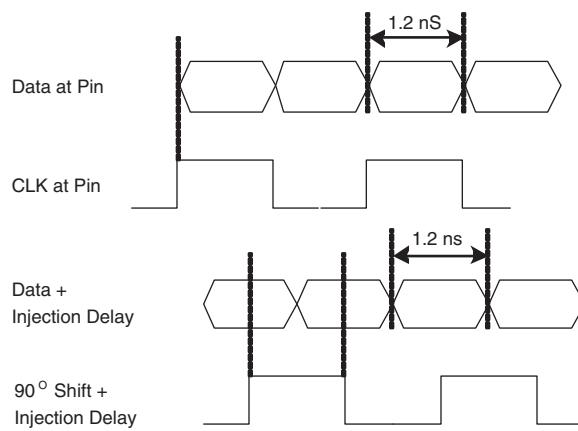
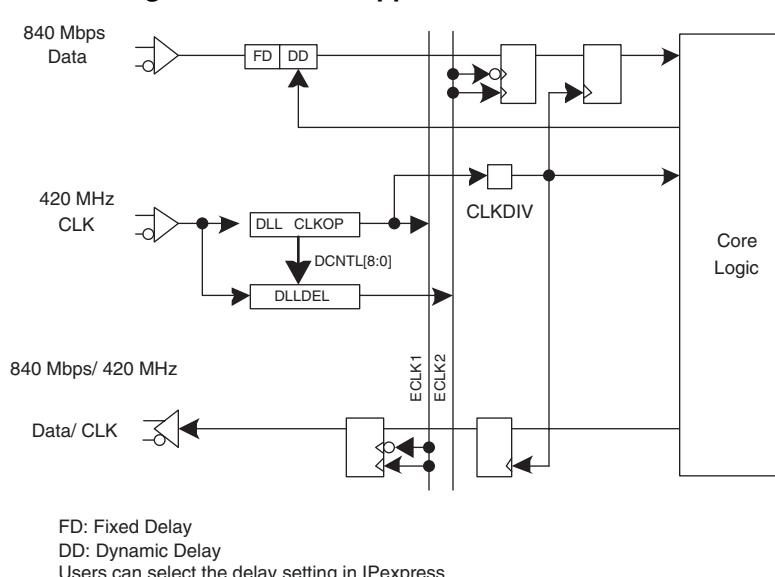
endmodule

```

**DLLDELA Application Example**

Figure 10-25 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 10-25. SPI4.2 and DDR Registers Interface Application**

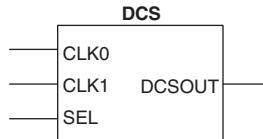
**DQS DLL and DQS DEL**

There is another combination of DLL and Slave Delay Line, DQS DLL and DQS DEL, in the LatticeECP2/M 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 Lattice Technical Note, TN1102, *LatticeECP2/M sysIO Usage Guide*.

## 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. The outputs of the DCS then reach primary clock distribution via the feed-lines. Figure 10-26 shows the block diagram of the DCS.

**Figure 10-26. DCS Primitive Symbol**



### DCS Primitive Definition

Table 10-14 defines the I/O ports of the DCS block. There are eight modes available. Table 10-15 describes how each mode is configured.

**Table 10-14. DCS I/O Definition**

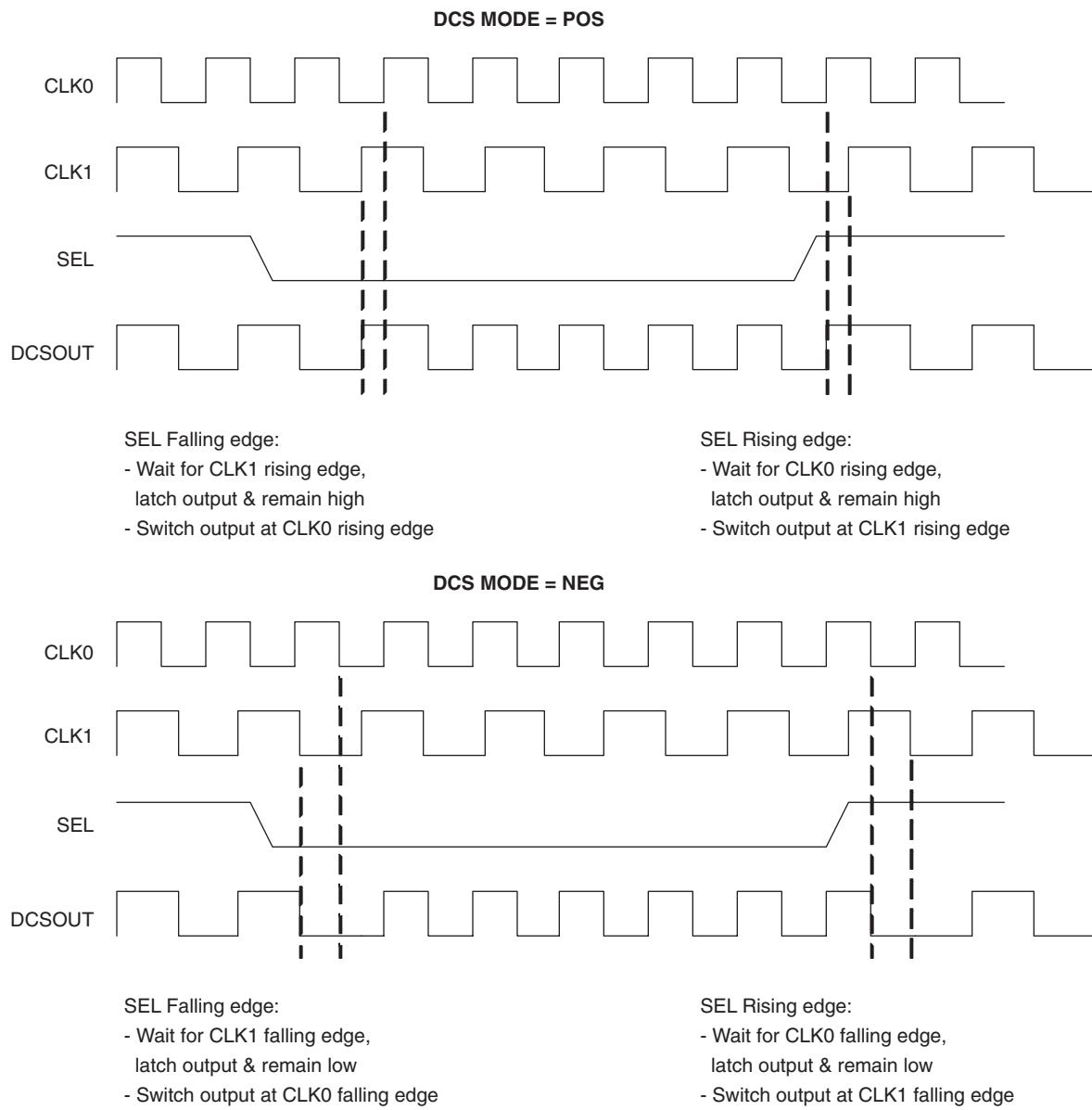
I/O	Name	Description
Input	SEL	Input Clock Select
	CLK0	Clock input 0
	CLK1	Clock Input 1
Output	DCSOUT	Clock Output

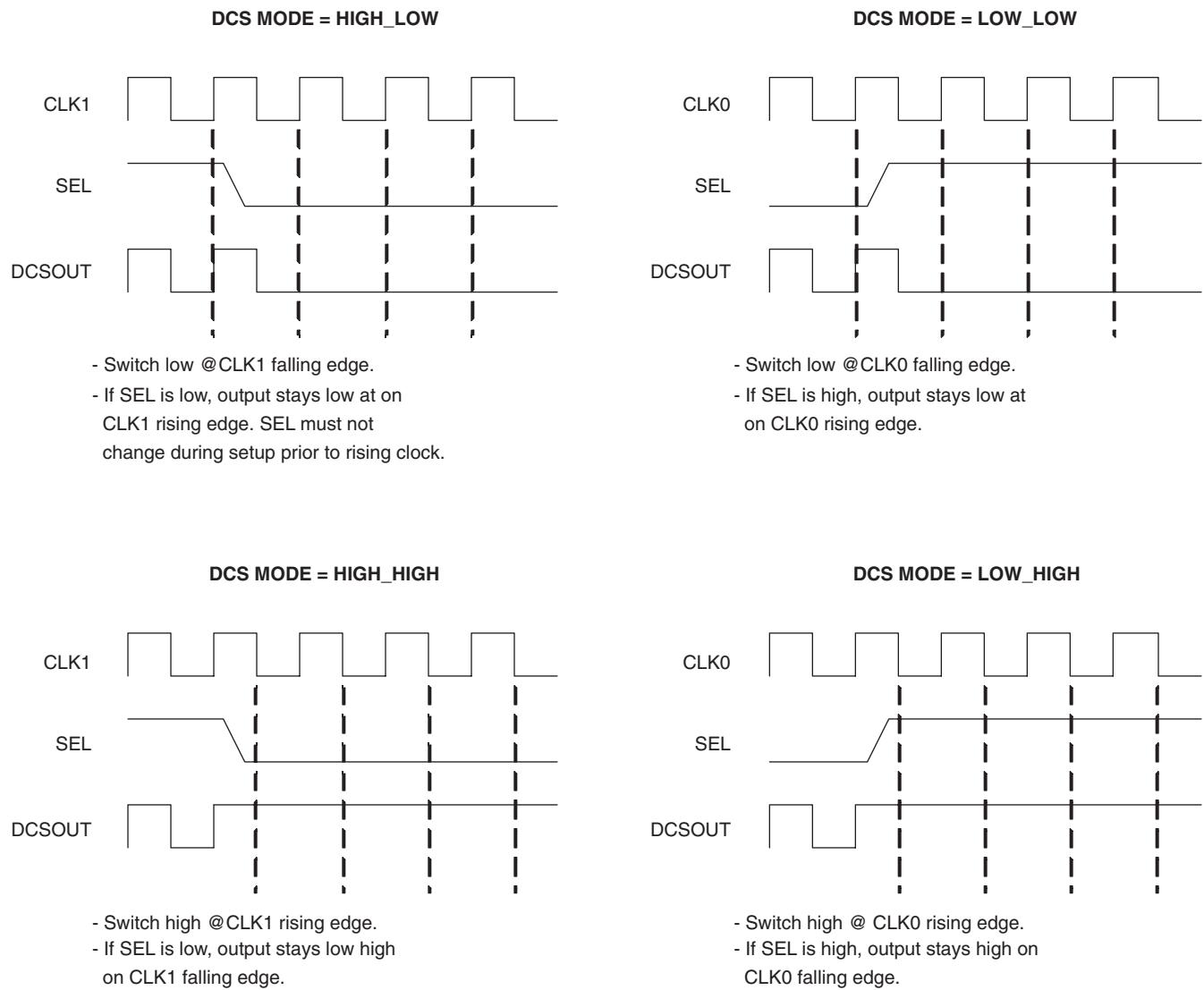
**Table 10-15. DCS Modes of Operations**

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 its unique operation. Clock output timing is determined by input clocks and the edge of the SEL signal. Figure 10-27 describes the timing of each mode.

**Figure 10-27. Timing Diagrams by DCS MODE**

**Figure 8-24. 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 = "CLK0";

endmodule

```

**Oscillator (OSCD)**

There is a dedicated oscillator in the LatticeECP2/M devices whose output is made available for users. The oscillator frequency is programmable with a range of 2.5 to 130MHz. The output of the oscillator can also be routed as an input clock to the clock tree. 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 for use as an extra clock that does not require accurate clocking.

Primitive Name: OSCD

**Table 10-16. OSCD Port Definition**

I/O	Name	Description
Output	OSC	Oscillator Clock Output

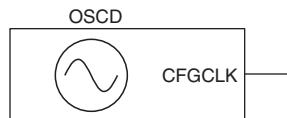
**Table 10-17. OSCD Attribute Definition**

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, 55, 60, 130	2.5

Please refer to the LatticeECP2/M Family Data Sheet for detailed specifications.

## OSC Primitive Symbol (OSCD)

**Figure 10-28. OCS Symbol**



## OSC Usage with VHDL - Example

```

COMPONENT OSCD
-- synthesis translate_off
    GENERIC(NOM_FREQ: string);
-- synthesis translate_on
PORT (CFGCLK:OUT    std_logic);

END COMPONENT;

attribute NOM_FREQ : string;
attribute NOM_FREQ of OSCInst0 : label is "2.5";
begin

OSCInst0: OSCD
-- synthesis translate_off
    GENERIC MAP (NOM_FREQ => "2.5")
-- synthesis translate_on
    PORT MAP ( CFGCLK=> osc_int);

```

## OSC Usage with Verilog - Example

```

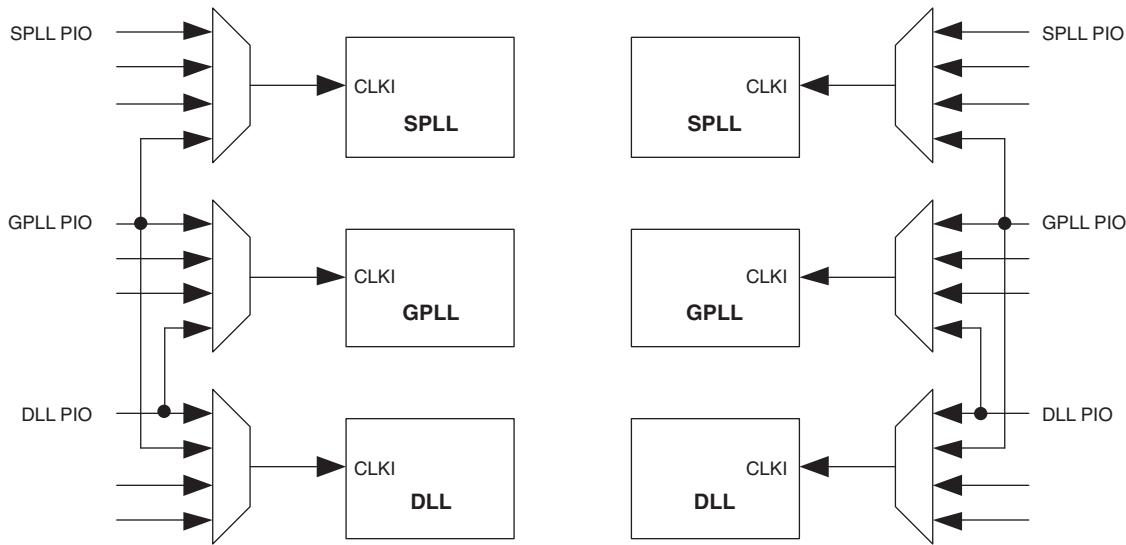
module OSC_TOP(OSC_CLK);
output OSC_CLK;
OSCD OSCInst0 (.CFGCLK(OSC_CLK));
defparam OSCInst0.NOM_FREQ = "2.5";
endmodule

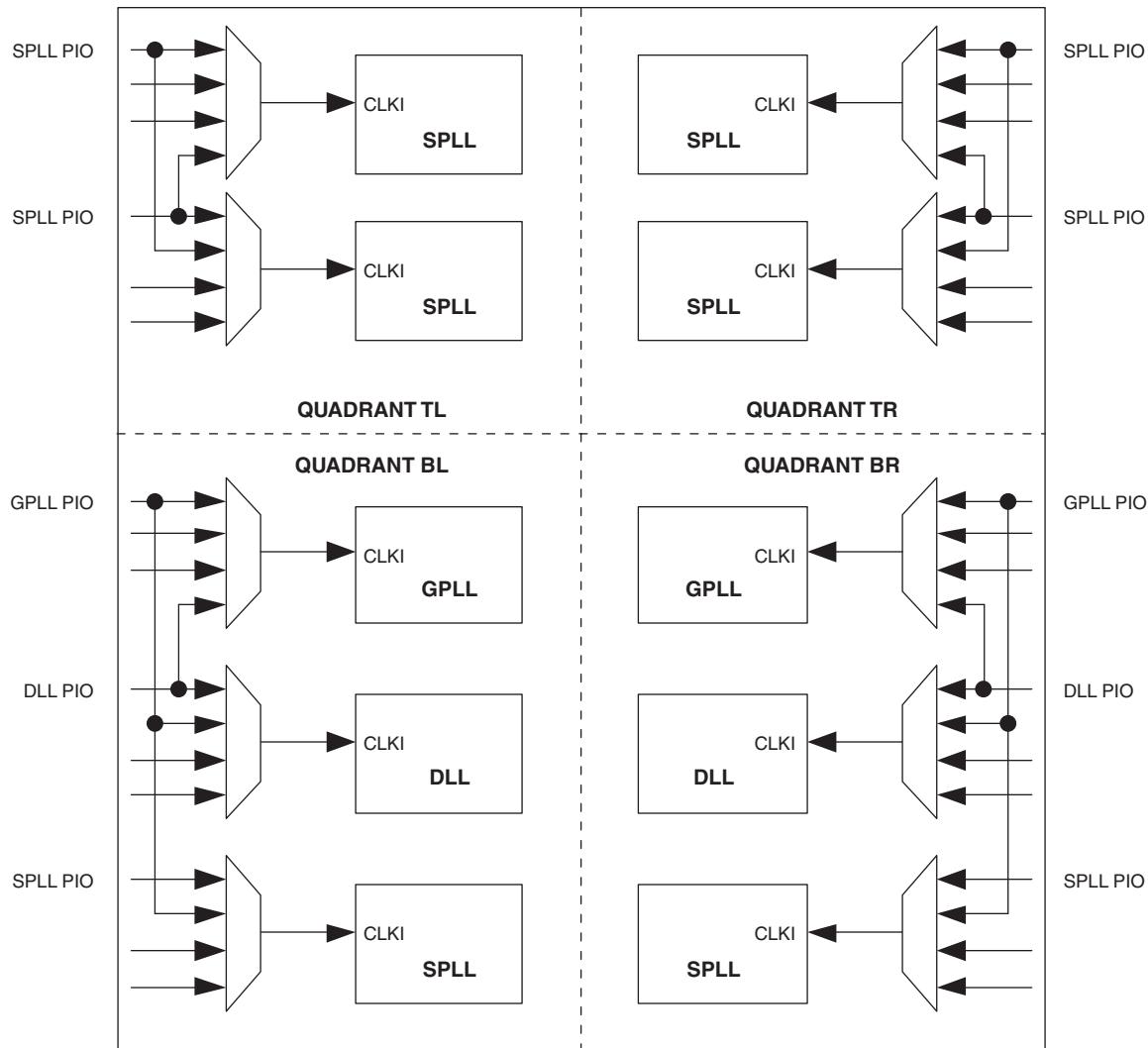
```

## Input Clock Sharing

The reference clock from the pads can be shared in LatticeECP2/M PLLS and DLLs as shown in Figures 10-29 and 10-30. This feature is useful when only one clock source is available for multiple PLLs/DLLs.

**Figure 10-29. Input Clock Sharing (LatticeECP2)**



**Figure 10-30. Input Clock Sharing (LatticeECP2M with Six SPLLs)**

## Setting Clock Preferences

Clock preferences allow designers to implement clocks to the desired performance. Preferences can be set in the Pre-Map Preference Editor in ispLEVER or in preference files. Frequently used preferences are described in Appendix C. For additional information see the ispLEVER on-line Help system.

## Power Supplies

Each PLL has its own power supply pin, VCCPLL. Since VCC and VCCPLL are normally the same 1.2V, it is recommended that they are driven from the same power supply on the circuit board, thus minimizing leakage. In addition, each of these supplies should be independently isolated from the main 1.2V supply on the board using proper board filtering techniques to minimize the noise coupling between them.

The DLL is powered from the FPGA core power supply.

## Technical Support Assistance

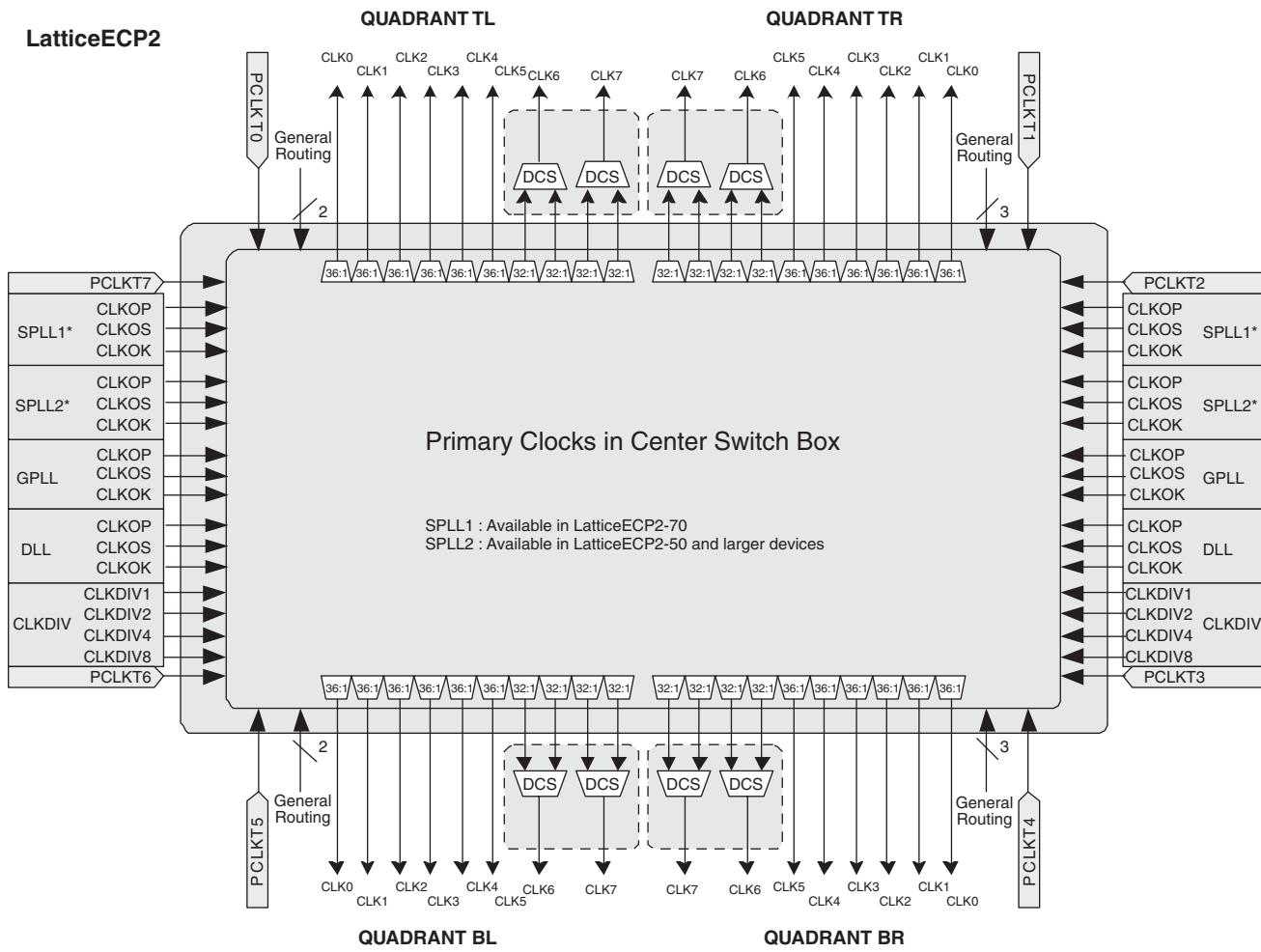
Hotline: 1-800-LATTICE (North America)  
+1-503-268-8001 (Outside North America)  
e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

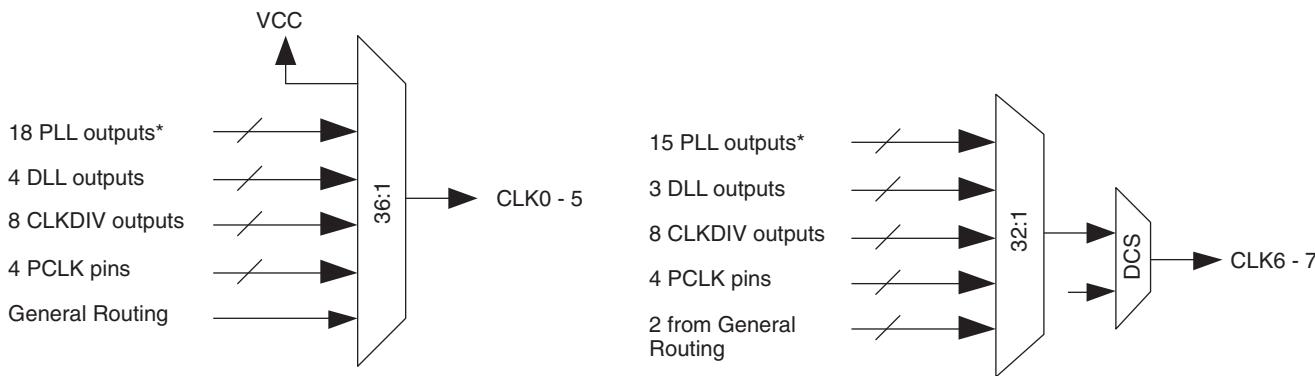
Date	Version	Change Summary
February 2006	01.0	Initial release.
April 2006	01.1	Removed unsupported devices, removed DLL SMI phrases, rephrased DDUTY support due to software incomplete.
September 2006	01.2	Added detailed clock network descriptions. Added IPexpress GUI quick reference table. Added LatticeECP2M information throughout. OSC divider value range updated.

## Appendix A. Primary Clock Sources and Distribution

**Figure 10-31. LatticeECP2 Primary Clock Sources and Distribution**



**Figure 10-32. LatticeECP2 Primary Clock Muxes**



\*LatticeECP2-50 has twelve PLL outputs

\*LatticeECP2-50 has ten PLL outputs

Figure 10-33. LatticeECP2M Primary Clock Sources and Distribution

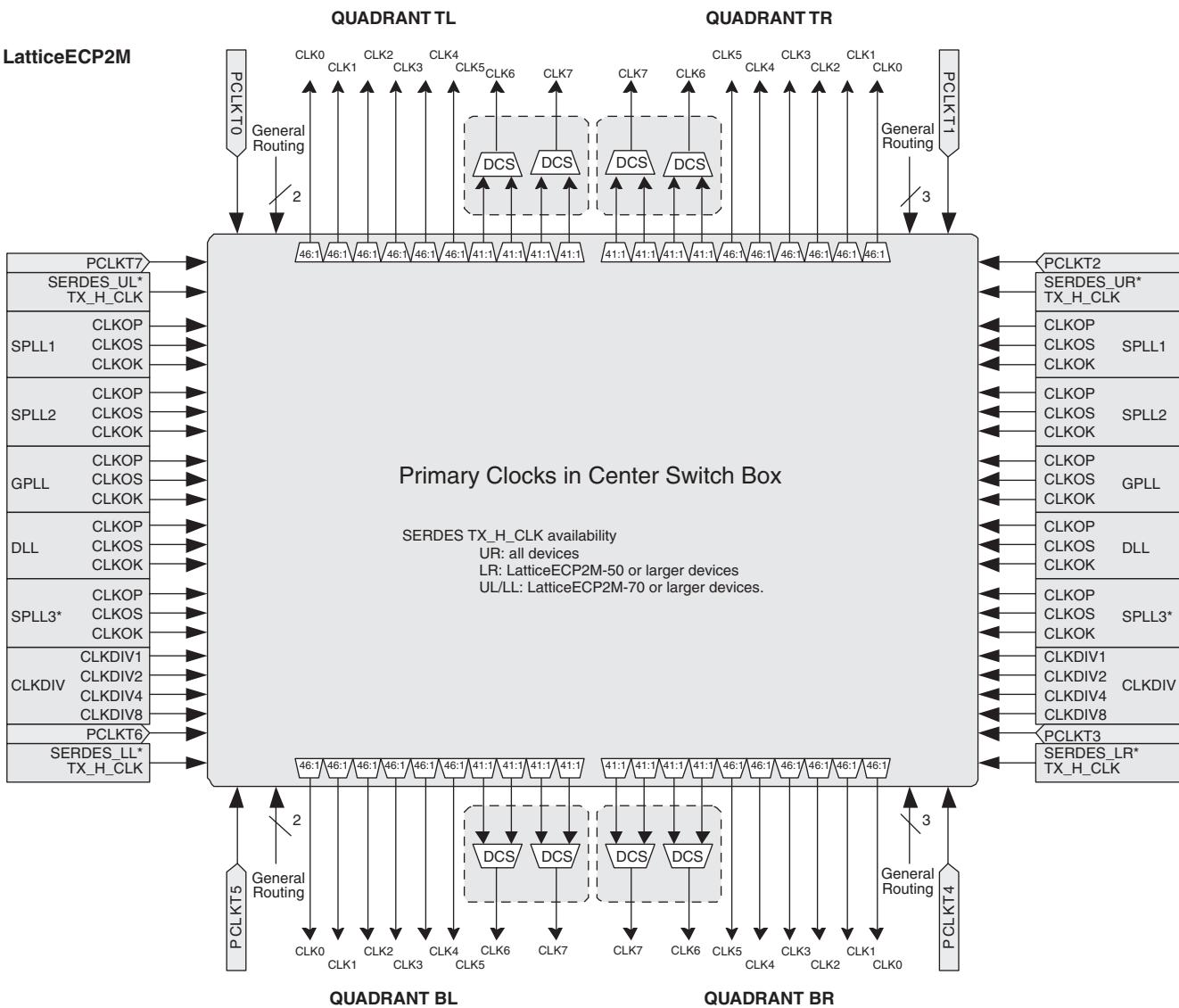
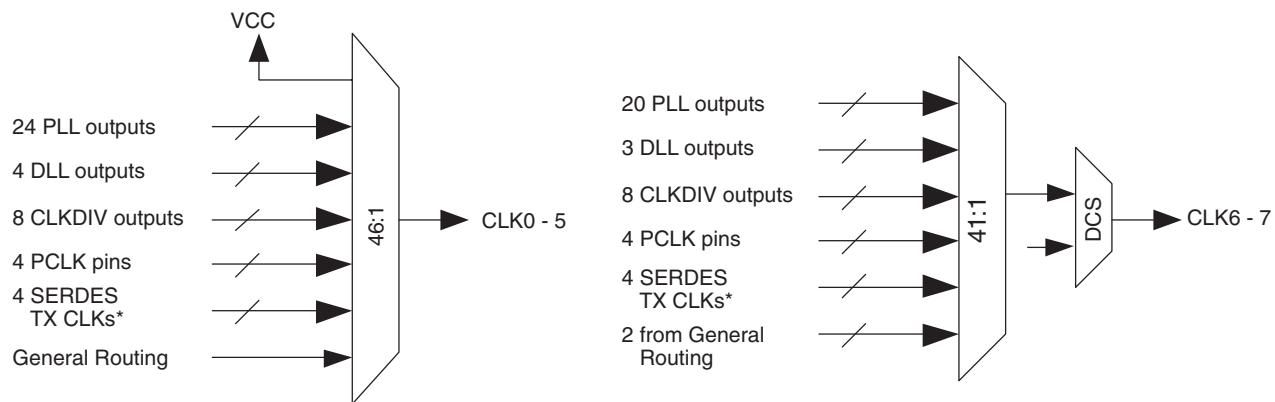


Figure 10-34. LatticeECP2M Primary Clock Muxes

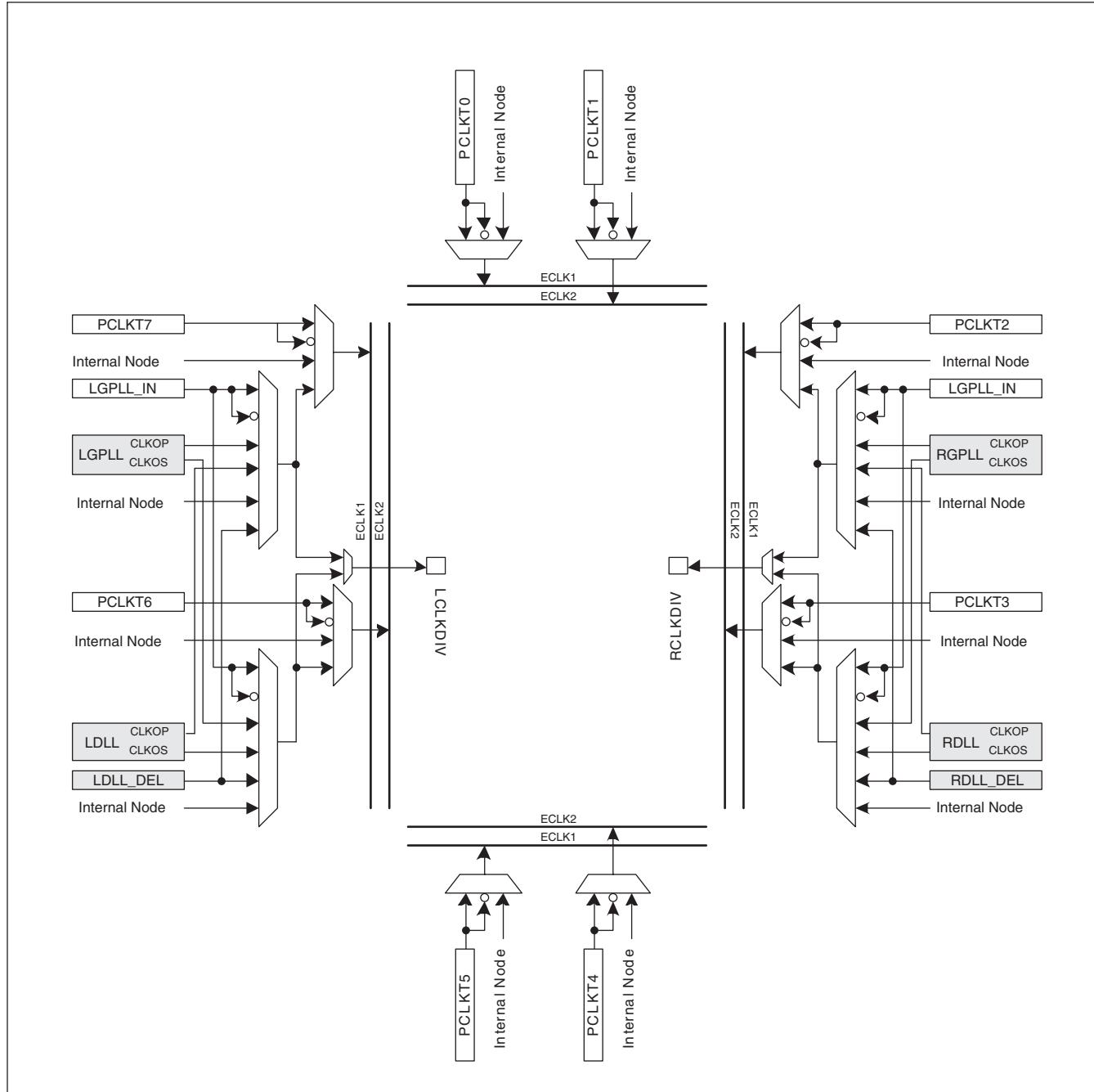


\*LatticeECP2M-50 has two SERDES TX CLK outputs

## Appendix B. PLL, DLL, CLKIDV and ECLK Locations and Connectivity

Figure 10-35 shows the locations, site names, and connectivity of the PLLs, DLLs, CLKDIVs and ECLKs

**Figure 10-35. PLL, DLL, CLKIDV and ECLK Locations and Connectivity**



## Appendix C. Clock Preferences

A few key clock preferences are introduced below. Refer to the software “Help” file for other preferences and detailed information.

### ASIC

The following preference command assigns a phase of 90° to the CIMDLLA CLKOP:

```
ASIC "my_dll" TYPE "CIMDLLA" CLKOP_PHASE=90;
```

### FREQUENCY

The following physical preference command assigns a frequency of 100 MHz to a net named clk1:

```
FREQUENCY NET "clk1" 100 MHz;
```

The following preference specifies a hold margin value for each clock domain:

```
FREQUENCY NET "RX_CLKA_CMOS_C" 100.000 MHz HOLD_MARGIN 1 ns;
```

### MAXSKEW

The following command assigns a maximum skew of 5ns to a net named NetB:

```
MAXSKEW NET "NetB" 5 NS;
```

### MULTICYCLE

The following command will relax the period to 50ns for the path starting at COMPA to COMPB (NET1):

```
MULTICYCLE "PATH1" START COMP "COMPA" END COMP "COMPB" NET "NET1" 50 NS ;
```

### PERIOD

The following command assigns a clock period of 30ns to the port named Clk1:

```
PERIOD PORT "Clk1" 30 NS;
```

### PROHIBIT

This command prohibits the use of a primary clock to route a clock net named bf\_clk:

```
PROHIBIT PRIMARY NET "bf_clk";
```

### USE PRIMARY

Use a primary clock resource to route the specified net.

```
USE PRIMARY NET clk_fast;  
USE PRIMARY DCS NET "bf_clk";  
USE PRIMARY PURE NET "bf_clk" QUADRANT_TL;
```

### USE SECONDARY

Use a secondary clock resource to route the specified net.

```
USE SECONDARY NET "clk_lessfast" QUADRANT_TL;
```

### USE EDGE

Use a edge clock resource to route the specified net.

```
USE EDGE NET "clk_fast";
```

### CLOCK\_TO\_OUT

Specifies a maximum allowable output delay relative to a clock.

Here are two preferences using both the CLKPORT and CLKNET keywords showing the corresponding scope of TRACE reporting.

The CLKNET will stop tracing the path before the PLL, so you will not get PLL compensation timing numbers.

```
CLOCK_TO_OUT PORT "RxAddr_0" 6.000000 ns CLKNET "pll_rxclk" ;
```

The above preference will yield the following clock path:

```
Physical Path Details:  
Clock path pll_inst/pll_utp_0_0 to PFU_33:  
Name Fanout Delay (ns) Site Resource  
ROUTE 49 2.892 ULPPLL.MCLK to R3C14.CLK0 pll_rxclk  
-----  
2.892 (0.0% logic, 100.0% route), 0 logic levels.
```

If CLKPORT is used, the trace is complete back to the clock port resource and provides PLL compensation timing numbers.

```
CLOCK_TO_OUT PORT "RxAddr_0" 6.000000 ns CLKPORT "RxClk" ;
```

The above preference will yield the following clock path:

```
Clock path RxClk to PFU_33:  
Name Fanout Delay (ns) Site Resource  
IN_DEL --- 1.431 D5.PAD to D5.INCK RxClk  
ROUTE 1 0.843 D5.INCK to ULPPLL.CLKIN RxClk_c  
MCLK_DEL --- 3.605 ULPPLL.CLKIN to ULPPLL.MCLK pll_inst/pll_utp_0_0  
ROUTE 49 2.892 ULPPLL.MCLK to R3C14.CLK0 pll_rxclk  
-----  
8.771 (57.4% logic, 42.6% route), 2 logic levels.
```

## **INPUT\_SETUP**

Specifies an setup time requirement for input ports relative to a clock net.

```
INPUT_SETUP PORT "datain" 2.000000 ns HOLD 1.000000 ns CLKPORT "clk"  
PLL_PHASE_BACK ;
```

## **PLL\_PHASE\_BACK**

This preference is used with INPUT\_SETUP when the user needs a Trace calculation based on the previous clock edge.

This preference is useful when setting the PLL output Phase Adjustment. Since there is no negative phase adjustment provided, the PLL\_PHASE\_BACK preference works as if negative phase adjustment is available.

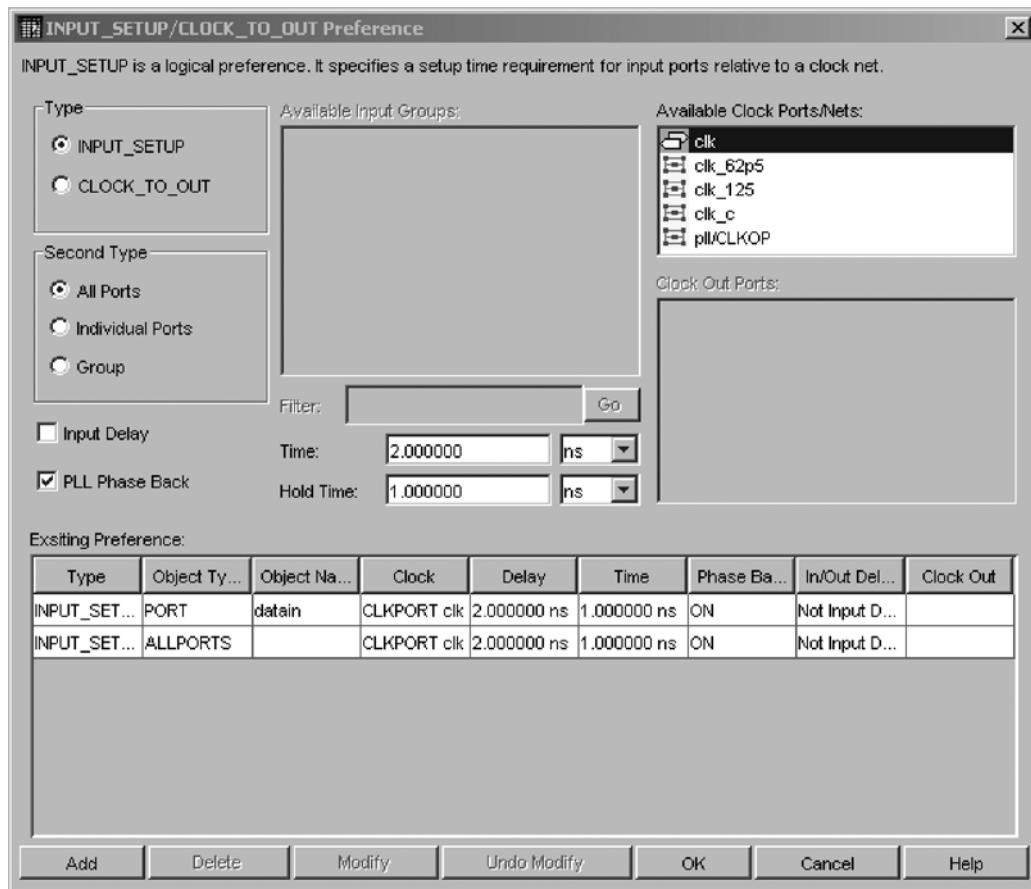
For example:

If a Phase Adjustment of -90° of CLKOS is desired, the user can set the Phase to 270° and set the INPUT\_SETUP preference with PLL\_PHASE\_BACK.

**PLL\_PHASE\_BACK Usage in Pre-Map Preference Editor:** The Pre-Map Preference Editor can be used to set the PLL\_PHASE\_BACK attribute.

1. Open the Design Planner (Pre-Map).
2. In the Design Planner control window, select **Spreadsheet View** under **View**.
3. In the Spreadsheet View window, select **Input\_setup/Clock\_to\_out...**
4. The INPUT\_SETUP/CLOCK\_TO\_OUT Preference window is shown in Figure 10-36.

Figure 10-36. INPUT\_SETUP/CLOCK\_TO\_OUT Preference Window



## Introduction

This technical note discusses memory usage for the LatticeECP2™ and LatticeECP2M™ device families. It is intended to be used by design engineers as a guide for integrating the EBR- (Embedded Block RAM) and PFU-based memories in this device family using the ispLEVER® design tool.

The architecture of these devices provides resources for FPGA on-chip memory applications. The sysMEM™ EBR complements the distributed PFU-based memory. Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM, FIFO and ROM memories can be constructed using the EBR. LUTs and PFU can implement Distributed Single-Port RAM, Dual-Port RAM and ROM.

The capabilities of the EBR RAM and PFU RAM are referred to as primitives and are described later in this document. Designers can utilize the memory primitives in two ways via the IPexpress™ tool in the ispLEVER software. The IPexpress GUI allows users to specify the memory type and size required. IPexpress takes this specification and constructs a netlist to implement the desired memory by using one or more of the memory primitives.

The remainder of this document discusses the use of IPexpress, memory modules and memory primitives.

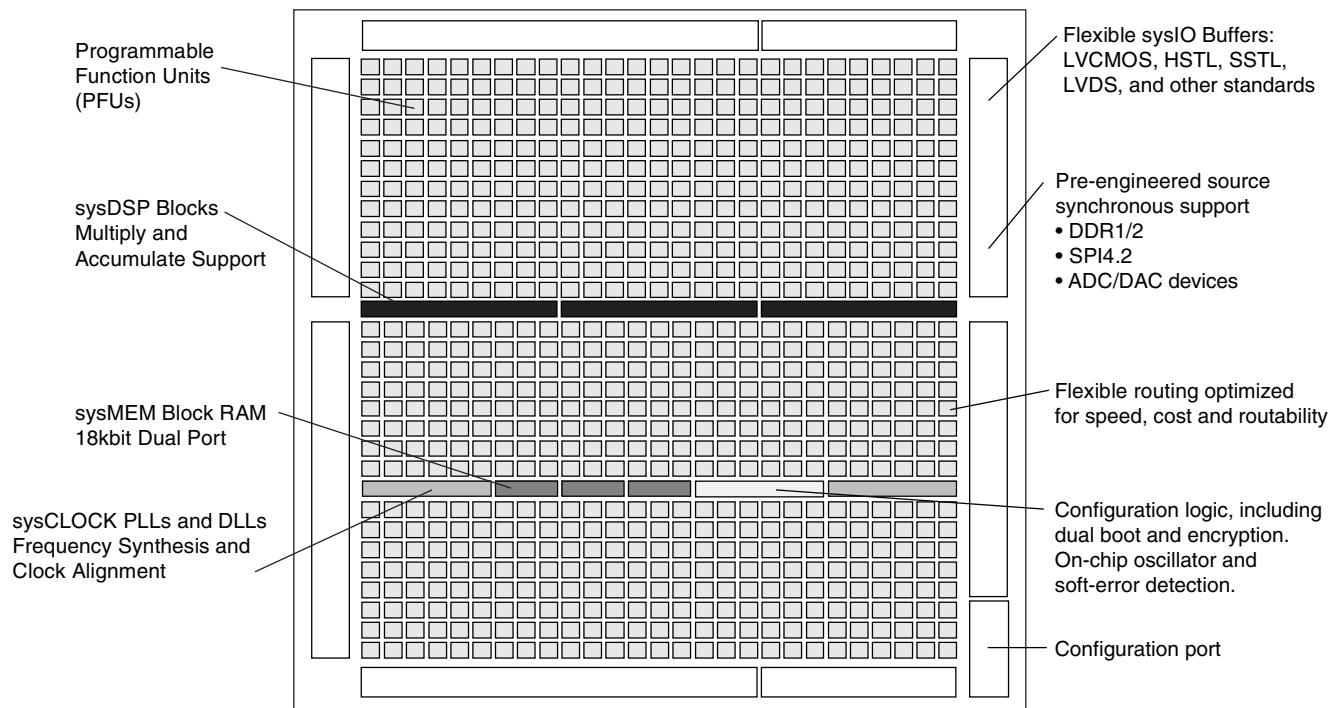
## Memories in LatticeECP2/M Devices

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, ROM and register 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 LatticeECP2 family of devices contains up to two rows of sysMEM EBR blocks and the LatticeECP2M family of devices contains up to seven rows of sysMEM EBR blocks. sysMEM EBRs are large, dedicated 18K fast memory blocks. Each sysMEM block can be configured in a variety of depths and widths of RAM or ROM. In addition, LatticeECP2/M devices contain up to two rows of sysDSP™ blocks. Each sysDSP block has multipliers and accumulators, which are the building blocks for complex signal processing capabilities

**Table 11-1. LatticeECP2/M LUT and Memory Densities**

LUTs	6 K	12 K	20 K		35 K		50 K		70 K		100 K
Device	ECP2-6	ECP2-12	ECP2-20	ECP2M-20	ECP2-35	ECP2M-35	ECP2-50	ECP2M-50	ECP2-70	ECP2M-70	ECP2M-100
LUTs (K)	6	12	21	19	32	34	48	48	68	67	95
Distributed RAM (Kbits)	12	24	42	41	65	71	96	101	136	145	202
EBR SRAM Blocks	3	12	15	66	18	114	21	225	60	246	288
EBR SRAM (Kbits)	55	221	276	1217	332	2101	387	4147	1106	4534	5308

**Figure 11-1. Simplified Block Diagram, LatticeECP2-6 Device (Top Level)**

## Utilizing IPexpress

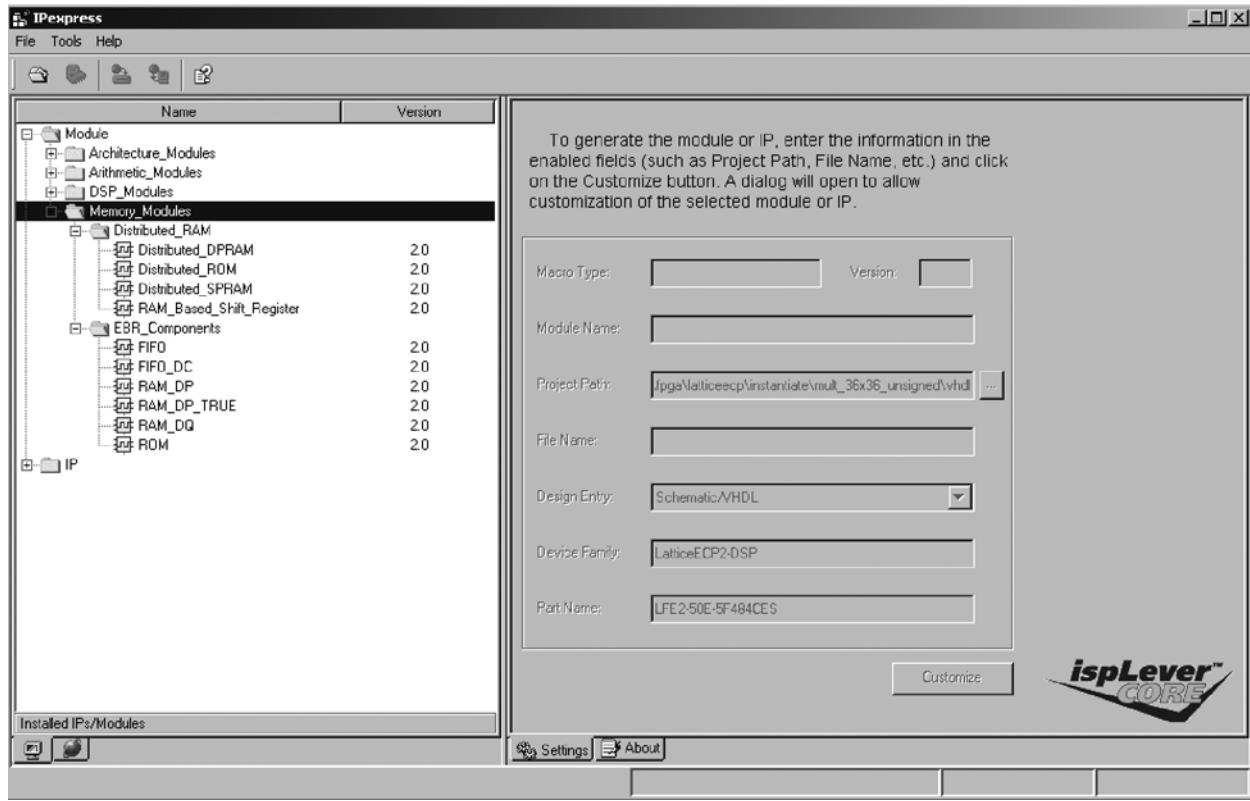
Designers can utilize IPexpress to easily specify a variety of memories in their designs. These modules are constructed using one or more memory primitives along with general purpose routing and LUTs, as required. The available primitives are:

- 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 (Dual Clock) (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

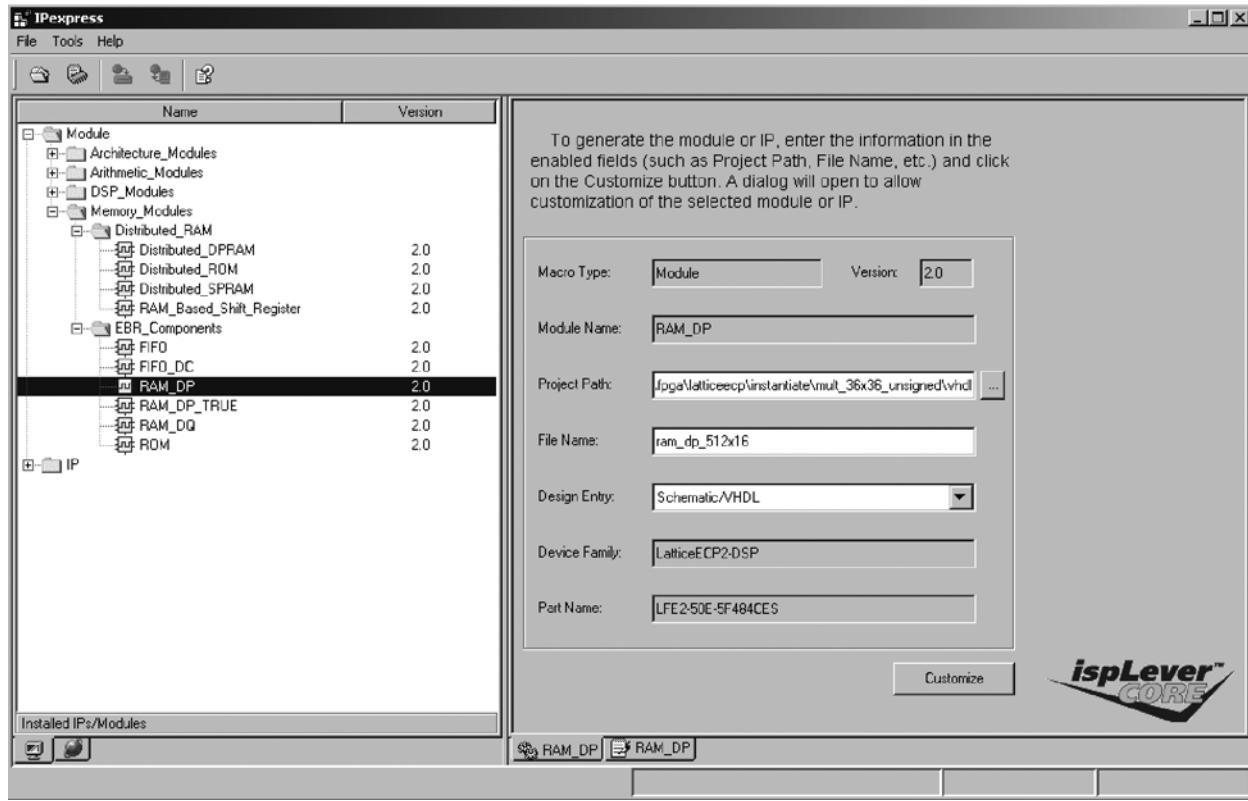
For generating any of these memories, create (or open) a project for the LatticeECP2/M devices.

From the Project Navigator, select **Tools > IPexpress** or click on the button in the toolbar when LatticeECP2/M devices are targeted in the project. This opens the IPexpress main window as shown in Figure 11-2.

**Figure 11-2. IPExpress - Main Window**

The left pane of this window includes the Module Tree. The EBR-based Memory Modules are under the **EBR\_Components** and the PFU-based Distributed Memory Modules are under **Storage\_Components**, as shown in Figure 11-2.

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 11-3.

**Figure 11-3. Example Generating Pseudo Dual Port RAM (RAM\_DP) Using IPexpress**

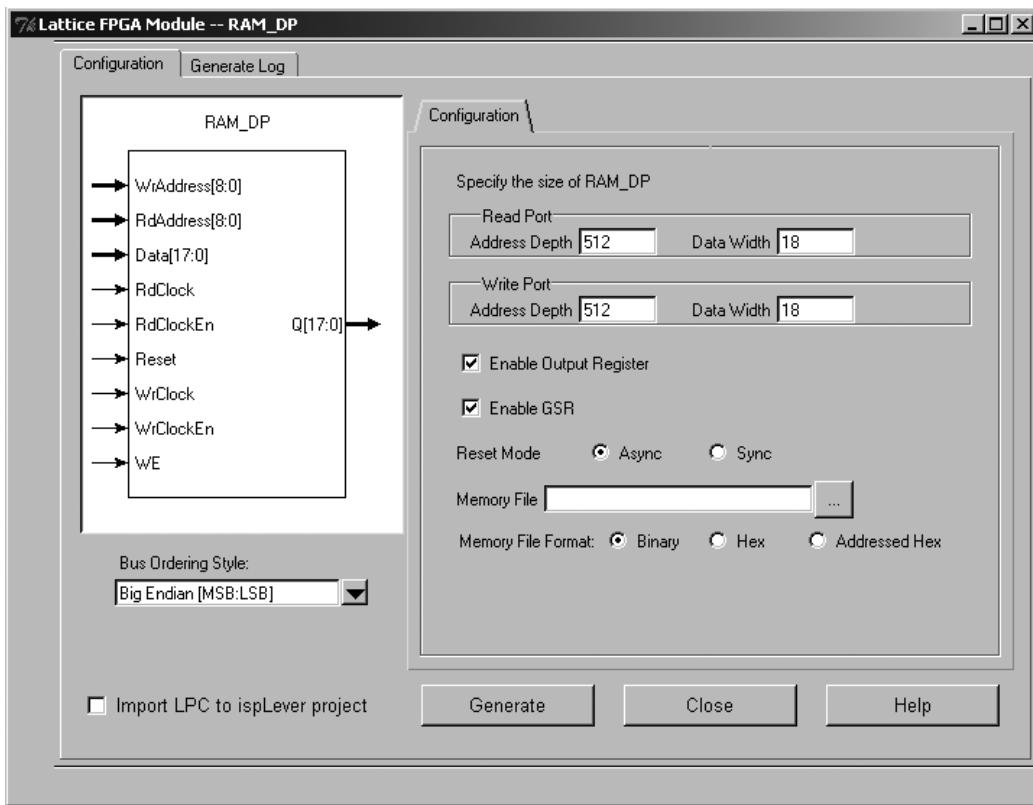
In the right pane, options like the **Device Family**, **Macro Type**, **Category**, and **Module\_Name** are device and selected module dependent. These cannot be changed in IPexpress.

Users can change the directory where the generated module files will be placed by clicking the **Browse** button in the **Project Path**.

The **Module Name** text box allows users to specify an entity name for the module they are about to generate. 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, LatticeECP2/M in this example. By clicking the **Customize** button, another window opens where users can customize the RAM (Figure 11-4).

**Figure 11-4. Example Generating Pseudo Dual Port RAM (RAM\_DP) Module Customization**

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 check box **Enable Output Registers** inserts the output registers in the Read Data Port. Output registers are optional for EBR-based RAMs.

Users have the option to set the **Reset Mode** as Asynchronous Reset or Synchronous Reset. **Enable GSR** can be checked to enable the Global Set Reset.

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.

Once the module is generated, users can either instantiate the \*.lpc or the Verilog-HDL/ VHDL file in top-level module of their design.

The various memory modules, both EBR and distributed, are discussed in detail in this document.

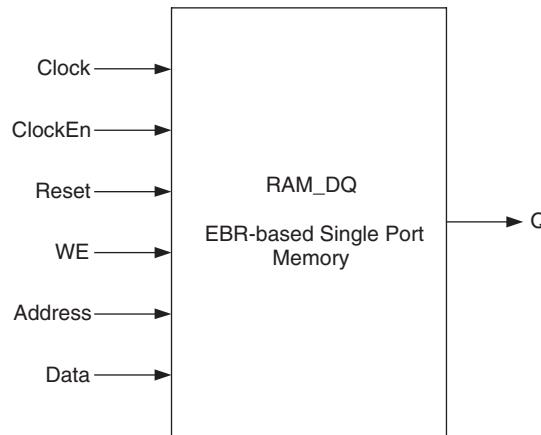
## Memory Modules

### Single Port RAM (RAM\_DQ) – EBR Based

The EBR blocks in LatticeECP2/M devices can be configured as Single Port RAM or RAM\_DQ. IPexpress allows users to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirements.

IPexpress generates the memory module as shown in Figure 11-5.

**Figure 11-5. 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 memory sizes smaller than an EBR block, the module will be created in one EBR 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 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 the Single Port Memory are listed in Table 11-2. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM\_DQ primitive.

**Table 11-2. EBR-based Single Port Memory Port Definitions**

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
Clock	CLK	Clock	Rising Clock Edge
ClockEn	CE	Clock Enable	Active High
Address	AD[x:0]	Address Bus	—
Data	DI[y:0]	Data In	—
Q	DO[y:0]	Data Out	—
WE	WE	Write Enable	Active High
Reset	RST	Reset	Active High
—	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.

Chip Select (CS) is a useful port in the EBR primitive 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. 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 ispLEVER 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) for each EBR block for the devices are listed in Table 11-3.

**Table 11-3. Single Port Memory Sizes for 16K Memories for LatticeECP2/M**

Single Port Memory Size	Input Data	Output Data	Address [MSB:LSB]
16K x 1	DI	DO	AD[13:0]
8K x 2	DI[1:0]	DO[1:0]	AD[12:0]
4K x 4	DI[3:0]	DO[3:0]	AD[11:0]
2K x 9	DI[8:0]	DO[8:0]	AD[10:0]
1K x 18	DI[17:0]	DO[17:0]	AD[9:0]
512 x 36	DI[35:0]	DO[35:0]	AD[8:0]

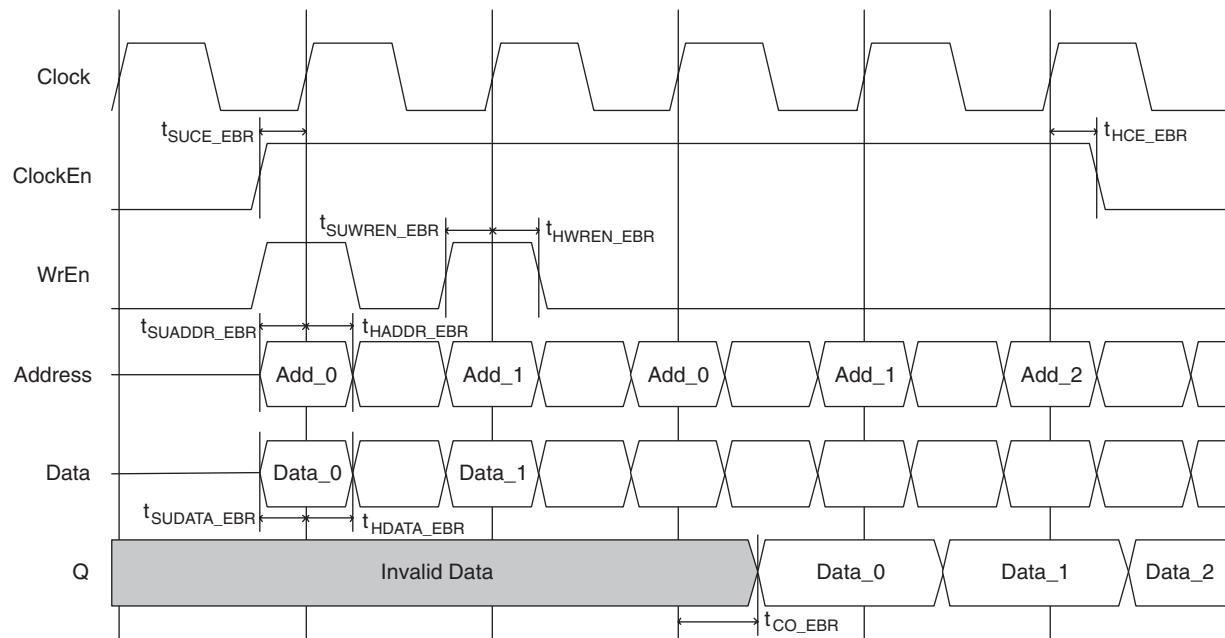
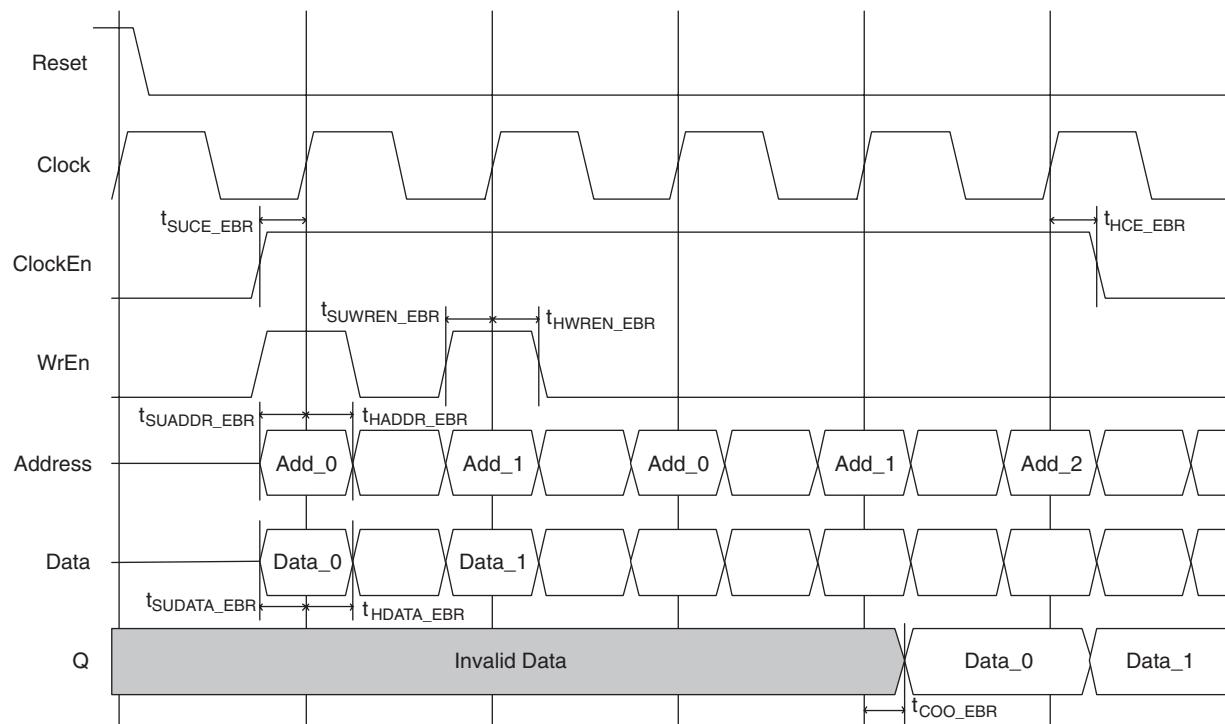
Table 11-4 shows the various attributes available for the Single Port Memory (RAM\_DQ). Some of these attributes are user-selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

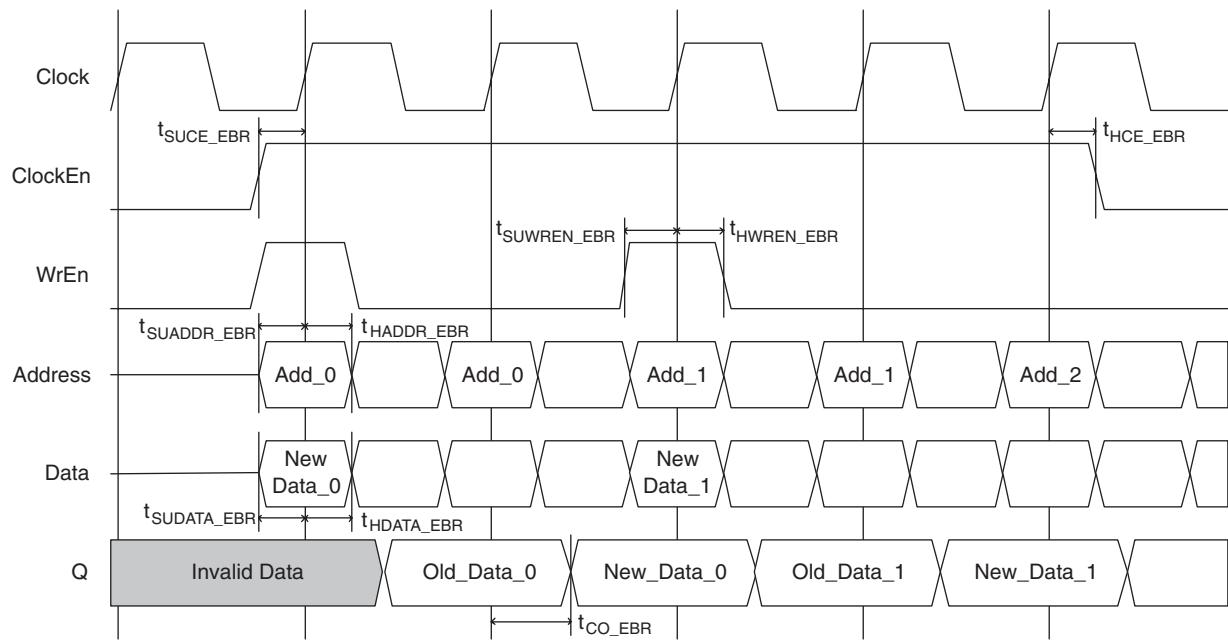
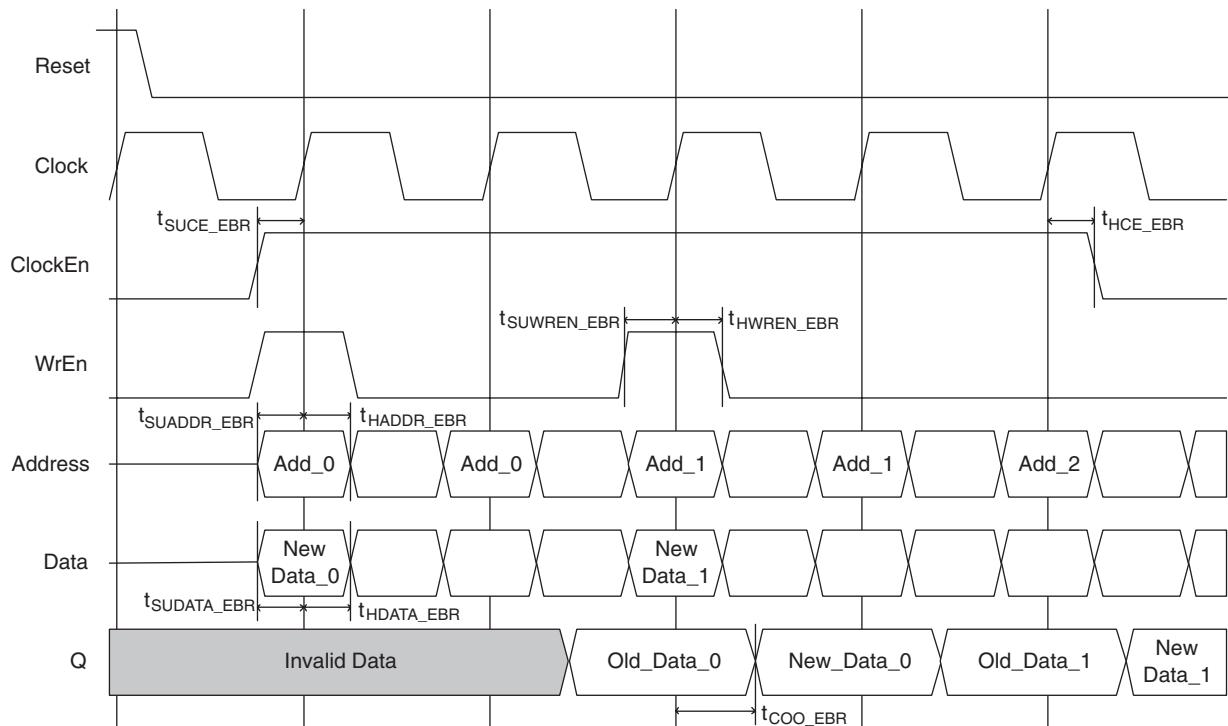
**Table 11-4. Single Port RAM Attributes for LatticeECP2/M**

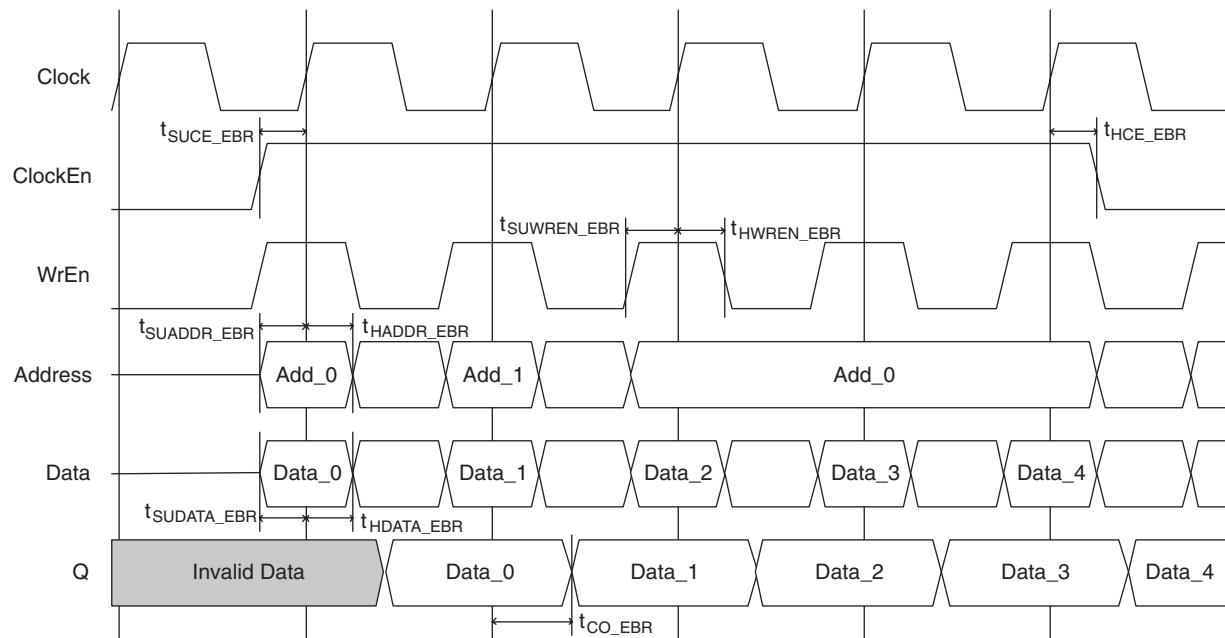
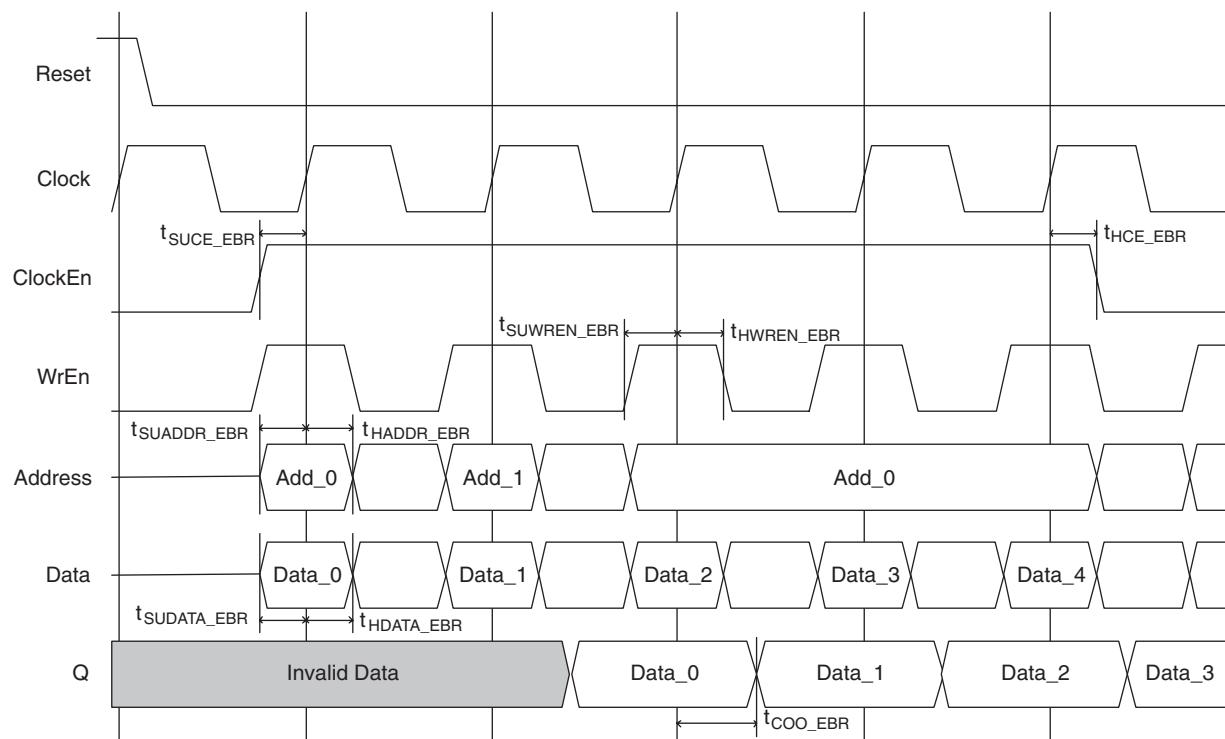
Attribute	Description	Values	Default Value	User Selectable Through IPexpress	
Address depth	Address Depth Read Port	16K, 8K, 4K, 2K, 1K, 512		YES	
Data Width	Data Word Width Read Port	1, 2, 4, 9, 18, 36	1	YES	
Enable Output Registers	Register Mode (Pipelining) for Write Port	NOREG, OUTREG	NOREG	YES	
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, WRITETHROUGH, READ-BEFOREWRITE	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 FFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFF FFFFFFFFFFFF	0x0000000000 000000000000 000000000000 000000000000 000000000000 000000000000 000000000000 000000000000	00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000 00000000000000000000000000000000	NO

The Single Port RAM (RAM\_DQ) can be configured as NORMAL, READ BEFORE WRITE or WRITE THROUGH 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 11-6-11-11 show the internal timing waveforms for the Single Port RAM (RAM\_DQ) with these options.

**Figure 11-6. Single Port RAM Timing Waveform - NORMAL Mode, without Output Registers****Figure 11-7. Single Port RAM Timing Waveform - NORMAL Mode, with Output Registers**

**Figure 11-8. Single Port RAM Timing Waveform - READ BEFORE WRITE Mode, without Output Registers****Figure 11-9. Single Port RAM Timing Waveform - READ BEFORE WRITE Mode, with Output Registers**

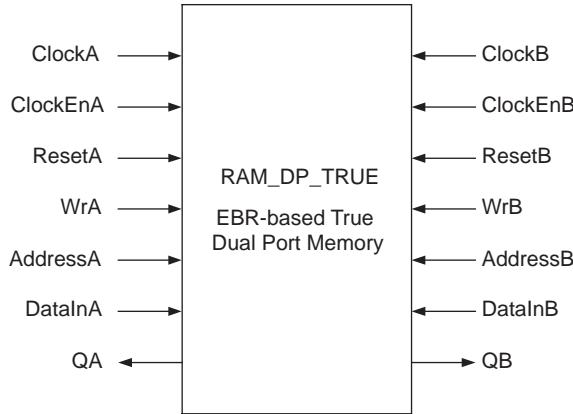
**Figure 11-10. Single Port RAM Timing Waveform - WRITE THROUGH Mode, without Output Registers****Figure 11-11. Single Port RAM Timing Waveform - WRITE THROUGH Mode, with Output Registers**

## True Dual Port RAM (RAM\_DP\_TRUE) – EBR Based

The EBR blocks in the LatticeECP2/M 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 11-12.

**Figure 11-12. True Dual Port Memory Module Generated by IPexpress**



The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. When 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 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 Single Port Memory are listed in Table 11-5. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM\_DP\_TRUE primitive.

**Table 11-5. EBR-based True Dual Port Memory Port Definitions**

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
ClockA, ClockB	CLKA, CLKB	Clock for PortA and PortB	Rising Clock Edge
ClockEnA, ClockEnB	CEA, CEB	Clock Enables for Port CLKA and CLKB	Active High
AddressA, AddressB	ADA[x1:0], ADB[x2:0]	Address Bus port A and port B	—
DataA, DataB	DIA[y1:0], DIB[y2:0]	Input Data port A and port B	—
QA, QB	DOA[y1:0], DOB[y2:0]	Output Data port A and port B	—
WrA, WrB	WEA, WEB	Write enable port A and port B	Active High
ResetA, ResetB	RSTA, RSTB	Reset for PortA and PortB	Active High
—	CSA[2:0], CSB[2:0]	Chip Selects for each port	—

Reset (or 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 in the EBR primitive 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.

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 are listed in Table 11-6.

**Table 11-6. True Dual Port Memory Sizes for 16K Memory for LatticeECP2/M**

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]
16K x 1	DIA	DIB	DOA	DOB	ADA[13:0]	ADB[13:0]
8K x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	ADA[12:0]	ADB[12:0]
4K x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	ADA[11:0]	ADB[11:0]
2K x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	ADA[10:0]	ADB[10:0]
1K x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	ADA[9:0]	ADB[9:0]

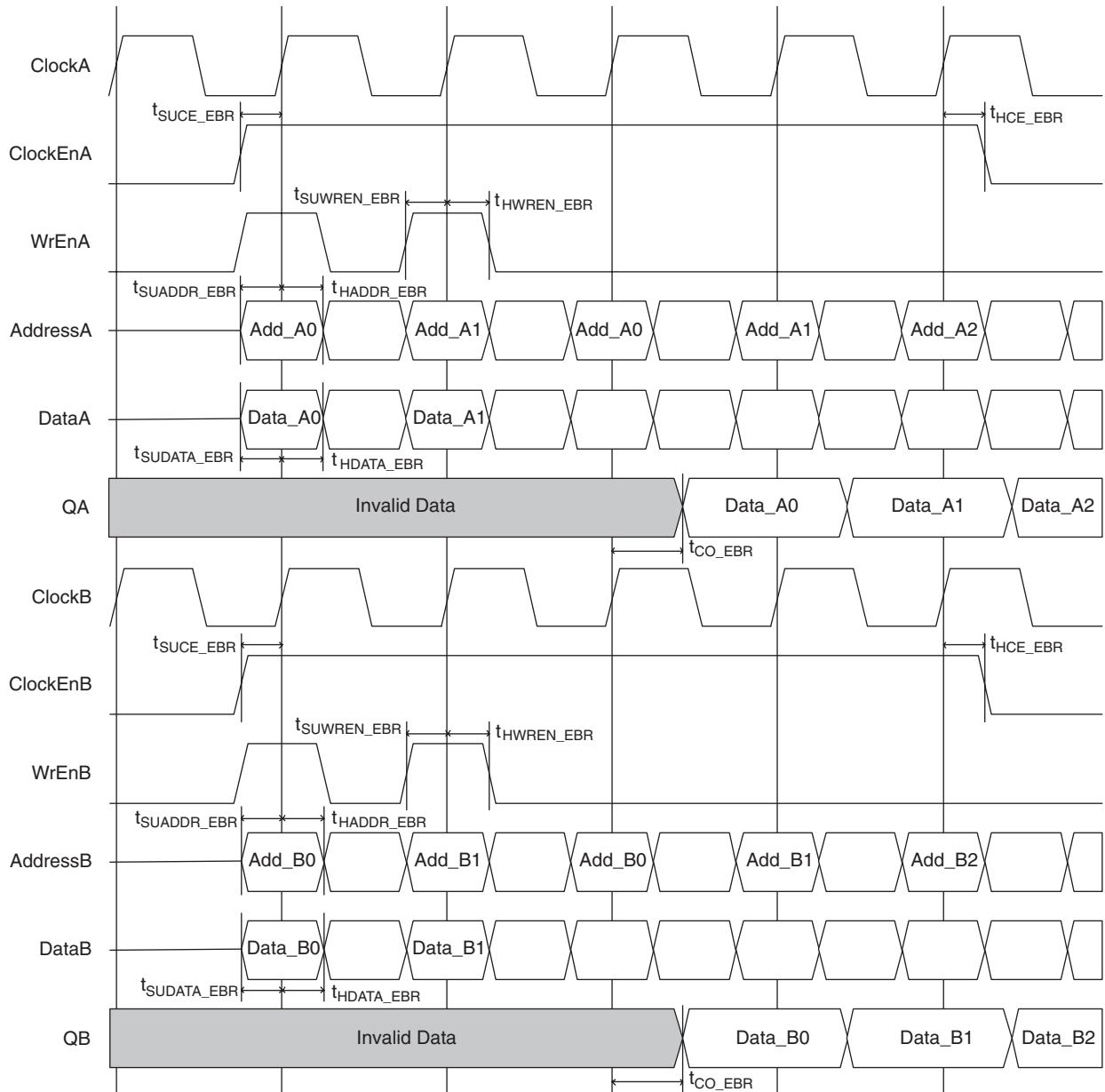
Table 11-7 shows the various attributes available for the Single Port Memory (RAM\_DQ). Some of these attributes are user-selectable through the IPExpress GUI. For detailed attribute definitions, refer to the Appendix A.

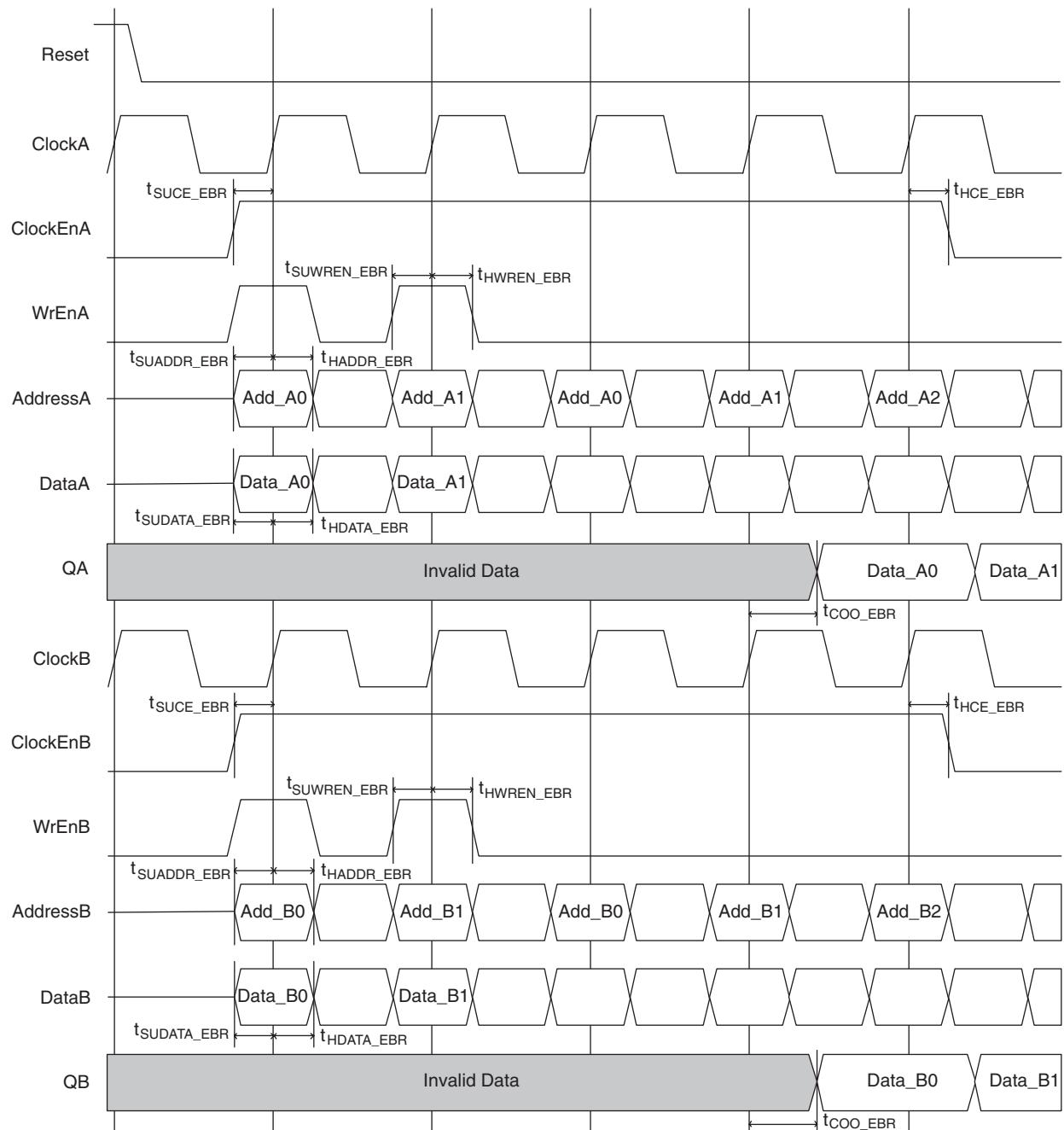
**Table 11-7. True Dual Port RAM Attributes for LatticeECP2/M**

The True Dual Port RAM (RAM\_DP\_TRUE) can be configured as NORMAL, READ BEFORE WRITE or WRITE THROUGH 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.

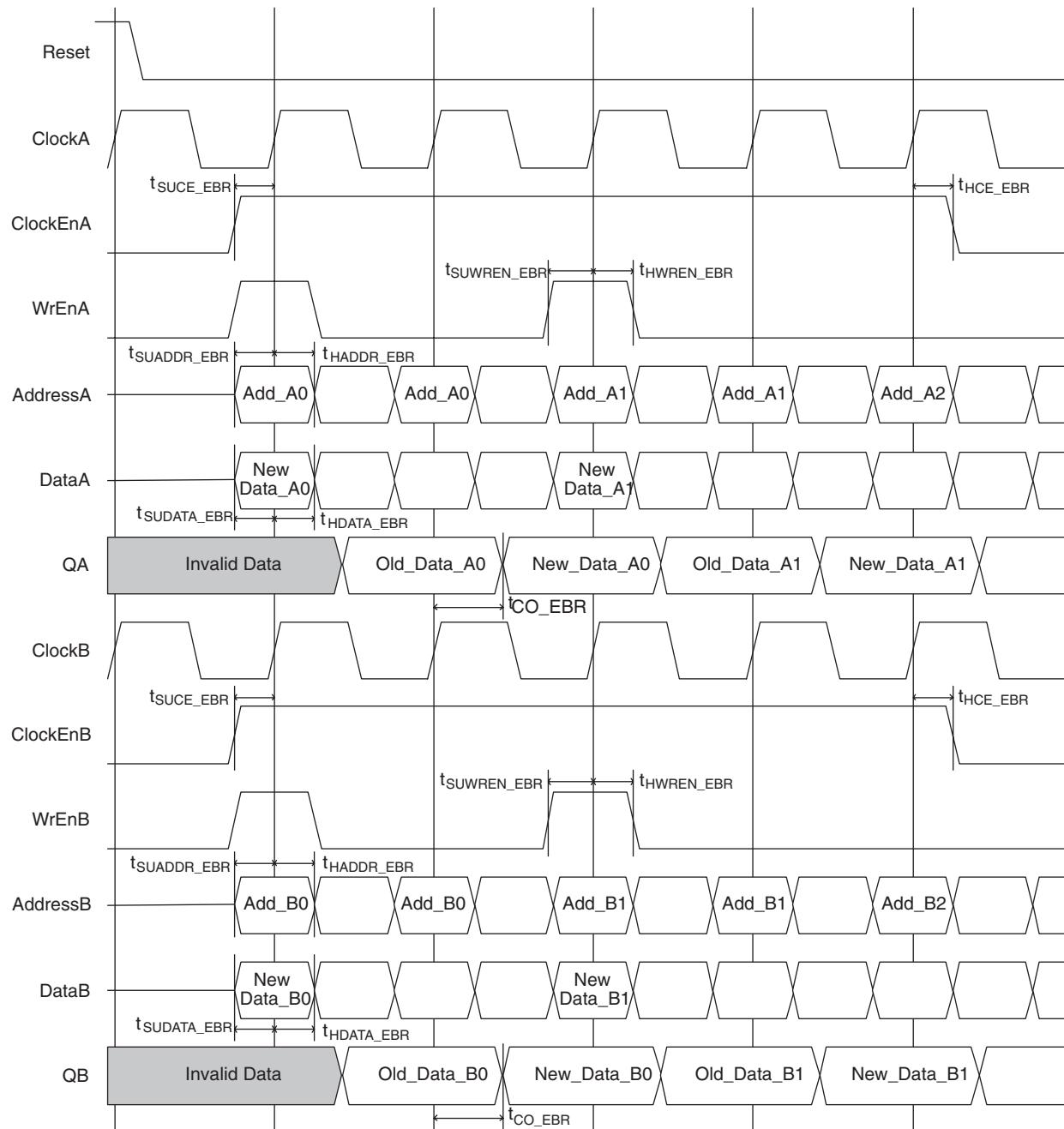
Additionally, users can select to enable the output registers for RAM\_DP\_TRUE. Figures 11-13 through 11-18 show the internal timing waveforms for the True Dual Port RAM (RAM\_DP\_TRUE) with these options.

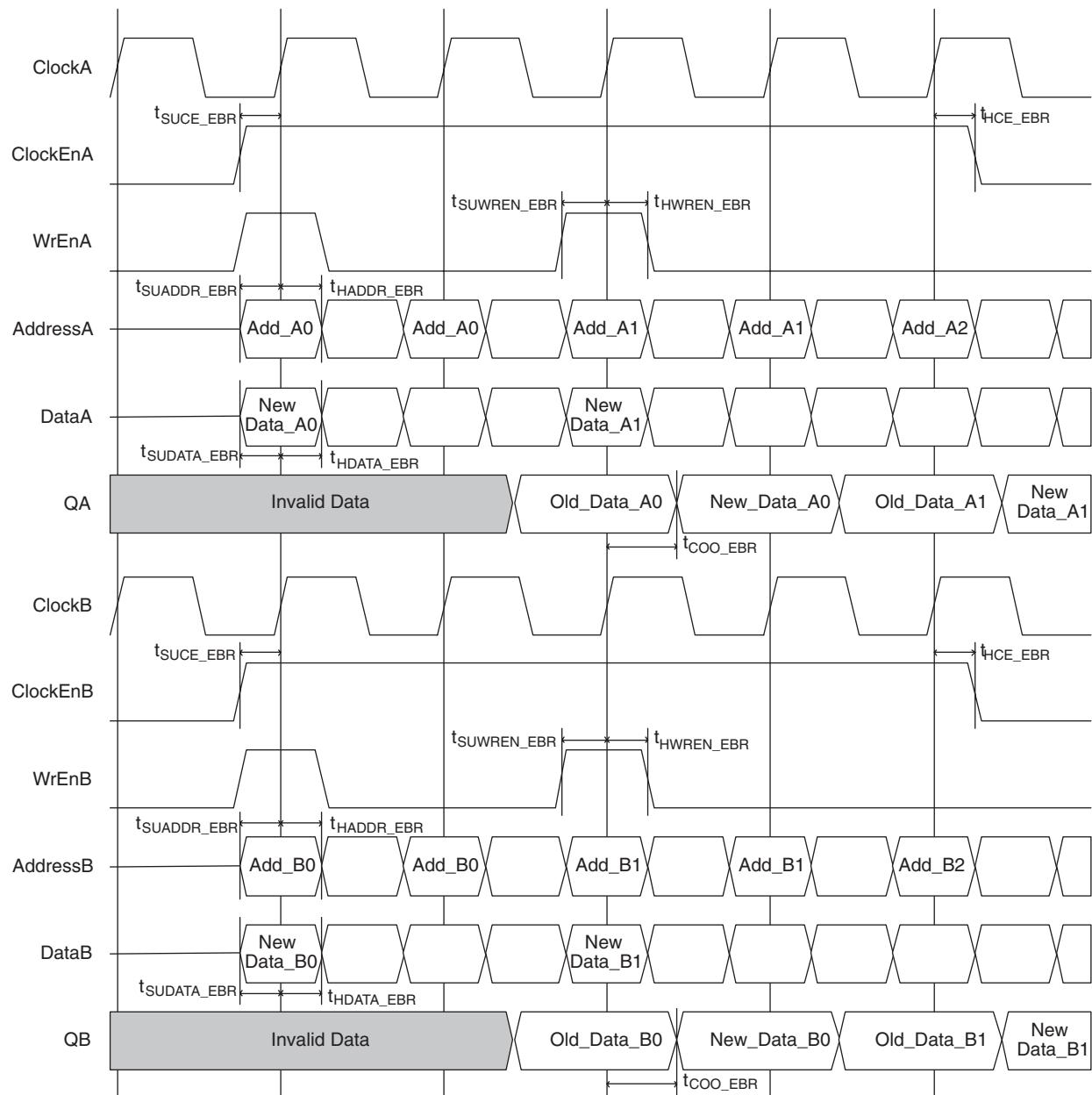
**Figure 11-13. True Dual Port RAM Timing Waveform - NORMAL Mode, without Output Registers**

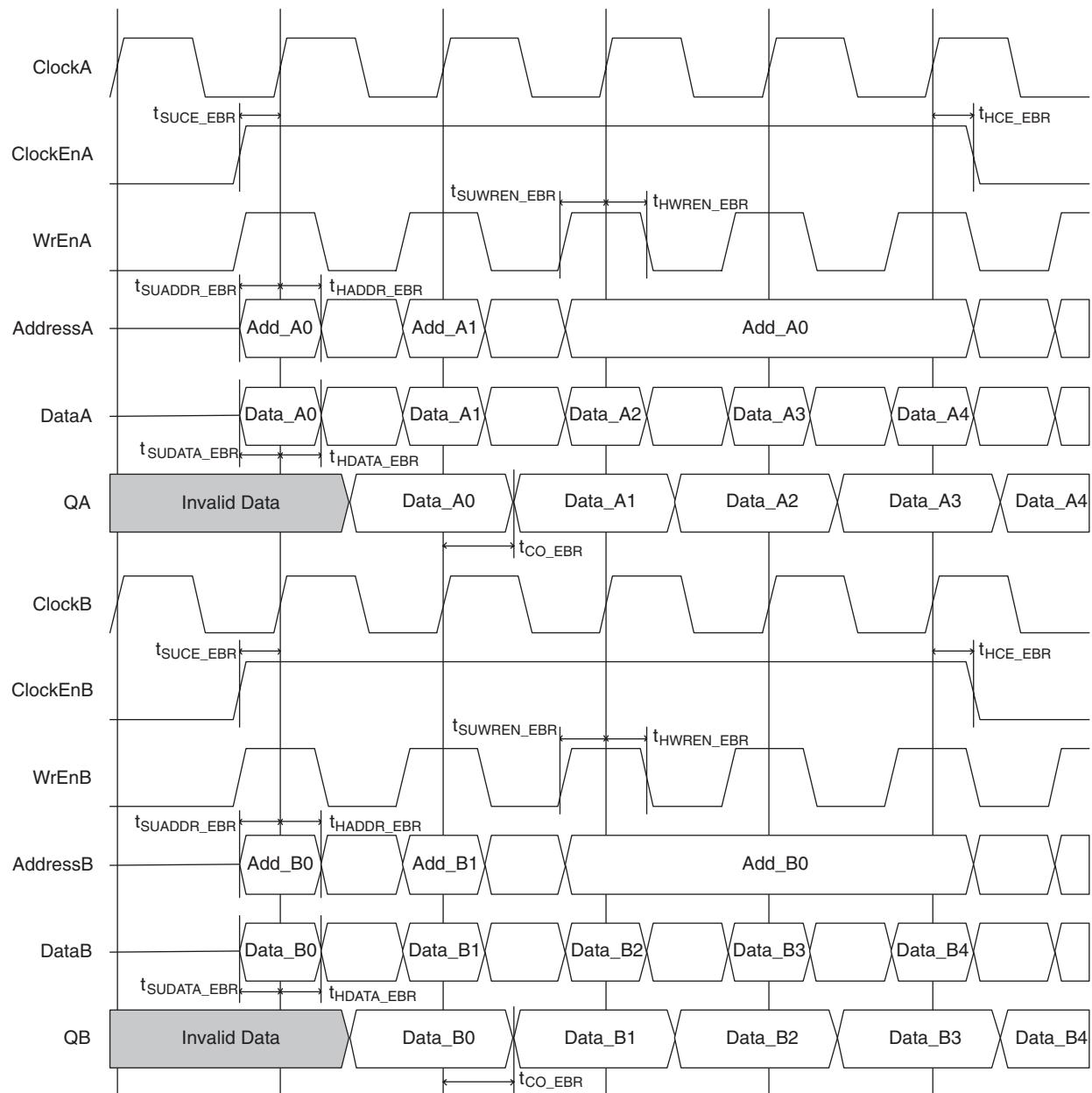


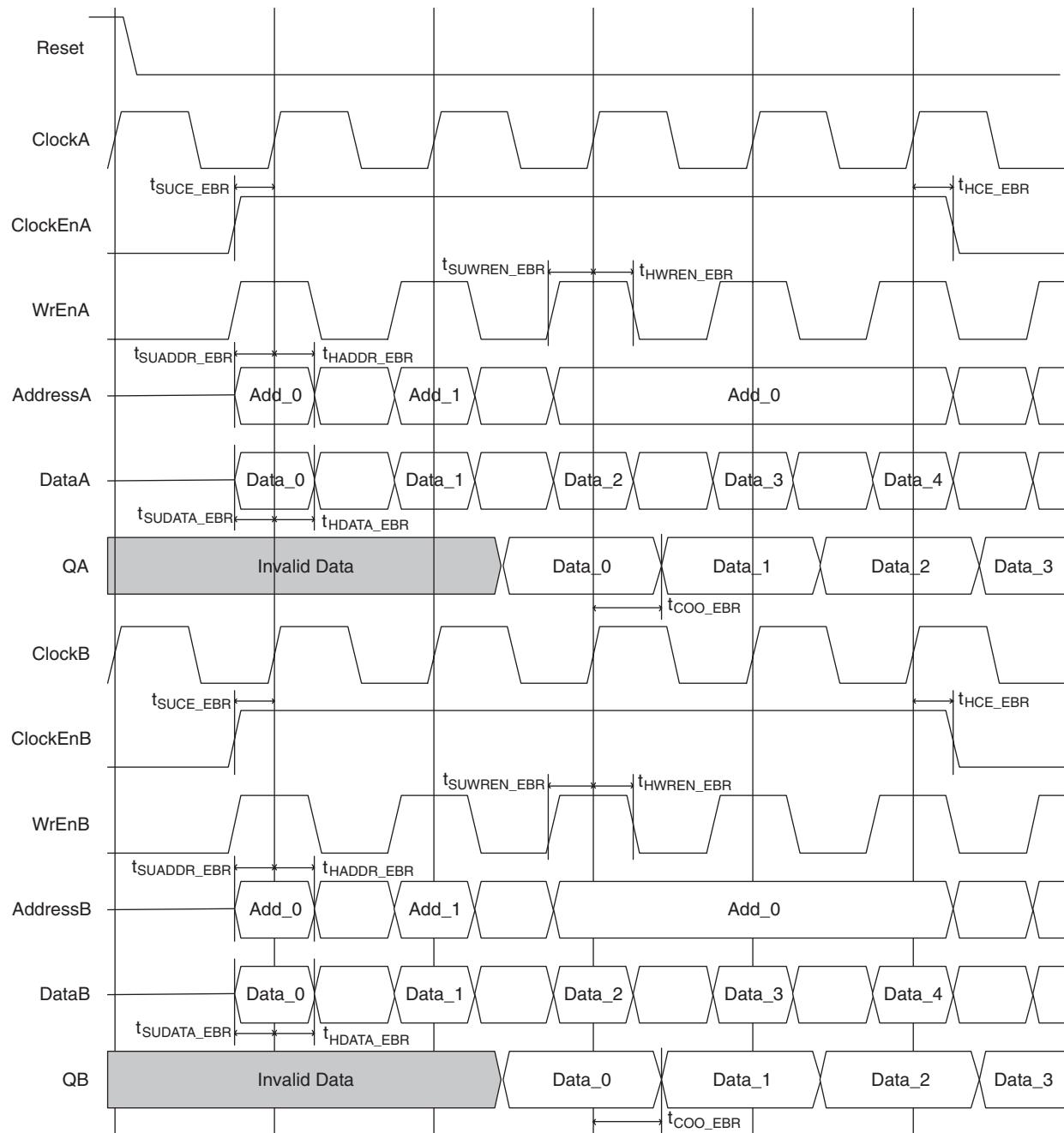
**Figure 11-14. True Dual Port RAM Timing Waveform - NORMAL Mode with Output Registers**

**Figure 11-15. True Dual Port RAM Timing Waveform - READ BEFORE WRITE Mode, without Output Registers**



**Figure 11-16. True Dual Port RAM Timing Waveform - READ BEFORE WRITE Mode, with Output Registers**

**Figure 11-17. True Dual Port RAM Timing Waveform - WRITE THROUGH Mode, without Output Registers**

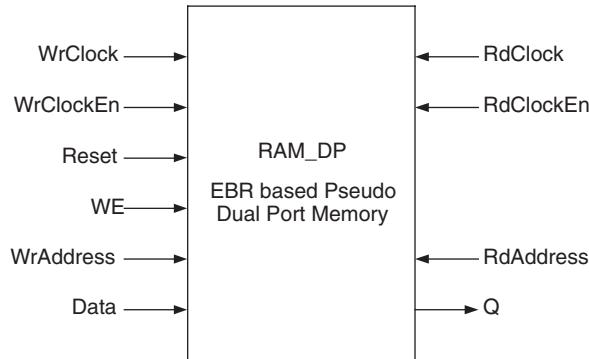
**Figure 11-18. True Dual Port RAM Timing Waveform - WRITE THROUGH Mode, with Output Registers**

## Pseudo Dual Port RAM (RAM\_DP) – EBR Based

The EBR blocks in LatticeECP2/M 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 as shown in Figure 11-19.

**Figure 11-19. Pseudo Dual Port Memory Module Generated by IPExpress**



The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. If 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 the Single Port Memory are listed in Table 11-8. The table lists the corresponding ports for the module generated by IPExpress and for the EBR RAM\_DP primitive.

**Table 11-8. EBR-based Pseudo-Dual Port Memory Port Definitions**

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
RdAddress	ADR[x1:0]	Read Address	—
WrAddress	ADW[x2:0]	Write Address	—
RdClock	CLKR	Read Clock	Rising Clock Edge
WrClock	CLKW	Write Clock	Rising Clock Edge
RdClockEn	CER	Read Clock Enable	Active High
WrClockEn	CEW	Write Clock Enable	Active High
Q	DO[y1:0]	Read Data	—
Data	DI[y2:0]	Write Data	—
WE	WE	Write Enable	Active High
Reset	RST	Reset	Active High
—	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.

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 are as in Table 11-9.

**Table 11-9. Pseudo-Dual Port Memory Sizes for 16K Memory for LatticeECP2/M**

Pseudo-Dual Port Memory Size	Input Data Port A	Input Data Port B	Output Data Port A	Output Data Port B	Read Address Port A [MSB:LSB]	Write Address Port B [MSB:LSB]
16K x 1	DIA	DIB	DOA	DOB	RAD[13:0]	WAD[13:0]
8K x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	RAD[12:0]	WAD[12:0]
4K x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	RAD[11:0]	WAD[11:0]
2K x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	RAD[10:0]	WAD[10:0]
1K x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	RAD[9:0]	WAD[9:0]
512 x 36	DIA[35:0]	DIB[35:0]	DOA[35:0]	DOB[35:0]	RAD[8:0]	WAD[8:0]

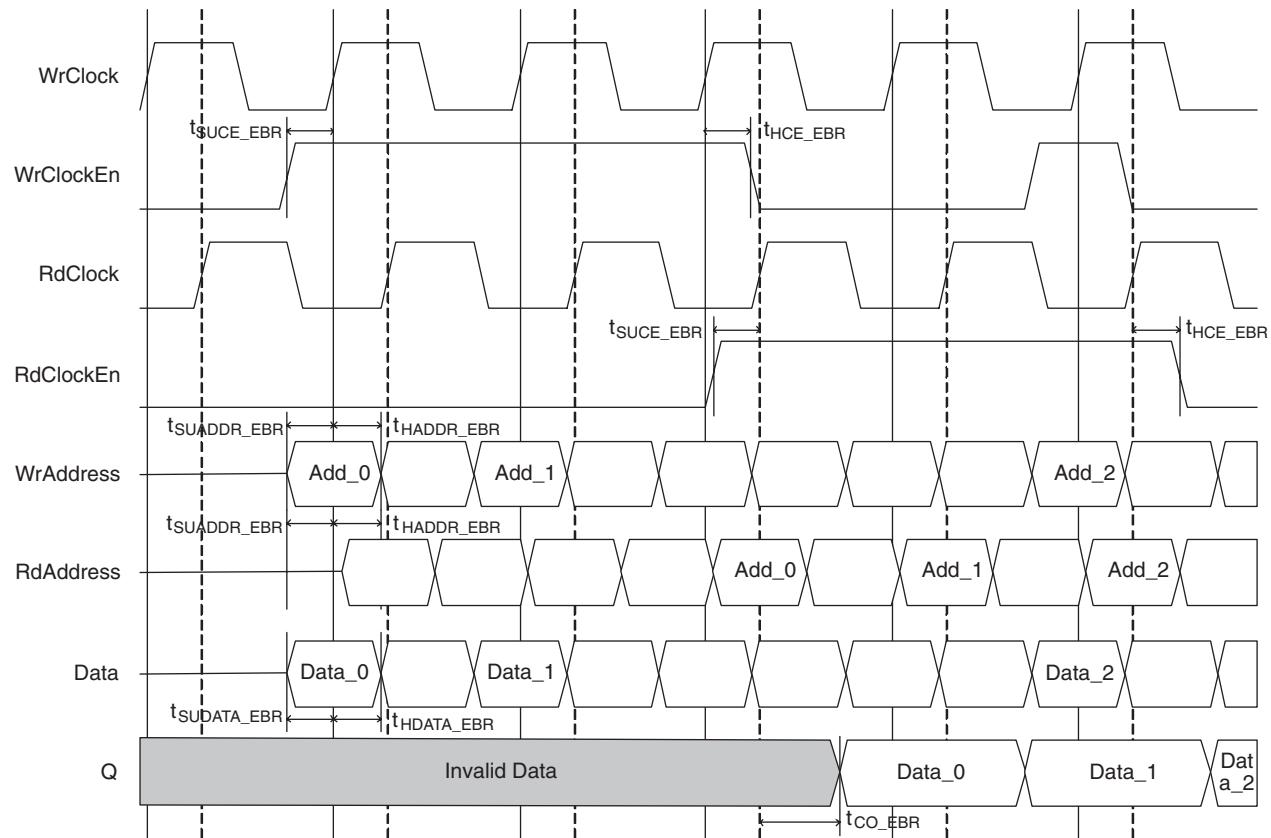
Table 11-10 shows the various attributes available for the Pseudo-Dual Port Memory (RAM\_DP). Some of these attributes are user-selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

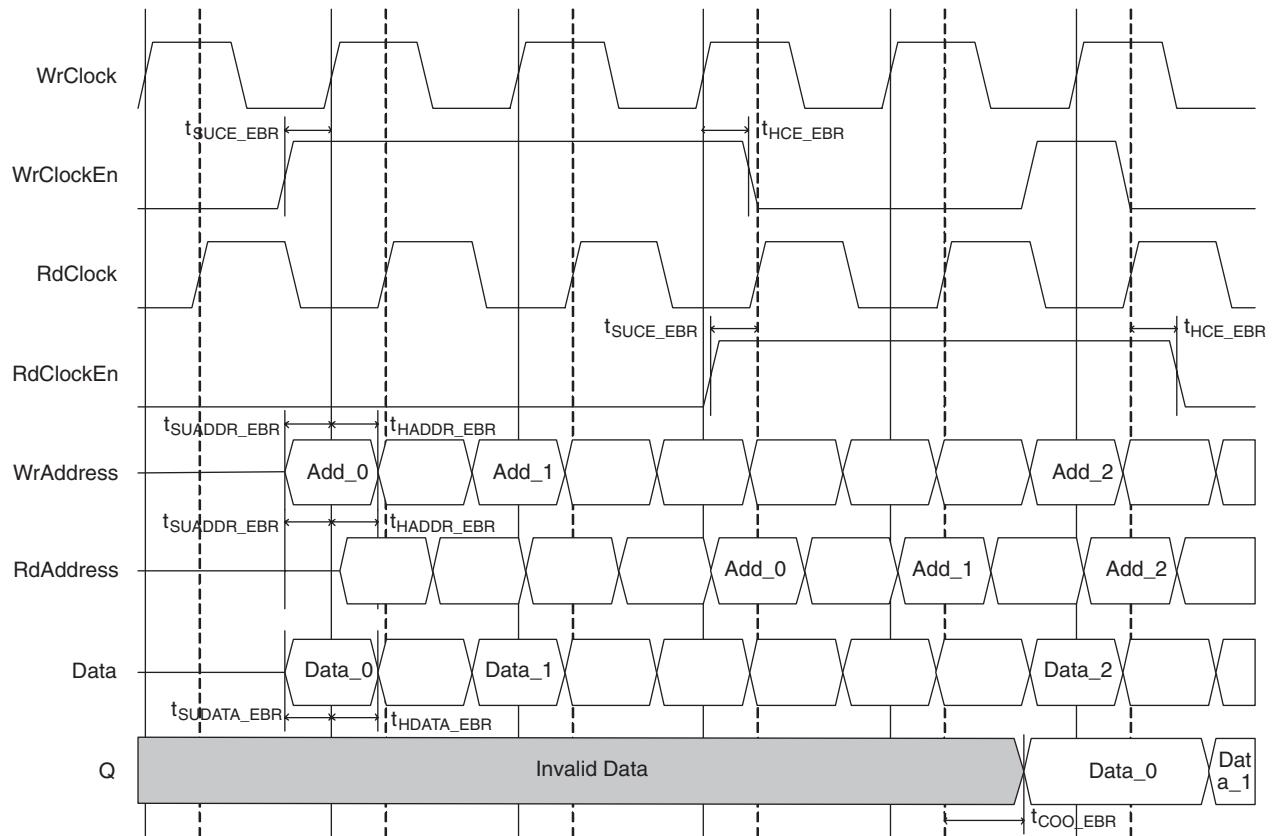
**Table 11-10. Pseudo-Dual Port RAM Attributes for LatticeECP2/M**

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, WRITETHROUGH, READBEFOREWRITE	NORMAL	YES
Write Port Write Mode	Read / Write Mode for Write Port	NORMAL, WRITETHROUGH, READBEFOREWRITE	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.....0xFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFFFFFFFF FFFFFFFFFFFFF	0x0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000 0000000000000000	NO

Users have the option to enable the output registers for Pseudo-Dual Port RAM (RAM\_DP). Figures 11-20 and 11-21 show the internal timing waveforms for Pseudo-Dual Port RAM (RAM\_DP) with these options.

**Figure 11-20. PSEUDO DUAL PORT RAM Timing Diagram - without Output Registers**

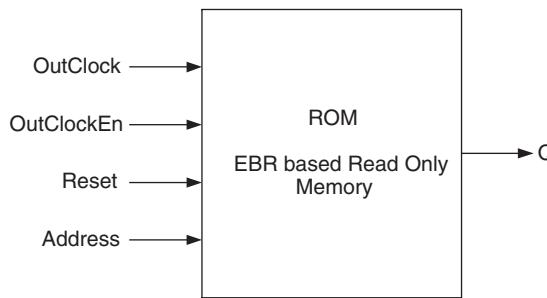


**Figure 11-21. PSEUDO DUAL PORT RAM Timing Diagram - with Output Registers**

### Read Only Memory (ROM) - EBR Based

The EBR blocks in the LatticeECP2/M devices can be configured as Read Only Memory or ROM. IPexpress allows users to generate the Verilog-HDL or VHDL and the EDIF netlist for the memory size, as per design requirements. Users are required to provide the ROM memory content in the form of an initialization file.

IPexpress generates the memory module as shown in Figure 11-22.

**Figure 11-22. Read-Only Memory Module Generated by IPexpress**

The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. If 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 ROM mode, the address for the port is 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 the ROM are listed in Table 11-11. The table lists the corresponding ports for the module generated by IPexpress and for the ROM primitive.

**Table 11-11. EBR-based ROM Port Definitions**

Port Name in Generated Module	Port Name in the EBR block Primitive	Description	Active State
Address	AD[x:0]	Read Address	—
OutClock	CLK	Clock	Rising Clock Edge
OutClockEn	CE	Clock Enable	Active High
Reset	RST	Reset	Active High
—	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.

While generating the ROM using IPexpress, the user 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 the 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 11-23 and 11-24 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 are as per Table 11-12.

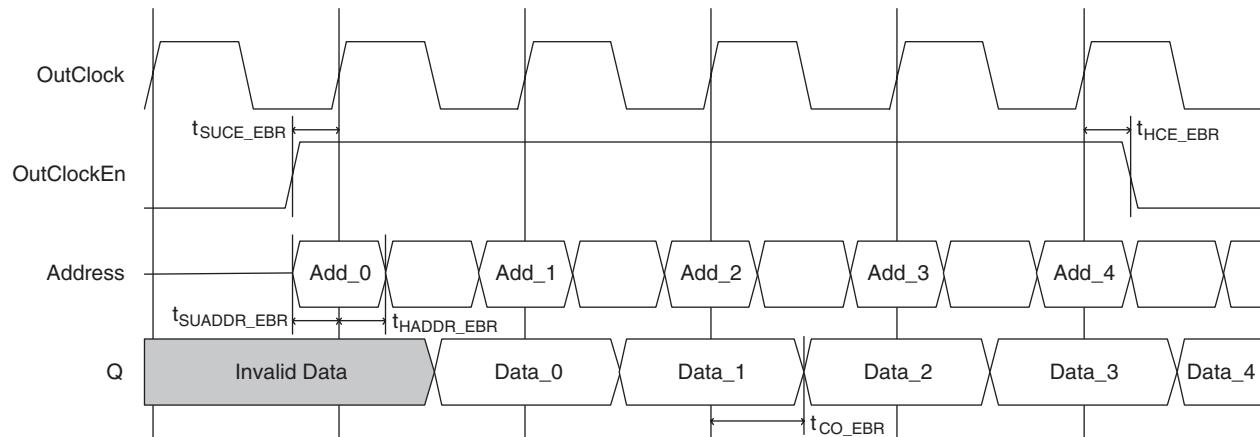
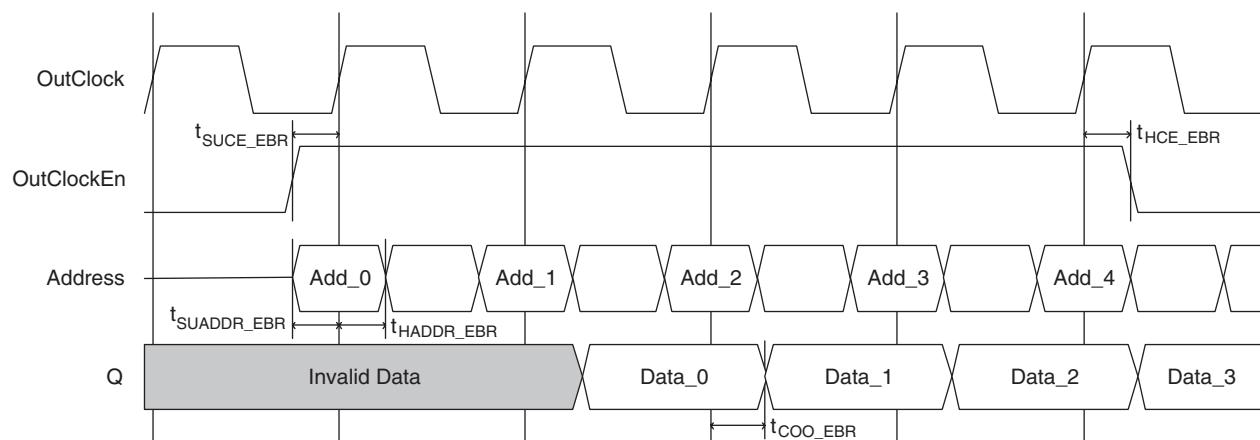
**Table 11-12. ROM Memory Sizes for 16K Memory for LatticeECP2/M**

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]

Table 11-13 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 11-13. Pseudo-Dual Port RAM Attributes for LatticeECP2/M**

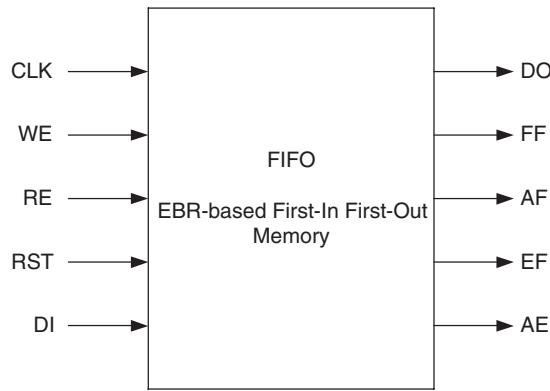
Attribute	Description	Values	Default Value	User Selectable Through IPexpress
Address depth	Address Depth Read Port	16K, 8K, 4K, 2K, 1K, 512		YES
Data Width	Data Word Width Read Port	1, 2, 4, 9, 18, 36	1	YES
Enable Output Registers	Register Mode (Pipelining) for Write Port	NOREG, OUTREG	NOREG	YES
Enable 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

**Figure 11-23. ROM Timing Waveform - without Output Registers****Figure 11-24. ROM Timing Waveform - with Output Registers**

### First In First Out (FIFO, FIFO\_DC) – EBR Based

The EBR blocks in LatticeECP2/M devices can be configured as Dual Clock First In First Out Memories, FIFO\_DC. IPexpress allows users to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size, according to design requirements.

IPexpress generates the FIFO\_DC memory module as shown in Figure 11-25.

**Figure 11-25. FIFO Module Generated by IPexpress**

The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. If 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).

A clock is always required, as only synchronous write is supported. The various ports and their definitions for the FIFO\_DC are listed in Figure 11-14.

**Table 11-14. EBR based FIFO\_DC Memory Port Definitions**

Port Name in Generated Module	Port Name in Primitive	Description	Active State
CLKR		Read Port Clock	Rising Clock Edge
CLKW		Write Port Clock	Rising Clock Edge
WE		Write Enable	Active High
RE		Read Enable	Active High
RST		Reset	Active High
DI		Data Input	—
DO		Data Output	—
FF		Full Flag	Active High
AF		Almost Full Flag	Active High
EF		Empty Flag	Active High
AE		Almost Empty	Active High

Reset (or RST) only resets the input and output registers of the RAM. It does not reset the contents of the memory.

The various supported sizes for the FIFO\_DC for LatticeECP2/M are as per Table 11-15.

**Table 11-15. FIFO\_DC Data Widths Sizes for LatticeECP2/M**

FIFO Size	Input Data	Output Data
16K x 1	DI	DO
8K x 2	DI[1:0]	DO[1:0]
4K x 4	DI[3:0]	DO[3:0]
2K x 9	DI[8:0]	DO[8:0]
1K x 18	DI[17:0]	DO[17:0]
512 x 36	DI[35:0]	DO[35:0]

Table 11-16 shows the various attributes available for the FIFO\_DC. Some of these attributes are user-selectable through the IPExpress GUI. For detailed attribute definitions, refer to the Appendix A.

**Table 11-16. FIFO\_DC Attributes for LatticeECP2/M**

Attribute	Description	Values	Default Value	User Selectable Through IPExpress
Write Port Depth	Number of Address locations	16K, 8K, 4K, 2K, 1K, 512		YES
Write Port Width	Data Port Width	1, 2, 4, 9, 18, 36	1	YES
Read Port Depth	Number of Address locations	16K, 8K, 4K, 2K, 1K, 512		YES
Read Port Width	Data Port Width	1, 2, 4, 9, 18, 36	1	YES
Enable Output Registers	Register Mode (Pipelining) for Write Port	NOREG, OUTREG	NOREG	YES
Reset Mode	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
Almost Full	Almost Full Flag value			YES
Almost Empty	Almost Empty Flag value			YES

### FIFO\_DC Flags

The FIFO\_DC have four flags available: Empty, Almost Empty, Almost Full and Full. Almost Empty and Almost Full flags have a programmable range.

The program ranges for the four FIFO\_DC flags are specified in Table 11-17.

**Table 11-17. FIFO Flag Settings**

FIFO Attribute Name	Description	Programming Range	Program Bits
FF	Full flag setting	$2^N - 1$	15
AFF	Almost full setting	1 to (FF-1)	15
AEF	Almost empty setting	1 to (FF-1)	15
EF	Empty setting	0	5

The only restriction is that the values must be in an order (Empty=0, Almost Empty next, followed by Almost Full and Full, respectively). The value of Empty is not equal to the value of Almost Empty (or Full is equal to Almost Full). In case, a warning is generated and the value of Empty (or Full) is used in place of Almost Empty (or Almost Full). When coming out of reset, the active high flags empty and almost empty are set to high, since they are true.

The user should specify the absolute value of the address at which the Almost Empty and Almost Full flags will go true. For example, if the Almost Full flag is required to go true at the address location 500 for a FIFO of depth 512, the user should specify a value of 500 in IPExpress.

The Empty and Almost Empty flags are always registered to the read clock and the Full and Almost Full flags are always registered to the write clock.

At reset, both the write and read counters are pointing to address zero. After reset is de-asserted data can be written into the FIFO\_DC to the address pointed to by the write counter at the positive edge of the write clock when the write enable is asserted.

Similarly, data can be read from the FIFO\_DC from the address pointed to by the read counter at the positive edge of the read clock when read enable is asserted.

Read Pointer Reset (RPRReset) is used to indicate a retransmit, and is more commonly used in “packetized” communications. In this application, the user must keep careful track of when a packet is written into or read from the FIFO\_DC.

The data output of the FIFO\_DC can be registered or non-registered through a selection in IPexpress. The output registers are enabled by read enable. A reset will clear the contents of the FIFO\_DC by resetting the read and write pointers and will put the flags in the initial reset state.

### FIFO\_DC Operation

If the output registers are not enabled it will take two clock cycles to read the first word out. The register for the flag logic causes this extra clock latency. In the architecture of the emulated FIFO\_DC, the internal read enables for reading the data out is controlled not only by the read enable provided by the user but also the empty flag. When the data is written into the FIFO, an internal empty flag is registered using write clock that is enabled by write enable (WrEn). Another clock latency is added due to the clock domain transfer from write clock to read clock using another register which is clocked by read clock that is enabled by read enable.

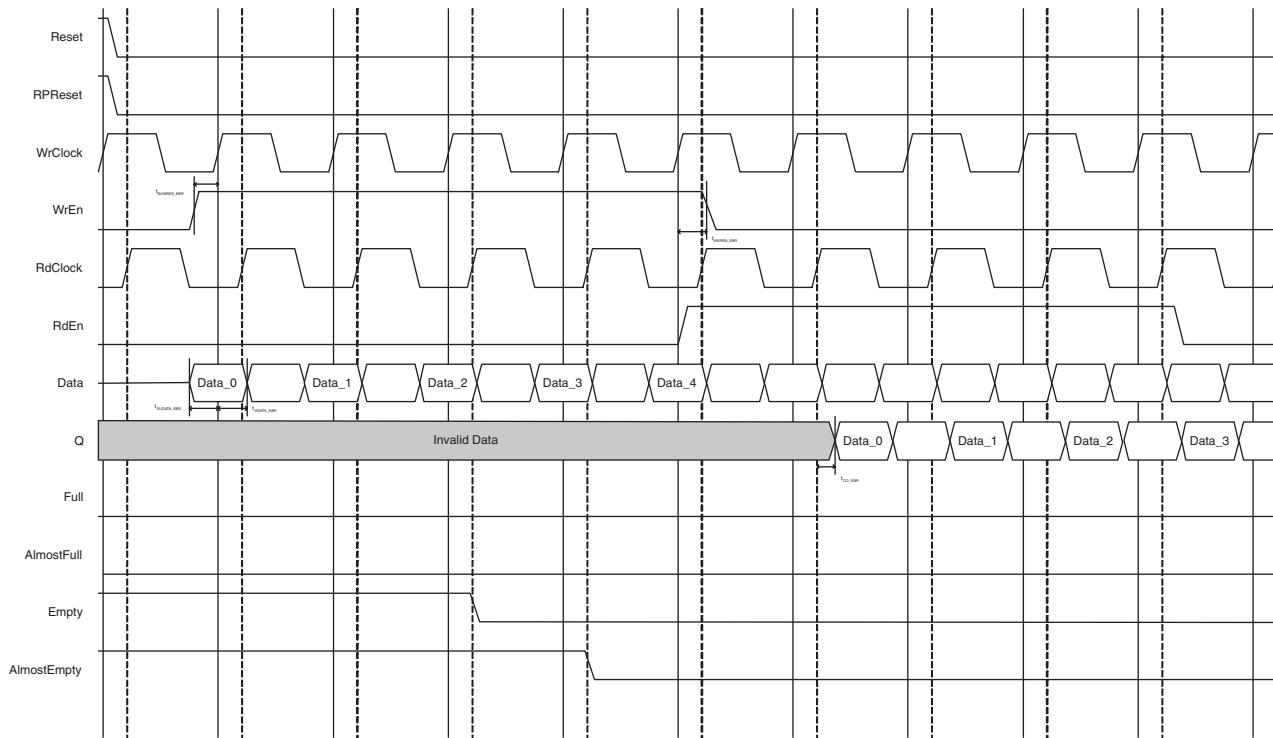
Internally, the output of this register is inverted and then ANDed with the user-provided read enable that becomes the internal read enable to the RAM\_DP which is at the core of the FIFO\_DC.

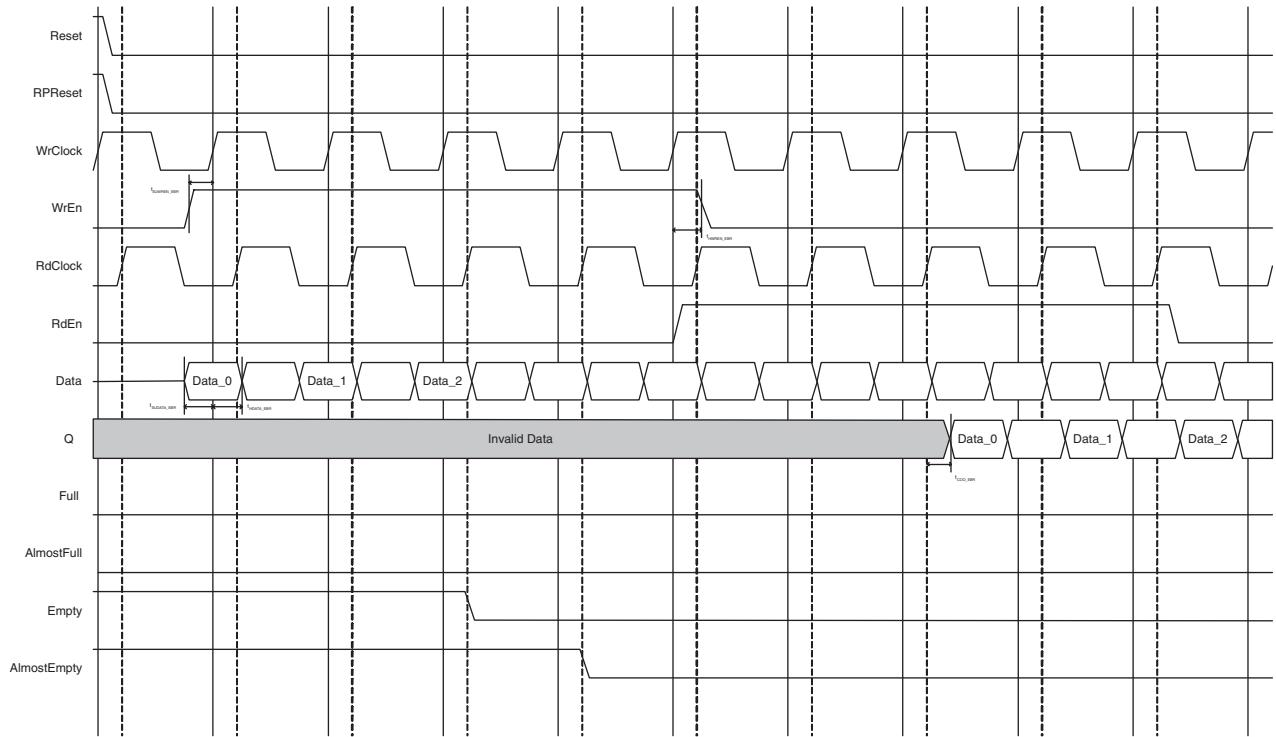
Thus, the first read data takes two clock cycles to propagate through. During the first data out, read enable goes high for one clock cycle, empty flag is de-asserted and is not propagated through the second register enabled by the read enable. The first clock cycle brings the Empty Low and the second clock cycle brings the internal read enable high (RdEn and !EF) and then the data is read out by the second clock cycle. Similarly, the first write data after the full flag has a similar latency.

If the user has enabled the output registers, the output registers will cause an extra clock delay during the first data out as they are clocked by the read clock and enabled by the read enable.

1. First RdEn and Clock Cycle to propagate the EF internally.
2. Second RdEn and Clock Cycle to generate internal Read Enable into the DPRAM.
3. Third RdEn and Clock Cycle to get the data out of the output registers.

**Figure 11-26. FIFO\_DC without Output Registers (Non-Pipelined)**

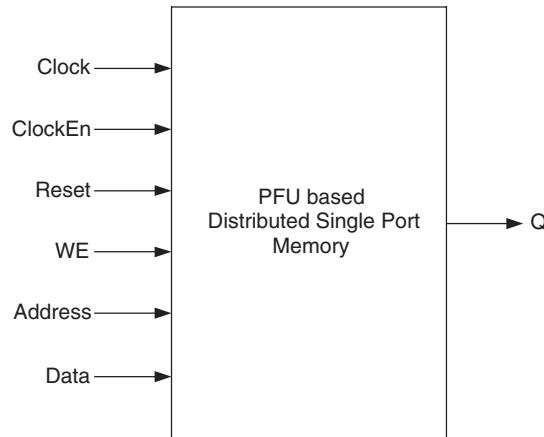


**Figure 11-27. FIFO\_CD with Output Registers (Pipelined)**

### Distributed Single Port RAM (Distributed\_SPRAM) – PFU Based

PFU-based Distributed Single Port RAM is created using the 4-input LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create larger Distributed Memory sizes.

Figure 11-28 shows the Distributed Single Port RAM module as generated by IPexpress.

**Figure 11-28. Distributed Single Port RAM Module Generated by IPexpress**

The generated module makes use 4-input LUT available in the PFU. Additional logic like Clock, 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 the to enable the output registers in their IPexpress configuration.

The various ports and their definitions for the memory are as per Table 11-18. The table lists the corresponding ports for the module generated by IPExpress and for the primitive.

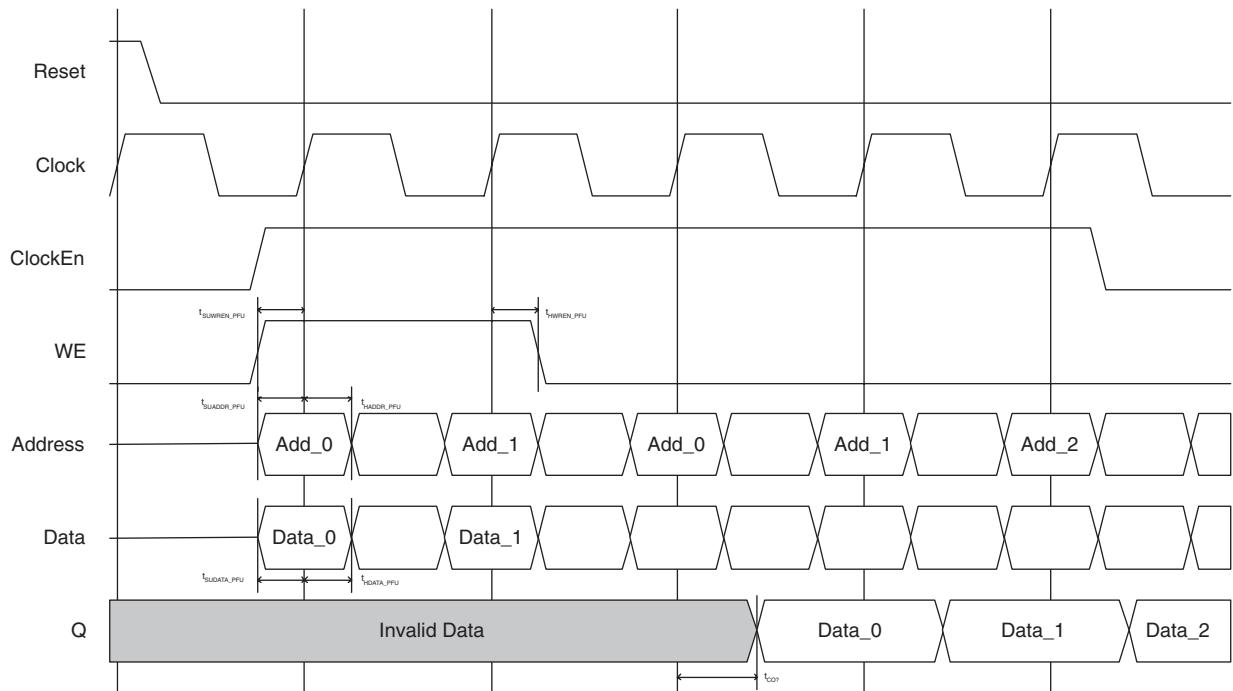
**Table 11-18. PFU-based Distributed Single Port RAM Port Definitions**

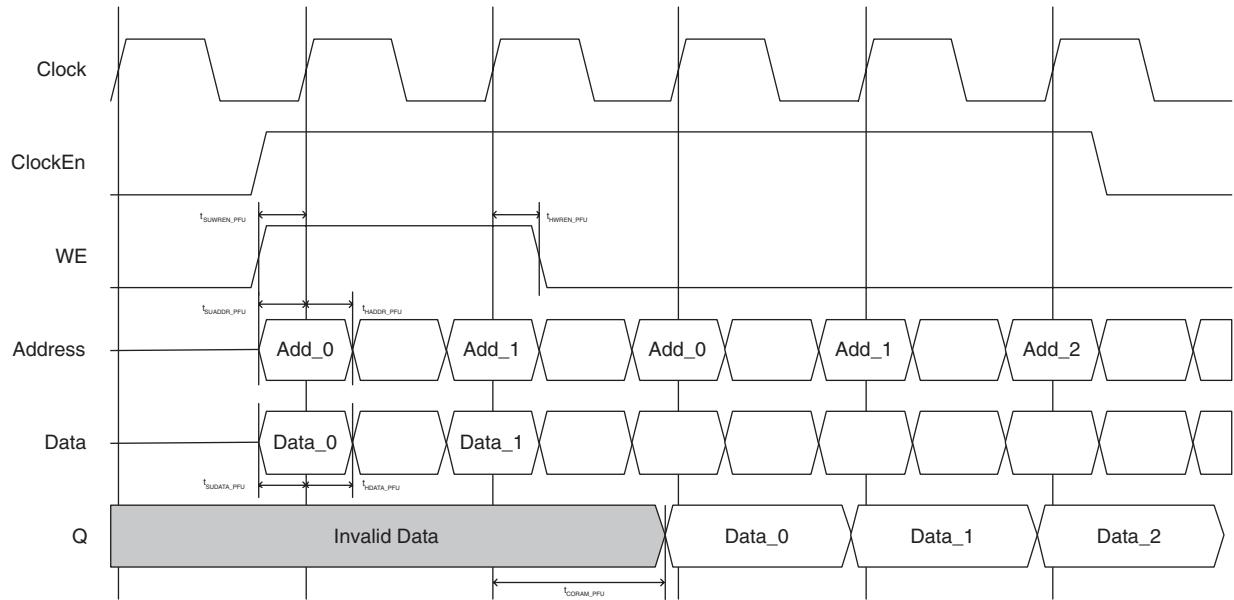
Port Name in Generated Module	Port Name in the PFU Primitive	Description	Active State
Clock	CK	Clock	Rising Clock Edge
ClockEn	—	Clock Enable	Active High
Reset	—	Reset	Active High
WE	WRE	Write Enable	Active High
Address	AD[3:0]	Address	—
Data	DI[1:0]	Data In	—
Q	DO[1:0]	Data Out	—

Ports such as Clock Enable (ClockEn) are not available in the hardware primitive. These are generated by IPExpress when the user wishes to enable the output registers in the IPExpress configuration.

Users have the option of enabling the output registers for Distributed Single Port RAM (Distributed\_SPRAM). Figures 11-29 and 11-30 show the internal timing waveforms for the Distributed Single Port RAM (Distributed\_SPRAM) with these options.

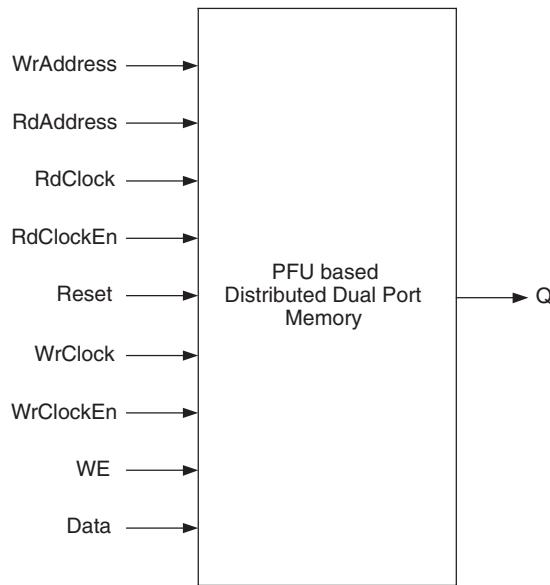
**Figure 11-29. PFU Based Distributed Single Port RAM Timing Waveform - without Output Registers**



**Figure 11-30. 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 LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create a larger Distributed Memory sizes.

**Figure 11-31. Distributed Dual Port RAM Module Generated by IPexpress**

The generated module makes use of the 4-input LUT available in the PFU. Additional logic like Clock and 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 the to enable the output registers in the IPexpress configuration.

The various ports and their definitions for memory are as per Table 11-19. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

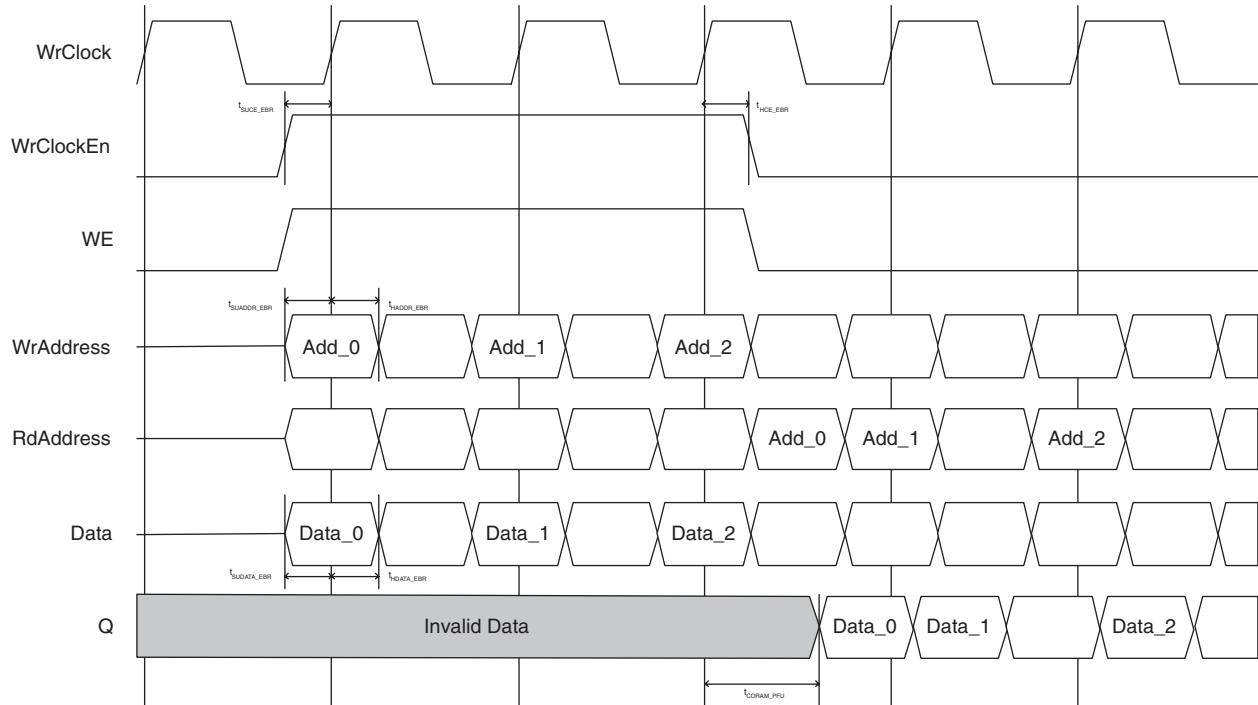
**Table 11-19. PFU-based Distributed Dual-Port RAM Port Definitions**

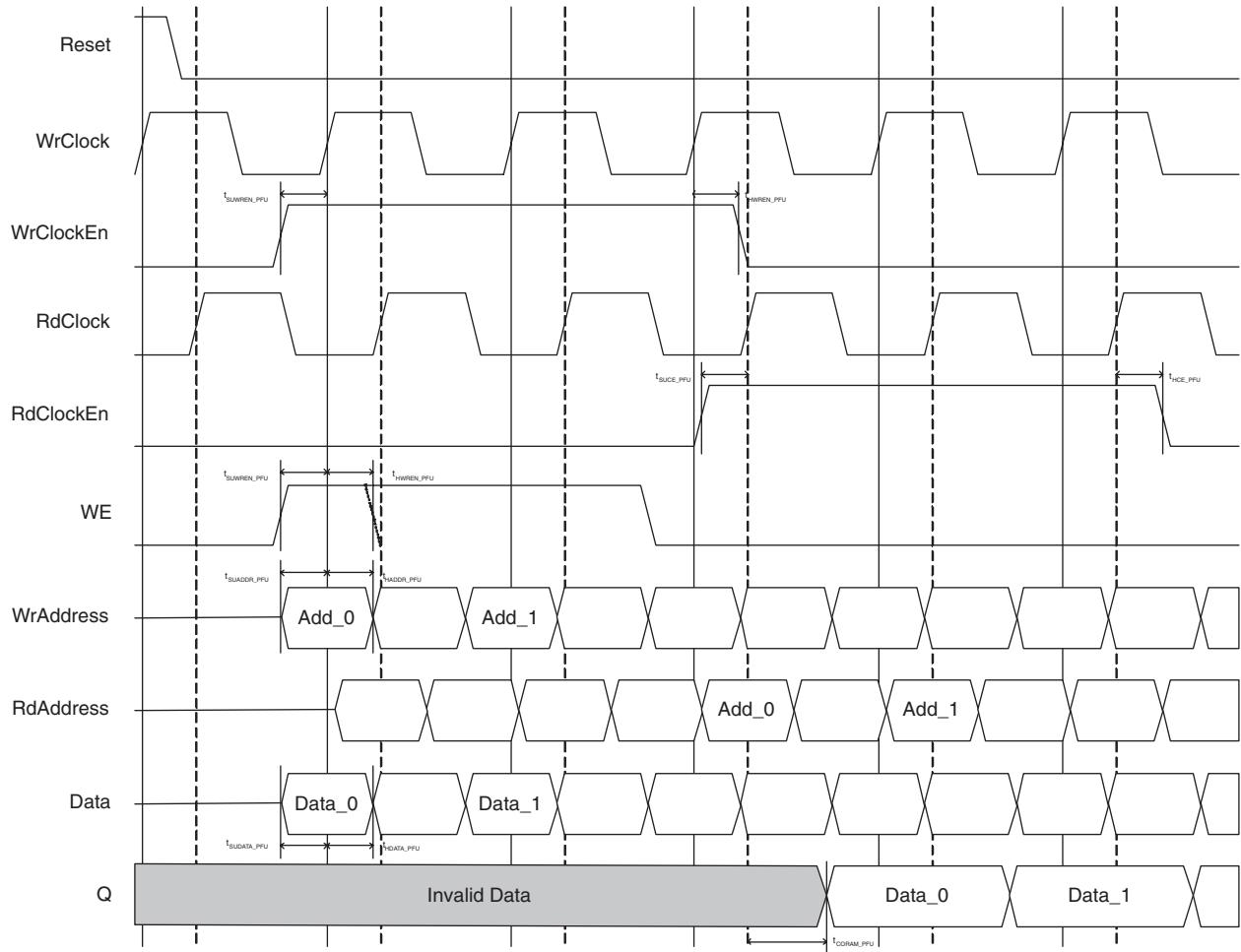
Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
WrAddress	WAD[3:0]	Write Address	—
RdAddress	RAD[3:0]	Read Address	—
RdClock	—	Read Clock	Rising Clock Edge
RdClockEn	—	Read Clock Enable	Active High
WrClock	WCK	Write Clock	Rising Clock Edge
WrClockEn	—	Write Clock Enable	Active High
WE	WRE	Write Enable	Active High
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 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 11-32 and 11-33 show the internal timing waveforms for the Distributed Dual Port RAM (Distributed\_DPRAM) with these options.

**Figure 11-32. PFU Based Distributed Dual Port RAM Timing Waveform - without Output Registers**

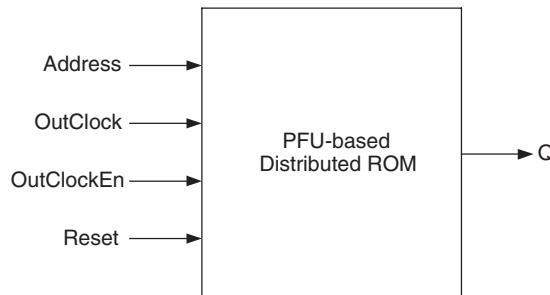


**Figure 11-33. PFU Based Distributed Dual Port RAM Timing Waveform - with Output Registers**

### Distributed ROM (Distributed\_ROM) – PFU Based

PFU-based Distributed ROM is also created using the 4-input LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create larger Distributed Memory sizes.

Figure 11-34 shows the Distributed ROM module as generated by IPexpress.

**Figure 11-34. Distributed ROM Generated by IPexpress**

The generated module makes use of the 4-input LUT available in the PFU. Additional logic like Clock and 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.

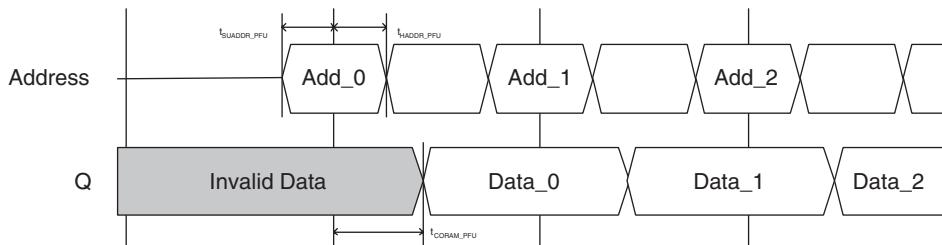
The various ports and their definitions for memory are as per Table 11-20. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

**Table 11-20. PFU-based Distributed ROM Port Definitions**

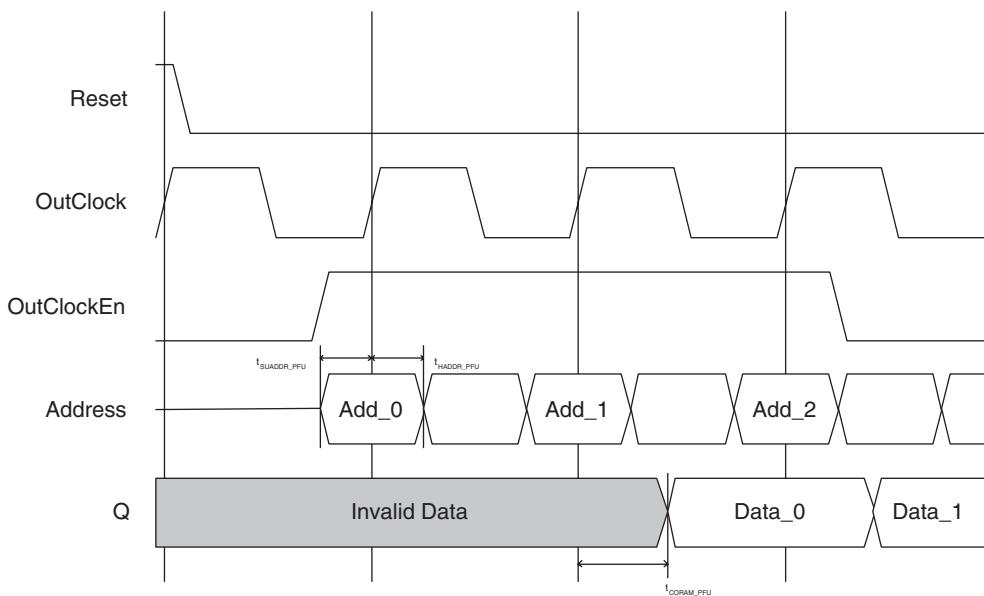
Port Name in Generated Module	Port Name in the PFU Block Primitive	Description	Active State
Address	AD[3:0]	Address	—
OutClock	—	Out Clock	Rising Clock Edge
OutClockEn	—	Out Clock Enable	Active High
Reset	—	Reset	Active High
Q	DO	Data Out	—

Users have the option to enable the output registers for Distributed ROM (Distributed\_ROM). Figures 11-35 and 11-36 show the internal timing waveforms for the Distributed ROM with these options.

**Figure 11-35. PFU Based ROM Timing Waveform – without Output Registers**



**Figure 11-36. PFU Based ROM Timing Waveform – with Output Registers**



## Initializing Memory

In the EBR based ROM or RAM memory modes and the PFU based ROM memory mode, it is possible to specify the power-on state of each bit in the memory array. Each bit in the memory array can have one of two values: 0 or 1.

### Initialization File Format

The initialization file is an ASCII file, which users can create or edit using any ASCII editor. IPExpress supports three types of memory file formats:

- Binary file
- Hex File
- Addressed Hex

The file name for the memory initialization file is \*.mem (<file\_name>.mem). Each row depicts the value to be stored in a particular memory location and 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. The EBR in RAM mode can optionally use this Initialization File also to preload the memory contents.

### Binary File

The file is essentially a text file of 0's and 1's. The rows indicate the number of words and columns indicate the width of the memory.

```
Memory Size 20x32
0010000001000000010000001000000
00000001000000010000000100000001
00000010000000100000001000000010
00000011000000110000001100000011
00000100000001000000010000000100
00000101000001010000010100000101
00000110000001100000011000000110
00000111000001110000011100000111
00001000010010000000100001001000
000010010010010000100101001001
000010100100100000101001001010
00001011010010110000101101001011
00001100000011000000110000001100
00001101001011010000110100101101
0000111000111100000111000111110
000011110011111000011100111111
000100000001000000001000000010000
00010001000100010001000100010001
00010010000100100001001000010010
00010011000100110001001100010011
```

## Hex File

The Hex file is essentially a text file of Hex characters arranged in a similar row-column arrangement. The number of rows in the file is 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 address and data. Each line starts with an address, followed by a colon, and any number of data. The format of memfile is address: data data data data ... where address and data are hexadecimal numbers.

```
-A0 : 03 F3 3E 4F
-B2 : 3B 9F
```

The first line puts 03 at address A0, F3 at address A1, 3E at address A2, and 4F at address A3. The second line puts 3B at address B2 and 9F at address B3.

There is no limitation on the values of address and data. 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. Users need not specify data at all address locations. If data is not specified at certain address, the data at that location is initialized to 0. IPexpress makes memory initialization possible both through the synthesis and simulation flows.

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)  
+1-503-268-8001 (Outside North America)  
e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2006	01.0	Initial release.
April 2006	01.1	Updated the Initializing Memory section
September 2006	01.2	Added LatticeECP2M device information. Added dual port memory access notes.

## Appendix A: Attribute Definitions

### DATA\_WIDTH

Data width is associated with the RAM and FIFO elements. The DATA\_WIDTH attribute will define the number of bits in each word. It takes the values as 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.

### RESETMODE

The RESETMODE attribute allows users to select the mode of reset in the memory. This attribute is associated with the block RAM elements. RESETMODE takes two parameters: SYNC and ASYNC. SYNC means that the memory reset is synchronized with the clock. ASYNC means that the memory reset is asynchronous to clock.

### CSDECODE

CSDECODE or the Chip Select Decode attributes are associated to block RAM elements. CS, or Chip Select, is the port available in the EBR primitive that is useful when memory requires multiple EBR blocks 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.

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.

WRITEMODE\_A and WRITEMODE\_B are used for dual port RAM elements and refer to the A and B ports in case of a True Dual Port RAM.

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 the designer implements control logic to identify this situation if it occurs and either:

1. Implement status signals to flag the read data as possibly invalid, or
2. Implement control logic to prevent the simultaneous access from both ports.

### GSR

GSR or the Global Set/ Reset attribute is used to enable or disable the global set/reset for RAM element.

---

## Introduction

LatticeECP2™ and LatticeECP2M™ devices support 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 the DDR interfaces capture data on both the rising and falling edges of the clock, thus doubling performance. The LatticeECP2/M I/Os also have dedicated circuitry to support DDR and DDR2 SDRAM memory interfaces. This technical note details the use of LatticeECP2/M devices to implement both a high-speed generic DDR interface and DDR and DDR2 memory interfaces.

## DDR and DDR2 SDRAM Interfaces Overview

A DDR SDRAM interface will transfer data at both the rising and falling edges of the clock. The DDR2 is the second generation of the DDR SRDRAM memory.

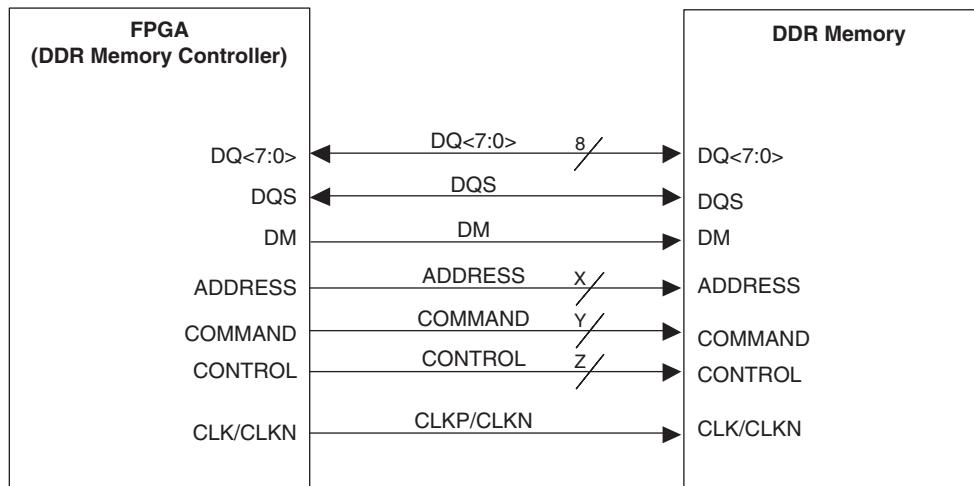
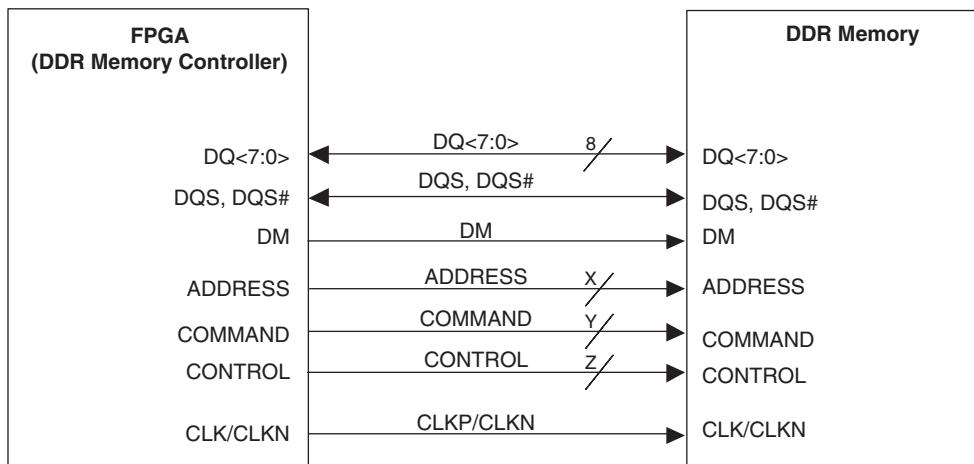
The DDR and DDR2 SDRAM interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. The DDR SDRAM interface uses a single-ended DQS strobe signal, whereas the DDR2 interface uses a differential DQS strobe. Figures 12-1 and 12-2 show typical DDR and DDR2 SDRAM interface signals. SDRAM interfaces are typically implemented with eight DQ data bits per DQS. An 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 used to both 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 degrees before FPGA logic can sample the read data. When writing to a DDR/DDR2 SDRAM, the memory controller (FPGA) must shift the DQS by 90 degrees 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. Figures 12-3 and 12-4 show DQ and DQS timing relationships for read and write cycles. For other detailed timing requirements, please refer to the DDR SDRAM JEDEC specification (JESD79C).

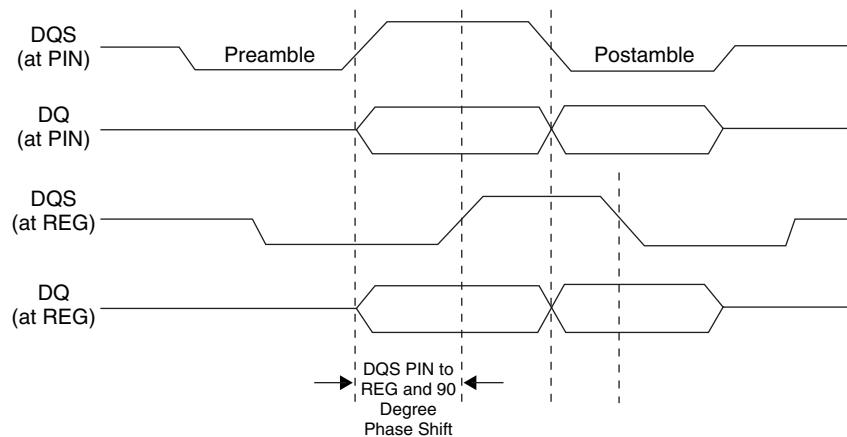
During read, the DQS signal is LOW for some duration after it comes out of tristate. This state is called Preamble. The state when the DQS is LOW before it goes into Tristate is the Postamble state. This is the state after the last valid data transition.

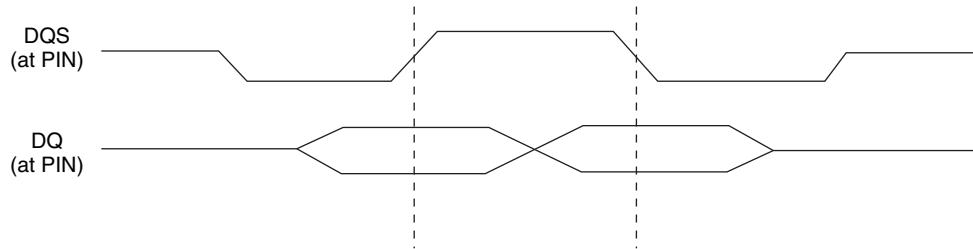
DDR SDRAM also requires a Data Mask (DM) signal to mask data bits during write cycles. Note that the ratio of DQS to data bits is independent of the overall width of the memory. An 8-bit interface will have one strobe signal.

DDR SDRAM interfaces use the SSTL25 Class I/II I/O standards whereas the DDR2 SDRAM interface uses the SSTL18 Class I/II I/O standards. The DDR2 SDRAM interface also supports differential DQS (DQS and DQS#).

**Figure 12-1. Typical DDR SDRAM Interface****Figure 12-2. Typical DDR2 SDRAM Interface**

The following two figures show the DQ and DQS relationship for memory read and write interfaces.

**Figure 12-3. DQ-DQS During READ**

**Figure 12-4. DQ-DQS During WRITE**

## Implementing DDR Memory Interfaces with LatticeECP2/M Devices

As described in the DDRSDRAM overview section, the DDR SDRAM interfaces rely primarily 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 LatticeECP2/M device is edge-aligned with respect to the DQS signal. Therefore, the LatticeECP2/M device needs to shift the DQS (a 90-degree phase shift) before using it to sample the read data. When writing to a DDR SDRAM, the memory controller from the LatticeECP2/M device must generate a DQS signal that is center-aligned with the DQ, the data signals. This is accomplished by ensuring the DQS strobe is 90 degrees ahead relative to DQ data.

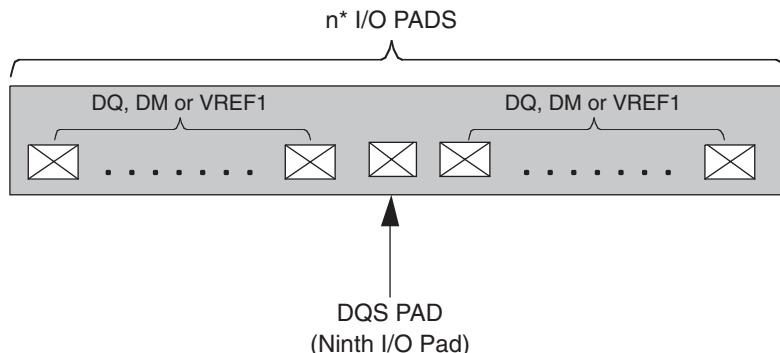
LatticeECP2/M devices have dedicated DQS support circuitry for generating the appropriate phase shifting for DQS. The DQS phase shift circuit uses a frequency reference DLL to generate delay control signals associated with each of the dedicated DQS pins and is designed to compensate for process, voltage 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 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 LatticeECP2/M devices.

### DQS Grouping

Each DQS group generally consists of at least 10 I/Os (one DQS, eight DQ and one DM) to implement a complete 8-bit DDR memory interface. LatticeECP2/M devices support DQS signals on the bottom, left and right sides of the device. Each DQS signal on the bottom half of the device will span across 18 I/Os and on the left and right sides of the device will span across 16 I/Os. Any 10 of these 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 12-5. DQ-DQS Grouping**

\*n=18 on bottom banks and n=16 on the left and right side banks.

Figure 12-5 shows a typical DQ-DQS group for a LatticeECP2/M device. The ninth I/O of this group of 16 I/Os (on the left and right side banks) and 14 I/Os (on the bottom bank) is the dedicated DQS pin. All eight pads before of the DQS and 7 (on the left and right side) and 9 (on the bottom bank) pads after the DQS are covered by this DQS bus span. The user can assign any eight of these 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. Each of the dedicated DQS pins is internally connected to the DQS phase shift circuitry. The pinout information contained in the LatticeECP2/M Family Data Sheet shows pin locations for the DQS pads. Table 12-1 shows an extract from the data sheet.

In this case, the DQS is marked as LDQS8 (L = left side, 8 = associated PFU row/column). Since DQS is always the fifth True Pad in the DQ-DQS group, counting from low to high PFU Row/Column number, LDQS6 will cover PL2A to PL11B. Following this convention, there are eight pads before and seven pads after DQS for DQ available following counter-clockwise for the left and bottom sides of the device and following clockwise for the top and right sides of the device. The user can assign any eight of these pads to be DQ data signals.

**Table 12-1. ECP2-50 672 fpBGA Pinout from LatticeECP2/M Family Data Sheet**

Ball Number	Ball Function	Bank	Dual Function	Differential
D2	PL2A	7	VREF2_7	T*
D1	PL2B	7	VREF1_7	C*
GND	GNDIO	7		
F6	PL5A	7		T
F5	PL5B	7		C
VCCIO	VCCIO	7		
E4	PL6A	7		T*
E3	PL6B	7		C*
VCC	VCC	7		
E2	PL7A	7		T
E1	PL7B	7		C
GND	GNDIO	7		
GND	GND	7		
H6	PL8A	7	LDQS8	T*
H5	PL8B	7		C*
F2	PL9A	7		T
VCCIO	VCCIO	7		
F1	PL9B	7		C
H8	PL10A	7		T*
J9	PL10B	7		C*
G4	PL11A	7		T
GND	GNDIO	7		
G3	PL11B	7		C
H7	PL12A	7		T*
VCCAUX	VCCAUX	7		

## DDR Software Primitives

This section describes the software primitives that can be used to implement DDR interfaces. These primitives include:

- **DQSDLL** – The DQS delay calibration DLL
- **DQSBUFC** – The DQS delay function and the clock polarity selection logic
- **IDDRXMX1A** – The DDR input and DQS to system clock transfer registers with half clock cycle transfer
- **IDDRXMF1A** – The DDR input and DQS to system clock transfer registers with full clock cycle transfer
- **ODDRMxa** – The DDR output registers

HDL usage examples for each of these primitives are listed in Appendices A and B.

### DQSDLL

The DQSDLL generates a 90-degree phase shift required for the DQS signal. This primitive implements 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 generates the delay based on this clock frequency and the update control input to this block. The DLL updates the dynamic delay control to the DQS delay block when this update control (UDDCNTL) input is asserted. Figure 12-6 shows the primitive symbol. The active low signal on UDDCNTL updates the DQS phase alignment and should be initiated at the beginning of READ cycles.

**Figure 12-6. DQSDLL Symbol**

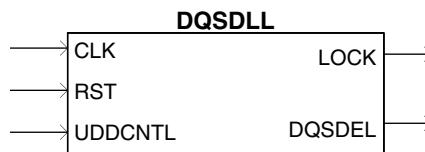


Table 12-2 provides a description of the ports.

**Table 12-2. DQSDLL Ports**

Port Name	I/O	Description
CLK	I	System CLK should be at the frequency of the DDR interface from the FPGA core.
RST	I	Resets the DQSDLL
UDDCNTL	I	Provides an update signal to the DLL that will update the dynamic delay. When held low, this signal will update the DQSDEL.
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.

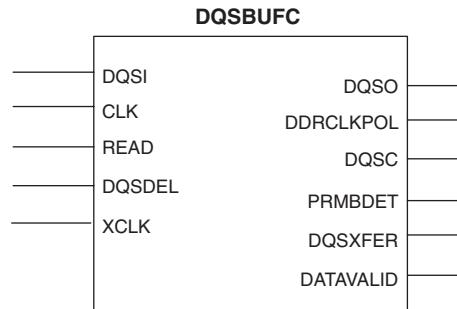
**DQSDLL Configuration:** By default, this DLL will generate a 90-degree phase shift for the DQS strobe based on the frequency of the input reference clock to the DLL. The user can control the sensitivity to jitter by using the **LOCK\_SENSITIVITY** attribute. This configuration bit can be programmed to be either “HIGH” or “LOW”.

The DLL Lock Detect circuit has two modes of operation controlled by the **LOCK\_SENSITIVITY** bit, which selects more or less sensitivity to jitter. If this DLL is operated at or above 150 MHz, it is recommended that the **LOCK\_SENSITIVITY** bit be programmed “HIGH” (more sensitive). When running at or below 100 MHz, it is recommended that the bit be programmed “LOW” (more tolerant). For 133 MHz, the **LOCK\_SENSITIVITY** bit can go either way.

**DQSBUFC**

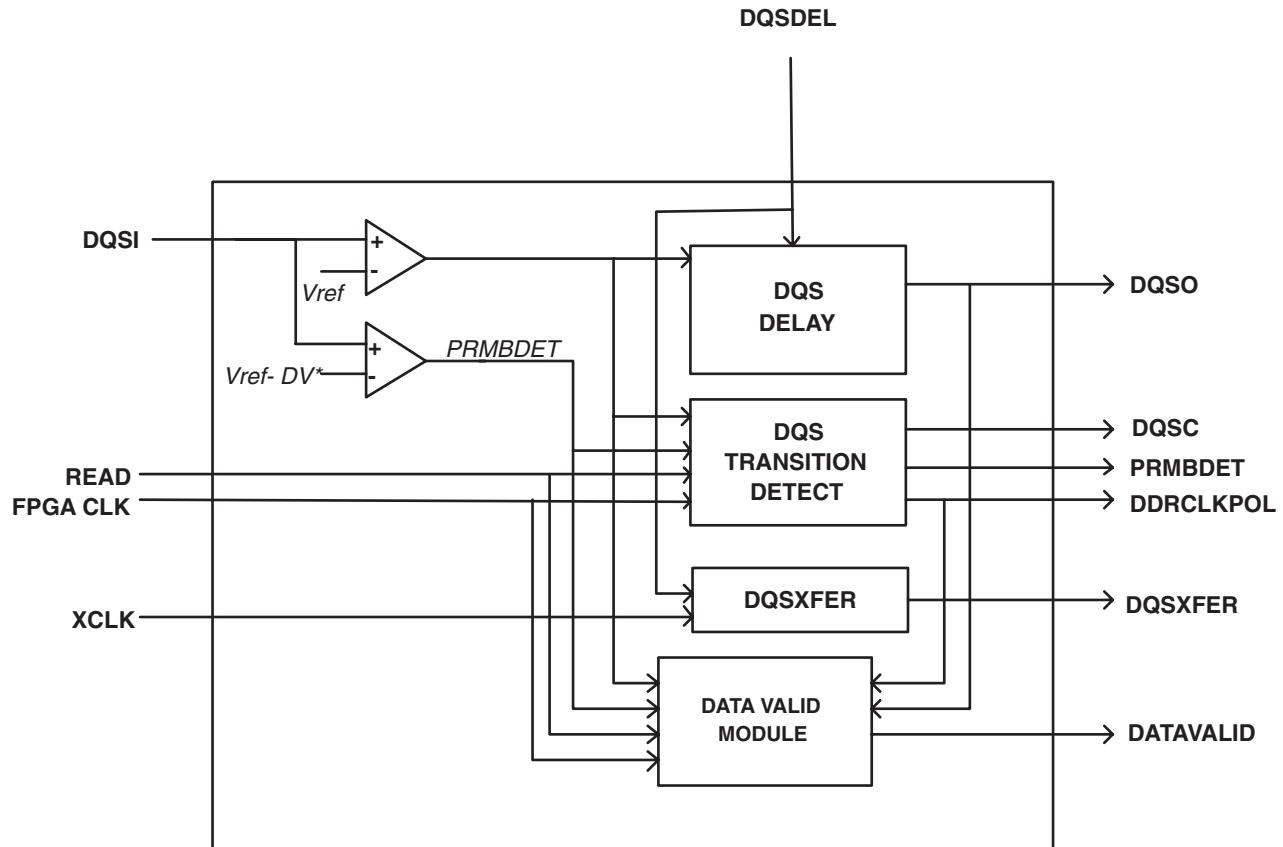
This primitive implements the DQS delay and the DQS transition detector logic. Figure 12-7 shows the primitive symbol.

**Figure 12-7. DQSBUFC Symbol**



The DQSBUFC is composed of the DQS Delay, the DQS Transition Detect and the DQSXFER block as shown in Figure 12-8. This block inputs the DQS and delays it by 90 degrees. It also generates the DDR Clock Polarity and the DQSXFER signal. The preamble detect (PRMBDET) signal is generated from the DQSI input using a voltage divider circuit.

**Figure 12-8. DQSBUFC Function**



\* $DV \sim 170mV$  for DDR1 (SSTL25 signaling)

\* $DV \sim 120mV$  for DDR2 (SSTL18 signaling)

**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 degrees. DQSO is the delayed DQS and is connected to the clock input of the first set of DDR registers.

**DQS Transition Detect:** The DQS Transition Detect block generates the DDR Clock Polarity signal based on the phase of the FPGA clock at the first DQS transition. The DDR READ control signal and FPGA CLK inputs to this coming and should be coming from the FPGA core.

**DQSXFER:** This block generates the 90-degree phase shifted clock to for the DDR Write interface. The input to this block is the XCLK. The user can choose to connect this either to the edge clock or the FPGA clocks. The DQSXFER is routed using the DQSXFER tree to all the I/Os spanned by that DQS.

**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.

Table 12-3 provides a description of the I/O ports associated with the DQSBUFC primitive.

**Table 12-3. DQSBUFC Ports**

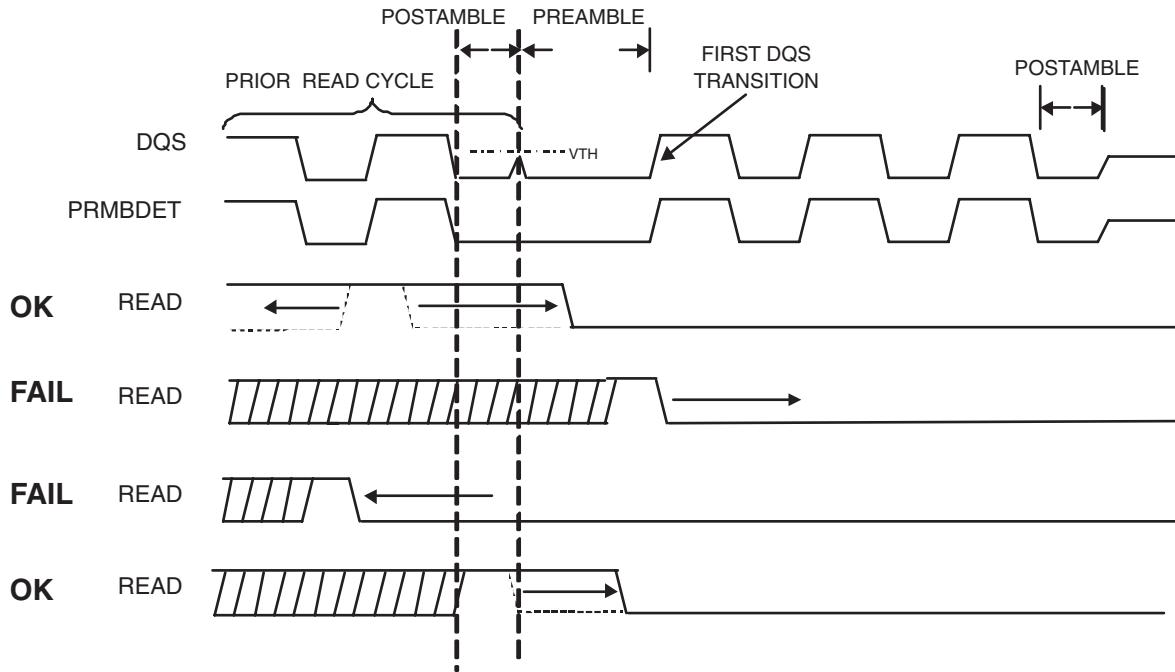
Port Name	I/O	Description
DQSI	I	DQS Strobe signal from memory
CLK	I	System CLK
READ	I	Read generated from the FPGA core
DQSDEL	I	DQS Delay from the DQSDLL primitive
XCLK	I	Edge Clock or System CLK
DQSO	O	Delayed DQS Strobe signal, to the input capture register block
DQSC	O	DQS Strobe signal before delay, going to the FPGA core logic
DDRCLKPOL	O	DDR Clock Polarity signal
PRMBDET	O	Preamble detect signal, going to the FPGA core logic
DQSXFER	O	90 degree shifted clock going to the Output DDR register Block
DATAVALID	O	Signal indicating transmission of Valid data to the FPGA core

### READ Pulse Generation

The READ signal to the DQSBUFC block is internally generated in the FPGA core. The READ signal goes high when the READ command to control the DDR-SDRAM is initially asserted. This precedes the DQS preamble by one cycle, yet may overlap the trailing bits of a prior read cycle. The DQS Detect circuitry of the LatticeECP2/M device requires the falling edge of the READ signal to be placed within the preamble stage.

The preamble state of the DQS can be detected using the CAS latency and the round trip delay for the signals between the FPGA and the memory device. Note that the internal FPGA core generates the READ pulse. The rise of the READ pulse should coincide with the initial READ command of the Read Burst and need to go low before the Preamble goes high.

Figure 12-9 shows a READ Pulse timing example with respect to the PRMBDET signal.

**Figure 12-9. READ Pulse Generation****IDDRMX1A**

This primitive will implement the input register block. The 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 (DQSBUFC primitive). 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 hardware. The DDRCLKPOL signal is used to choose the polarity of the SCLK to the synchronization registers. The LatticeECP2/M Family Data Sheet explains the Input Register Block in more detail. Figure 12-10 shows the primitive symbol and all the I/O ports.

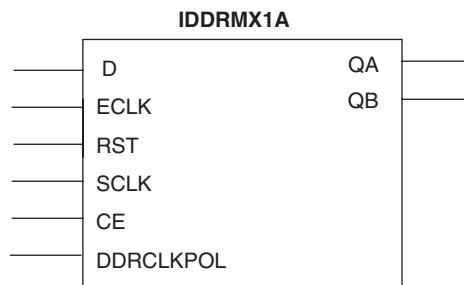
**Figure 12-10. IDDRMX1A Symbol**

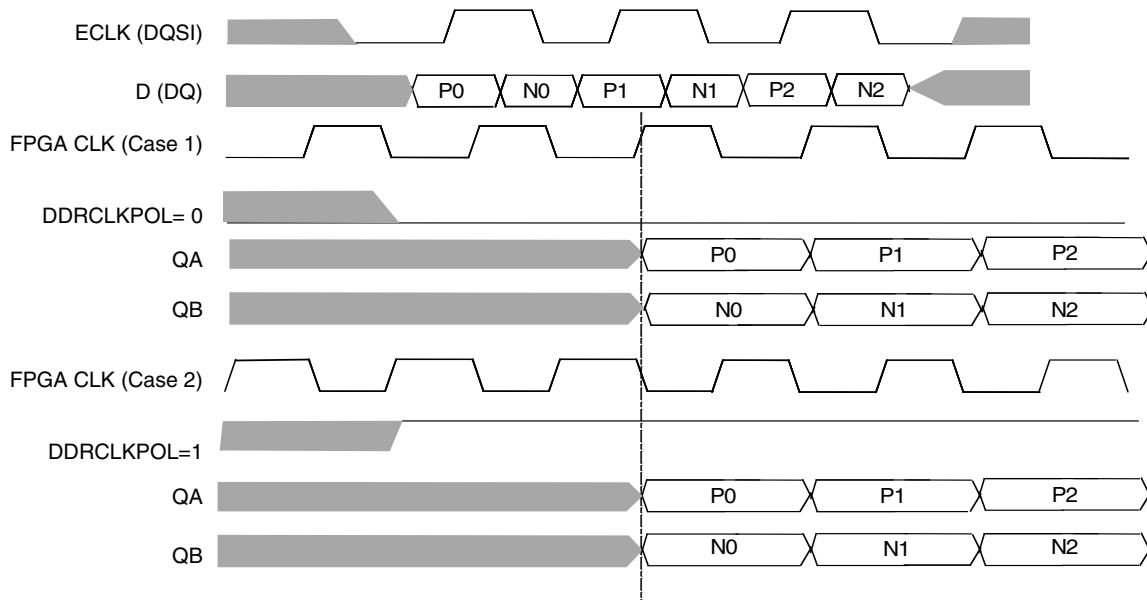
Table 12-4 provides a description of all I/O ports associated with the IDDRMX1A primitive.

**Table 12-4. IDDRMX1A Ports**

Port Name	I/O	Definition
D	I	DDR Data
ECLK	I	The phase shifted DQS should be connected to this input
RST	I	Reset
SCLK	I	System CLK
CE	I	Clock enable
DDRCLKPOL	I	DDR clock polarity signal
QA	O	Data at Positive edge of the CLK
QB	O	Data at the negative edge of the CLK

Note: The DDRCLKPOL input to IDDRMX1A should be connected to the DDRCLKPOL output of DQSBUFC.

Figure 12-11 shows the IDDRMX1A timing waveform.

**Figure 12-11. IDDRMX1A Waveform**

#### IDDRMX1A

This primitive implements a full clock cycle transfer as compared to the IDDRMX1A primitive that will only implement a half clock cycle transfer. The 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 (DQSBUFC primitive). The CLK1 and CLK2 inputs should be connected to the slow 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 hardware. The LatticeECP2/M Family Data Sheet explains the Input Register Block in more detail. Figure 12-12 shows the primitive symbol and all the I/O ports.

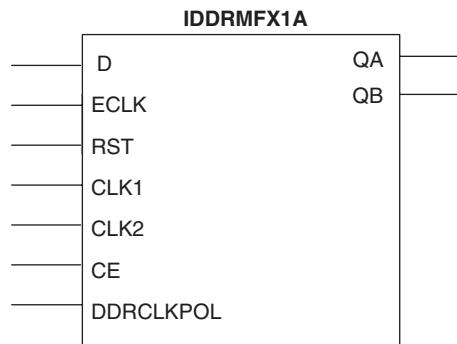
**Figure 12-12. IDDRMFX1A Symbol**

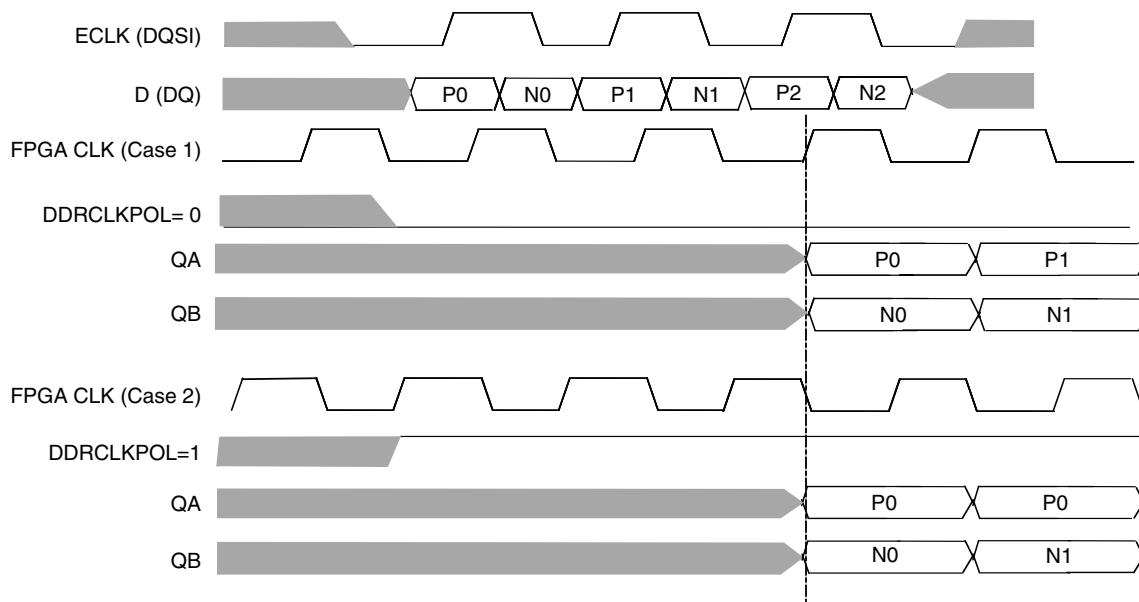
Table 12-5 provides a description of all I/O ports associated with the IDDRMFX1A primitive.

**Table 12-5. IDDRMFX1A Ports**

Port Name	I/O	Description
D	I	DDR Data
ECLK	I	The phase shifted DQS should be connected to this input
RST	I	Reset
CLK1	I	Slow FPGA CLK
CLK2	I	Slow FPGA CLK
CE	I	Clock enable
DDRCLKPOL	I	DDR clock polarity signal
QA	O	Data at the positive edge of the CLK
QB	O	Data at the negative edge of the CLK

Note: The DDRCLKPOL input to IDDRMFX1A should be connected to the DDRCLKPOL output of DQSBUFC.

Figure 12-13 shows the IDDRMFX1A timing waveform.

**Figure 12-13. IDDRMFX1A Waveform**

**ODDRMXA**

The ODDRMXA primitive implements the output register for both the write and tristate functions. This primitive is used to output DDR data and the D to the memory. All the DDR output tristate functions are also implemented using this primitive.

Figure 12-14 shows the ODDRMXA primitive symbol and its I/O ports.

**Figure 12-14. ODDRMXA Symbol**

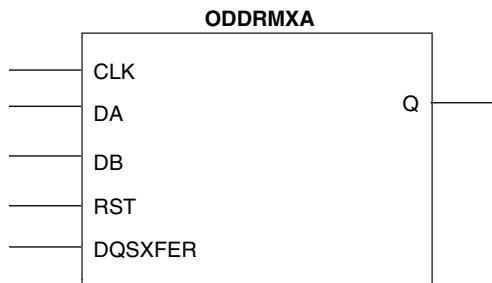


Table 12-6 provides a description of all I/O ports associated with the ODDRMXA primitive.

**Table 12-6. ODDRMXA Ports**

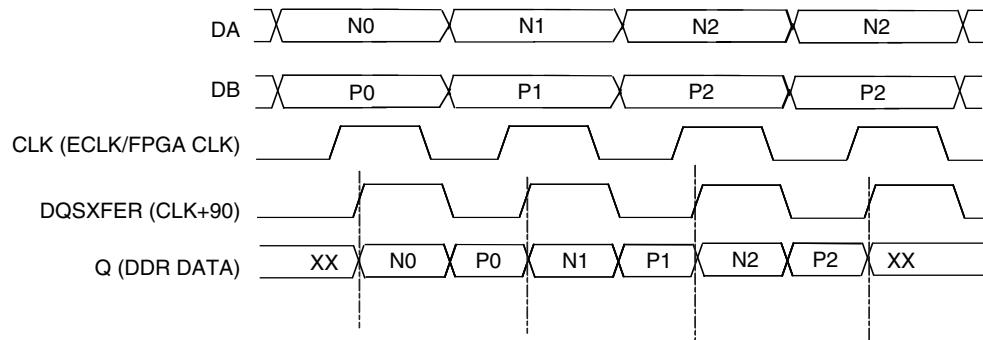
Port Name	I/O	Description
CLK	I	System CLK or ECLK
DA	I	Data at the negative edge of the clock
DB	I	Data at the positive edge of the clock
RST	I	Reset
DQSXFER	I	90-degree phase shifted clock coming from the DQSBUFC block
Q	I	DDR data to the memory

Notes:

1. RST should be held low during DDR Write operation.
2. DDR output and tristate registers do not have CE support. RST is available for the tristate DDRX mode (while reading). The LSR will default to set when used in the tristate mode.
3. When asserting reset during DDR writes, it is important to keep in mind that this only resets the flip-flops and not the latches.

Figure 12-15 shows the ODDRMXA timing waveform.

**Figure 12-15. ODDRMXA Waveform**



## Memory Read Implementation

LatticeECP2/M 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 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 6-bit delay control from the on-chip DQSDLL. The LatticeECP2/M devices support two DQSDLL, one on the left and one on the right side of the device. The DQSDEL generated by the DQSDLL on the left side is routed to all the DQS blocks on the left and bottom half of the device. The delay generated by the DQSDLL on the right side is distributed to all the DQS Delay blocks on the right side and the other bottom half of the device. There are no DQS pins on the top banks of the device. These digital delay control signals are used to delay the DQS from the memory by 90 degrees.

The DQS received from the memory is delayed in each of the DQS Delay blocks and this delayed DQS is used to clock the first set stage DDR input registers.

### DQS Transition Detect or Automatic Clock Polarity Select

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 (pulled by termination). 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.

### 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.

### DDR I/O Register Implementation

The first set of DDR registers is used to de-mux the DDR data at the positive and negative edge of the phase shifted DQS signal. The register 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.

The second stage of registers is clocked by the FPGA clock, the polarity of this clock is selected by the DDR Clock Polarity signal generated by the DQS Transition Detect Block.

The I/O Logic registers can be implemented in two modes:

- Half Clock Transfer Mode
- Full Clock Transfer Mode

In Half Clock Transfer mode the data is transferred to the FPGA core after the second stage of the register. In Full Clock Transfer mode, an additional stage of I/O registers clocked by the FPGA clock is used to transfer the data to the FPGA core.

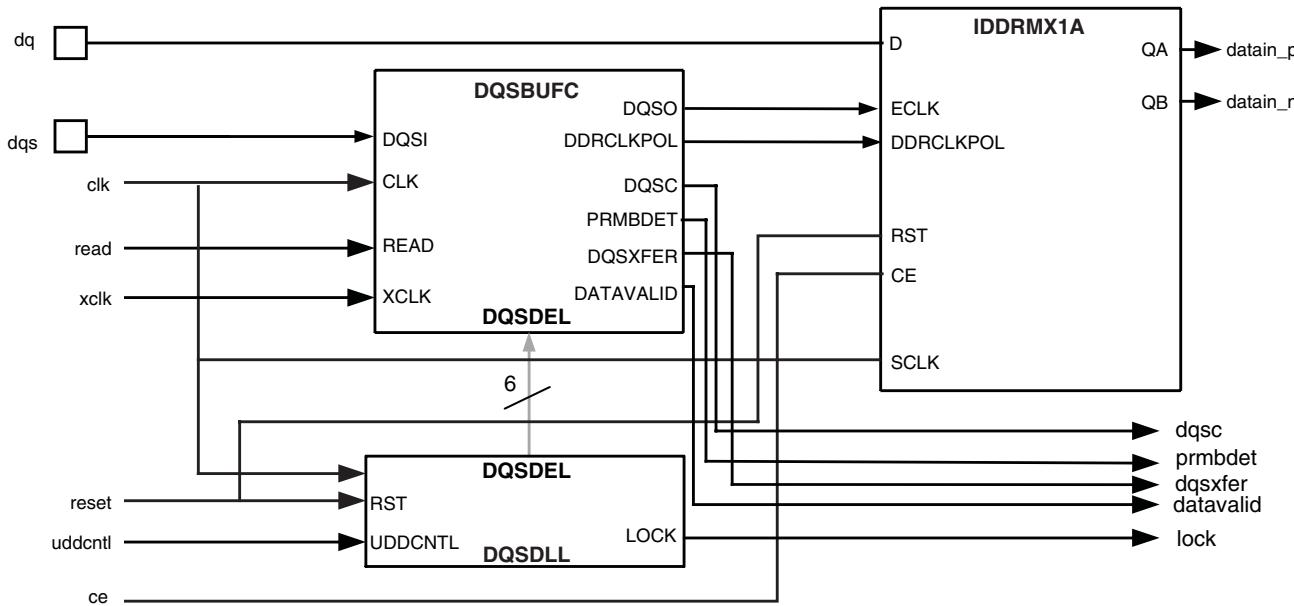
The LatticeECP2/M Family Data Sheet explains each of these circuit elements in more detail.

---

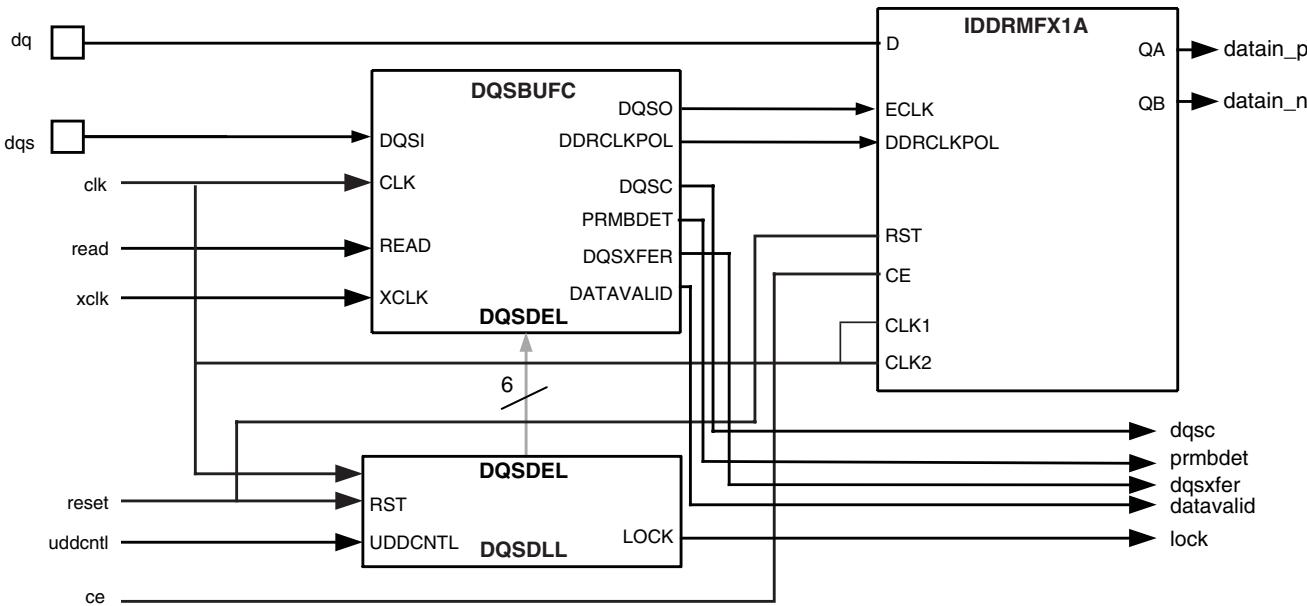
## Memory Read Implementation in Software

Three primitives in the ispLEVER® design tools represent the capability of these three elements. The DQSDLL represents the DLL used for calibration. The IDDRMX1A/IDDRMFX1A primitive represents the DDR input registers and clock domain transfer registers with or without full clock transfer. Finally, the DQSBUFC represents the DQS delay block, the clock polarity control logic and the Data Valid module. Figures 12-16 and 12-17 show the READ interface block generated using the IPExpress™ tool in the ispLEVER software.

**Figure 12-16. Software Primitive Implementation for Memory READ (Half Clock Transfer)**



**Figure 12-17. Software Primitive Implementation for Memory READ (Full Clock Transfer)**



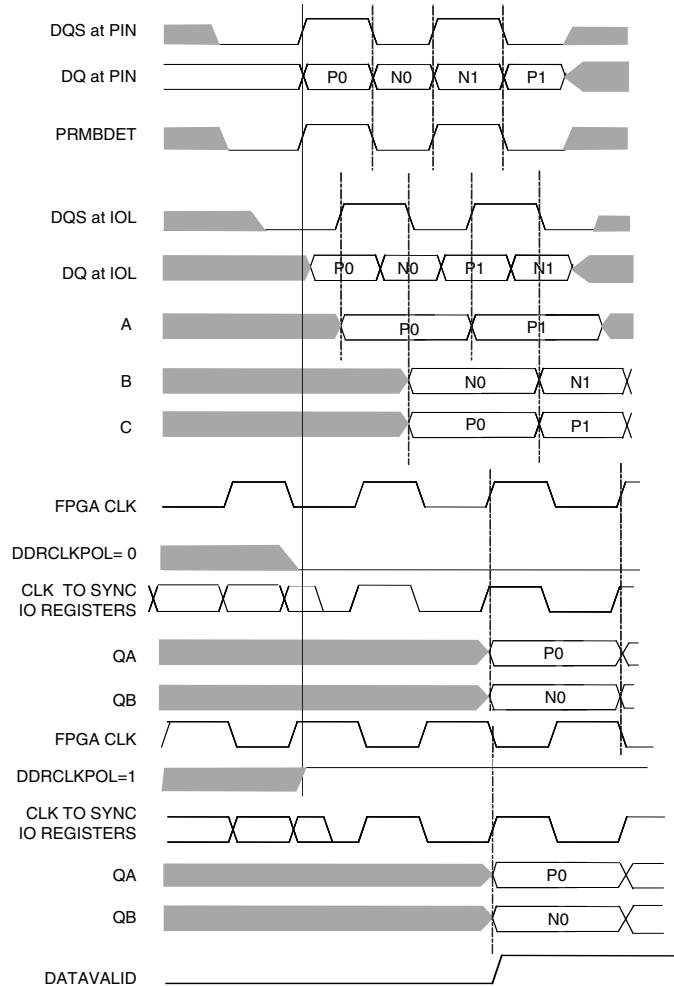
## Read Timing Waveforms

Figures 12-16 and 12-17 show READ data transfer for half and full clock cycle data transfer based on the results of the DQS Transition detector logic. This circuitry decides whether or not to invert the phase of FPGA system CLK to the synchronization registers based on the relative phases of PRMBDET and CLK.

- **Case 1:** If the FPGA clock is low on the first PRMBDET transition, then DDRCLKPOL is low and no inversion is required.
- **Case 2:** If the FPGA clock is high on the first PRMBDET, then DDRCLKPOL is high and the FPGA clock (CLK) needs to be inverted before it is used for synchronization.

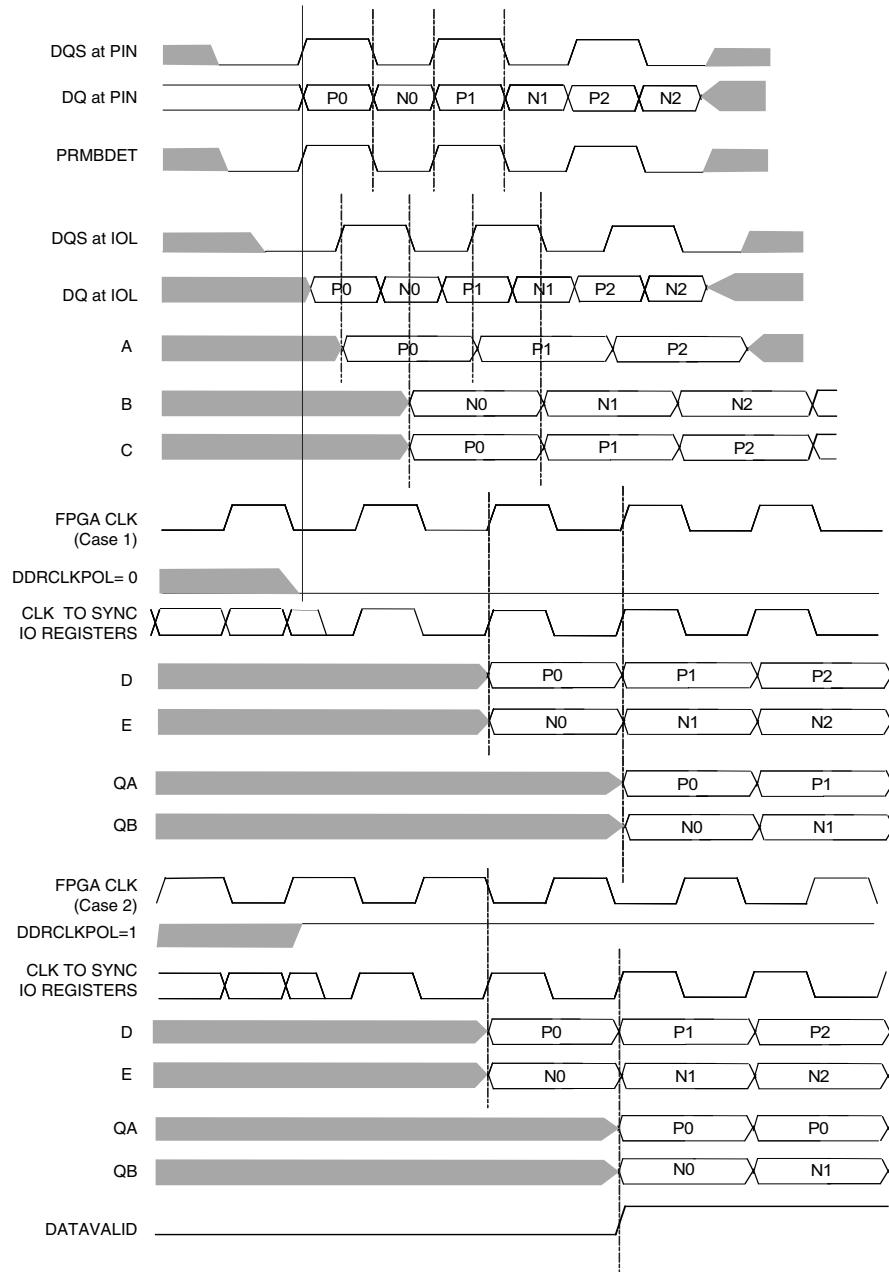
Figure 12-18 illustrates the DDR data timing using half clock transfer mode at different stages of the IDDRMX1A registers. The first stage of the register captures data on the positive edge as shown by signal A and the negative edge as shown by signal B. Data stream A goes through an additional half clock cycle transfer shown by signal C. Phase-aligned data streams B and C are presented to the next stage registers clocked by the FPGA clock.

Figure 12-19 illustrates the DDR data timing using full clock transfer mode at different stages of IDDRMFX1A registers. In addition to the first two register stages in the half clock mode, the full clock transfer mode has an additional stage register clocked by the FPGA clock. In this case, D and E are the data streams after the second register stage presented to the final stage of registers clocked by the FPGA clock.

**Figure 12-18. READ Data Transfer When Using IDDRMX1A****Notes:**

1. DDR memory sends DQ aligned to DQS Strobe.
2. The DQS Strobe is delayed by 90 degrees using the dedicated DQS logic.
3. DQ is now center aligned to DQS Strobe.
4. PRMBDET is the Preamble detect signal generated using the DQSBUFB primitive. This is used to generate the DDRCLKPOL signal.
5. The first set of I/O registers, A and B, capture data on the positive and negative edges of DQS.
6. I/O Register C transfers data so that both data are now aligned to negative edge of DQS.
7. DDCLKPOL signal generated will determine if the FPGA CLK going into the synchronization registers need to be inverted. The DDRCLKPOL=0 when the FPGA CLK is LOW at the first rising edge of PRMBDET. The clock to the synchronization registers is not inverted. The DDRCLKPOL=1 when the FPGA CLK is HIGH at the first rising edge of PRMBDET. In this case the clock to the synchronization register is inverted.
8. The I/O synchronization registers capture data on either the rising or falling edge of the FPGA clock.
9. DATAVALID signal goes HIGH when valid data enters the FPGA core.

Figure 12-19. Read Data Transfer When Using IDDRMFX1A



## Notes:

1. DDR memory sends DQ aligned to DQS Strobe.
2. The DQS Strobe is delayed by 90 degrees using the dedicated DQS logic.
3. DQ is now center-aligned to DQS Strobe.
4. PRMBDET is the Preamble detect signal generated using the DQSBUFB primitive. This is used to generate the DDRCLKPOL signal.
5. The first set of I/O registers, A and B, capture data on the positive and negative edges of DQS.
6. I/O register C transfers data so that both data are now aligned to the negative edge of DQS.
7. DDCLKPOL signal generated will determine if the FPGA clock going into the synchronization registers need to be inverted. The DDRCLKPOL=0 when the FPGA CLK is LOW at the first rising edge of PRMBDET. So the clock to the synchronization registers is not inverted. The DDRCLKPOL=1 when the FPGA CLK is HIGH at the first rising edge of PRMBDET. In this case the clock to the synchronization register is inverted.
8. Registers D and E capture data at the FPGA clock.
9. The data is then again registers at the FPGA clock to ensure a Full Clock Cycle transfer.
10. DATAVALID signal goes HIGH when valid data enters the FPGA core.

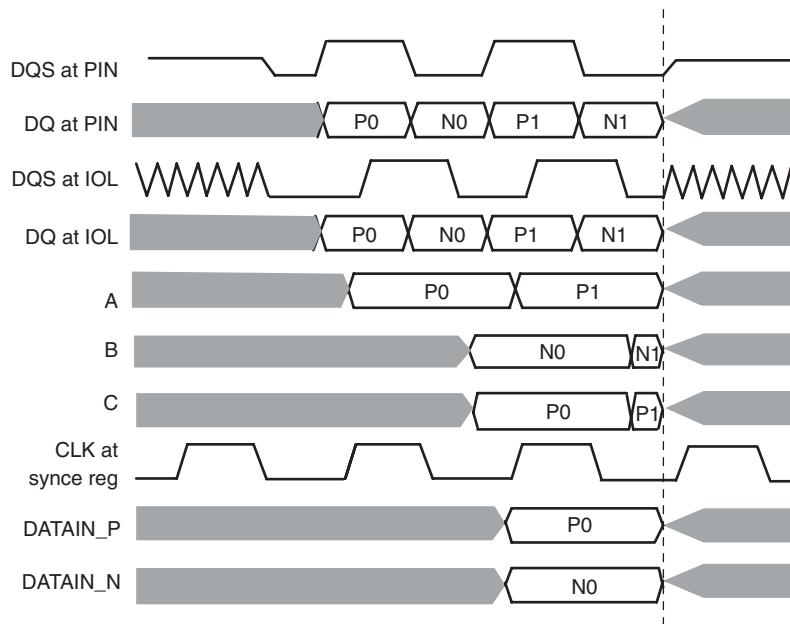
**Data Read Critical Path**

Data in the second stage DDR registers can be registered either on the positive edge or on the falling edge of FPGA clock depending on the DDRCLKPOL signal. In order to ensure that the data transferred to the FPGA core registers is aligned to the rising edge of the system clock, this path should be constrained with a half clock transfer. This half clock transfer can be forced in the software by assigning a multi-cycle constraint (multi-cycle of 0.5 X) on all the data paths to first PFU register.

**DQS Postamble**

At the end of a READ cycle, the DDR SDRAM device executes the READ cycle postamble and then immediately tristates both the DQ and DQS output drivers. Since neither the memory controller (FPGA) nor the DDR SDRAM device are driving DQ or DQS at that time, these signals float to a level determined by the off-chip termination resistors. While these signals are floating, noise on the DQS strobe may be interpreted as a valid strobe signal by the FPGA input buffer. This can cause the last READ data captured in the IOL input DDR registers to be overwritten before the data has been transferred to the free running resynchronization registers inside the FPGA.

**Figure 12-20. Postamble Effect on READ**



LatticeECP2/M devices have extra dedicated logic in the DQS Delay Block that prevents this postamble problem. The DQS postamble logic is automatically implemented when the user instantiates the DQS Delay logic (DQSBUFC software primitive) in the design.

**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 the data, DQ. Along with the DQS strobe and data this portion of the interface must also provide the CLKP, CLKN Address/Command and Data Mask (DM) signals to the memory.

It is the responsibility of the FPGA output control to edge-align the DDR output signals (ADDR,CMD, DQS, but not DQ, DM) to the rising edge of the outgoing differential clock (CLKP/CLKN).

Challenges encountered by the during Memory WRITE:

1. DQS needs to be center-aligned with the outgoing DDR Data, DQ.
2. Differential CLK signals (CLKP and CLKN) need to be generated.

3. The controller must meet the DDR interface specification for  $t_{DSS}$  and  $t_{DSH}$  parameters, defined as DQS falling to CLKP rising setup and hold times.
4. The DDR output data must be muxed from two SDR streams into a single outgoing DDR data stream.

All DDR output signals (“ADDR, CMD”, DQS, DQ, DM) are initially aligned to the rising edge of the FPGA clock inside the FPGA core. The relative phase of the signals may be adjusted in the IOL logic before departing the FPGA. These adjustments are shown in Figure 12-21.

LatticeECP2/M devices contain DDR output and tri-state registers along with the DQSXFER signal generated by the DQSBUFC that allows easy implementation of the write portion of the DDR memory interfaces. The DDR output registers can be accessed in the design tools via the ODDRMXA and the ODDRXC primitives.

The DQS signal and the DDR clock outputs are generated using the ODDRXC primitive. As shown in the figure, the CLKP and DQS signals are generated so that they are 180 degrees in phase with the clock. This is done by connecting “1” to the DA input and “0” to the DB inputs of the ODDRXC primitive. Refer to the DDR Generic Software Primitive section of this document to see the ODDRXC timing waveforms.

The DDR clock output is then fed into a SSTL differential output buffer to generate CLKP and CLKN differential clocks. Generating the CLKN in this manner prevents any skew between the two signals. When interfacing to DDR1, SDRAM memory CLKP should be connected to the SSTL25D I/O standard. When interfacing to DDR2 memory, it should be connected to the SSTL18D I/O standard.

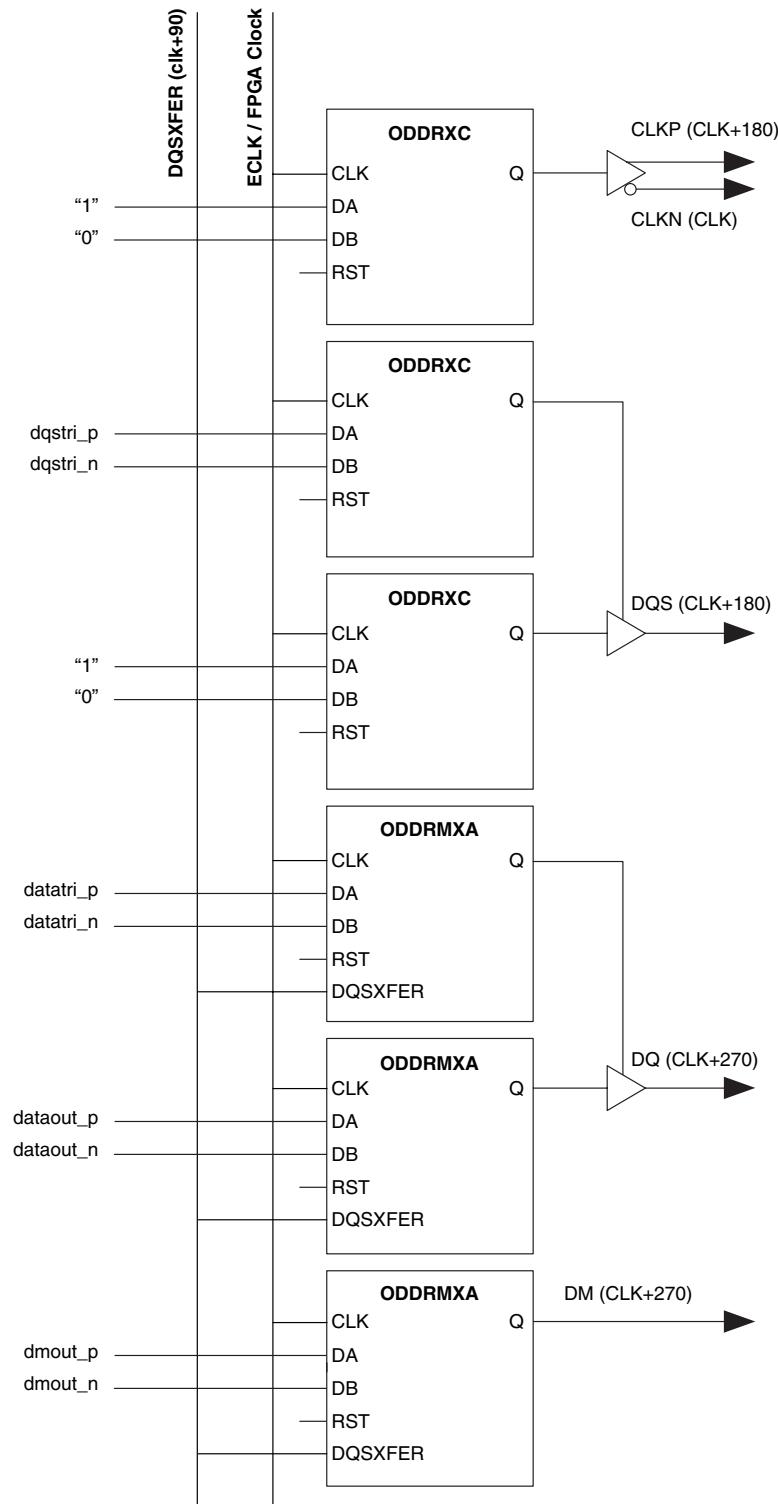
The DQSXFER output from the DQSBUFC block is the 90-degree phase shifted clock. This 90-degree phase shifted clock is used as an input to the ODDRMXA block. The ODDRMXA is used to generate the DQ and DM data outputs going to the memory. In the ODDRMXA module, the data is first registered using the ECLK or FPGA clock input and then shifted out using the DQSXFER signal. To ensure that the data going to the memory is center-aligned to the DQS, the DQSXFER is inverted inside the ODDRXC primitive. This will generate data that is center-aligned to the DQS. Refer to the Software Primitives section of this document for the ODDRXC timing waveforms.

The DDR interface specification for  $t_{DSS}$  and  $t_{DSH}$  parameters defined as DQS falling to CLKP rising setup and hold times must be met. This is accomplished by ensuring that the CLKP and DQS signals are identical in phase.

The tristate control for the DQS and DQ outputs can also be implemented using the ODDRXC primitive.

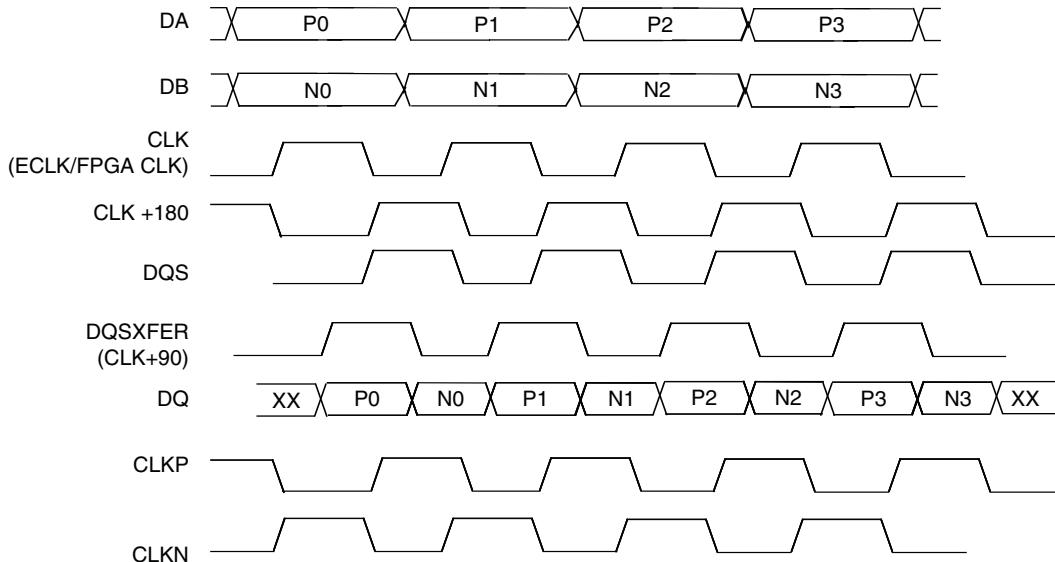
Figure 12-21 shows the DDR Write implementation using the DDR primitives.

Figure 12-21. Software Implementation for Memory Write



**Write Timing Waveforms**

Figure 12-22 shows the DDR write side data transfer timing for the DQ Data pad and the DQS Strobe Pad. When writing to the DDR memory device, the DM (Data Mask) and the ADDR/ CMD (Address and Command) signals are also sent to the memory device along with the data and strobe signals.

**Figure 12-22. DDR Write Data Transfer for DQ Data****Design Rules/Guidelines**

Listed below are some rules and guidelines to keep in mind when implementing DDR memory interfaces in the LatticeECP2/M devices.

- The LatticeECP2/M devices have dedicated DQ-DQS banks. Please refer to the logical signal connections of the groups in the LatticeECP2/M Family Data Sheet before locking these pins.
- There are two DQSDLL on the device, one for the left half and one for the right half of the device. Therefore, only one DQSDLL primitive should be instantiated for each half of the device. Since there is only one DQSDLL on each half of the device, all the DDR memory interfaces on that half of the device should run at the same frequency. Each of the DQSDLL will generate 90-degree digital delay bits for all the DQS delay blocks on that half of the device based on the reference clock input to the DLL.
- When implementing a DDR SDRAM interface, all interface signals should be connected to the SSTL25 I/O standard. In the case of the DDR2 SDRAM interface, the interface signal should be connected to SSTL18 I/O standard.
- For DDR2, the differential DQS signals need to be connected to SSTL18 the Differential I/O standard.
- When implementing the DDR interface, the VREF1 of the bank is used to provide the reference voltage for the interface pins.

**Generic High Speed DDR Implementation**

In addition to the DDR memory interface, the I/O logic DDR registers can be used to implement high speed DDR interfaces. The Input DDR registers can operate in full clock transfer and half clock transfer modes. The DDR input and output register also support x1 and x2 gearing ratios. A gearing capability is provided to Mux/DeMux the I/O data rate (ECLK) to the FPGA clock rate (SCLK). For DDR interfaces, this ratio is slightly different than the SDR ratio. A basic 2x DDR element provides four FPGA side bits for two I/O side bits at half the clock rate on the FPGA side.

The data going to the DDR registers can be optionally delayed before going to the DDR register block.

## Generic DDR Software Primitives

The IPExpress tool in the ispLEVER software can be used to generate the DDR modules. The various DDR modes described below can be configured in the IPExpress tool. The various modes are implemented using the following software primitives:

- IDDRXC - DDR Generic Input
- IDDRFXA - DDR Generic Input with full clock transfer (x1 gearbox)
- IDDRX2B - DDR Generic Input with 2x gearing ratio. DDRX2 inputs a double data rate signal as four data streams. Two stages of DDR registers are used to convert serial DDR data at the input pad into four SDR data streams entering FPGA core logic.
- ODDRXC- DDR Generic Output
- ODDRX2B - DDR Generic Output with 2x gearing ratio. The DDRX2 inputs four separate data streams and outputs a single data stream to the I/O buffer.
- DELAYB - The DDR input can be optionally delayed before it is input to the DDR registers. The user can choose to implement a fixed delay value or use a dynamic delay.

### IDDRXC

This primitive inputs DDR data at both edges of the CLK and generates two streams of data. The CLK to this module can be connected to either the edge clock or the primary FPGA clock.

Figure 12-23 shows the primitive symbol for IDDRXC mode.

**Figure 12-23. IDDRXC Symbol**

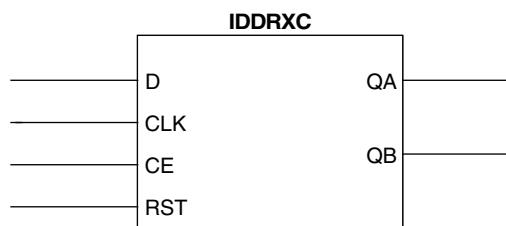
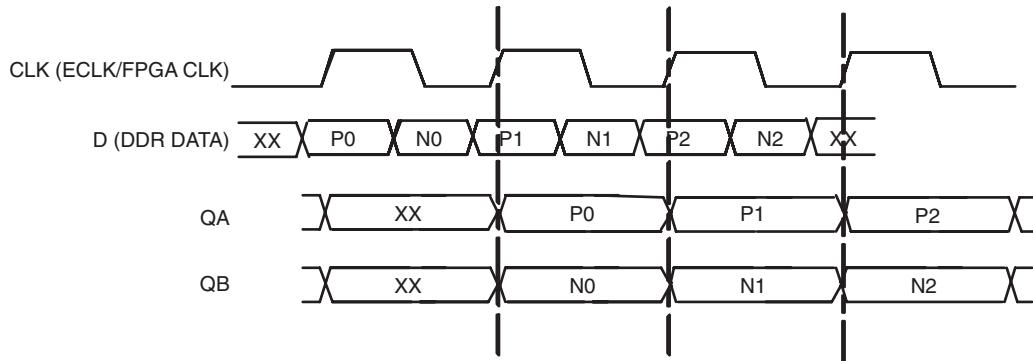


Table 12-7 lists the port names and descriptions for the IDDRXC primitive.

**Table 12-7. IDDRXC Port Names**

Port Name	I/O	Definition
D	I	DDR data
CLK	I	This clock can be connected to the ECLK or the FPGA clock
CE	I	Clock enable signal
RST	I	Reset to the DDR register
QA	O	Data at the positive edge of the clock
QB	O	Data at the negative edge of the clock

Figure 12-24 shows the timing waveform when using the IDDRXC module.

**Figure 12-24. IDDRXC Waveform****IDDRFXA**

This primitive inputs DDR data at both edges of clock CLK1 and generates two streams of data aligned to clock CLK2. CLK1 can be connected either to the edge clock or the internal FPGA clock. CLK1 is used to register the DDR registers and the first set of synchronization registers. CLK2 is used by the third stage of registers and should be clocked by the FPGA clock. The LatticeECP2/M Family Data Sheet explains the input register block in more detail.

Figure 12-25 shows the primitive symbol for the IDDRFXA mode.

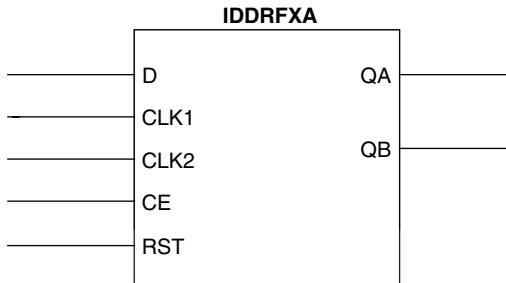
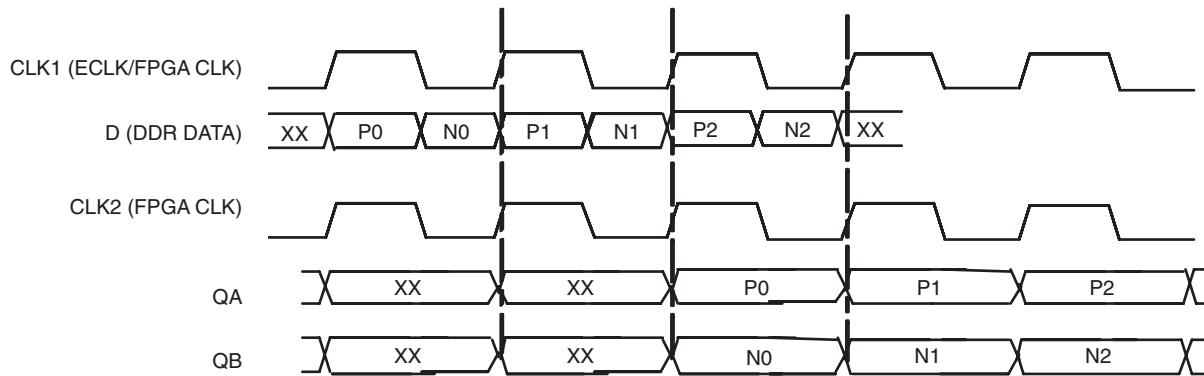
**Figure 12-25. IDDRFXA Symbol**

Table 12-8 lists the port names and descriptions for the IDDRFXA primitive.

**Table 12-8. IDDRFXA Port Names**

Port Name	I/O	Description
D	I	DDR data
CLK1	I	This clock can be connected to the ECLK or the FPGA clock
CLK2	I	This clock should be connected to the FPGA clock
CE	I	Clock Enable signal
RST	I	Reset to the DDR register
QA	O	Data at the positive edge of the clock
QB	O	Data at the negative edge of the clock

Figure 12-26 shows the timing waveform when using the IDDRFXA module.

**Figure 12-26. IDDRFXA Waveform****IDDRX2B**

This primitive is used when a gearing function is required. This primitive inputs the DDR data at both edges of the edge clock and generates four streams of data. The DDR registers and the first set of synchronization registers are clocked by the ECLK input of the primitive, which should be connected to the fast ECLK. The SCLK is used to clock the third stage of registers and should be connected to the FPGA clock. This primitive will output four streams of data. The 2x gearing function is implemented by using the synchronization registers of the complementary PIO.

Figure 12-27 shows the primitive symbol for the IDDRX2B mode.

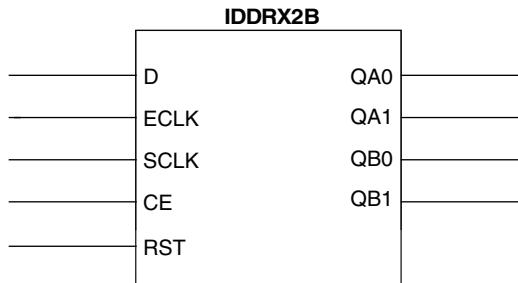
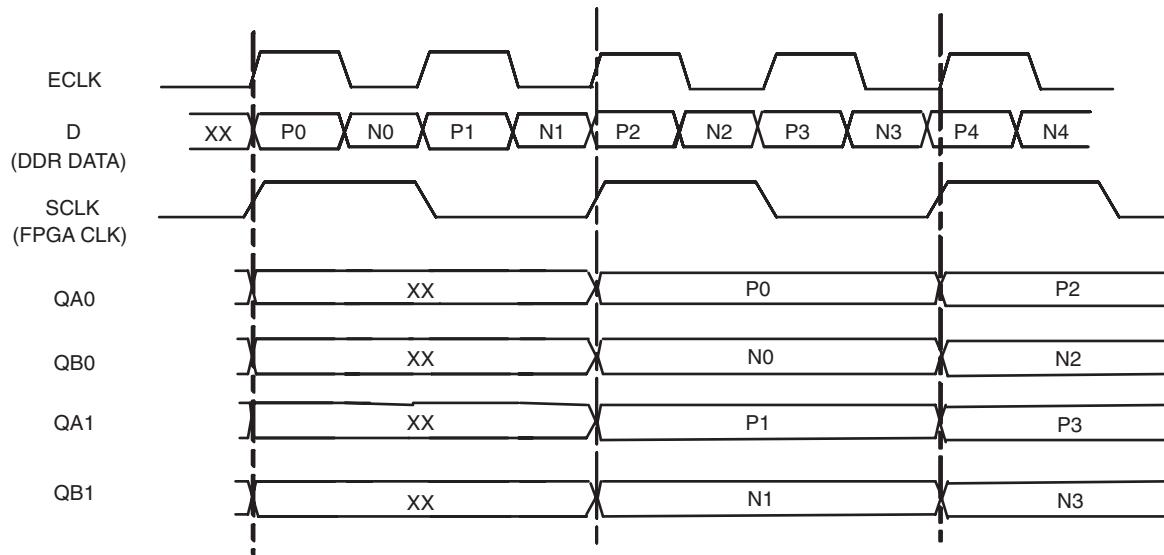
**Figure 12-27. IDDRX2B Symbol**

Table 12-9 lists the port names and descriptions for the IDDRX2B primitive.

**Table 12-9. IDDRX2B Port Names**

Port Name	I/O	Description
D	I	DDR data
ECLK	I	This clock can be connected to the fast edge clock
SCLK	I	This clock should be connected to the FPGA clock
CE	I	Clock enable signal
RST	I	Reset to the DDR register
QA0, QA1	O	Data at the positive edge of the clock
QB0, QB1	O	Data at the negative edge of the clock

Figure 12-28 shows the timing waveform when using the IDDRX2B module.

**Figure 12-28. IDDRX2B Waveform****ODDRXC**

This is the DDR output module. This primitive will input two data streams and mux them together to generate a single stream of data going to the sysIO™ buffer. The CLK to this module can be connected to the edge clock or to the FPGA clock. This primitive is also used for when DDR function is required for the tristate signal.

Figure 12-29 shows the primitive symbol for the ODDRXC mode.

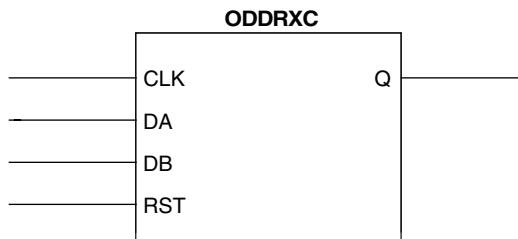
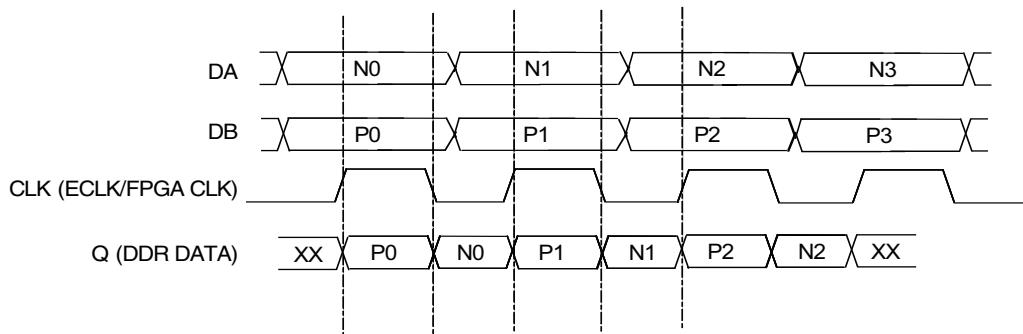
**Figure 12-29. ODDRXC Symbol**

Table 12-10 lists the port names and descriptions for the ODDRXC primitive.

**Table 12-10. ODDRXC Port Names**

PORT NAME	I/O	Definition
DA	I	Data at the negative edge of the clock
DB	I	Data at the positive edge of the clock
CLK	I	This clock can be connected to the edge clock or to the FPGA clock
RST	I	Reset signal
Q	O	DDR data output

Figure 12-30 shows the timing waveform when using the ODDRXC module.

**Figure 12-30. ODDRXC Waveform****ODDRX2B**

This DDR output module can be used when a gearbox function is required. This primitive inputs four data streams and muxes them together to generate a single stream of data going to the sysIO buffer.

DDR registers of the complementary PIO are used in this mode. The complementary PIO register can no longer be used to perform the DDR function. There are two clocks going to this primitive. The ECLK is connected to the faster edge clock and the SCLK is connected to the slower FPGA clock. The DDR data output of this primitive is aligned to the faster edge clock.

Figure 12-31 shows the primitive symbol for the ODDRX2B mode.

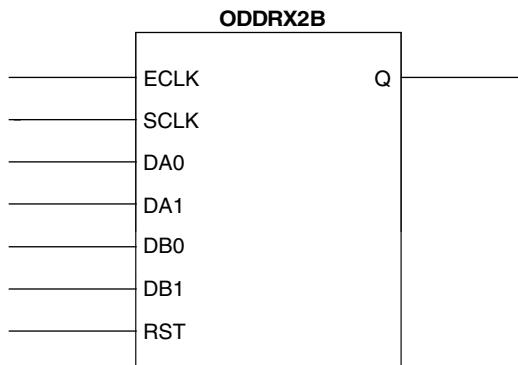
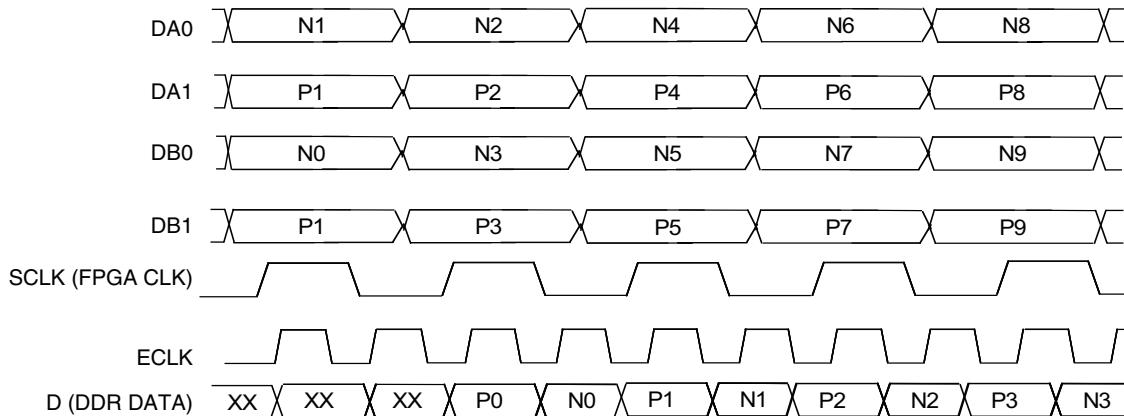
**Figure 12-31. ODDRX2B Symbol**

Table 12-11 lists the port names and descriptions for the ODDRX2B primitive.

**Table 12-11. ODDRX2B Port Names**

Port Name	I/O	Description
DA0, DB0	I	Data at the negative edge of the clock
DA1, DB1	I	Data at the positive edge of the clock
ECLK	I	This clock should be connected to the faster edge clock
SCLK	I	This clock should be connected to the slower FPGA clock
RST	I	Reset signal
Q	O	DDR data output

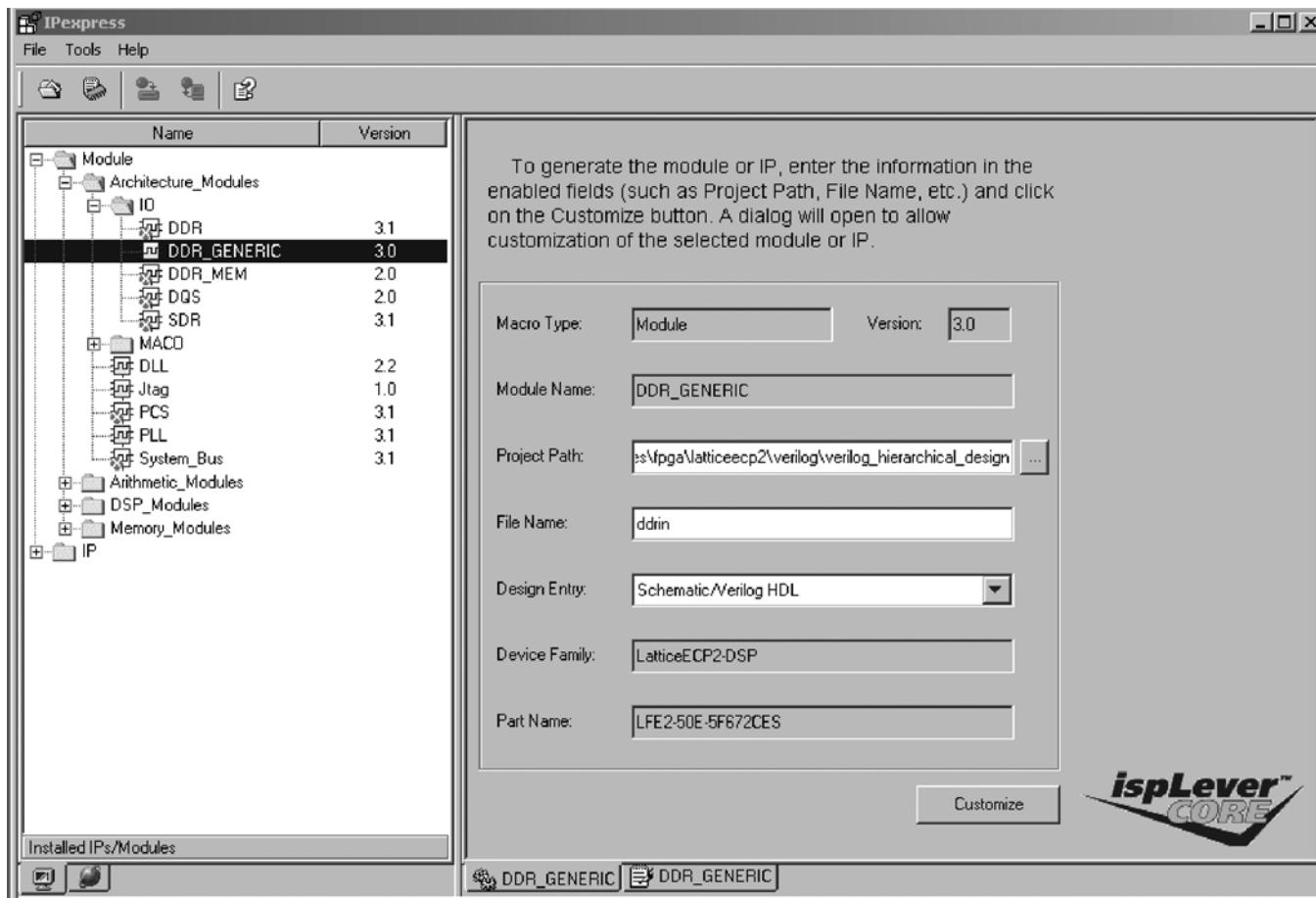
Figure 12-32 shows the timing waveform when using the ODDRXC module.

**Figure 12-32. ODDRX2B Waveform**

## DDR Generic Usage In IPexpress

IPexpress is used to create and configure generic DDR I/O modules. The user can choose the mode to configure the DDR block. The result in an HDL module to be used in a top level design.

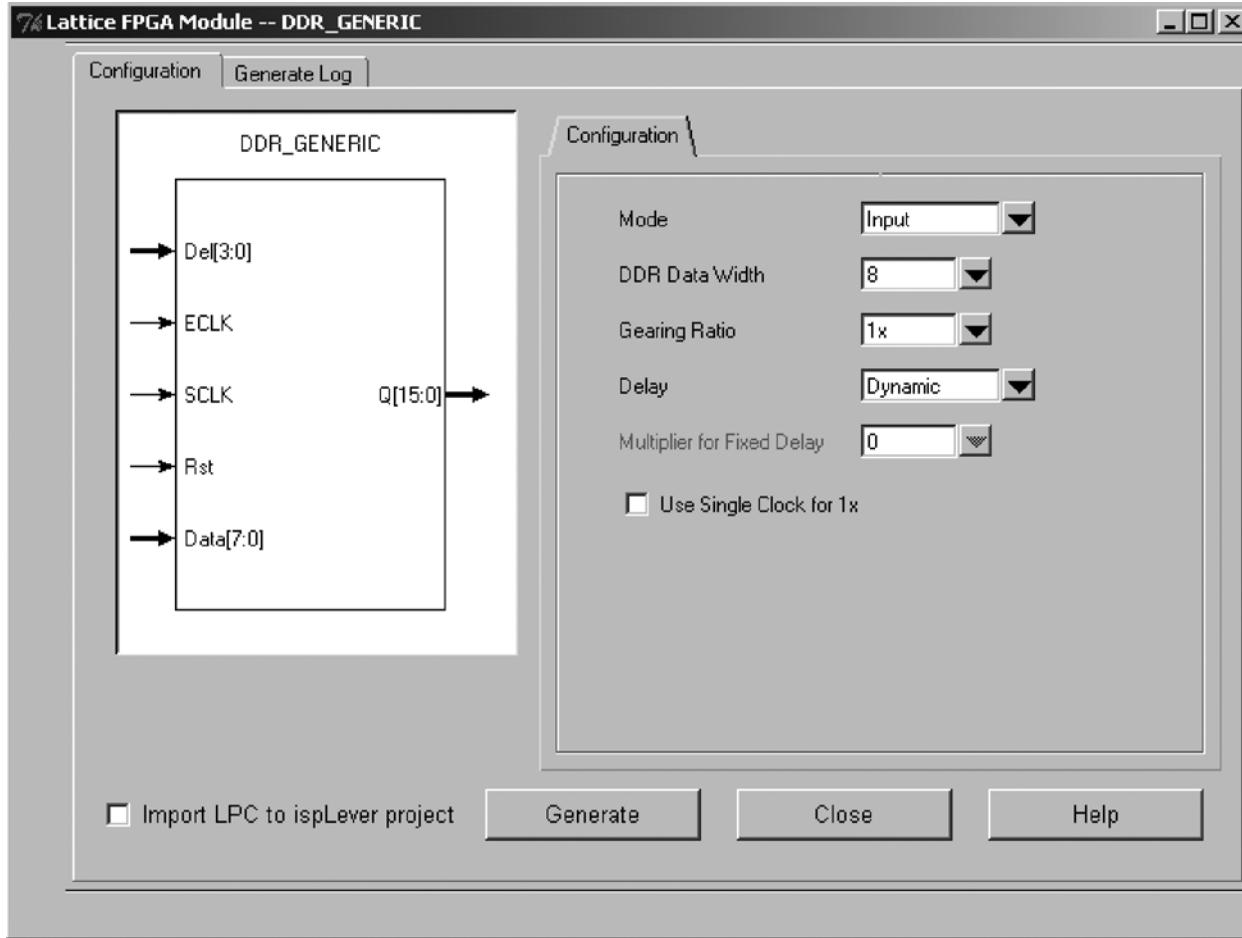
Figure 12-33 shows the main window when DDR\_Generic is selected. The only entry required in this window is the module name. Other entries are set to the project settings. The user may change these entries if desired. After entering the module name, click on **Customize** to open the Configuration Tab window as shown in Figure 12-34.

**Figure 12-33. 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 12-34. Configuration Tab for DDR\_Generic**



The user can change the Mode parameter to choose either Input, Output, Bidirection or Tristate DDR module. The other configuration parameters will change according to the Mode selected. The Delay parameter is only available for Input and Bidirectional modes. Similarly, the Multiplier for Fixed Delay parameter is only available when the Delay parameter is configured to Fixed.

Table 12-12 describes all user parameters in the IPexpress GUI and their usage.

**Table 12-12. User Parameters in the IPexpress GUI**

User Parameters	Description	Values/Range	Default
Mode	Mode selection for the DDR block	Input, Output, Bidirectional, Tristate	Input
Data Width	Width of the data bus	1-64	8
Gearing Ratio	Gearing ratio selection	1x, 2x <sup>1</sup>	1x
Delay	Input Delay configuration	Dynamic, Fixed, FixedXGMII	Dynamic
Multiplier for Fixed Delay	Fixed delay setting. Available only when delay is configured to fixed	0-15	0
Use Single Clk for 1x	Allows user to select a single clock for the gearing logic	On/Off	Off

1. Only 1x available when Mode is Bidirection or Tristate.

## DELAYB

Data going to the DDR registers can be optionally delayed using the Delay block. The Delay block receives 4-bit delay control. The 4-bit delay can be set using fixed multiplier values or it can be controlled by the user. The DELAYB block is available for use with the input DDR registers.

The DELAYB block can be configured when generating the DDR input modules in the IPexpress tool in the ispLEVER software. The user can choose from three types of delay values:

1. Dynamic – The delay value is controlled by the user logic using the DEL[3:0] input of the DELAYB block.
2. Fixed – When choosing the fixed value, the user will also need to choose from one of the 16 multiplier values. This will tie the inputs DEL[3:0] of the DELAYB block to a fixed value depending on the multiplier value chosen.
3. FIXED\_XGMII – The DEL [3:0] will be configured with the delay value required when implementing a XGMII interface.

Figure 12-35 shows the primitive symbol for the DELAYB mode.

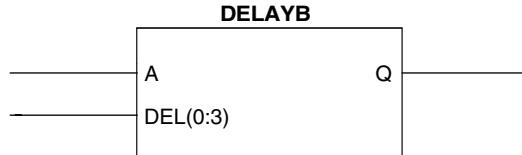
**Figure 12-35. DELAYB Symbol**

Table 12-13 lists the port names and descriptions for the DELAYB primitive.

**Table 12-13. 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

## Design Rules/Guidelines

Listed below are some rules and guidelines for implementing generic DDR interfaces in LatticeECP2/M devices.

- When implementing a 2x gearing mode, the complement PIO registers are used. This complementary PIO register can no longer be used and should not be connected.

## FCRAM (“Fast Cycle Random Access Memory”) Interface

FCRAM is a DDR-type DRAM, which performs data output at both the rising and falling edges of the clock. FCRAM devices operate at a core voltage of 2.5V with SSTL Class II I/O. It has enhanced both the core and peripheral logic of the SDRAM. In FCRAM the address and command signals are synchronized with the clock input, and the data pins are synchronized with the DQS signal. Data output takes place at both the rising and falling edges of the DQS. DQS is in phase with the clock input of the device. The DDR SDRAM and DDR FCRAM controller will have different pinouts.

LatticeECP2/M devices can implement the FCRAM interface using dedicated DQS logic, input DDR registers and output DDR registers, as described in the Implementing Memory Interfaces section of this document. Generation of address and control signals for FCRAM are different than in DDR SDRAM devices. Please refer to the FCRAM data sheets to see detailed specifications. Toshiba, Inc. and Fujitsu, Inc. offer FCRAM devices in 256Mb densities. They are available in x8 or x16 configurations.

## Board Design Guidelines

The most common challenge associated with implementing DDR memory interfaces is the board design and layout. It is required that users strictly follow the guidelines recommended by memory device vendors.

Some of the common recommendations include matching trace lengths of interface signals to avoid skew, proper DQ-DQS signal grouping, proper termination of the SSTL2 or SSTL18 I/O Standard, proper VREF and VTT generation decoupling and proper PCB routing.

The following documents include board layout guidelines:

- [www.idt.com](http://www.idt.com), IDT, *PCB Design for Double Data Rate Memory*
- [www.motorola.com](http://www.motorola.com), AN2582, *Hardware and Layout Design Considerations for DDR Interfaces*

## References

- [www.jedec.org](http://www.jedec.org), JEDEC Standard 79, Double Data Rate (DDR) SDRAM Specification
- [www.micron.com](http://www.micron.com), DDR SDRAM Data Sheets
- [www.infinion.com](http://www.infinion.com), DDR SDRAM Data Sheets
- [www.samsung.com](http://www.samsung.com), DDR SDRAM Data Sheets
- [www.latticesemi.com](http://www.latticesemi.com), RD1019, *QDR Memory Controller Reference Design for Lattice ECP/EC Devices*
- [www.toshiba.com](http://www.toshiba.com), DDR FCRAM Data Sheet
- [www.fujitsu.com](http://www.fujitsu.com), DDR FCRAM Data Sheet
- [www.latticesemi.com](http://www.latticesemi.com), *LatticeEC Advanced Evaluation Board User's Guide*
- [www.latticesemi.com](http://www.latticesemi.com), *DDR SDRAM Controller (Pipelined Version for Lattice ECP/EC and LatticeXP™ Devices) User's Guide*

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)  
+1-503-268-8001 (Outside North America)  
e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
Internet: [www.latticesemi.com](http://www.latticesemi.com)

---

## Revision History

Date	Version	Change Summary
February 2006	01.0	Original version.
September 2006	01.1	Some figures updated. Added information on DELAYB block.
September 2006	01.2	Updated for LatticeECP2M. Added "DDR Generic Usage in IPexpress" section.

## Introduction

Power considerations in FPGA design are critical for determining the maximum system power requirements and sequencing requirements of the FPGA on the board. This technical note provides users with detailed power considerations such as sequencing. Also included are instructions for calculating power consumption in LatticeECP2™ and LatticeECP2M™ devices using the Power Calculator available in the Lattice ispLEVER® design tool. General guidelines for reducing power consumption are also discussed.

## Power Supply Sequencing

### Power-Up Sequencing

There are three main power supplies that are required to power-up the LatticeECP2/M device for proper operation:  $V_{CC}$ ,  $V_{CCAUX}$  and  $V_{CCIO8}$ . Bank 8, or  $V_{CCIO8}$ , powers the sysCONFIG™ port and configuration circuitry and is therefore required during power-up.

The nominal voltages for these power supplies are 1.2V for  $V_{CC}$ , 3.3V for  $V_{CCAUX}$  and 1.2V to 3.3V for  $V_{CCIO8}$ . The nominal trip points for these power supplies are 0.6V to 0.8 V for  $V_{CC}$  and  $V_{CCIO8}$ , and 2.2V to 2.5 V for  $V_{CCAUX}$ . These power supplies are recommended to transition from the trip point to the corresponding minimum operating voltage (refer to device data sheet for the minimum voltage supply values) no faster than 2ms. This transition recommendation is most critical for the last power supply that reaches the trip point.

Also, each power supply must follow a monotonically clean ramp between the trip points and the minimum required supply voltage. Slow power supply ramps in the 10's of milliseconds to 100's of milliseconds are critical to ensure that the transitions around trip points are monotonic. Multiple transitions through the trip point may cause multiple internal power-on reset sequencing.

The rest of the bank voltages can come up in any sequence.

After initialization is complete, if any  $V_{CC}$ ,  $V_{CCAUX}$  or  $V_{CCIO8}$  drops below its power-down trip point, the device will reset. Any  $V_{CCIO[7:0]}$  can be removed without resetting the device after initialization is complete.

### Power-Down Sequencing

During power-down, power should be removed from one of the supplies'  $V_{CC}$ ,  $V_{CCAUX}$  or  $V_{CCIO8}$  first to ensure that no high currents are seen on the input pins as the other  $V_{CCIO}$  supplies are removed. This only applies when input signals are still being driven, such as in hot-socketing applications.

For non-hot-socketing applications, the input signals are likely to be powered from the same supply as  $V_{CCIO}$ . Therefore, they will usually be less than or equal to  $V_{CCIO}$  during power down.

### Power Sequencing Recommendations

The recommended power supply sequencing for LatticeECP2/M devices is  $V_{CC}$  before  $V_{CCAUX}$  or  $V_{CCIO8}$ . It is suggested that  $V_{CC}$  should reach its minimum voltage value before  $V_{CCAUX}$  and  $V_{CCIO8}$  reach their minimum values. It should also follow the rule of a minimum of 2ms during transition from the trip point to the minimum voltage.

## Power Calculator Hardware Assumptions

The Power Calculator reports the power dissipation in terms of:

1. DC portion of the power consumption
2. AC portion of the power consumption

Total power dissipation is the sum of the static (DC) and dynamic (AC) power dissipations of a device. While DC power depends upon voltage, temperature and process variation, AC power is a strong function of the frequency and the activity of the resources and a weak function of voltage, temperature and process.

### Static Power or DC Power

DC Power can be further subdivided into the power consumption of the used and unused resources. Another important term is Quiescent Power, the DC power for a blank (BE or Bulk Erase) device. In the Bulk Erased mode, none of the resources are used, so it is the total DC power of an unused device.

The AC portion of the power consumption, associated with used resources, is the dynamic part of the power consumption. AC power dissipation is directly proportional to the frequency and activity at which the resource is running and the number of resource units used.

### Junction Temperature

For a fixed temperature, voltage and device package combination, quiescent power is fixed.

Ambient temperature that affects the junction temperature is a factor that contributes to the final power consumption.

Power Calculator models this ambient to junction temperature dependency. When a user provides an ambient temperature, it is rolled into an algorithm which calculates the junction temperature and quiescent power through an iterative process.

### Typical and Worst Case Process Power/I<sub>CC</sub>

Another factor that affects the DC power is process variation. This variation in turn causes variation in quiescent power.

Power Calculator takes these factors into account and allows users to specify either a typical process or a worst case process.

- A typical process selection under Device Variation allows users to calculate the power dissipation of a design using a typical process device.
- The worst case selection under the same option provides the maximum power dissipation for the device and package combination. This information is particularly useful for FPGA power budgeting for the entire system.

### Dynamic Power Budgets and Maximum Operating Temperature

When designing a system, designers 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 device 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 100°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.

Let us consider an example for how to determine and use the Power Calculator for thermal analysis. Once the user has imported or provided all the required information in the Power Calculator, the software will provide the power estimation and predict the Junction Temperature (TJ). Any time this junction temperature is outside the limits specified in the device 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 in this technical note. The four factors listed earlier in this section, namely 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 the overall system reliability.

## Power Calculator

Power Calculator is a powerful tool that allows users to estimate power consumption at two different levels:

1. Estimate of the utilized resources before completing place and route
2. Post place and route design

At a coarse level of estimation, the user provides estimates of device usage in the Power Calculator Wizard and the tool provides a rough estimate of the power consumption.

For a more accurate approach, a designer can import actual device utilization by importing the post Place and Route netlist (NCD) file.

## Power Calculation Equations

The following are the power equations used in the Power Calculator:

Total DC Power (Resource)

$$\begin{aligned} &= \text{Total DC Power of Used Portion} + \text{Total DC Power of Unused Portion} \\ &= [\text{DC Leakage per Resource when Used} * \text{NRESOURCE}] \\ &\quad + [\text{DC Leakage per Resource when Unused} * (\text{N}_{\text{TOTAL RESOURCE}} - \text{NRESOURCE})] \end{aligned}$$

Where:

$\text{N}_{\text{TOTAL RESOURCE}}$  is the total number of Resources in a device,  
 $\text{NRESOURCE}$  is the number of Resources used in the design,

The total DC power consumption for all the resources as per the design data is the sum of the quiescent power and the individual DC power of the resources in the Power Calculator.

The AC Power, on the other hand, is governed by the following equation:

Total AC Power (Resource)

$$= \text{K}_{\text{RESOURCE}} * f_{\text{MAX}} * \text{AF}_{\text{RESOURCE}} * \text{NRESOURCE}$$

Where:

$\text{NRESOURCE}$  is the number of resources used in the design,

---

$K_{RESOURCE}$	is the power constant for the resource in mW/MHz.
$f_{MAX}$	is the max frequency at which the resource is running. Frequency is measured in MHz.
$AF_{RESOURCE}$	is the activity factor for the resource group. The Activity Factor is a percentage of the switching frequency.

For example, the power consumption of the LUT is calculated as per the following equation,

Total AC Power (LUT)

$$= K_{LUT} * f_{MAX} * AF_{LUT} * N_{LUT}$$

$N_{LUT}$	is the number of LUTs used in the design.
$K_{LUT}$	is the Power constant for the LUT blocks in mW/MHz.
$f_{MAX}$	is the max frequency of the LUT clock measured in MHz
$AF_{LUT}$	is the activity factor for the LUT. The Activity Factor is a percentage of the switching frequency.

Another example is the power consumption of the EBR block, which is calculated as follows:

Total AC Power (EBR)

$$= K_{EBR} * f_{MAX} * AF_{EBR} * N_{EBR}$$

$N_{EBR}$	is the number of EBR blocks used in the design.
$K_{EBR}$	is the Power constant for the EBR blocks in mW/MHz.
$f_{MAX}$	is the max frequency of the EBR clock measured in MHz.
$AF_{EBR}$	is the activity factor for the Read and Write ports of the EBR. The Activity Factor is a percentage of the switching frequency.

Also note that the LUT can be configured in Logic, Ripple or Distributed RAM modes. Each of these modes has a different power constant/power coefficient. However, the equations stay the same.

The AC power of some of the dedicated blocks can be calculated using the following equation:

Total AC Power (Dedicated Resource)

$$= K_{RESOURCE} * f_{MAX} * N_{RESOURCE}$$

Where:

$N_{RESOURCE}$	is the number of resources used in the design.
$K_{RESOURCE}$	is the power constant for the resource in mW/MHz.
$f_{MAX}$	is the max frequency at which the resource is running measured in MHz.

## 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 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:

$$\text{Toggle Rate (MHz)} = 1/2 * f_{MAX} * AF\%$$

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.

## Ambient and Junction Temperatures and Airflow

A common method of characterizing a packaged device's thermal performance is with Thermal Resistance,  $\theta$ . In 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 °C/W.

The most common examples are  $\theta_{JA}$ , Thermal Resistance Junction-to-Ambient (in °C/W) and  $\theta_{JC}$ , Thermal Resistance Junction-to-Case (also in °C/W). Another factor is  $\theta_{JB}$ , Thermal Resistance Junction-to-Board (in °C/W).

Knowing the reference (i.e. ambient, case or board) temperature, the power and the relevant  $\theta$  value, the junction temperature can be calculated as follows.

$$T_J = T_A + \theta_{JA} * P \quad (1)$$

$$T_J = T_C + \theta_{JC} * P \quad (2)$$

$$T_J = T_B + \theta_{JB} * P \quad (3)$$

Where  $T_J$ ,  $T_A$ ,  $T_C$  and  $T_B$  are the Junction, Ambient, Case (or Package) and Board temperatures (in °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 heat sink.  $\theta_{JB}$  applies when the board temperature adjacent to the package is known.

Power Calculator utilizes the Ambient Temperature (°C) to calculate the Junction Temperature (°C) based on the  $\theta_{JA}$  for the targeted device, per equation 1 above. Users can also provide the Airflow values (in LFM) to get a more accurate value of the Junction temperature.

## Managing Power Consumption

One of the most critical factors in design today is reducing the system power consumption. Low power consumption is especially important for hand-held devices and other modern electronic products. There are several design techniques that can significantly reduce overall system power consumption. 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 a portion of their design can be clocked at a lower rate, which will reduce power.
4. Reducing the span of the design across the device. A more closely placed design utilizes fewer routing resources for less power consumption.
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 Linear Feedback Shift Register can toggle as much as 50% Activity Factor, 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:
  - a. Use packages that can better dissipate heat. For example, packages with lower thermal impedance.
  - b. Place heat sinks and thermal planes around the device on the PCB.
  - c. Better airflow techniques using 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. The user 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 supplies that are  $V_{CC}$ ,  $V_{CCIO}$ ,  $V_{CCJ}$ , and  $V_{CCAUX}$ .
5. The nominal  $V_{CC}$  is used by default to calculate the power consumption. A lower or higher  $V_{CC}$  can be chosen from a list of available values.
6. Users can enter an Airflow in Linear Feet per Minute (LFM) along with a Heat Sink option to calculate the Junction Temperature.
7. The default value of the I/O types for the LatticeECP2/M devices is LVCMOS12, 6mA.
8. 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 100MHz is 50MHz.

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)  
+1-503-268-8001 (Outside North America)  
e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2006	01.0	Initial release.
September 2006	01.1	Updated for LatticeECP2/M. Added discussion on Dynamic Power Budgets and Junction Temperature.

September 2006

Technical Note TN1107

## Introduction

This technical note discusses how to access the features of the LatticeECP2™ and LatticeECP2M™ sysDSP™ (Digital Signal Processing) Block described in the LatticeECP2/M Family Data Sheet. Designs targeting the sysDSP Block 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 Block vs. LUT-based Multipliers**

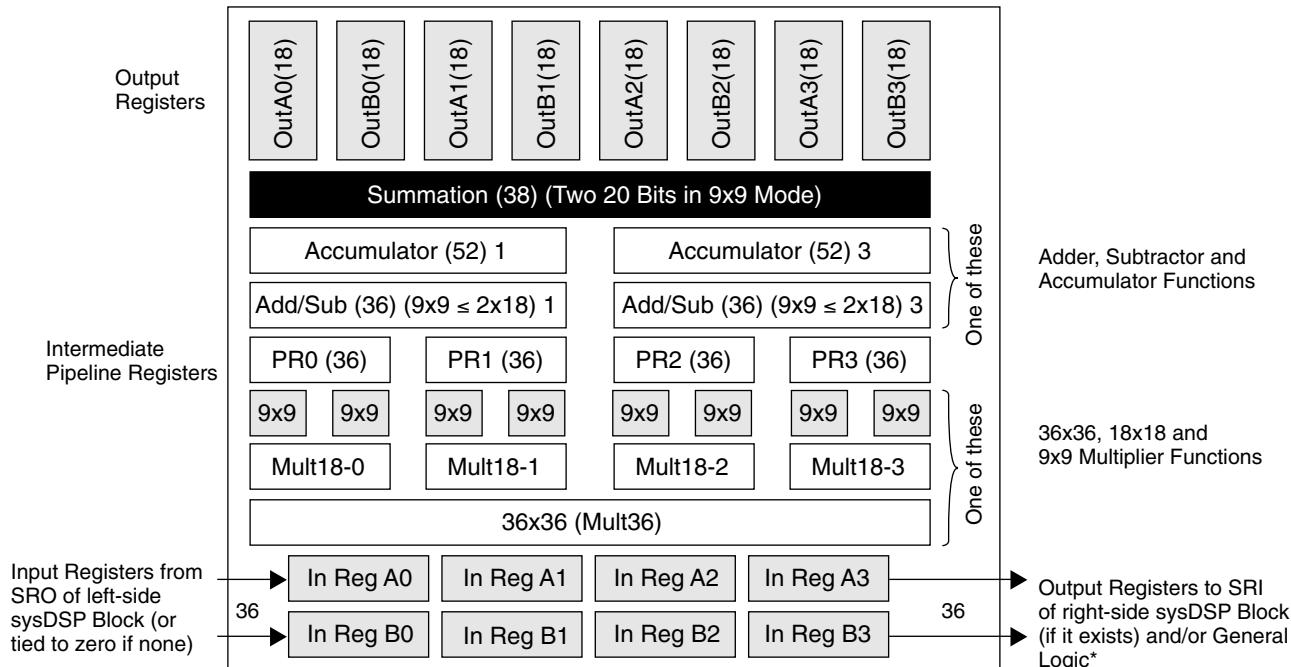
Multiplier Width	Register Pipelining	ECP2-50-7 Uses One sysDSP Block		ECP2-50-7 Uses LUTs	
		f <sub>MAX</sub> (MHz) <sup>1</sup>	LUTs	f <sub>MAX</sub> (MHz) <sup>1</sup>	LUTs
9x9	Input, Multiplier, Output	404	0	124	192
18x18	Input, Multiplier, Output	420	0	95	698
36x36	Input, Multiplier, Output	371	0	61	2732

1. These timing numbers were generated using the ispLEVER® design tool. Exact performance may vary with design and tool version.

## sysDSP Block Hardware

The LatticeECP2/M sysDSP Blocks are located in rows throughout device. Below is a block diagram of one of the sysDSP Blocks:

**Figure 14-1. LatticeECP2/M sysDSP Block**



\*Can only be routed to general logic routing when configured with less than three MULT18X18.

Note: Each sysDSP Block spans nine columns of PFUs.

The sysDSP Block can be configured as:

- One 36x36 Multiplier
  - Basic multiplier, no add/sub/accum/sum blocks
- Four 18x18 Multipliers
  - Two add/sub/accum blocks
  - One summation block for adding four multipliers
- Eight 9x9 Multipliers
  - Four add/sub blocks
  - Two summation blocks

Note that a sysDSP block can only be configured in one mode at a time.

## sysDSP Block Software

### Overview

The sysDSP Block of the LatticeECP2/M device can be targeted in a number of ways.

- The IPexpress™ tool in the ispLEVER software allows the rapid creation of modules implementing sysDSP elements. These modules can then be used in HDL designs as appropriate.
- The coding of certain functions into a design's HDL and allowing the synthesis tools to Infer the use of a sysDSP block.
- The implementation of designs in The MathWorks® Simulink® tool using a Lattice block set. The ispLEVER sys-DSP portion of the ispLEVER design tools will then convert these blocks into HDL as appropriate.
- Instantiation of sysDSP primitives directly in the source code

### Targeting sysDSP Block Using IPexpress

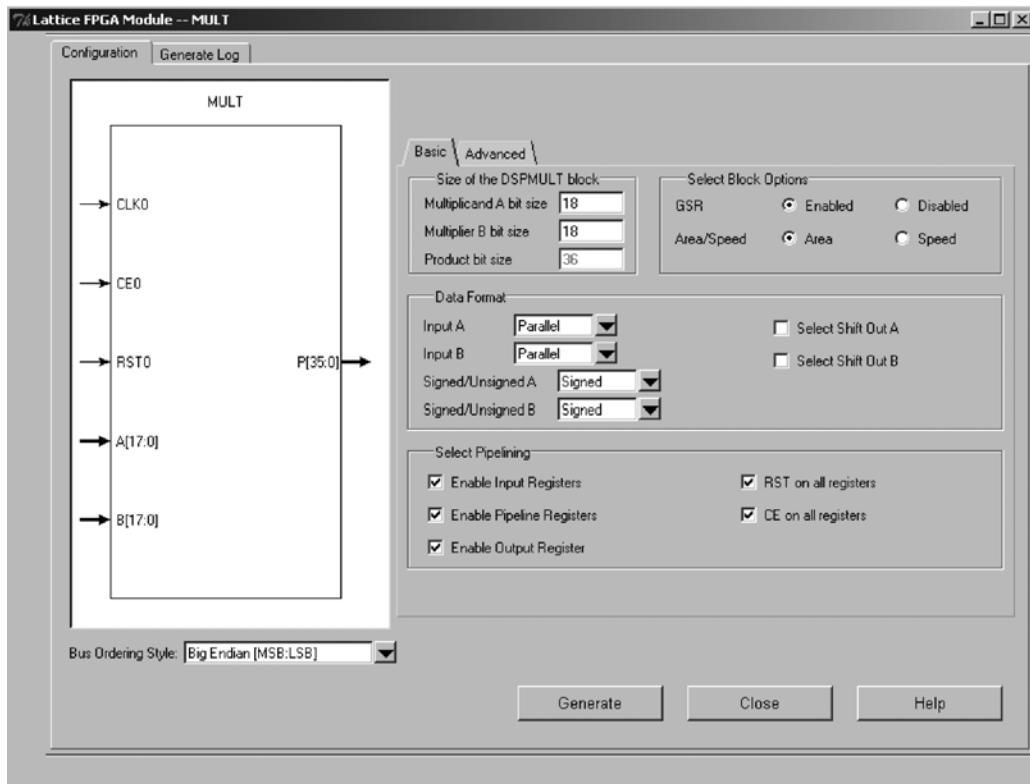
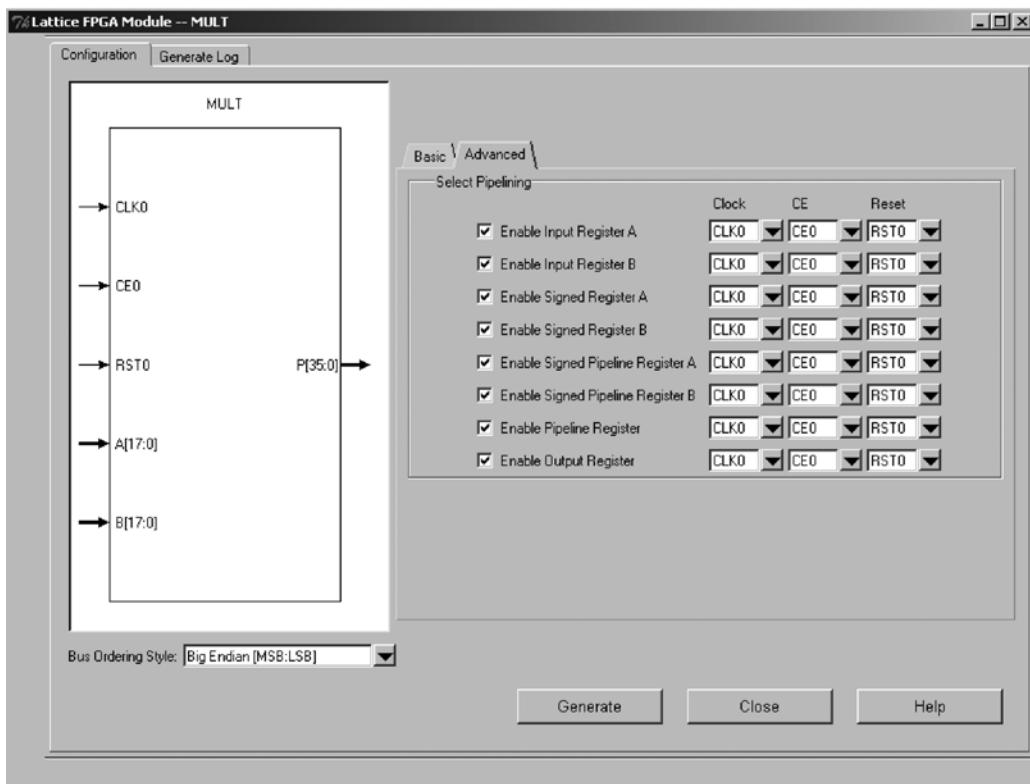
IPexpress allows you to graphically specify sysDSP elements. Once the element is specified, a HDL file is generated, which can be instantiated in a design. IPexpress allows users to configure all ports and set all available parameters. The following modules target the sysDSP Block. For design examples using IPexpress, refer to EXAMPLES in the ispLEVER Software (from the Project Navigator pull down-menu, go to **File** and open **Example**). The following four types of elements can be specified via IPexpress:

- MULT (Multiplier)
- MAC (Multiplier Accumulate)
- MULTADDSUB (Multiplier Add/Subtract)
- MULTADDSUBSUM (Multiply Add/Subtract and SUM)

#### MULT Module

The MULT Module configures elements to be packed into the sysDSP primitives. The Basic mode screen illustrated in Figure 14-2 consists of an optional one clock, one clock enable and one reset tied to all registers. Multiple sys-DSP Blocks can be spanned to accommodate large multiplications. Additional LUTs may be required if multiple sysDSP blocks are needed. Select **Area/Speed** to determine the LUT implementation. 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. The Advanced mode screen, illustrated in Figure 14-3, allows finer control over the register. In the Advanced mode, users can control each register with independent clocks, clock enables and resets. MULT inputs can be from 2 to 72 bits.

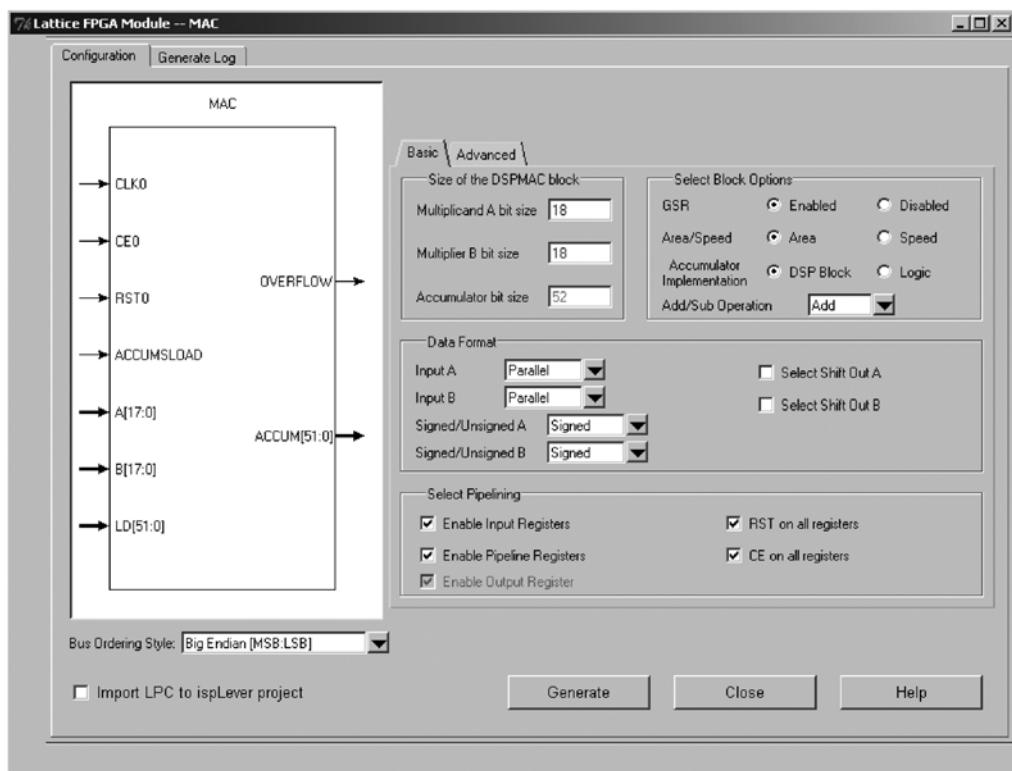
---

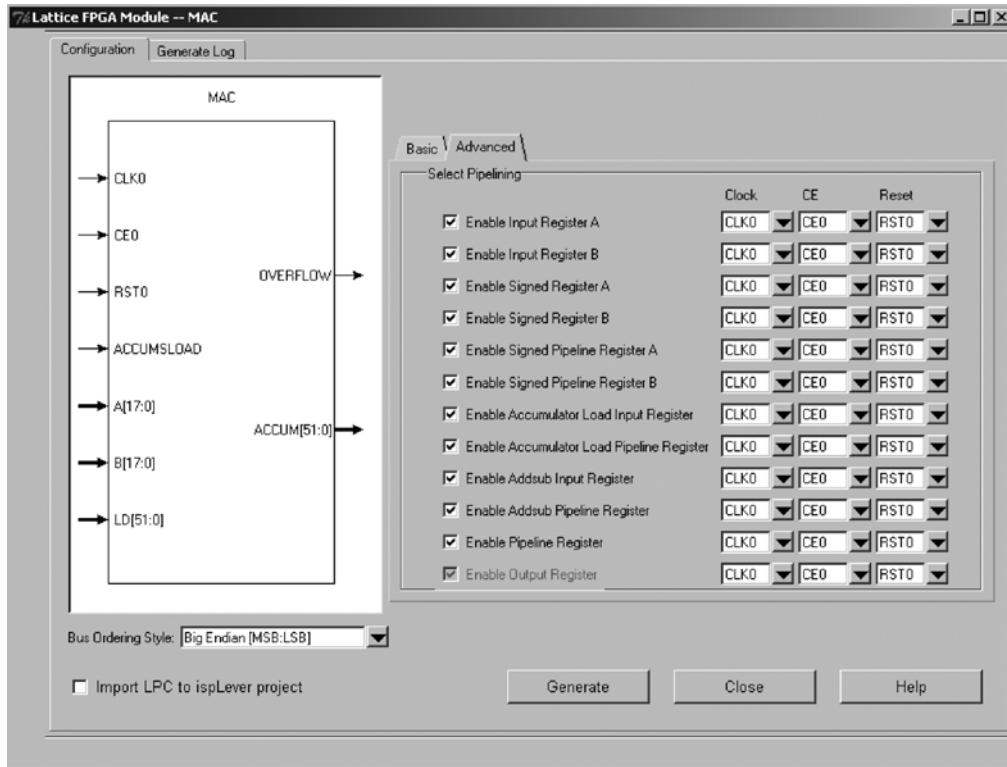
**Figure 14-2. MULT Mode Basic Set-up****Figure 14-3. MULT Mode Advanced Set-up**

## MAC Module

The MAC Module configures multiply accumulate elements to be packed into the primitive MULT18X18MACB. The Basic mode, shown in Figure 14-4, consists of an optional one clock, one clock enable and one reset tied to all registers. Because of the accumulator, the output register is automatically enabled. Multiple sysDSP Blocks can be spanned to accommodate large multiplications. The accumulator of the sysDSP block is 52 bits deep and additional LUTs can be used if a larger accumulation is required. If sysDSP blocks are spanned, additional LUT logic may be required. Select **Area/Speed** to determine the LUT implementation. 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 Accumsload loads the accumulator with the value from the LD port. This is required to initialize and load the first value of the accumulation. The Advanced mode, shown in Figure 14-5, allows finer control over the registers. In the advanced mode, users can control each register with independent clocks, clock enables and resets. MAC inputs can be from 2 to 72 bits.

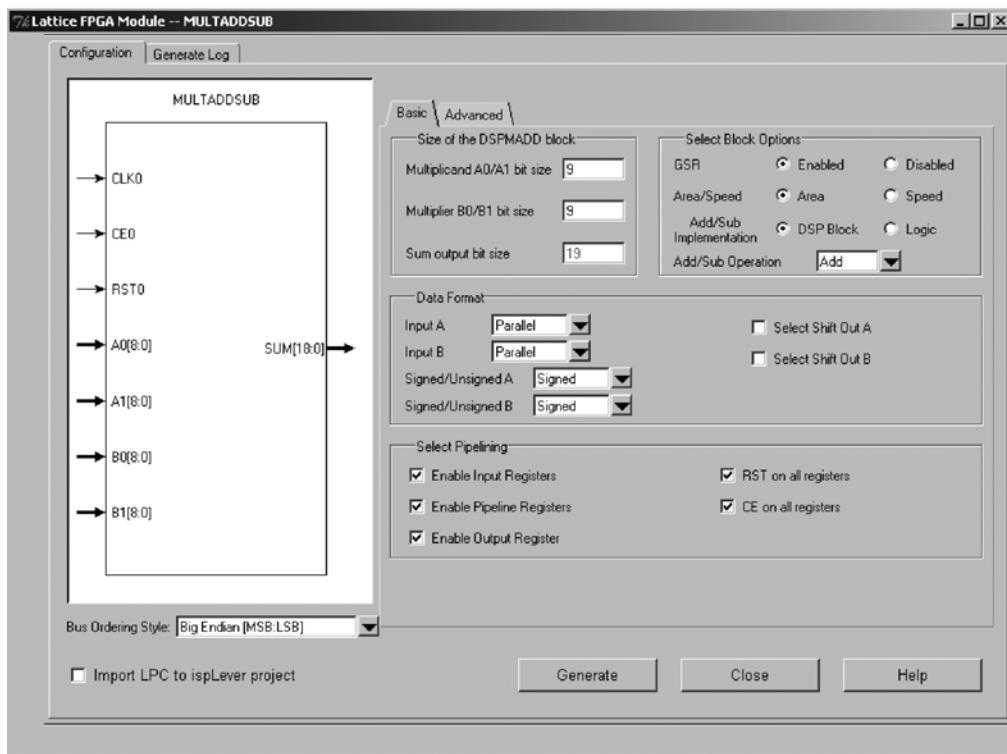
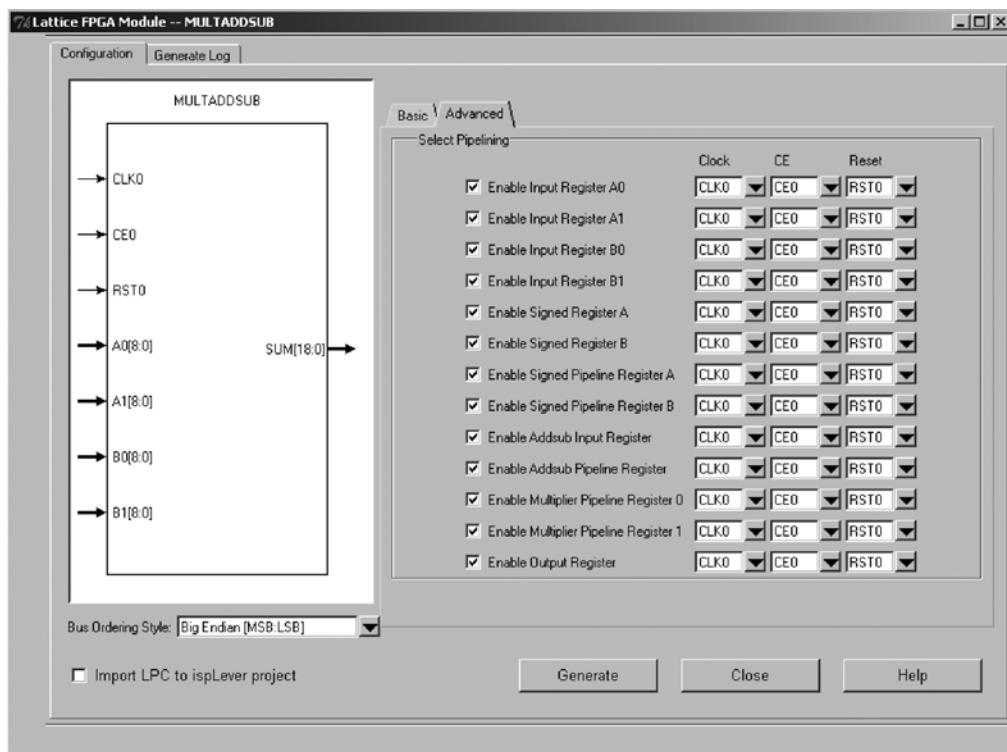
**Figure 14-4. MAC Mode Basic Set-up**



**Figure 14-5. MAC Mode Advanced Set-up**

### MULTADDSSUB Module

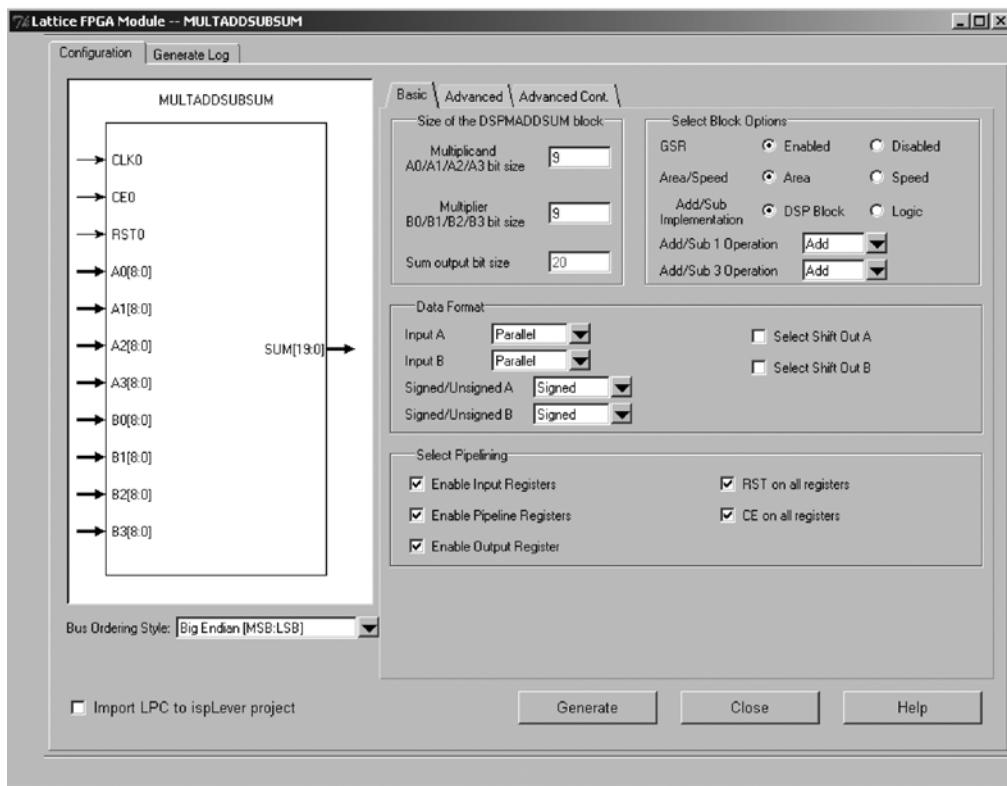
The MULTADDSSUB GUI configures multiplier addition/subtraction elements to be packed into the primitives MULT18X18ADDSUBB or MULT9X9ADDSUBB. The Basic mode, shown in Figure 14-6, consists of an optional one clock, one clock enable and one reset tied to all registers. Multiple sysDSP Blocks can be spanned to accommodate large multiplications. If sysDSP blocks are spanned, additional LUT logic may be required. Select **Area/Speed** to determine the LUT implementation. 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. The Advanced mode, shown in Figure 14-7, provides finer control over the registers. In the advanced mode, users can control each register with independent clocks, clock enables and resets. MULTADDSSUB inputs can be from 2 to 72 bits.

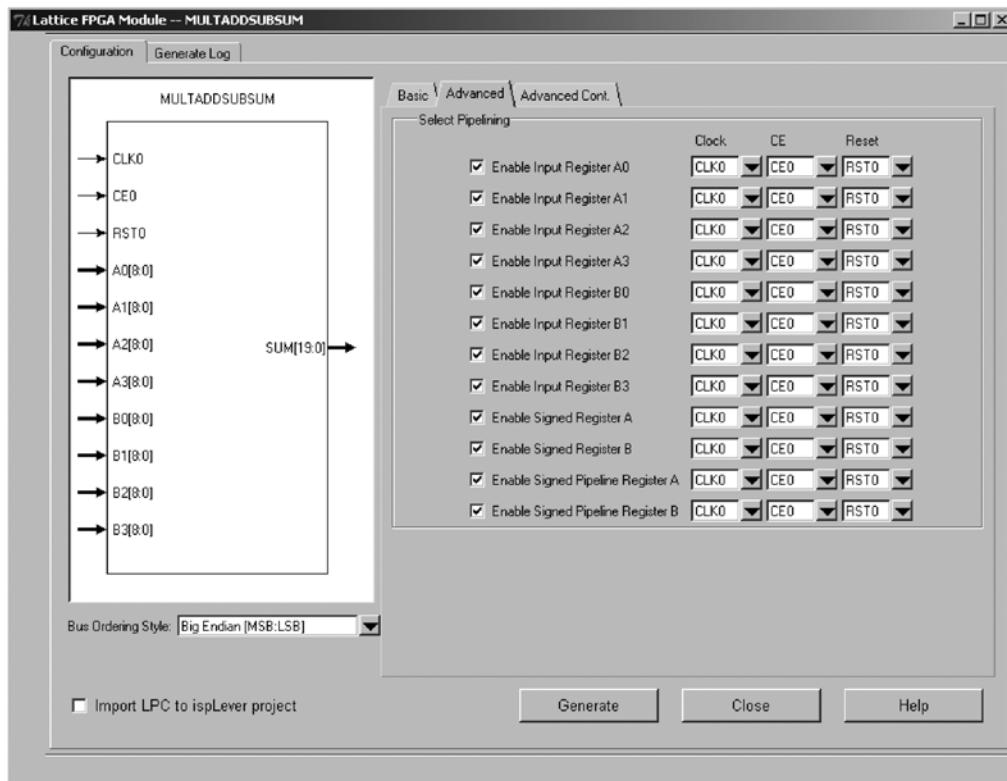
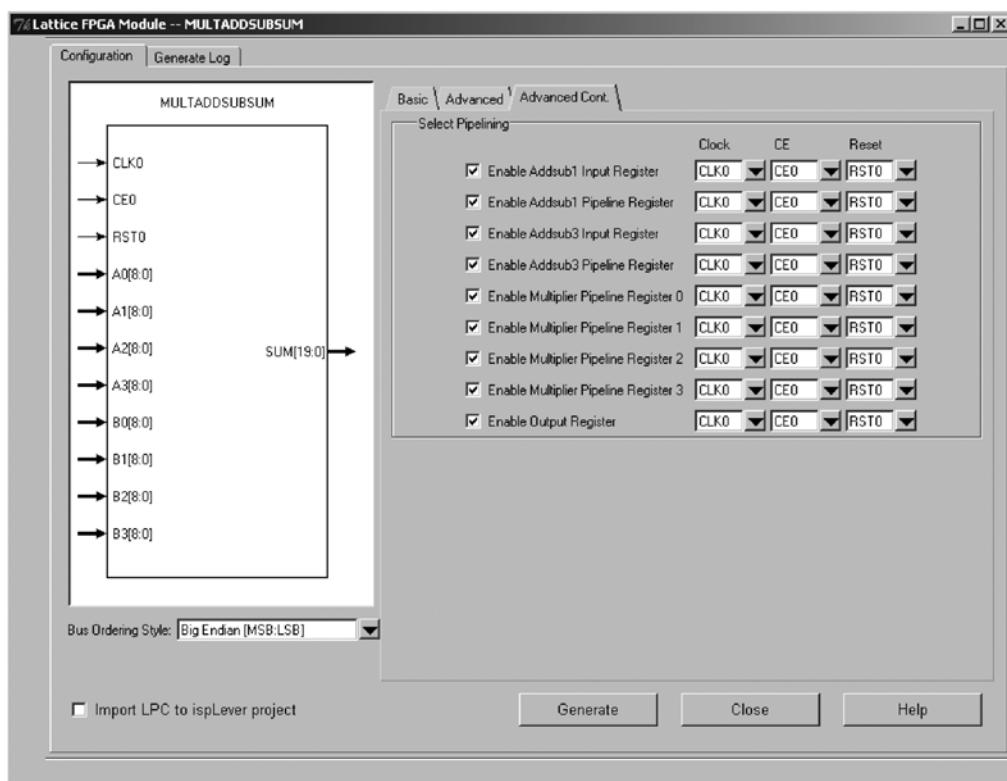
**Figure 14-6. MULTADDSSUB Mode Basic Set-up****Figure 14-7. MULTADDSSUB Mode Advanced Set-up**

### MULTADDSSUBSUM Module

The MULTADDSSUBSUM GUI configures Multiplier Addition/Subtraction Addition elements to be packed into the primitives MULT18X18ADDSSUBSUM or MULT9X9ADDSSUBSUM. The Basic mode, shown in Figure 14-8, consists of an optional one clock, one clock enable and one reset tied to all registers. Multiple sysDSP Blocks can be spanned to accommodate large multiplications. If sysDSP blocks are spanned, additional LUT logic may be required. Select **Area/Speed** to determine the LUT implementation. The input data format can be selected as Parallel, Shift or Dynamic. The Shift format is 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. The Advanced mode, shown in Figure 14-9, provides finer control over the registers. In the advanced mode, users can control each register with independent clocks, clock enables and resets. MULTADDSSUBSUM inputs can be from 2 to 72 bits.

**Figure 14-8. MULTADDSSUBSUM Mode Basic Set-up**



**Figure 14-9. MULTADDSSUBSUM Mode Advance****Figure 14-10. MULTADDSSUBSUM Mode Advance**

## Targeting the sysDSP Block by Inference

The Inferencing flow enables the design tools to infer sysDSP Blocks from a HDL design. It is important to note that when using the Inferencing flow, unless the code style matches the sysDSP Block, results will not be optimal. Consider the following Verilog and VHDL examples:

```
// This Verilog example will be mapped into single MULT18X18MACB with the output register enabled
module mult_acc (dataout, dataax, dataay, clk);
    output [16:0] dataout;
    input [7:0] dataax, dataay;
    input clk;
    reg [16:0] dataout;

    wire [15:0] multa = dataax * dataay; // 9x9 Multiplier
    wire [16:0] adder_out;
    assign adder_out = multa + dataout; // Accumulator
    always @(posedge clk)
    begin
        dataout <= adder_out; // Output Register of the Accumulator
    end
endmodule

-- This VHDL example will be mapped into single MULT18X18MACB with all the registers enabled
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity mac is
port (clk, reset : in std_logic;
      dataax, dataay : in std_logic_vector(8 downto 0);
      dataout : out std_logic_vector(17 downto 0));
end;

architecture arch of mac is
signal dataax_reg, dataay_reg : std_logic_vector(8 downto 0);
signal multout, multout_reg : std_logic_vector(17 downto 0);
signal addout : std_logic_vector(17 downto 0);
signal dataout_reg : std_logic_vector(17 downto 0);
begin

dataout <= dataout_reg;

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;

multout <= dataax_reg * dataay_reg;

process (clk, reset)
begin
if (reset = '1') then
    multout_reg <= (others => '0');
elsif (clk'event and clk='1') then
    multout_reg <= multout;
end if;
```

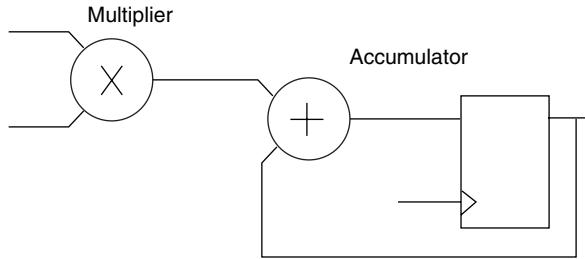
```
end process;

addout <= multout_reg + dataout_reg;

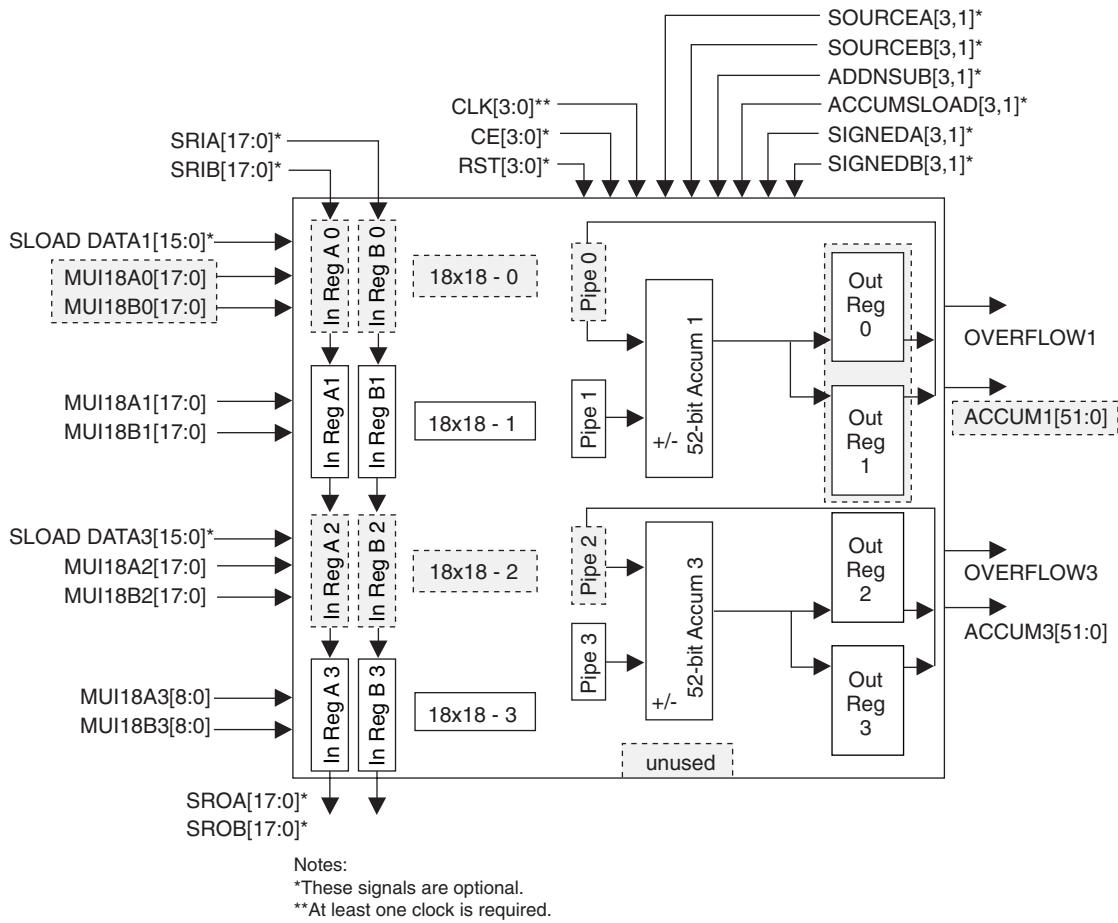
process (clk, reset)
begin
if (reset = '1') then
  dataout_reg <= (others => '0');
elsif (clk'event and clk='1') then
  dataout_reg <= addout;
end if;
end process;
end arch;
```

The above RTL will infer the following block diagram:

**Figure 14-11. MULT18X18MACB Block Diagram**



This block diagram can be mapped directly into the sysDSP primitives. Note that if a test point were added between the multiplier and the accumulator, or two output registers, etc. the code could not be mapped into a MULT18X18MACB of a sysDSP Block. Therefore, options that could be included in a design are input registers, pipeline registers, etc. For more Inferring design examples refer to EXAMPLES in the ispLEVER software.

**Figure 14-12. MAC18X18MACB Packed into a sysDSP Block**

## sysDSP Blocks in the Report File

To check the configuration of the sysDSP Blocks in your design you can look at the MAP and Post PAR report files. The MAP report file shows the mapped sysDSP components/primitives in your design. The Post PAR report file shows the number of components in each sysDSP Block. The report files that follow show how the inferred MAC was used.

### MAP Report File

```
. MULT18X18MACB addout_17_0:

Multiplier
  Operation          Unsigned
  Operation Registers    CLK      CE      RST
  -----
  Input
  Pipeline
  Operation Registers    CLK      CE      RST
  -----
  Input
  Pipeline
```

```

AddSub
      Operation          Add
      Operation Registers   CLK    CE     RST
-----
      Input
      Pipeline

Data
      Input Registers      CLK    CE     RST
-----
      A                  CLK0   CE0    RST0
      B                  CLK0   CE0    RST0
      Pipeline Registers  CLK    CE     RST
-----
      Pipe
      Output Register     CLK0   CE0    RST0
-----
      Output             CLK0   CE0    RST0

Other
      GSR      ENABLED
      Number Of Mapped DSP Components:
-----
      MULT36X36B        0
      MULT18X18B         0
      MULT18X18MACB      1
      MULT18X18ADDSUBB   0
      MULT18X18ADDSUBSUMB 0
      MULT9X9B           0
      MULT9X9ADDSUBB     0
      MULT9X9ADDSUBSUMB   0
-----
```

## Post PAR Report File

### DSP Utilization Summary:

DSP Block #:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
--------------	---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----

# of MULT36X36B

# of MULT18X18B

# of MULT18X18MACB

1

# of MULT18X18ADDSUBB

# of MULT18X18ADDSUBSUMB

# of MULT9X9B

# of MULT9X9ADDSUBB

# of MULT9X9ADDSUBSUMB

DSP Block 1 Component\_Type

Instance\_Name

DSP Block 2 Component\_Type

Instance\_Name

DSP Block 3 Component\_Type

Instance\_Name

DSP Block 4 Component\_Type

Instance\_Name

DSP Block 5 Component\_Type

Instance\_Name

DSP Block 6 Component\_Type

Instance\_Name

DSP Block 7 Component\_Type

Instance\_Name

DSP Block 8 Component\_Type

Instance\_Name

DSP Block 9 Component\_Type

Instance\_Name

R45C81 MULT18X18MACB addout\_17\_0

DSP Block 10 Component\_Type

Instance\_Name

DSP Block 11 Component\_Type

Instance\_Name

DSP Block 12 Component\_Type

Instance\_Name

DSP Block 13 Component\_Type

Instance\_Name

DSP Block 14 Component\_Type

Instance\_Name

DSP Block 15 Component\_Type

Instance\_Name

DSP Block 16 Component\_Type

Instance\_Name

DSP Block 17 Component\_Type

Instance\_Name

DSP Block 18 Component\_Type

Instance\_Name

## Targeting the sysDSP Block Using Simulink

### Simulink Overview

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/).

#### Why is Simulink used?

- It allows users to create algorithms using floating point numbers.
- It helps users convert floating point algorithms into fixed point algorithms.

#### How does Simulink fit into the normal ispLEVER design flow?

- Once you have converted your algorithm working in fixed point. You can use the Lattice ispDSP Block to create HDL files, which can be instantiated in your HDL design. Currently there is only support for VHDL.

#### What does Lattice provide?

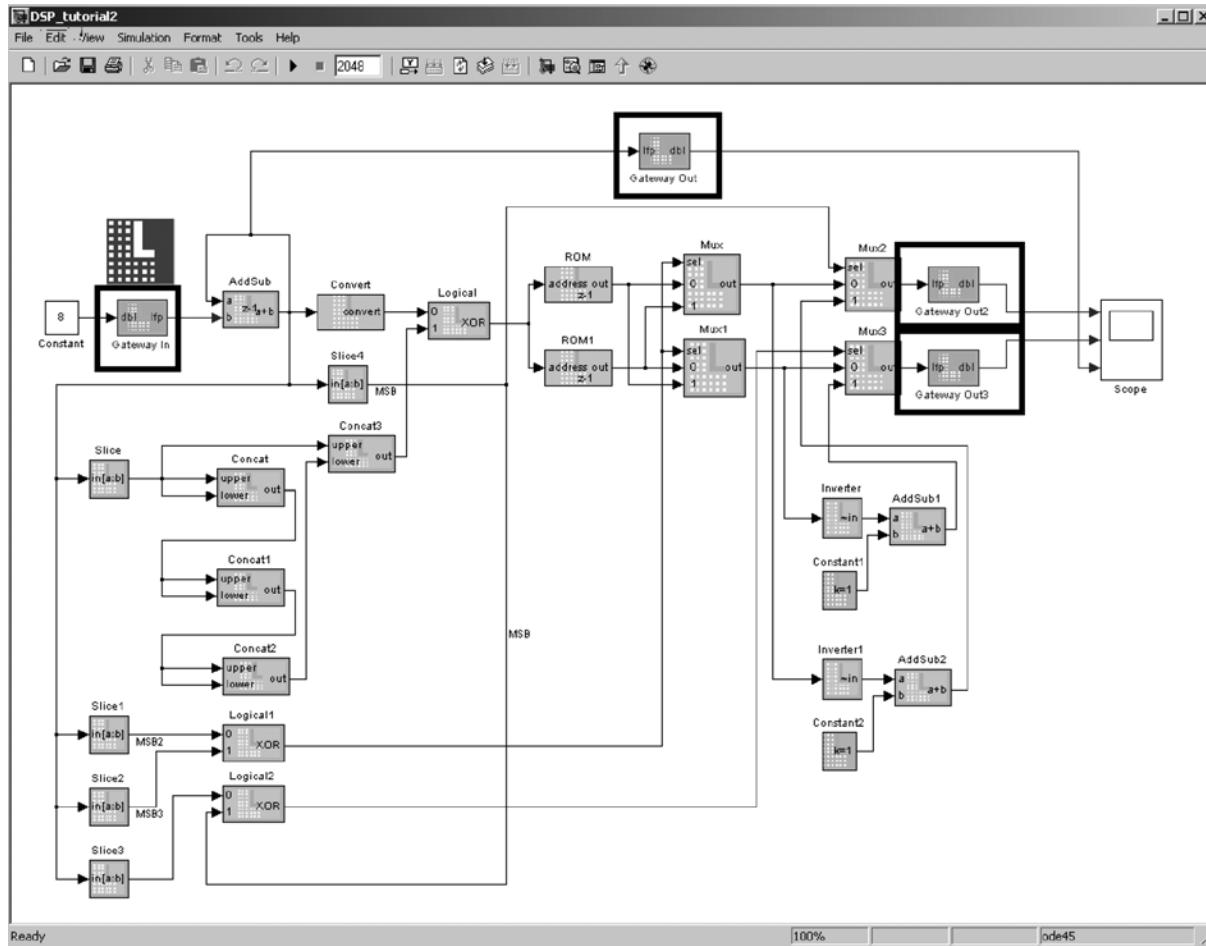
- Lattice provides a library of blocks for the Simulink tool, which include Multipliers, Adders, Registers, and other standard building blocks. Besides the basic building blocks there are a couple of unique Lattice blocks:

##### Gateways In and Out

Everything between Gateways In and Out represents the HDL code. Everything before a Gateway In is the stimulus your test bench. Everything after the Gateway Out are the signals you will be monitoring in the test bench. Below is an example. The box on the left contains Gateways In's and the three boxes on the right contain Gateway Outs in Figure 14-13.

##### Generate

The Generate block is used to convert Fixed point Simulink design into HDL files which can be instantiated in a HDL design. The Generate Block is identified by the Lattice logo and can be seen in Figure 14-13.

**Figure 14-13. Simulink Design**

## Targeting the sysDSP Block by Instantiating Primitives

The sysDSP Block can be targeted by instantiating the sysDSP Block primitives into the 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 Block primitives.

## sysDSP Block 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

- Hotline: 1-800-LATTICE (North America)  
+1-503-268-8001 (Outside North America)
- e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)
- Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
February 2006	01.0	Initial release.
September 2006	01.1	Updated table 1 with new software numbers.
		Updated figure 1 summation size from 37 to 38

## Appendix A. DSP Block Primitives

### MULT18X18B

```

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 CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
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 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;
parameter REG_INPUTA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";

```

### MULT18X18ADDSUBB

```

input A017,A016,A015,A014,A013,A012,A011,A010,A09;
input A08,A07,A06,A05,A04,A03,A02,A01,A00;
input A117,A116,A115,A114,A113,A112,A111,A110,A19;
input A18,A17,A16,A15,A14,A13,A12,A11,A10;
input B017,B016,B015,B014,B013,B012,B011,B010,B09;
input B08,B07,B06,B05,B04,B03,B02,B01,B00;
input B117,B116,B115,B114,B113,B112,B111,B110,B19;
input B18,B17,B16,B15,B14,B13,B12,B11,B10;
input SIGNEDA, SIGNEDB, SOURCEA0, SOURCEA1, SOURCEB0, SOURCEB1, ADDNSUB;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
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
SUM36,SUM35,SUM34,SUM33,SUM32,SUM31,SUM30,SUM29,SUM28,SUM27,SUM26,SUM25,SUM24,SUM23,SUM22,SUM21,SU
M20,SUM19,SUM18,SUM17,SUM16,SUM15,SUM14,SUM13,SUM12,SUM11,SUM10,SUM9,SUM8,SUM7,SUM6,SUM5,SUM4,SUM3
,SUM2,SUM1,SUM0;

```

```

parameter REG_INPUTA0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";

```

## MULT18X18ADDSUBSUMB

```

input A017,A016,A015,A014,A013,A012,A011,A010,A09;
input A08,A07,A06,A05,A04,A03,A02,A01,A00;
input A117,A116,A115,A114,A113,A112,A111,A110,A19;
input A18,A17,A16,A15,A14,A13,A12,A11,A10;
input A217,A216,A215,A214,A213,A212,A211,A210,A29;
input A28,A27,A26,A25,A24,A23,A22,A21,A20;
input A317,A316,A315,A314,A313,A312,A311,A310,A39;
input A38,A37,A36,A35,A34,A33,A32,A31,A30;
input B017,B016,B015,B014,B013,B012,B011,B010,B09;
input B08,B07,B06,B05,B04,B03,B02,B01,B00;
input B117,B116,B115,B114,B113,B112,B111,B110,B19;
input B18,B17,B16,B15,B14,B13,B12,B11,B10;
input B217,B216,B215,B214,B213,B212,B211,B210,B29;
input B28,B27,B26,B25,B24,B23,B22,B21,B20;
input B317,B316,B315,B314,B313,B312,B311,B310,B39;
input B38,B37,B36,B35,B34,B33,B32,B31,B30;
input SIGNEDA, SIGNEDB, ADDNSUB1, ADDNSUB3;
input SOURCEA0, SOURCEA1, SOURCEA2, SOURCEA3;
input SOURCEB0, SOURCEB1, SOURCEB2, SOURCEB3;

```

```
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
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
SUM37,SUM36,SUM35,SUM34,SUM33,SUM32,SUM31,SUM30,SUM29,SUM28,SUM27,SUM26,SUM25,SUM24,SUM23,SUM22,SU
M21,SUM20,SUM19,SUM18,SUM17,SUM16,SUM15,SUM14,SUM13,SUM12,SUM11,SUM10,SUM9,SUM8,SUM7,SUM6,SUM5,SUM
4,SUM3,SUM2,SUM1,SUM0;
parameter REG_INPUTA0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
```

---

```

parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB1_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB1_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB1_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB1_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB1_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB1_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB3_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB3_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB3_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB3_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB3_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB3_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";

```

## MULT18X18MACB

```

input A17,A16,A15,A14,A13,A12,A11,A10,A9;
input A8,A7,A6,A5,A4,A3,A2,A1,A0;
input B17,B16,B15,B14,B13,B12,B11,B10,B9;
input B8,B7,B6,B5,B4,B3,B2,B1,B0;
input ADDNSUB, SIGNEDA, SIGNEDB,ACCUMSLOAD;
input SOURCEA, SOURCEB;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
input LD51, LD50, LD49, LD48, LD47, LD46, LD45, LD44, LD43, LD42, LD41, LD40
input LD39, LD38, LD37, LD36, LD35, LD34, LD33, LD32, LD31, LD30
input LD29, LD28, LD27, LD26, LD25, LD24, LD23, LD22, LD21, LD20
input LD19, LD18, LD17, LD16, LD15, LD14, LD13, LD12, LD11, LD10
input LD9, LD8, LD7, LD6, LD5, LD4, LD3, LD2, LD1, LD0;
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
ACCUM51,ACCUM50,ACCUM49,ACCUM48,ACCUM47,ACCUM46,ACCUM45,ACCUM44,ACCUM43,ACCUM42,ACCUM41,ACCUM40,AC
CUM39,ACCUM38,ACCUM37,ACCUM36,ACCUM35,ACCUM34,ACCUM33,ACCUM32,ACCUM31,ACCUM30,ACCUM29,ACCUM28,ACC
M27,ACCUM26,ACCUM25,ACCUM24,ACCUM23,ACCUM22,ACCUM21,ACCUM20,ACCUM19,ACCUM18,ACCUM17,ACCUM16,ACCUM1
5,ACCUM14,ACCUM13,ACCUM12,ACCUM11,ACCUM10,ACCUM9,ACCUM8,ACCUM7,ACCUM6,ACCUM5,ACCUM4,ACCUM3,ACCUM2,
ACCUM1,ACCUM0,OVERFLOW;
parameter REG_INPUTA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";

```

---

---

```

parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ACCUMSLOAD_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ACCUMSLOAD_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ACCUMSLOAD_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ACCUMSLOAD_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ACCUMSLOAD_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ACCUMSLOAD_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";

```

## MULT36X36B

```

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 SIGNEDA, SIGNEDB;
input CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3;
output P71,P70,P69,P68,P67,P66,P65,P64,P63,P62,P61,P60,P59,P58,P57,P56,P55,P54;
output P53,P52,P51,P50,P49,P48,P47,P46,P45,P44,P43,P42,P41,P40,P39,P38,P37,P36;
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;
parameter REG_INPUTA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";

```

---

**MULT9X9B**

```

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 P17,P16,P15,P14,P13,P12,P11,P10,P9,P8,P7,P6,P5,P4,P3,P2,P1,P0;
parameter REG_INPUTA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";

```

**MULT9X9ADDSUBB**

```

input A08,A07,A06,A05,A04,A03,A02,A01,A00;
input A18,A17,A16,A15,A14,A13,A12,A11,A10;
input B08,B07,B06,B05,B04,B03,B02,B01,B00;
input B18,B17,B16,B15,B14,B13,B12,B11,B10;
input SIGNEDA, SIGNEDB, ADDNSUB;
input SOURCEA0, SOURCEA1, SOURCEB0, SOURCEB1;
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 SUM18,SUM17,SUM16,SUM15,SUM14,SUM13,SUM12,SUM11,SUM10,SUM9,SUM8,SUM7,SUM6,SUM5,SUM4,SUM3,SUM2,SUM1
,SUM0;
parameter REG_INPUTA0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE0_RST = " RST0, RST1, RST2, RST3 ";

```

---

```

parameter REG_PIPELINE1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";

```

## MULT9X9ADDSUBSUMB

```

input A08,A07,A06,A05,A04,A03,A02,A01,A00;
input A18,A17,A16,A15,A14,A13,A12,A11,A10;
input A28,A27,A26,A25,A24,A23,A22,A21,A20;
input A38,A37,A36,A35,A34,A33,A32,A31,A30;
input B08,B07,B06,B05,B04,B03,B02,B01,B00;
input B18,B17,B16,B15,B14,B13,B12,B11,B10;
input B28,B27,B26,B25,B24,B23,B22,B21,B20;
input B38,B37,B36,B35,B34,B33,B32,B31,B30;
input SIGNEDA, SIGNEDB,ADDNSUB1,ADDNSUB3;
input SOURCEA0, SOURCEA1, SOURCEA2, SOURCEA3;
input SOURCEB0, SOURCEB1, SOURCEB2, SOURCEB3;
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
SUM19,SUM18,SUM17,SUM16,SUM15,SUM14,SUM13,SUM12,SUM11,SUM10,SUM9,SUM8,SUM7,SUM6,SUM5,SUM4,SUM3,SUM2,SUM1,SUM0;
parameter REG_INPUTA0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTA3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTA3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTA3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB0_RST = " RST0, RST1, RST2, RST3 ";

```

---

```
parameter REG_INPUTB1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_INPUTB3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_INPUTB3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_INPUTB3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE2_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE2_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE2_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_PIPELINE3_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_PIPELINE3_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_PIPELINE3_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_OUTPUT_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_OUTPUT_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_OUTPUT_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDA_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDA_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDA_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_SIGNEDB_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_SIGNEDB_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_SIGNEDB_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB1_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB1_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB1_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB1_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB1_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB1_1_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB3_0_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB3_0_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB3_0_RST = " RST0, RST1, RST2, RST3 ";
parameter REG_ADDNSUB3_1_CLK = " NONE, CLK0, CLK1, CLK2, CLK3 ";
parameter REG_ADDNSUB3_1_CE = " CE0, CE1, CE2, CE3 ";
parameter REG_ADDNSUB3_1_RST = " RST0, RST1, RST2, RST3 ";
parameter GSR = " Enabled, Disabled ";
```

## Introduction

The configuration memory in the LatticeECP2™ and LatticeECP2M™ 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 LatticeECP2/M when it powers-up, or any other time the device needs to be updated.

To support multiple configuration options the LatticeECP2/M supports the Lattice sysCONFIG™ interface, as well as the dedicated ispJTAG™ port. The available configuration options, or ports, are listed in Table 15-1.

**Table 15-1. Supported Configuration Ports**

Interface	Port
sysCONFIG	SPI
	SPI <sup>m</sup>
	Slave Serial
	Slave Parallel
ispJTAG	JTAG (IEEE 1149.1 and IEEE 1532 compliant)

This technical note covers all of the configuration options available for LatticeECP2/M.

## General Configuration Flow

The LatticeECP2/M will enter configuration mode when one of three things happens, power is applied to the chip, the PROGRAMN pin is driven low, or when a JTAG 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 the LatticeECP2/M has finished initializing, 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 LatticeECP2/M which port it is going to configure from. The LatticeECP2/M then begins reading data from the selected port and starts looking for the preamble, BDB3 (hex). All data after the preamble is valid configuration data.

When the LatticeECP2/M has finished reading all of the configuration data, assuming there have been no errors, the DONE pin goes high and the LatticeECP2/M enters user mode, in other words the device begins to function according to the user's design.

Note that the LatticeECP2/M may also be programmed via JTAG. When programming via JTAG, the PROGRAMN, INITN, and DONE signals have no meaning, because JTAG, per the IEEE standard, takes complete control of the chip and its I/Os.

The following sections define each configuration pin, each configuration mode, and all of the configuration options for the LatticeECP2/M.

## Configuration Pins

The LatticeECP2/M supports two types of configuration pins, dedicated and dual-purpose. The dedicated pins are used exclusively for configuration; the dual-purpose pins, when not being used for configuration, 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 the user must adhere to the following:

- The general purpose I/O (GPIO) must maintain the same I/O direction as it has during configuration, in other words, if the pin is an input during configuration it must remain an input as a GPIO, if an output during configuration it must remain an output as a GPIO, if a bi-directional it must remain a bi-directional as a GPIO.
- The I/O type must remain the same, in other words if the pin is a 3.3V CMOS pin (LVCMOS33) during configuration it must remain a 3.3V CMOS pin as a GPIO.
- The user must select the correct CONFIG\_MODE setting using the Design Planner (formerly called the Preference Editor).
- The user is responsible for insuring that no internal or external logic will interfere with device configuration.

Also, if slave parallel configuration mode is not being used then one or both of the parallel port chip selects (CSN, CS1N) must be high or tri-state during configuration.

Programmable options control the direction and type of each dual-purpose configuration pin. These options are controlled via pin preferences in Lattice's ispLEVER® software, or as HDL source file attributes.

The LatticeECP2/M also supports ispJTAG 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. Configuration Pins for the LatticeECP2/M**

Pin Name	I/O Type	Pin Type	Mode Used
CFG[2:0]	Input, weak pull-up	Dedicated	All
PROGRAMN	Input, weak pull-up	Dedicated	All
INITN	Bi-Directional Open Drain, weak pull-up	Dedicated	All
DONE	Bi-Directional Open Drain with weak pull-up, or Active Drive	Dedicated	All
CCLK	Input or Output	Dedicated	All
DI/CSSPI0N	Input, weak pull up	Dual-Purpose	Serial, SPI, SPIm
DOUT/CS0N : ECP2	Output	Dual-Purpose	Parallel, Serial, SPI
DOUT/CS0N/ CSSPI1N : ECP2M	Output	Dual-Purpose	Parallel, Serial, SPI, SPIm
CSN	Input, weak pull-up	Dual-Purpose	Parallel
CS1N	Input, weak pull-up	Dual-Purpose	Parallel
WRITEN	Input, weak pull-up	Dual-Purpose	Parallel
BUSY/SISPI	Output, tri-state, weak pull-up	Dual-Purpose	Serial, SPI, SPIm
D[0]/SPIFASTN	Input or Output	Dual-Purpose	Parallel, SPI, SPIm
D[1:5]			Parallel
D[6]/SPID1			Parallel, SPIm
D[7]/SPID0			Parallel, SPI, SPIm
TDI	Input, weak pull-up	Dedicated	JTAG
TDO	Output, weak pull-up	Dedicated	JTAG
TCK	Input with Hysteresis	Dedicated	JTAG
TMS	Input, weak pull-up	Dedicated	JTAG

Note: Weak pull-ups consist of a current source of 30µA to 150µA. The pull-ups for sysCONFIG dedicated and dual-purpose pins track V<sub>CCIO8</sub>; the pull-ups for TDI, TDO, and TMS track V<sub>CCU</sub>.

## Dedicated Control Pins

The following sub-sections describe the LatticeECP2/M dedicated sysCONFIG pins. These pins are powered by V<sub>CCIO8</sub>.

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.

### **CFG[2:0]**

The Configuration Mode pins, CFG[2:0], are dedicated inputs with weak pull-ups. The CFG pins are sampled on the rising edge of INITN and are used to select the configuration mode, i.e. what type of device the LatticeECP2/M will configure from. As a consequence the CFG pins determine which groups of dual-purpose pins will be used for device configuration (see the right most column in Table 15-2). See Table 15-3 for a list of Configuration Modes.

**Table 15-3. Configuration Modes**

<b>Configuration Mode</b>		<b>CFG[2]</b>	<b>CFG[1]</b>	<b>CFG[0]</b>	<b>D[0]/SPIFASTN</b>
SPI	Normal (0x03)	0	0	0	Pull-Up
	Fast (0x0B)	0	0	0	Pull-Down
Reserved		0	0	1	X
SPIm	Normal (0x03)	0	1	0	Pull-Up
	Fast (0x0B)	0	1	0	Pull-Down
Reserved		0	1	1	X
Reserved		1	0	0	X
Slave Serial		1	0	1	X
Reserved		1	1	0	X
Slave Parallel		1	1	1	D0

Notes:

JTAG is always available for IEEE 1149.1 and 1532 support.

### **PROGRAMN**

The PROGRAMN pin is a dedicated input with a weak pull-up. This pin is used to initiate a non-JTAG SRAM configuration sequence.

A high to low signal applied to PROGRAMN takes the device out of user mode and sets it into configuration mode. The low to high transition will initiate the configuration process. Holding the PROGRAMN pin low will delay the configuration process. The PROGRAMN pin can be used to trigger configuration at any time. If the device is being accessed by JTAG then PROGRAMN will be ignored until the device is released from JTAG mode.

### **INITN**

The INITN pin is a bidirectional open drain control pin. INITN is capable of driving a low pulse out as well as detecting a low pulse driven in.

When the PROGRAMN pin is driven low, or a JTAG Reset instruction is received, or after the internal Power-On-Reset signal is released during power-up, the INITN pin will be driven low to reset the internal configuration circuitry. Once the PROGRAMN pin is driven high, the configuration initialization begins. Once the configuration initialization is completed, the INITN pin will go high. To delay configuration the INITN pin can be held low externally. The device will not enter configuration mode as long as the INITN pin is held low. A low to high transition on INITN causes the CFG pins to be sampled, telling the LatticeECP2/M which port to use, and starts configuration.

During configuration the INITN pin becomes an error detection pin. If a Verify ID or CRC error is detected during configuration INITN will be driven low. The error will be cleared at the beginning of the next configuration.

### **DONE**

The DONE pin is a dedicated bi-directional open drain with a weak pull-up (default), or it is an actively driven pin.

DONE goes low when INITN goes low, when INITN and PROGRAMN go high, and the internal Done bit is programmed at the end of configuration, the DONE pin will be released (or driven high, if it is an actively driven pin). The DONE pin can be held low externally and, depending on the wake-up sequence selected, the device will not

become functional until the DONE pin is externally brought high. Externally delaying the wake-up sequence using the DONE pin is a good way to synchronize the wake-up of multiple FPGAs; it is also required when configuring multiple FPGAs from a single configuration device.

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 reverts to internal logic and will be high).

### CCLK

CCLK is a dedicated bi-directional pin; direction depends on whether a Master or Slave mode is selected. If a Master mode (SPI or SPIm) is selected, via the CFG pins, the CCLK pin becomes an output; otherwise CCLK is an input.

If the CCLK pin becomes an output, the internal programmable oscillator is connected to CCLK and is driven out to slave devices. CCLK will stop 100 to 500 clock cycles after the DONE pin is brought high. The extra clock cycles ensure that enough clocks are provided to wake-up other devices in the chain. When stopped, CCLK becomes an input (tri-stated output). CCLK will restart (become an output again) on the next configuration initialization sequence.

The MCCLK\_FREQ parameter (one of the global settings in the Design Planner of ispLEVER) controls the CCLK master frequency (see data sheet On-Chip Oscillator section for the frequency selection). The software default setting for the configuration CCLK is 2.5 MHz. One of the first operations during configuration is the MCCLK\_FREQ parameter; once this parameter is loaded the frequency changes to the selected value. Care should be exercised not to exceed the frequency specification of the slave devices or the signal integrity capabilities of the PCB layout.

### Dual-Purpose sysCONFIG Pins

The following is a list of the dual-purpose sysCONFIG pins. If any of these pins are used for configuration and for user I/O, the user must adhere to the requirements listed at the start of the Configuration pin sections.

These pins are powered by V<sub>CCIO8</sub>.

#### DI/CSSPI0N

The DI/CSSPI0N dual-purpose pin is designated as DI (Data Input) for Serial configurations. DI has an internal weak pull-up. DI captures data on the rising edge of CCLK.

In SPI or SPIm mode the DI/CSSPI0N becomes a low true Chip Select output that drives the SPI Serial Flash chip select.

#### DOUT/CS0N/CSSPI1N

The DOUT/CS0N/CSSPI1N pin is an output pin and has three purposes.

For serial and parallel configuration modes, when BYPASS mode is selected, this pin becomes DOUT (see Figure 15-9). 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.

For parallel configuration mode there is a FLOW\_THROUGH option as well. When FLOW\_THROUGH mode is selected this pin becomes Chip Select Out (CS0N). When the device is fully configured, and the Flow-Through instruction in the bitstream is executed, the CS0N pin is driven low to enable the next device. The data pins, D[0:7], are wired in parallel to each device in the chain (see Figure 15-10).

In SPIm mode, the sysCONFIG daisy chaining mode of configuration is not supported. For the LatticeECP2M devices only this pin becomes a low chip select (CSSPI1N) for the second SPI Serial Flash chip select. This pin is not used with LatticeECP2 devices when in SPIm mode.

The DOUT/CS0N drives out a high on power-up and will continue to do so until the execution of the Bypass/Flow Through instruction within the bitstream, or until the I/O Type is changed by the user code.

### **CSN and CS1N**

Both CSN and CS1N are active low input pins with weak pull-ups and are used in parallel mode only. These inputs are OR'ed and used to enable the D[0:7] data pins to receive or output a byte of data.

When CSN or CS1N is high, the D[0:7], INITN, and BUSY pins are tri-stated. CSN and CS1N are interchangeable when controlling the D[0:7], INITN, and BUSY pins. Driving both CSN and CS1N high causes the LatticeECP2/M to exit Bypass or Flow-Through mode and resets the Bypass register. If Bypass or Flow-Through mode will not be used then CSN or CS1N may be tied low, i.e. in this case it is only required that one of these pins be driven.

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 preference must be set to ON to preserve these pins as CSN and CS1N. Note that SRAM may only be read using JTAG or Slave Parallel mode.

### **WRITEN**

The WRITEN pin is an active low input with a weak pull-up and used for parallel mode only.

The WRITEN pin is used to determine the direction of the data pins D[0:7]. The WRITEN pin must be driven low in order to clock a byte of data into the device and driven high to clock data out of the device.

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 preference must be set to ON to preserve this pin as WRITEN. Note that SRAM may only be read using JTAG or Slave Parallel mode.

### **BUSY/SISPI**

The BUSY/SISPI pin has two functions.

In parallel configuration mode, the BUSY pin is a tri-stated output. The BUSY pin will be driven low by the device only when it is ready to receive a byte of data on D[0:7] or a byte of data is ready for reading. The BUSY pin allows the LatticeECP2/M to pause transfers on the parallel port.

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 preference must be set to ON to preserve this pin as BUSY. Note that SRAM may only be read using JTAG or Slave Parallel mode.

In SPI or SPI<sub>M</sub> configuration modes, the BUSY/SISPI pin becomes an output that drives control and data to the SPI Serial Flash. Control and data are output on the falling edge of CCLK.

### **D[0]/SPIFASTN**

The D[0]/SPIFASTN pin has two functions.

In parallel mode this pin is D[0] and operates in the same way as D[1:5] below. Taken together D[0:7] form the parallel data bus, D[0] is the most significant bit in the byte. As with D[1:5], 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 preference must be set to ON to preserve this pin as D[0]. Note that SRAM may only be read using JTAG or Slave Parallel mode.

In SPI or SPI<sub>M</sub> mode the D[0]/SPIFASTN pin becomes an input. SPIFASTN is sampled on the rising edge of INITN. If SPIFASTN is high the LatticeECP2/M will use SPI Serial Flash read op-code 03 (hex). Read op-code 03 (hex) is the standard read command used by all “25” series SPI Serial Flash. If SPIFASTN is low the LatticeECP2/M will use SPI Serial Flash fast read op-code 0B (hex). The fast read op-code 0B (hex) accommodates higher frequency read clocks, exact clock speeds can be found in the SPI Serial Flash manufacturer's data sheet.

Not all SPI Serial Flash support the 0B (hex) fast read op-code, consult the manufacturer's data sheet. Care must also be taken not to exceed the signal integrity capabilities of the PCB layout.

### D[1:5]

The D[1:5] pins support parallel mode only. The D[1:5] pins are tri-statable bi-directional I/O pins used for data write and read. When the WRITEN signal is low, and the CSN and CS1N pins are low, the D[1:5] pins become data inputs. When the WRITEN signal is driven high, and the CSN and CS1N pins are low, the D[1:5] pins become data outputs. If either CSN or CS1N is high D[1:5] will be tri-state.

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 preference must be set to ON to preserve this pin as D[1:5]. Note that SRAM may only be read using JTAG or Slave Parallel mode.

Care must be exercised during read back of EBR or PFU memory. It is up to the user to ensure that reading these RAMs will not cause data corruption; corruption may be caused when these RAMs are read while being accessed by user code.

### D[7]/SPID0 and D[6]/SPID1

The D[7]/SPID0 and D[6]/SPID1 pins each have two functions.

In parallel mode these pins are D[7] and D[6] and operate in the same way as D[1:5] above. Taken together D[0:7] form the parallel data bus, D[7] is the least significant bit in the byte. As with D[1:5], 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 preference must be set to ON to preserve this pin as D[7] or D[6]. Note that SRAM may only be read using JTAG or Slave Parallel mode.

In SPI or SPI<sub>M</sub> mode the D[7]/SPID0 pin becomes an input and should be wired to the output data pin of the SPI Serial Flash. The data on SPID0 is clocked in on the rising edge of CCLK.

For the LatticeECP2M only, the D[6]/SPID1 pin becomes an input and should be wired to the output data pin of the second SPI Serial flash when using the multi boot mode. The data on SPID1 is clocked on the rising edge of CCLK.

## ispJTAG Pins

The ispJTAG pins are standard IEEE 1149.1 TAP (Test Access Port) pins. The ispJTAG pins are dedicated pins and are always accessible when the LatticeECP2/M 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. While the LatticeECP2/M is under JTAG control the PROGRAMN pin will be ignored.

These pins are powered by V<sub>CCJ</sub>.

### 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<sub>CCJ</sub>.

### 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<sub>CCJ</sub>.

### 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<sub>CCJ</sub>.

**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 device data sheet. The TCK pin supports hysteresis; the value is shown in the DC parameter table of the data sheet. 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 LatticeECP2/M.

 **$V_{CCJ}$** 

Having a separate JTAG  $V_{CC}$  ( $V_{CCJ}$ ) pin lets the user 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).

JTAG voltage levels and thresholds are determined by the  $V_{CCJ}$  pin, allowing the LatticeECP2/M to accommodate JTAG chain voltages from 1.2V to 3.3V.

## Configuration Modes

The LatticeECP2/M devices support many different configuration modes, utilizing either serial or parallel data paths. On power-up, when a JTAG Refresh instruction is issued, or when the PROGRAMN pin is toggled, the CFG[2:0] pins are sampled to determine the configuration mode. See Table 15-3 above for a list of available configuration modes.

The following sub-sections break down each configuration mode. For more information on each mode's options, see the section below entitled Configuration Options.

### SPI Mode

The LatticeECP2/M offers a direct connection to memories that support the SPI Serial Flash standard (see Table 15-5). By setting the configuration pins, CFG[2:0], to all zeros the LatticeECP2/M will configure from the SPI interface. The SPI interface supports two configuration topologies:

- One FPGA configured from one SPI Serial Flash
- Multiple FPGAs configured from one SPI Serial Flash

The required boot memory size for each of the ECP2/M device sizes is shown in Table 15-4. The values shown are for a single LatticeECP2/M device. The size for a dual-boot application would be twice that shown.

**Table 15-4. Maximum Configuration Bits**

Density	Bitstream Size (Mb)	Required Boot Memory (Mb)
ECP2-6	1.6	2
ECP2-12	3	4
ECP2-20	4.7	8
ECP2-35	6.6	8
ECP2-50	9.4	16
ECP2-70	14	16
ECP2M35	10.3	16

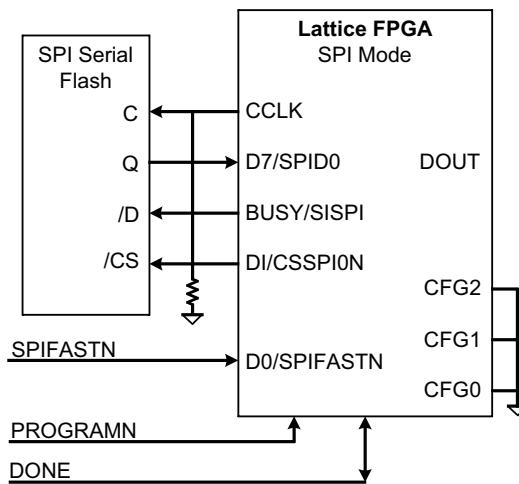
**Table 15-5. SPI Serial Flash Vendor List**

Vendor	Part Number
ST Microelectronics	M25Pxx
Winbond	W25Pxx
Silicon Storage Technology	SST25VFxx, SST25LFxx
Spansion	S25FLxx
Atmel	AT25Fxx
NexFlash	NX25Pxx
Macronix	MX25Lxx

Note: This is not meant to be an exhaustive list and may be updated from time to time.

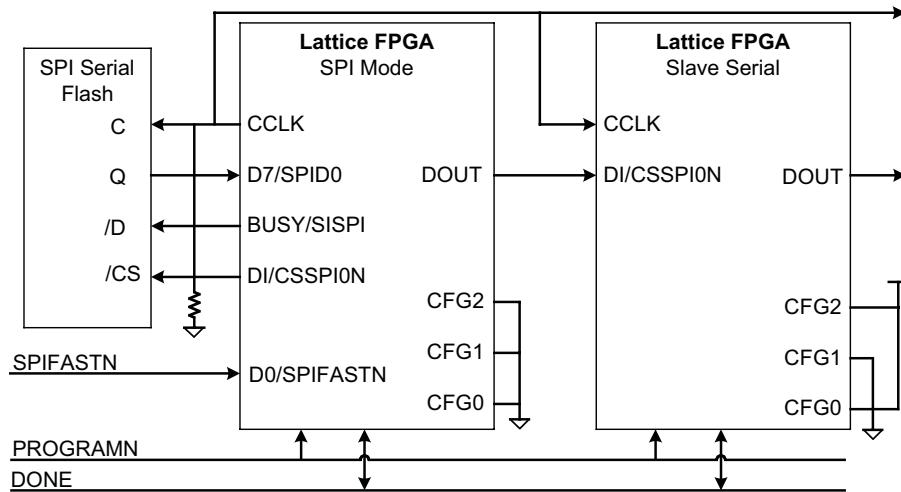
### One FPGA, One SPI Flash

The simplest SPI configuration consists of one SPI Serial Flash connected to one LatticeECP2/M, as shown in Figure 15-1.

**Figure 15-1. One FPGA, One SPI Serial Flash**

### Multiple FPGA, One SPI Flash

With a sufficiently large SPI Flash multiple FPGAs can be configured as shown in Figure 15-2. The first FPGA is configured in SPI mode; the following FPGAs are configured in Slave Serial mode.

**Figure 15-2. Multiple FPGAs, One SPI Serial Flash**

### SPIm Mode

Externally, except for the CFG pins, SPIm mode looks like SPI mode, they use the same SPI Serial Flash devices, and are wired to the FPGA in the same way (see Figure 15-1). The SPIm mode does not support multiple FPGAs being configured from one SPI serial Flash as shown in Figure 15-2. Internally the two modes are treated differently.

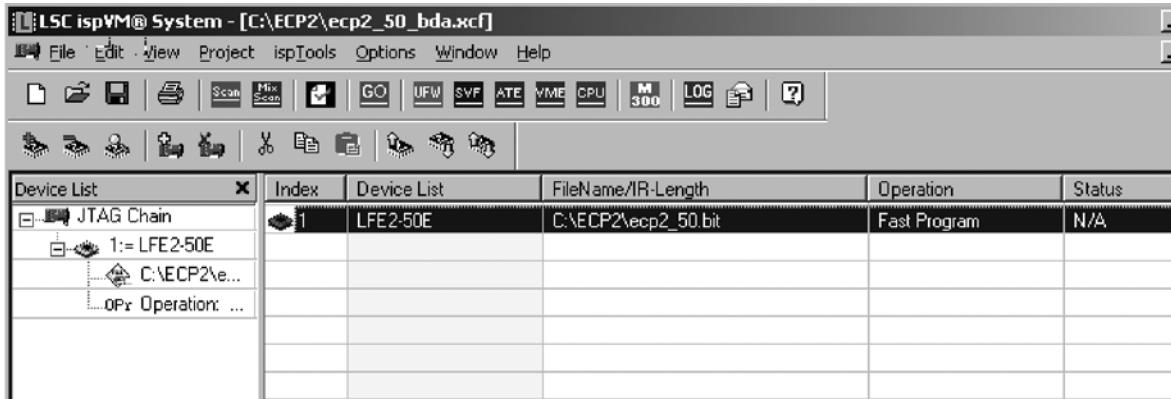
SPI mode treats the SPI Serial Flash as a single block of storage starting at address zero. The SPIm mode treats the SPI flash memory as discrete blocks of memory rather than a single block of memory. This allows the storage of separate configuration images in separate blocks of memory. SPIm supports dual configuration (or boot) images, here referred to as a “golden” image and a primary image.

A golden image is used when there is the possibility that corrupt data could be inadvertently loaded into the SPI Serial Flash, such as during remote updates. For instance, if the Flash erase or program procedure is interrupted, perhaps due to a power failure, then the Flash will contain corrupt data and the system could be rendered inoperable. Ideally the FPGA should detect that the data is corrupt and boot from a known good, or golden, boot image. This is exactly what SPIm does.

The golden image is stored at the beginning of the Flash address space; the updatable, or primary, image is above the golden image. During configuration, if the FPGA detects data corruption in the primary image, it will automatically reboot from the golden image. Each time the FPGA powers up, the PROGRAMN pin is toggled, or a JTAG Refresh instruction is issued, it will try to configure from the primary image first. Note that if the LatticeECP2/M detects that the data in the golden image is corrupt as well INITN will be driven low and the part will stop trying to configure.

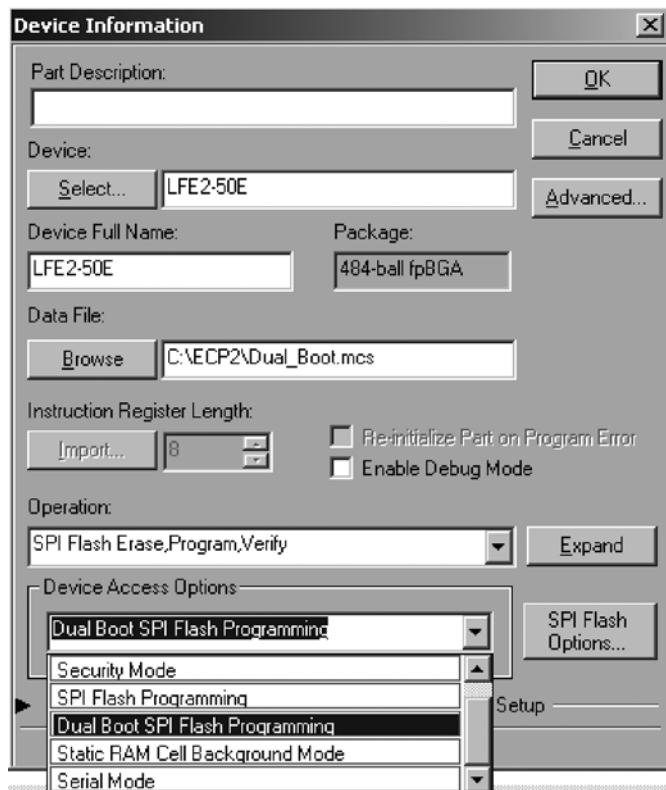
### Dual Boot Image Setup

In order to use dual configuration files the files must first be properly stored within the SPI Serial Flash. Lattice's ispVM® software makes this easy.

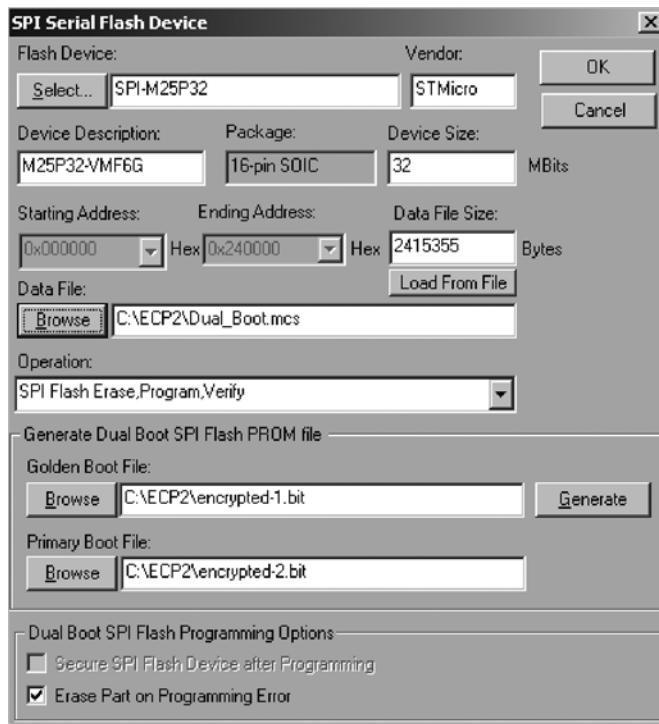
**Figure 15-3. ispVM Main Window**

First, create the desired files using ispLEVER. Depending on the application, these files might be the same design or different designs. There is nothing special about these files, in other words they contain all of the information needed to fully configure the FPGA.

Next, open ispVM (see Figure 15-3), do a scan of the board by clicking on the **SCAN** button on the toolbar, and then double click on the row in the chain that has the LatticeECP2/M. You will now see the Device Information window (see Figure 15-4).

**Figure 15-4. Device Information Window**

From the Device Options drop-down box select **Dual Boot SPI Flash Programming**. Then click on the **SPI Flash Options...** button to open the SPI Serial Flash Device window (see Figure 15-5). In this window click on **Select** and choose the SPI Flash that's mounted on the board.

**Figure 15-5. SPI Serial Flash Device Window**

Second, click on **Browse** to select an output file name.

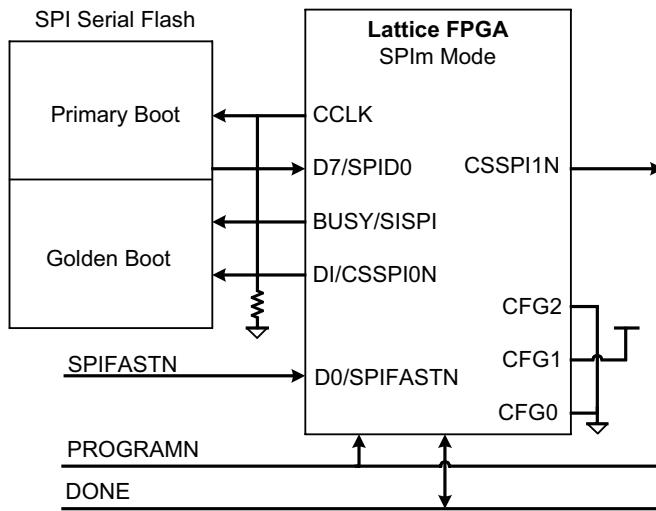
Third, select the operation, such as **SPI Flash Erase, Program, Verify**.

Fourth, select the golden file and primary file. Note that these may be encrypted or non-encrypted files. If they are encrypted files they must both be encrypted with the same key.

And finally, click on the **Generate** button. When file generation is complete, click on **OK** to get back to the main ispVM window. To program the file into the SPI Serial Flash just click on the **GO** button on the toolbar.

At power-up, when the PROGRAMN pin is toggled, or when a JTAG Refresh instruction is issued, the LatticeECP2/M reads the primary file from SPI Serial Flash. If a CRC error is found then the LatticeECP2/M will reboot automatically, reading configuration data from the golden file instead of the primary file. Note that if an error is found in the golden file the LatticeECP2/M will drive the INITN pin low and stop trying to configure.

*Note that the flow specified here is for initial programming of the SPI Serial Flash. During a field update of the SPI Serial Flash it is expected that only the primary configuration file will be updated, not the golden file. This is important because it guarantees the availability of a known good configuration file.*

**Figure 15-6. SPIm, Dual Configuration Images**

Note: The CSSPI1N pin should not be connected when using LatticeECP2. This pin is used for LatticeECP2M when using 2 SPI serial flash chips in SPIm mode

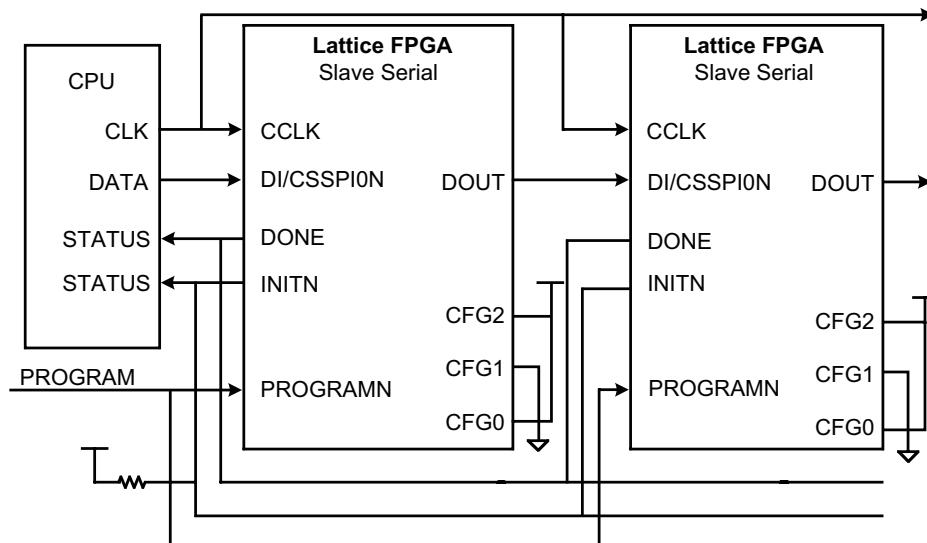
## Programming SPI Serial Flash

The LatticeECP2/M contains dedicated hardware that allows JTAG to access the SPI port, allowing ispVM, embedded hardware, or ATE equipment to program the Flash while it is on the board. In order to program SPI Serial Flash using JTAG, the CFG pins must be set to SPI or SPIm (see Table 15-3). Please refer to ispVM's help facility for more information.

## Slave Serial Mode

In Slave Serial mode the CCLK pin becomes an input, receiving the clock from an external device. The LatticeECP2/M accepts data on the DI pin on the rising edge of CCLK. Slave Serial only supports writes to the FPGA, it does not support reading from the Flash.

After the device is fully configured, if the Bypass option has been set, any additional data clocked into DI will be presented to the next device via the DOUT pin, as shown in Figure 15-7.

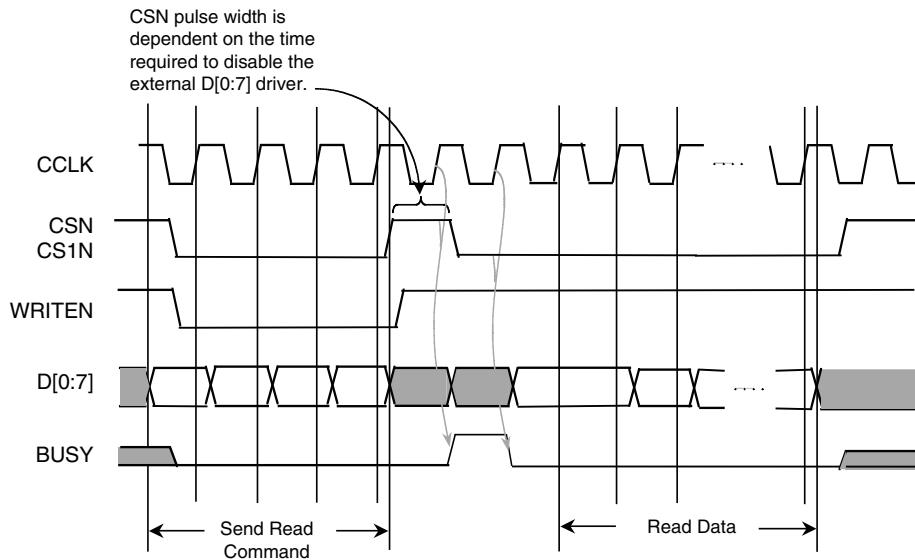
**Figure 15-7. Serial Mode Daisy Chain**

## Slave Parallel Mode

In Slave Parallel mode a host system sends the configuration data in a byte-wide stream to the LatticeECP2/M. The CCLK, CSN, CS1N, and WRITEN pins are driven by the host system. WRITEN, CSN, and CS1N must be held low to write to the device; data is input from D[0:7]. D0 is the MSb and D7 is the LSb.

Slave Parallel mode can also be used for readback of the internal configuration. By driving the WRITEN pin low, and CSN and CS1N low, the device will input the readback instructions on the D[0:7] pins; WRITEN is then driven high and read data is output on D[0:7] (see Figure 15-8). In order to support readback the PERSISTENT bit in ispLEVER's Design Planner must be set to ON.

**Figure 15-8. Parallel Port Read Timing Diagram**



The host sends the preamble, BDB3 (hex), and then sends the read command. The LatticeECP2/M sends read data on D[0:7], driving BUSY high as needed to pause data flow. For an example bitstream see Table 15-6. Table 15-7 lists the various read commands. Note that the sample bitstream and the list of read commands are for reference only; to help the user better understand the flow. The actual bitstream, containing the read commands, is created by ispLEVER and ispVM, the host toggles the control signals and sends the bitstream.

**Table 15-6. Parallel Port Read Bitstream Example**

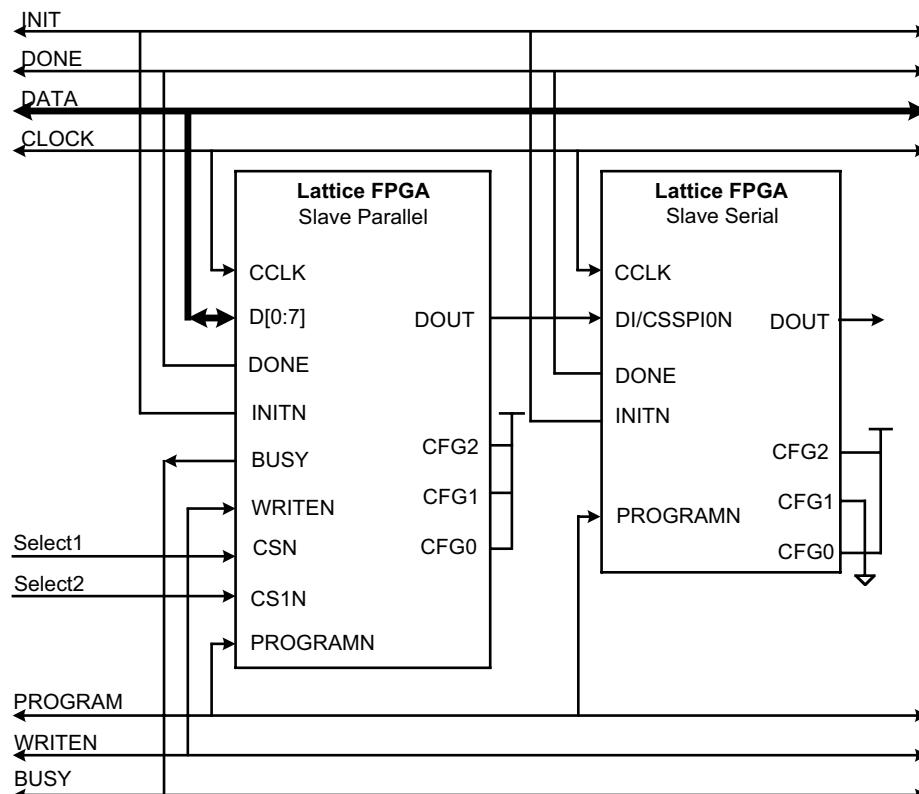
Frame	Contents	Description
Header	1111...1111	2 Dummy Bytes
	10111101 10110011	2-byte Preamble (BDB3)
Verify ID		8 bytes of command and data
Reset Address		4 bytes of command and data
Read Increment		4 bytes of command and data

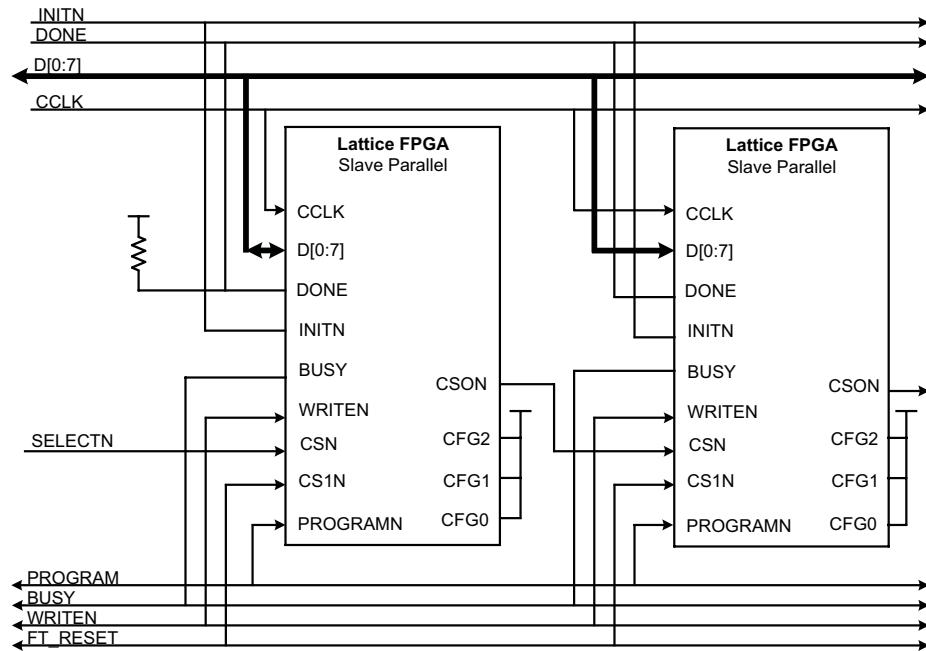
**Table 15-7. Parallel Port Read Commands**

Command	32-bit Opcode	Function
Reset Address	62xxxxxx	Reset address register to point to the first data frame
Read Increment	01vvvvv	Read back the configuration memory frame selected by the address register and post increment the address
Read Usercode	03xxxxxx	Read the content of the USERCODE register
Read Ctrl Reg 0	04xxxxxx	Read the content of Control Register 0
Read CRC	06xxxxxx	Read CRC register content
Read ID Code	07xxxxxx	Read ID code
NO OP	FFxxxxxx	No operation

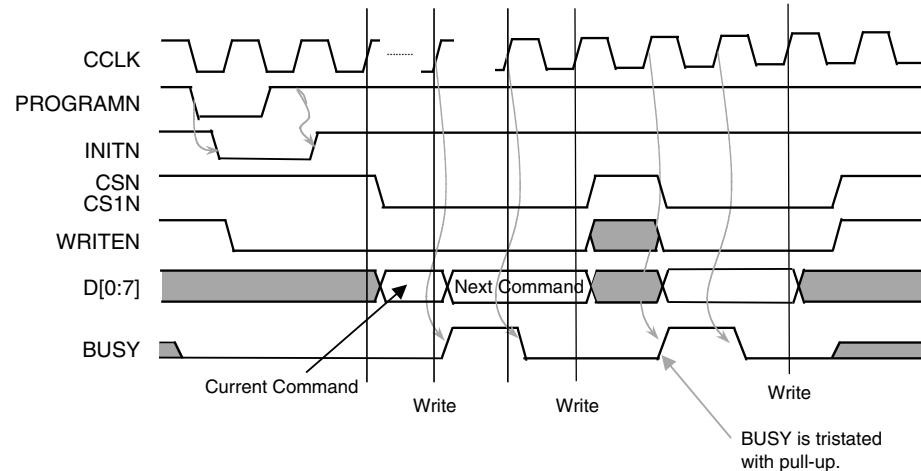
Note: x = don't care, v = variable.

Slave Parallel mode can support two types of overflow, Bypass and Flow-Through. After the first device has received all of its configuration data, and the Bypass command is detected in the bitstream, the data presented to the D[0:7] pins will be serialized and bypassed to the DOUT pin (see Figure 15-9). If the Flow-Through command is detected in the bitstream, instead of the bypass command, the CS0N signal will drive the following parallel mode device's chip select low as shown in Figure 15-10. If either type of overflow is active, driving both the CSN and CS1N pins high will reset overflow, i.e. take the device out of overflow.

**Figure 15-9. Slave Parallel with Bypass Option**

**Figure 15-10. Slave Parallel with Flow-Through**

To support asynchronous configuration, where the host may provide data faster than the FPGA can accept it, Slave Parallel mode can use the BUSY signal. By driving the BUSY signal high the Slave Parallel device tells the host to pause sending data. Please note that all data and control are still synchronous with CCLK, asynchronous refers to the ability to throttle the data transfer using BUSY. See Figure 15-11.

**Figure 15-11. Parallel Port Write Timing Diagram**

The CSN or CS1N signal can be used to temporarily stop the write process by setting either signal to a high state. The LatticeECP2/M will resume configuration when both CSN and CS1N are set low again (see Figure 15-11).

### ispJTAG Mode

The LatticeECP2/M device can be configured through the ispJTAG port using either Fast Program or IEEE 1532 mode. The JTAG port is always on and available, regardless of the configuration mode selected. The IEEE 1532 mode is called Erase, Program, Verify in ispVM System.

### Fast Program

Fast Program can be thought of as serial configuration using the JTAG port. The data file used for Fast Program is the same as a file used for a sysCONFIG Serial mode configuration, in other words there is a header, a preamble, and configuration data. Fast Program will result in a faster configuration time since the bitstream contains CRC checking so a time consuming post programming bit by bit verification is not required. The SVF and VME file formats also support Fast programming.

During JTAG configuration the Boundary Scan cells take control of the LatticeECP2/M I/Os. The Boundary Scan cells will usually drive the I/Os to a tri-state level but this can be controlled and even customized using ispVM. Note that PROGRAMN, INITN, and DONE have no meaning since they are controlled by the Boundary Scan cells during JTAG configuration. However, the INITN and DONE do indicate configuration status after JTAG configuration is completed.

### IEEE 1532

Besides Fast Program the LatticeECP2/M can also be configured through JTAG using the IEEE 1532 Standard. The IEEE 1532 mode is called Erase, Program, Verify in ispVM System. IEEE 1532 configuration files contain JTAG instructions, as well as the configuration data. IEEE 1532 files, including ISC, SVF, and VME, can be created using ispVM's Universal File Writer (UFW). These files can be used by ispVM or by third party ATE equipment. These files can also be used in embedded situations, where an on-board processor provides the data while controlling the JTAG signals (this is called ispVM Embedded, more information can be found in ispVM's help facility). IEEE 1532 programming will be slower than the Fast Program mode since it requires a post programming bit by bit verification.

During JTAG configuration the Boundary Scan cells take control of the LatticeECP2/M I/Os. The Boundary Scan cells will usually drive the I/Os to a tri-state level but this can be controlled and even customized using ispVM.

Note that PROGRAMN, INITN, and DONE have no meaning since they are controlled by the Boundary Scan cells during JTAG configuration. However, the DONE pin will indicate configuration status after IEEE 1532 configuration is completed.

### Transparent Read Back

The ispJTAG transparent read back mode allows the user to read the content of the device while the device remains fully functional. All I/O, as well as the non-JTAG configuration pins, remain under internal logic control during a Transparent Read Back. The device enters the Transparent Read Back mode through a JTAG instruction. The user must ensure that Transparent Read Back does not access EBR or distributed RAM at the same time internal logic is accessing these resources or corruption of the RAM may occur.

### Boundary Scan and BSDL Files

The LatticeECP2/M BSDL files can be found on the Lattice Semiconductor web site. 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.

### Configuration Options

Several configuration options are available for each configuration mode.

- When daisy chaining multiple FPGA devices an overflow option is provided for serial and parallel configuration modes
  - When using SPI or SPI<sub>M</sub> mode, the master clock frequency can be set
  - A security bit can be set to prevent SRAM readback
  - The bitstream can be compressed
  - 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.

### Bypass Option

The Bypass option can be set by using ispLEVER's Bitgen properties, or a chain of bitstreams can be assembled and Bypass set using ispVM. The Bypass option can be used in parallel and serial mode daisy chains. The Bypass option is not supported when using SPIm configuration.

When the first device completes configuration, and a Bypass command is input from the bitstream, any additional data coming into the FPGA configuration port will overflow serially on DOUT. This data is applied to the DI pin of the next device (downstream devices must be set to slave serial mode).

In serial configuration mode the Bypass option connects DI to DOUT via a bypass register. The bypass register is initialized with a '1' at the beginning of configuration and will stay at that value until the Bypass command is executed. In parallel configuration mode the Bypass option causes the excess data coming in on D[0:7] to be serially shifted to DOUT. The serialized data is shifted to DOUT through a bypass register. D0 will be shifted out first followed by D1, D2, and so on. Once the Bypass option starts the device will remain in Bypass until the Wake-up sequence completes. In parallel mode, if Bypass needs to be aborted, drive both CSN and CS1N high, this acts as a Bypass reset signal.

### Flow-Through Option

As with Bypass, Flow-Through can be found in the software's Bitgen properties. The Flow-Through option can be used with parallel daisy chains only. The Flow-Through option is not supported when using SPIm configuration.

When the first device completes configuration, and a Flow-Through command is input from the bitstream, the CSON pin is driven low. In addition to driving CSON low, Flow-Through also tri-states the device's D[0:7] and BUSY pins in order to avoid contention with the other daisy chained devices. Once the Flow-Through option starts the device will remain in Flow-Through until the Wake-up sequence completes. If Flow-Through needs to be aborted drive both CSN and CS1N high, this acts as a Flow-Through reset signal.

### Master Clock

If the CFG pins indicate an SPI or SPIm mode the CCLK pin will become an output, with the frequency set by the user. The default Master Clock Frequency is 3.1 MHz.

The user can change the Master Clock frequency by setting the MCCLK\_FREQ global preference in the Lattice ispLEVER Design Planner. One of the first things loaded during configuration is the MCCLK\_FREQ parameter; once this parameter is loaded the frequency changes to the selected value using a glitchless switch. 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 15-4 above, by the Master Clock frequency.

### 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 fuse is updated as the last operation of SRAM configuration. If a secured device is read it will output all zeros.

The CONFIG\_SECURE option is accessed via the Design Planner in ispLEVER, the default is OFF.

### Compress Bitstream

Setting the global COMPRESS\_CONFIG option to ON in ispLEVER's Design Planner will cause the software to generate a compressed bitstream. The LatticeECP2/M will automatically decompress the bitstream as it comes into the device. The actual amount of compression varies according to the data pattern in the uncompressed bit-

stream. Though unlikely, it is theoretically possible for the compressed bitstream to be larger than the uncompressed bitstream.

Compressing the bitstream can result in faster configuration. The default setting is OFF.

### Persistent Option

The PERSISTENT Option is set using ispLEVER's Design Planner (default is OFF). PERSISTENT serves two purposes.

Setting PERSISTENT ON tells the software's place and route tools that it may not use any of the sysCONFIG pins associated with the parallel port (all of the dual-purpose pins except DI).

Setting PERSISTENT ON also sets a hardware fuse. So, not only are the pins reserved in software, they are also reserved in hardware.

PERSISTENT is set to ON only when the user wants to be able to read the SRAM configuration memory using the Slave Parallel port. In order to perform a read using the parallel port the user must first send a read command, setting PERSISTENT ON allows the parallel port to listen for this command while in user mode (the DONE pin is high). If the design does not require this function, the PERSISTENT option should be set to OFF.

### Configuration Mode

Just as the CFG pins tell the hardware which port to configure from, the CONFIG\_MODE option tells the software which port will be used. CONFIG\_MODE allows the user to protect dual-purpose sysCONFIG pins. For example setting CONFIG\_MODE to SPI will keep the Place and Route tools from using the SPI pins as general purpose I/O. The user, however, is still free to assign these pins as GPIO, but a warning will be generated as a reminder that there are certain precautions (see the section above entitled Configuration Pins).

Available options are None, JTAG, SPI, SPIIm, Slave Serial, and Slave Parallel. The default is Slave Serial.

### DONE OD, DONE EX

During configuration the DONE pin is low. Once configuration is complete, indicated by the setting of the internal Done bit, the device wake-up sequence takes place and then the DONE pin goes high. Under most circumstances this flow is exactly what is needed, however, if there are several devices in one configuration chain, delay of the wake-up sequence may be desirable in order to "synchronize" the wake-up of all devices in the chain. There are two options in the Design Planner that allow for this synchronization.

DONE OD defaults to ON, DONE OD ON forces the DONE pin type to be open- drain. When connecting multiple DONE pins together all of the pins should be open- drain, however it may be advantageous to have an actively driven DONE pin on the last device. Setting DONE OD to OFF makes the DONE pin an actively driven pin, rather than open-drain.

DONE EX defaults to OFF. Setting DONE EX to ON will cause LatticeECP2/M to sample the DONE pin and, if the DONE pin is held low externally, delay the wake-up sequence. Setting DONE EX to OFF will cause the device to wake-up as soon as the internal Done bit is set.

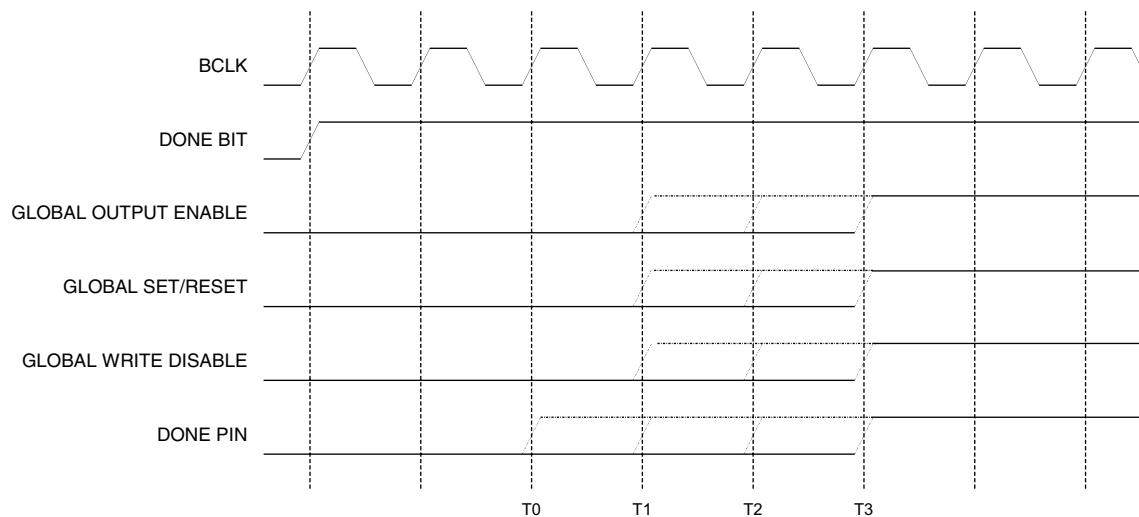
### Device Wake-Up

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 provides a list of the Wake-Up sequences supported by the LatticeECP2/M; Figure 15-12 shows the Wake-Up timing. The Wake-Up defaults work fine for the vast majority of applications.

**Table 15-8. 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	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
18		GOE, GSR	DONE	GWDIS
19			GOE, GWDIS, GSR	DONE
20		GOE, GWDIS, GSR		DONE
21 (Default)		GOE	GWDIS, GSR	DONE
22		GOE, GWDIS	GSR	DONE
23		GWDIS	GOE, GSR	DONE
24		GWDIS, GSR	GOE	DONE
25		GOE, GSR	GWDIS	DONE

**Figure 15-12. Wake-Up Timing Diagram**

## Synchronizing Wake-Up

The internal Wake-Up sequence clock source can be chosen as well as how the device wakes up relative to other devices.

### Wake-Up Clock Selection

The Wake-Up sequence is synchronized to a clock source that is user selectable. The clock sources are External (default) and User Clock.

When External is selected the LatticeECP2/M will use one of two clocks during the Wake-Up sequence, depending on the configuration data source. If the LatticeECP2/M is being configured from JTAG then JTAG's TCK will be used for the Wake-Up sequence, if configuration data is coming from a sysCONFIG port (serial, parallel, or SPI) then CCLK will be used.

When User is selected, any of the design's clock signals can be used as the clock source.

### Synchronous to Internal Done Bit

If the LatticeECP2/M 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 the configuration chain. If DONE\_EX (see the DONE OD, DONE EX section above) is ON then the DONE pin is a bi-directional pin. If an external device drives the DONE pin low then the Wake-Up sequence will be delayed; configuration can complete but Wake-Up is delayed. 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 15-2), 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 OD to OFF and DONE\_EX to OFF on the last device while setting DONE OD to ON and DONE\_EX to ON for the other devices.

### Wake-Up Sequence Options

The Wake-Up sequence options shown in Table 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.
- When high, the GOE signal prevents the device's I/O buffers from driving the pins.
- When high, the DONE pin indicates that configuration is complete and that no errors were detected.

If DONE\_EX (see DONE OD, DONE EX above) is OFF then sequence 21 is the default, but the user can select any sequence from 8 to 25; if DONE\_EX is ON the default sequence is 4, but the user can select any sequence from 1 to 7.

## Configuration FAQs

Here are some of the more common questions regarding device configuration.

### General

- **Q. Other than JTAG, what is the least expensive method of configuration?**

**A.** If you already have a processor and extra storage on the board you can use the processor to feed configuration data to the LatticeECP2/M. The least expensive stand-alone configuration option is SPI Serial Flash.

---

**• Q. I have created my bitstream, now how do I load the bitstream into the LatticeECP2/M?**

A. Use the free Lattice ispVM tool (from [www.latticesemi.com](http://www.latticesemi.com)), and a Lattice ispDOWNLOAD® cable.

**• Q. I can't read the LatticeECP2/M device ID using JTAG. What could be wrong?**

A. This is the most basic of JTAG operations. If you are having trouble reading the device ID then something basic is wrong. Check that the JTAG connections are correct, and that  $V_{CCJ}$  and the download cable  $V_{CC}$  are correct (and the same). Make sure that the XRES pin is connected to ground through a 10K resistor. Check that all LatticeECP2/M  $V_{CC}$  and ground pins are properly connected. Check for noise on the JTAG signals. Sometimes touching a properly grounded 'scope probe to TCK will change the symptoms; if so, you have signal noise issues. Check for excessive noise on the  $V_{CC}$  pins.

**• Q. Is there anything I can do during board design that will make debugging easier?**

A. Bring the dedicated configurations pins, PROGRAMN, INITN, DONE, and CCLK out to accessible test points.

**• Q. Do I need external pull-up resistors on PROGRAMN, INITN, or DONE?**

A. All of these signals have internal weak pull-ups, however if you have a noisy environment, or have several devices connected to these pins, I recommend adding a 10K pull-up to the signal.

**• Q. How can I get assistance with configuration issues?**

A. Use the On-line Assistant feature in ispVM. You will find it under the Help menu.

## Mode Specific

### SPI/SPI<sub>m</sub>

**• Q. How do I program the SPI Serial Flash once it's on the board?**

A. Connect the SPI Serial Flash to the LatticeECP2/M as shown in this document, then use ispVM, and a Lattice ispDOWNLOAD cable connected to the JTAG port, to program the bitstream into the Flash.

**• Q. Are there any special requirements for wiring the SPI Flash to the LatticeECP2/M?**

A. Other than connecting the Flash to the right pins the only other suggestion is to add a 4.7K pull-down resistor between CCLK and ground. This keeps CCLK quite during  $V_{CC}$  ramp-up.

**• Q. Can I use 2.5V to power the SPI Flash?**

A. Today all SPI Serial Flash of the "25" type are 3.3V, so the Flash, and  $V_{CCIO8}$ , must be connected to 3.3V.

**• Q. Can I use something other than a "25" type SPI Serial Flash?**

A. Only devices that recognize a read op-code of 03h may be used with the LatticeECP2/M. Please refer to Table 15-5 for a list of vendors.

**• Q. My design is small, can I use a smaller-than-recommended SPI Flash?**

A. The state of all of the device fuses is contained in the bitstream, whether they are part of the design or not. The size of the design does not affect the size of the bitstream.

### Serial

**• Q. Can I use a free running clock for Slave Serial mode?**

A. The LatticeECP2/M clocks data in on every rising edge of CCLK so there should only be one rising clock edge for each data bit.

**• Q. Is the bitstream for the serial modes different from the bitstream for other modes?**

A. All sysCONFIG bitstreams are the same, they can be different file types, such as hex or binary, but the data is the same.

**Parallel**

- Q. My processor is generating all of the proper control signals but the LatticeECP2/M won't configure, and INITN goes high and stays high while DONE stays low. What's wrong?  
A. D0 is the MSb and D7 is the LSb. Try reversing the bit order for each byte in the bitstream. You can do this using your processor or you can generate a bit mirrored file using ispVM. Lattice recommends using your processor so that you don't have to remember to bit mirror the file.

**Technical Support Assistance**

Hotline: 1-800-LATTICE (North America)  
+1-503-268-8001 (Outside North America)  
e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
Internet: [www.latticesemi.com](http://www.latticesemi.com)

**Revision History**

Date	Version	Change Summary
February 2006	01.0	Initial release.
February 2006	01.1	Added ECP2-6 to Maximum Configuration Bits table and changed ECP2-22 to ECP2-20.
April 2006	01.2	Updated section on SPIm. Also added screen shots to SPIm section.
September 2006	01.3	Updated Table "Maximum Configuration Bits" with LatticeECP2M data.

## Introduction

All Lattice FPGAs provide configuration data read security, meaning that a fuse can be set so that when the device is read all zeros will be output instead of the actual configuration data. This kind of protection is common in the industry and provides very good security if the configuration data storage is on-chip, such as with the LatticeXP™ and MachXO™ device families. However, if the configuration bitstream comes from an external boot device it is quite easy to read the configuration data, allowing access to the FPGA design.

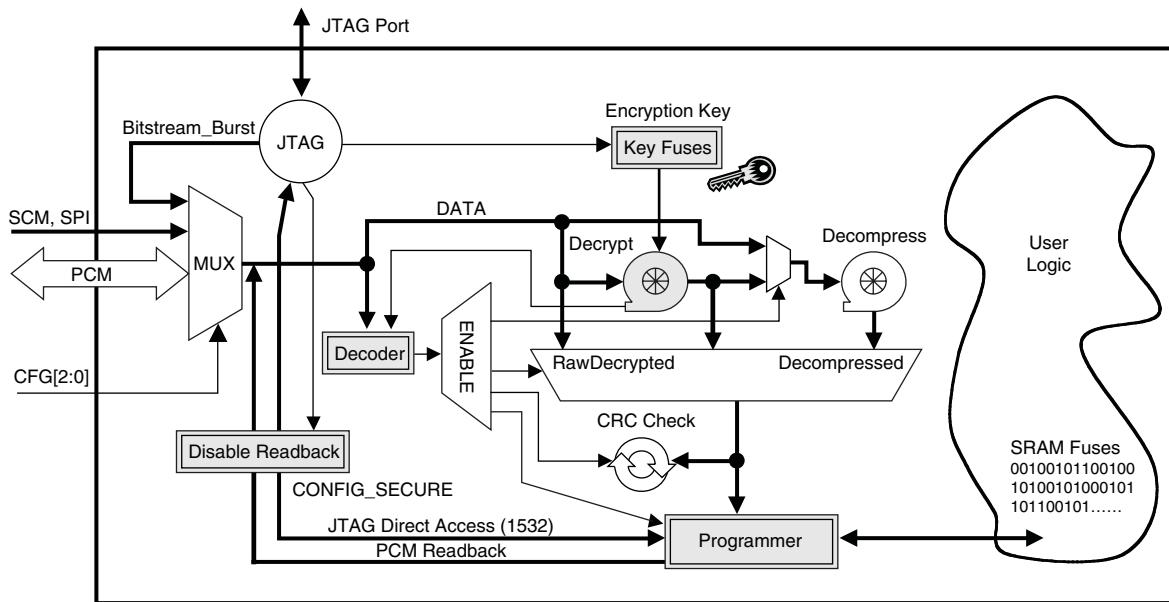
For this reason the latest FPGAs from Lattice, the LatticeECP2™ and LatticeECP2M™, offer the 128-bit Advanced Encryption Standard (AES) to protect the bitstream. The user selects and has total control over the 128-bit key and no special voltages are required to maintain the key within the FPGA.

This document explains the capabilities of this new security feature and how to take advantage of it.

## General Configuration Process

Figure 16-1 is a block diagram describing the LatticeECP2/M encryption data paths. Refer to this figure as you read the following sections.

**Figure 16-1. LatticeECP2/M Encryption Block Diagram**



Lattice FPGAs are configured by using the sysCONFIG™ interface or the JTAG interface (see Table 16-1).

**Table 16-1. Configuration Ports**

Interface	Port
sysCONFIG	SPI
	SPI <sup>M</sup>
	Slave Serial (SCM)
	Slave Parallel (PCM)
ispJTAG™	IEEE 1532 (Erase, Program, Verify) <sup>1</sup>
	Bitstream Burst (Fast Program)

1. Does not support encrypted bitstreams

The sysCONFIG interface allows the user to input data serially, using serial configuration mode (SCM) or SPI Serial Flash, or in parallel, using the parallel configuration mode (PCM). In general, the connection between the FPGA and the configuration device consists of a clock, chip select(s), a write signal (in PCM), and data. During configuration all data written to the FPGA is ignored until a special preamble is detected in the bitstream. Everything after the preamble is configuration data. The normal preamble is BDB3 (hex), however encrypted bitstreams contain a different preamble.

The JTAG port, which conforms to IEEE 1149.1 and IEEE 1532 standards, can input data in Bitstream-Burst mode (Fast Program) or 1532 mode (Erase, Program, Verify). Configuration bitstreams created for Bitstream-Burst mode (Fast Program) are identical to the configuration bitstreams created for sysCONFIG mode. The bitstream contains a header, a preamble, configuration data, and frame data CRC. However, 1532 mode makes use of standard JTAG instructions to configure the device. In other words, the configuration data file contains configuration data only. Because 1532 mode data files do not contain a preamble, they cannot be used to input encrypted configuration files.

In addition to being a configuration interface, JTAG also allows the user to program the 128-bit encryption key. In fact, JTAG is the only way to program the key.

For detailed information on LatticeECP2/M configuration, refer to Lattice technical note TN1108, *LatticeECP2/M sysCONFIG Usage Guide*, available on the Lattice web site at [www.latticesemi.com](http://www.latticesemi.com).

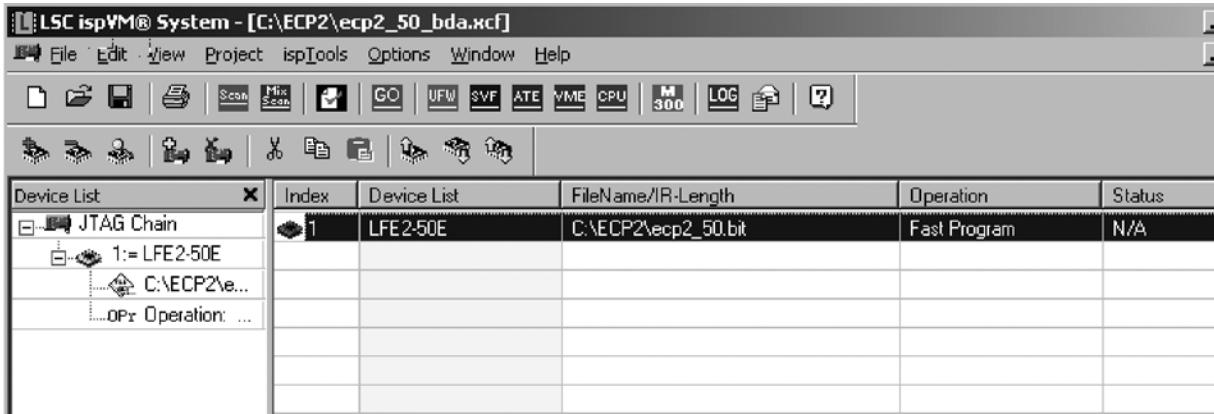
## Encryption/Decryption Flow

The LatticeECP2/M supports both encrypted and non-encrypted bitstreams. Since the non-encrypted flow is covered in technical note TN1108, this document will concentrate on the additional steps needed for the encrypted flow. 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 LatticeECP2/M.

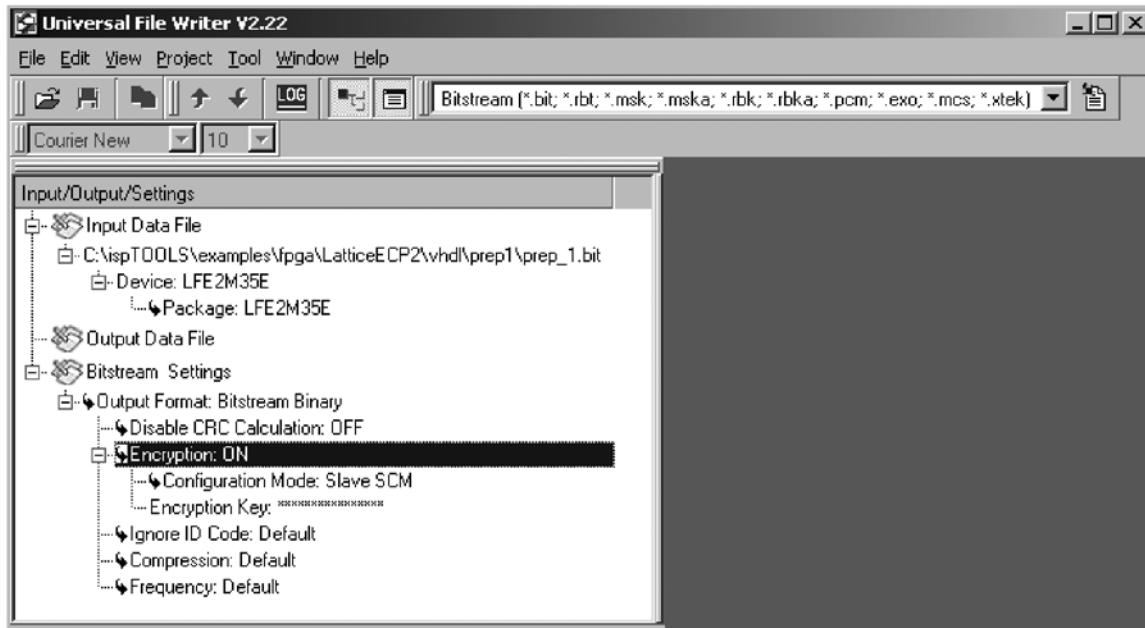
### Encrypting the Bitstream

As with any other Lattice FPGA design flow, the engineer must first create the design using the ispLEVER® tool suite. 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.

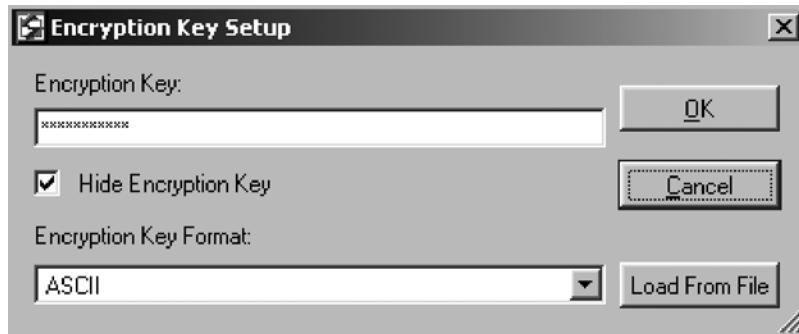
The bitstream can be encrypted using ispLEVER by going to the Tools pull-down menu and selecting Security features or by using the Universal File Writer (UFW), which is part of the Lattice ispVM® System tool suite. The file is encrypted using ispVM as follows.

**Figure 16-2. ispVM Main Window**

1. Start ispVM. You can start ispVM from within ispLEVER or from the **Start -> Programs** menu in Windows. You should see a window that looks similar to Figure 16-2. Click on the **UFW** button on the toolbar. You will see a window similar to Figure 16-3.

**Figure 16-3. Universal File Writer**

2. Double click on **Input Data File** and browse to the non-encrypted bitstream created using ispLEVER. 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 16-4.

**Figure 16-4. Encryption Key Dialog Window**

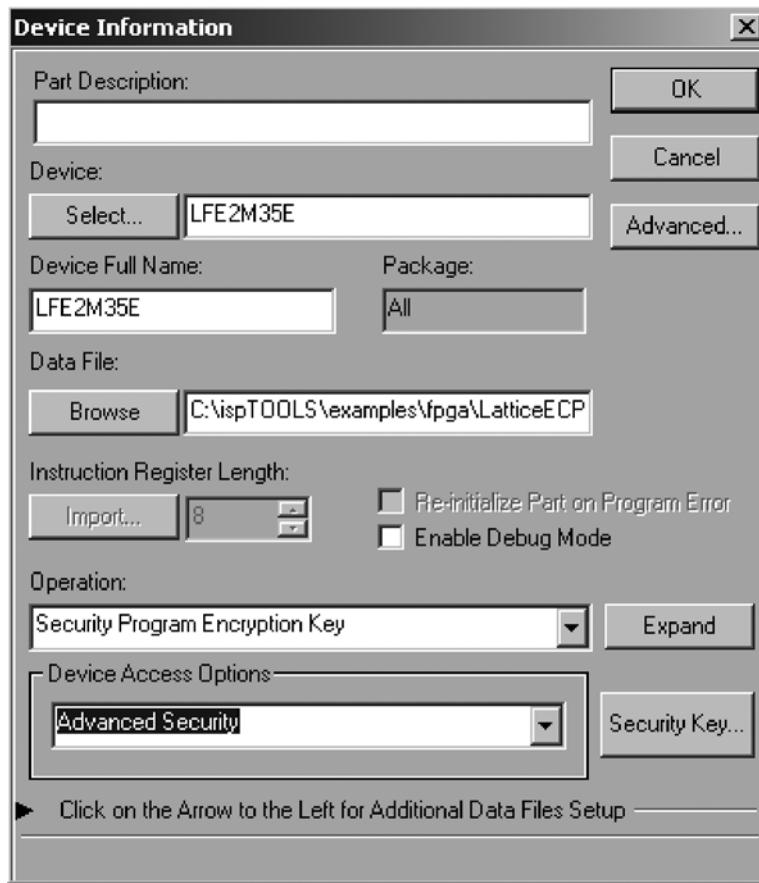
3. Enter the desired 128-bit key. The key can be entered in Hexadecimal or ASCII. Hex supports 0 through f and is not case sensitive. ASCII supports all alphanumeric characters, as well as spaces, and is case sensitive. Note: be sure to remember this key. Lattice cannot recover an encrypted file if the 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® Cable, a third-party programmer, or any other method normally used to program a non-encrypted bitstream. However, before the LatticeECP2/M can configure from the encrypted file the 128-bit key used to encrypt the file must be programmed into the one-time programmable fuses on the FPGA.

### Programming the 128-bit Key

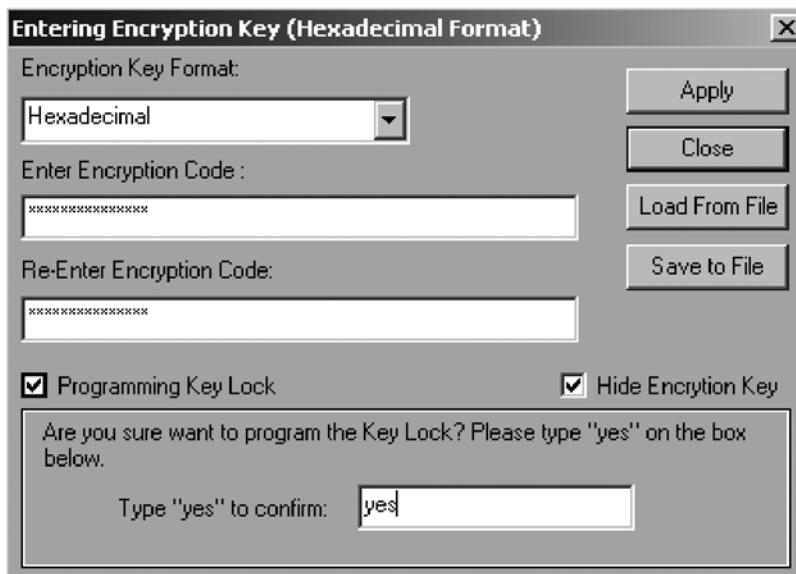
The next step is to program the 128-bit encryption key into the one-time programmable fuses on the LatticeECP2/M. Note that this step is separated from file encryption to allow flexibility in the manufacturing flow. For instance, the board manufacturer might program the encrypted file into the SPI Serial Flash, but the key might be programmed at the user's facility. This flow adds to design security and it allows the user 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 LatticeECP2/M 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 LatticeECP2/M, proceed as follows.

1. Attach a Lattice ispDOWNLOAD cable from a PC to the JTAG connector wired to the LatticeECP2/M (note that the 128-bit key can only be programmed into the LatticeECP2/M using the JTAG port). Apply power to the board.
2. Start the ispVM System software. ispVM can be started from within the ispLEVER design tool or from the **Start -> Programs** menu in Windows. You should see a window that looks similar to Figure 16-2. 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 16-2 has only one device, the LatticeECP2.

**Figure 16-5. Device Information Window**

3. Double-click on the line in the chain containing the LatticeECP2. This will open the Device Information window (see Figure 5). 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 16-6.

**Figure 16-6. 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 LatticeECP2/M will only configure from an encrypted file whose encryption key matches the one loaded into the FPGA's one-time programmable fuses. *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 the user selects. 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 or the PROGRAMN pin is toggled, 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 16-2) click on the green **GO** button on the toolbar to program the key into the LatticeECP2/M one-time programmable fuses. When complete, the LatticeECP2/M 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 zeros. 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 CRC on the data. 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 (for more information on this type of error, refer to Lattice technical note TN1108, *LatticeECP2/M sysCONFIG Usage Guide*).

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 16-2 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 16-2. 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
End	1111...1111	32-bit Terminator (all ones)

Note: The data in this table is intended for reference only.

Table 16-3 shows a bitstream that is built for encryption but has not yet been encrypted. The highlighted areas will be encrypted. The changes between Table 16-2 and Table 16-3 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 the user a method to identify an encrypted file. For example, the usercode could be used as a file index or a "hint". 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 the user 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 ispLEVER's help facility.
- A copy of CONFIG\_MODE, one of the global ispLEVER preferences, is placed in the non-encrypted comment string. CONFIG\_MODE can be SPI/SPIIm, Slave SCM, or Slave PCM.

Note that if the global COMPRESS\_CONFIG option is turned ON using ispLEVER's Design Planner or UFW, data compression will be performed before encryption.

**Table 16-3. 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
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 16-4, 16-5, and 16-6).

- 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 16-3. 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 SPI<sub>M</sub>) 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 16-4, 16-5, and 16-6). 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. On average it will take about 50% longer to configure from an encrypted data file, meaning that slave mode encrypted file sizes will be about 50% larger than the non-encrypted version.

**Table 16-4. Encrypted File Format for a Master Mode**

Frame	Contents	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator.
Header	1111...1111	16 Dummy bits.
		16-bit Encryption Preamble.
30,000 Filler Bits		This allows time for the device to load and hash the 128-bit encryption key.
Alignment Preamble		16-bit Alignment Preamble.
	1	1-bit Dummy Data.
Data		There are no dummy filler bits when the bitstream is generated for master programming modes. 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 - Encrypted.
Program Done		32-bit Program Done Command - Encrypted.
End	1111...1111	32-bit Terminator (all ones) - Encrypted.
Filler Bits		Filler to meet the bound requirement.
Dummy Data	1111...1111	200 bits of Dummy Data (all ones). Provides a delay to turn off the decryption engine.

Note: The data in this table is intended for reference only. The shaded area is encrypted data.

**Table 16-5. Encrypted File Format for a Slave Serial Mode**

Frame	Contents	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator.
Header	1111...1111	2 Dummy Bytes.
		16-bit Encryption Preamble
30,000 Filler Bits		This allows time for the device to load and hash the 128-bit encryption key.
Alignment Preamble		16-bit Alignment Preamble.
	1	1-bit Dummy Data.
Data		128 bits of Configuration Data.
		64 bits of all ones data. Provides a delay for the decryption engine to decrypt the 128 bits of data just received. If the peripheral device can provide the needed 64 clocks while pausing data, then the 64 bits of dummy data are not required, saving file size.
	...	Last 128 bits of the last Frame of Configuration Data.
		64 bits of all ones data. Provides a delay for the decryption engine to decrypt the 128 bits of data just received. If the peripheral device can provide the needed 64 clocks while pausing data, then the 64 bits of dummy data are not required, saving file size.
Program Done		32-bit Program Done Command - Encrypted.
End		32-bit Terminator (all ones) - Encrypted.
Filler Bits		Filler to meet the bound requirement.
Delay		64 bits of all ones data. Delay to decrypt the Program Done command and the filler.
Dummy Data	1111...1111	200 bits of Dummy Data (all ones), to provide delay to turn off the decryption engine.

Note: The data in this table is intended for reference only. The shaded area is encrypted data.

**Table 16-6. Encrypted File Format for a Slave Parallel Mode**

Frame	Contents	Description
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator.
Header	1111...1111	2 Dummy Bytes.
		2-byte Encryption Preamble.
30,000 Filler Bytes		This allows time for the device to load and hash the 128-bit encryption key.
Alignment Preamble		2-byte Alignment Preamble.
	11111111	1-byte Dummy Data.
Data		16 bytes of Configuration Data.
		64 bytes (clocks) of all ones data. Provides a delay for the decryption engine to decrypt the 16 bytes of data just received. If the peripheral device can provide the needed 64 clocks while pausing data, then the 64 bytes of dummy data are not required, saving file size.
	...	
		16 bytes of Configuration Data.
		64 bytes (clocks) of all ones data. Provides a delay for the decryption engine to decrypt the 16 bytes of data just received. If the peripheral device can provide the needed 64 clocks while pausing data, then the 64 bytes of dummy data are not required, saving file size.
Program Done		4-byte Program Done Command - Encrypted.
End		4-byte Terminator (all ones) - Encrypted.
Filler Bits		Filler to meet the bound requirement.
Delay		64 bytes of all ones data. Delay to decrypt the Program Done command and the filler.
Dummy Data	1111...1111	200 bytes of Dummy Data (all ones), to provide delay to turn off the decryption engine.

Note: The data in this table is intended for reference only. The shaded area is encrypted data.

## Decryption Flow

From the user's point of view, as 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 (see Figure 16-1) and all information before the preamble is ignored. The preamble, along with the compression bit in Control Register 0, 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 examines Control Register 0 in the bitstream to determine if the file has been compressed. If the file has not been compressed then the Raw data path is selected (see Figure 16-1). If the file has been compressed then the Decompressed path is selected; CRC is then checked and the SRAM fuses programmed.

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 fuses 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. The decoder then examines the decrypted Control Register 0 contents to determine if the file has been compressed. If the file has not been compressed then the Decrypted data path is used, if the file has been compressed then the decrypted data is passed through the decompression engine and the Decompressed path is selected (refer to the block diagram, Figure 16-1). CRC is then checked and the SRAM fuses programmed once the bitstream preamble is read. The decryption and decompression engines are turned off when the internal Done bit is set at the end of configuration. This is done so that if

---

there is any data overflow (to other devices in a chain) the downstream devices will receive raw data from configuration storage.

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 (see Lattice technical note TN1108, *LatticeECP2/M sysCONFIG Usage Guide*, for more information on INITN and DONE).

## Reference Documents

- Lattice Technical Note TN1108, *LatticeECP2/M sysCONFIG Usage Guide*
- Federal Information Processing Standard Publication 197, Nov. 26, 2001. Advanced Encryption Standard (AES)

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)  
+1-503-268-8001 (Outside North America)  
e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
April 2006	01.0	Initial release.
September 2006	01.1	Added information throughout for LatticeECP2M support. Updated screen shots based on the latest software version. Provided clarification in Table 1. Changed Bitstream Preamble to Alignment Preamble through out the document. Reworded sections of the document to provide additional information/clarification.

## 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.

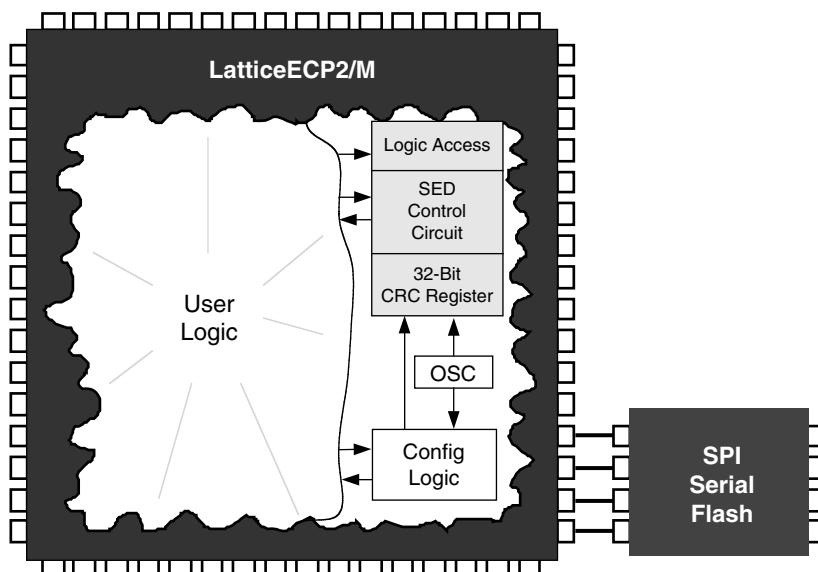
SRAM-based FPGAs store logic configuration data in SRAM cells. As the number and density of SRAM cells in an FPGA increase, the probability that a soft error will alter the programmed logical behavior of the system increases. A number of approaches have been taken to address this issue, but most involve 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 the new LatticeECP2™ and LatticeECP2M™ FPGAs.

## SED Overview

The SED hardware in the LatticeECP2/M 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 1). The SED hardware reads serial data from the FPGA's configuration memory and calculates a CRC. The data that is read, and the CRC that is calculated, does not include EBR memory or PFUs used as RAM. 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 1. System Block Diagram<sup>1</sup>**



1. Any kind of configuration memory can be used, including the SPI configuration shown.

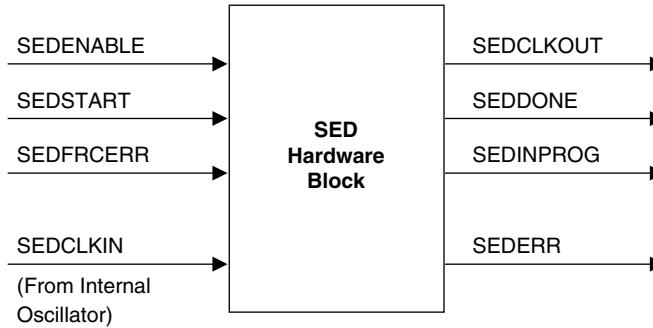
Note that the calculated CRC is based on the particular arrangement of configuration memory for a particular design. Consequently, the expected CRC results cannot be specified until after the design is placed and routed. The ispLEVER® bitstream generation software analyzes the configuration of a placed and routed design and updates the 32-bit SED CRC register contents during bitstream generation.

The following sections describe the LatticeECP2/M SED implementation and flow, along with some sample code to get started with.

## Hardware Description

As shown in Figure 2, the LatticeECP2/M SED hardware has several inputs and outputs that allow the user to control, and monitor, SED behavior.

**Figure 2. Signal Block Diagram**



## Signal Description

**Table 1. SED Signal Description**

Signal Name	Direction	Active	Description
SEDCLKIN	Input	N/A	Clock
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

### SEDCLKIN

Clock input to the SED hardware.

This clock is derived from the LatticeECP2/M's on-chip oscillator. The on-chip oscillator's output goes through a divider to create MCCLK. MCCLK goes through another divider to create SEDCLKIN.

The software default for MCCLK is 2.5 MHz, but this can be modified using the MCCLK\_FREQ global preference in ispLEVER's pre-map Design Planner (see Lattice technical note TN1108, *LatticeECP2/M sysCONFIG Usage Guide*, for possible values of MCCLK).

The divider for SEDCLKIN can be set to 1, 2, 4, 8, 16, 32, 64, 128, or 256. The default is 1, so the default SEDCLKIN frequency is 3.1 MHz. The divider value can be set using a parameter, see the example code at the end of this document.

Note that SEDCLKIN is an internally generated signal, so it should not be included as an input in the user design. See the examples at the end of this document. Also note that while inputs to the SED block are clocked using SED-CLKIN, no attempt has been made to synchronize between clock domains. If this is a concern for a particular design then the designer will need to provide synchronization.

## SEDENABLE

Active high input to the SED hardware, sampled on the rising edge of SEDCLKIN.

**Table 2. SEDENABLE**

State	Description
1	Enables output of SEDCLKOUT, arms SED hardware.
0	Aborts SED and forces all SED hardware outputs low.

## SEDCLKOUT

Gated version of SEDCLKIN, SEDCLKOUT is gated by SEDENABLE.

## SEDSTART

Active high input to the SED hardware, sampled on the rising edge of SEDCLKIN.

**Table 3. SEDSTART**

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, sampled on the rising edge of SEDCLKIN.

**Table 4. SEDFRCERR**

State	Description
1	Forces SEDERR high, simulating an SED error.
0	No action.

## SEDINPROG

Active high output from the SED hardware, clocked out on the rising edge of SEDCLKOUT.

**Table 5. SEDINPROG**

State	Description
1	SED checking is in progress, goes high on the clock following SED-START high.
0	SED checking is not active.

## SEDDONE

Active high output from the SED hardware, clocked out on the rising edge of SEDCLKOUT.

**Table 6. SEDDONE**

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

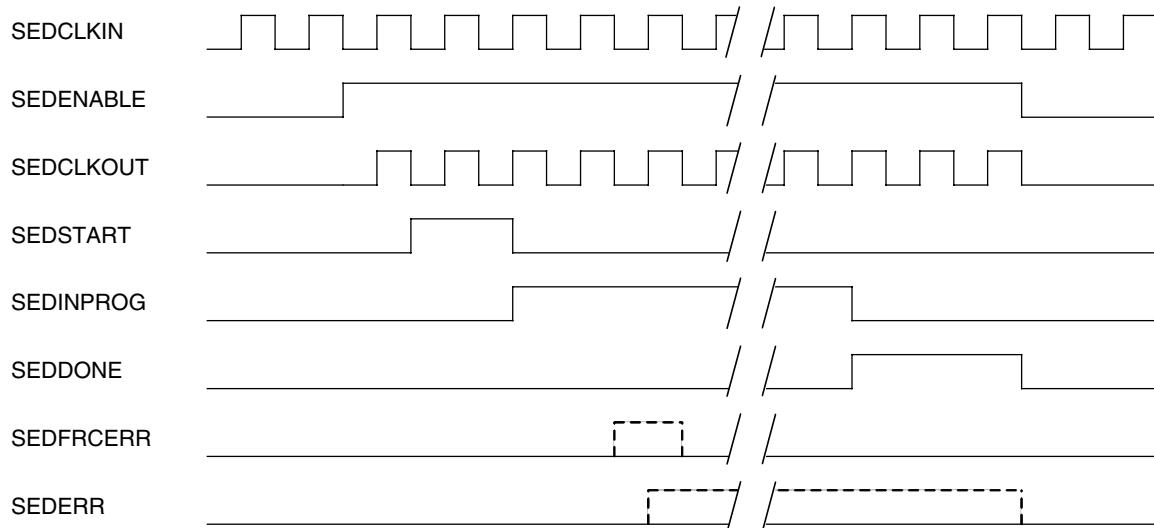
Active high output from the SED hardware, clocked out on the rising edge of SEDCLKOUT.

**Table 7. SEDERR**

State	Description
1	SED has detected an error. Reset by SEDENABLE going low.
0	SED has not detected an error.

## SED Flow

**Figure 3. Timing Diagram**



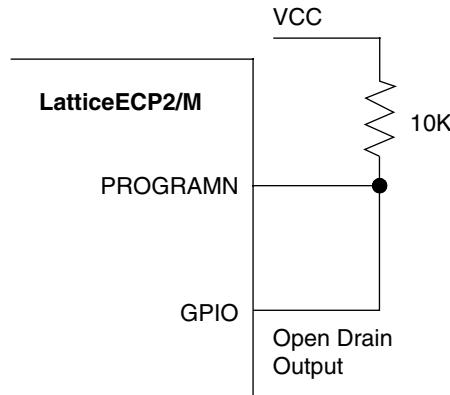
The general SED flow is as follows.

1. User logic sets SEDENABLE high. This signal may be tied high if desired.
2. User logic sets SEDSTART high. SEDINPROG goes high. If SEDDONE is already high it is driven low. SEDSTART may be tied high to enable continuous SED checking.
3. SED starts reading back data from the configuration SRAM.
4. SED finishes checking. SEDERR is updated, SEDINPROG goes low, and SEDDONE goes high.
5. If SEDERR is driven high there are only two ways to reset it, drive SEDENABLE low or reconfigure the FPGA.

The user has two choices when an error is detected, ignore the error, and possibly log it, or reconfigure the FPGA. Reconfiguration can be accomplished by driving the PROGRAMN pin low; this can be done with external logic or by wiring one of the FPGA's general purpose I/Os to the PROGRAMN pin and toggling the pin with user logic, per-

haps something as simple as inverting SEDERR. If a general purpose I/O is tied to PROGRAMN it is recommended that the I/O Type be set to open drain and an external pull-up resistor be connected to the pin.

**Figure 4. 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 SEDCLKIN. 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:

$$\text{Maxbits} / \text{SEDCLKIN} = \text{Time}$$

Maxbits is in mega-bits and depends on the density of the FPGA (see Table 8). SEDCLKIN is frequency in MHz. Time is in seconds

For example, if the design is using a LatticeECP2 with 50K look-up tables and the SEDCLKIN is the software default of 2.5 MHz:

$$9.4 \text{ Mbits} / 2.5 \text{ MHz} = 3.76 \text{ seconds}$$

In this example, SED checking will take approximately 3.76 seconds. Remember that this happens in the background and does not affect user logic performance.

Note that the internal oscillator used to generate SEDCLKIN can vary by  $\pm 30\%$ .

**Table 8. SED Run Time**

Density	Bitstream Size (Mb)	Run Time <sup>1</sup> (seconds)
ECP2-6	1.5	0.6
ECP2-12	2.9	1.16
ECP2-20	4.5	1.68
ECP2-35	6.3	2.52
ECP2-50	9.4	3.76
ECP2-70	13.3	5.32
ECP2M35	10.3	4.12

1. Based on SEDCLKIN = 2.5 MHz.

## Sample Code

The following simple example code shows how to instantiate the SED. In the example the SED is always on and always running, and the outputs of the SED hardware have been routed to FPGA output pins.

Note that the SEDAA primitive is part of ispLEVER 6.0 or later.

### VHDL Example

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity example is
    port (
        Sed_Done      : out std_logic;
        Sed_In_Prog  : out std_logic;
        Sed_Clk_out   : out std_logic;
        Sed_out       : out std_logic);
end;

architecture behavioral of example is

component SEDAA -- SED component
    generic (OSC_DIV : string := "1"); -- set SEDCLKIN divider
    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

    isnt1: SEDAA
    generic map (OSC_DIV=> "1")
    port map (
        SEDENABLE    => '1',    -- tied high
        SEDSTART     => '1',    -- tied high
        SEDFRCERR   => '0',    -- tied low
        SEDERR       => Sed_out, -- wired to an output
        SEDDONE      => Sed_Done, -- wired to an output
        SEDINPROG   => Sed_In_Prog, -- wired to an output
        SEDCLKOUT   => Sed_Clk_out ) ;    -- wired to an output

end behavioral ;
```

## Verilog Example

```
module example (
    Sed_Done,
    Sed_In_Prog,
    Sed_Clk_out,
    Sed_out) ;

output Sed_Done;
output Sed_In_Prog;
output Sed_Clk_out;
output Sed_out;

assign V_hi = 1'b1;
assign V_lo = 1'b0;

parameter OSC_DIV = "1"; // set SEDCLKIN divider

    SEDAA sed_ip (
        .SEDENABLE(V_hi), // always high
        .SEDSTART(V_hi), // always high
        .SEDFRCERR(V_lo), // always low
        .SEDERR(Sed_out), // wired to an output
        .SEDDONE(Sed_Done), // wired to an output
        .SEDINPROG(Sed_In_Prog), // wired to an output
        .SEDCLKOUT(Sed_Clk_out)); // wired to an output

endmodule
```

## Technical Support Assistance

Hotline: 1-800-LATTICE (North America)  
+1-503-268-8001 (Outside North America)  
e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)  
Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
April 2006	01.0	Initial release.
April 2006	01.1	Fixed VHDL code
September 2006	01.2	Changed FPGA family naming to show support for LatticeECP2M.



# LatticeECP2/M Family Handbook

## Revision History

September 2006

Handbook HB1003

### Revision History

Date	Handbook Revision Number	Change Summary
February 2006	01.0	Initial release.
August 2006	01.1	LatticeECP2 Family Data Sheet updated to version 01.1. Technical note TN1103 updated to version 01.1. Technical note TN1104 updated to version 01.1. Technical note TN1108 updated to version 01.2.
September 2006	02.0	LatticeECP2 Family Data Sheet updated to LatticeECP2/M Family Data Sheet version 02.0. Technical note TN1102 updated to version 02.0. Technical note TN1103 updated to version 01.2. Technical note TN1104 updated to version 01.2. Technical note TN1105 updated to version 02.0. Technical note TN1106 updated to version 02.0. Technical note TN1107 updated to version 01.1. Technical note TN1108 updated to version 01.3. Added technical note TN1109. Added technical note TN1113. Added technical note TN1124.

Note: For detailed revision changes, please refer to the revision history for each document.