



## LatticeECP5 Family Handbook

---

HB1012 Version 01.0, March 2014

**Section I. LatticeECP5 Family Data Sheet**
**Introduction**

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

**Architecture**

Architecture Overview .....	2-1
PFU Blocks .....	2-2
Slice .....	2-3
Modes of Operation.....	2-5
Routing.....	2-6
CLOCKING STRUCTURE .....	2-6
sysCLOCK PLL .....	2-6
Clock Distribution Network .....	2-8
Primary Clocks .....	2-8
Dynamic Clock Control (DCC) .....	2-9
Dynamic Clock Select (DCS) .....	2-9
Edge Clock.....	2-9
Clock Dividers .....	2-10
DDRDLL.....	2-11
sysMEM Memory .....	2-12
sysMEM Memory Block.....	2-12
Bus Size Matching .....	2-13
RAM Initialization and ROM Operation .....	2-13
Memory Cascading .....	2-13
Single, Dual and Pseudo-Dual Port Modes.....	2-13
Memory Core Reset.....	2-14
sysDSP™ Slice .....	2-14
sysDSP Slice Approach Compared to General DSP .....	2-14
ECP5 sysDSP Slice Architecture Features .....	2-15
Programmable I/O Cells (PIC) .....	2-19
PIO .....	2-21
Input Register Block.....	2-21
Output Register Block .....	2-22
Tristate Register Block .....	2-23
DDR Memory Support.....	2-24
DQS Grouping for DDR Memory.....	2-24
DLL Calibrated DQS Delay and Control Block (DQSBUF) .....	2-25
sysI/O Buffer .....	2-27
sysI/O Buffer Banks .....	2-27
Typical sysI/O I/O Behavior During Power-up.....	2-29
Supported sysI/O Standards .....	2-29
On-Chip Programmable Termination .....	2-29
Hot Socketing .....	2-30
SERDES and PCS (Physical Coding Sublayer).....	2-30
SERDES Block.....	2-33
PCS.....	2-34
SCI (SERDES Client Interface) Bus.....	2-34
Flexible Dual SERDES Architecture .....	2-35
IEEE 1149.1-Compliant Boundary Scan Testability .....	2-35
Device Configuration.....	2-36
Enhanced Configuration Options .....	2-36
Soft Error Detect (SED) Support.....	2-36
On-Chip Oscillator.....	2-37
Density Shifting .....	2-37

---

**DC and Switching Characteristics**

Absolute Maximum Ratings .....	3-1
Recommended Operating Conditions .....	3-1
Hot Socketing Specifications .....	3-2
Hot Socketing Requirements .....	3-2
ESD Performance .....	3-2
DC Electrical Characteristics.....	3-3
ECP5 Supply Current (Standby) .....	3-4
SERDES Power Supply Requirements.....	3-5
sysI/O Recommended Operating Conditions .....	3-6
sysI/O Single-Ended DC Electrical Characteristics .....	3-7
sysI/O Differential Electrical Characteristics .....	3-8
LVDS.....	3-8
SSTLD.....	3-8
LVCMOS33D .....	3-8
LVDS25E .....	3-9
BLVDS25 .....	3-10
LVPECL33 .....	3-11
MLVDS25.....	3-12
SLVS .....	3-13
ECP5 External Switching Characteristics .....	3-14
Timing Diagrams .....	3-20
ECP5 Family Timing Adders .....	3-21
ECP5 Maximum I/O Buffer Speed .....	3-24
sysCLOCK PLL Timing .....	3-25
DLL Timing.....	3-26
SERDES High-Speed Data Transmitter.....	3-26
SERDES/PCS Block Latency.....	3-28
SERDES High Speed Data Receiver.....	3-29
Input Data Jitter Tolerance.....	3-29
SERDES External Reference Clock.....	3-31
PCI Express Electrical and Timing Characteristics .....	3-32
AC and DC Characteristics .....	3-32
XAUJ/Serial Rapid I/O Type 3/CPRI LV E.30 Electrical and Timing Characteristics .....	3-33
AC and DC Characteristics .....	3-33
Serial Rapid I/O Type 2/CPRI LV E.24 Electrical and Timing Characteristics .....	3-34
AC and DC Characteristics .....	3-34
Gigabit Ethernet/Serial Rapid I/O Type 1/SGMII/CPRI LV E.12 Electrical and Timing Characteristics .....	3-35
AC and DC Characteristics .....	3-35
SMPTE SD/HD-SDI/3G-SDI (Serial Digital Interface) Electrical and Timing Characteristics.....	3-36
AC and DC Characteristics .....	3-36
ECP5 sysCONFIG Port Timing Specifications .....	3-37
JTAG Port Timing Specifications .....	3-42
Switching Test Conditions.....	3-43
Signal Descriptions .....	4-1

**Pinout Information**

PICs and DDR Data (DQ) Pins Associated with the DDR Strobe (DQS) Pin .....	4-4
Pin Information Summary.....	4-5
LFE5UM.....	4-5
LFE5U .....	4-7

**Ordering Information**

ECP5 Part Number Description .....	5-1
Ordering Part Numbers .....	5-2

---

Commercial.....	5-2
Industrial.....	5-4
<b>Supplemental Information</b>	
For Further Information .....	6-1
<b>Revision History</b>	
<b>Section II. LatticeECP5 Family Technical Notes</b>	
<b>ECP5 sysCONFIG Usage Guide</b>	
Introduction .....	8-1
Features .....	8-1
Definition of Terms .....	8-1
Configuration Details.....	8-2
Bitstream/PROM Sizes .....	8-2
Device Status Register.....	8-3
sysCONFIG™ Ports.....	8-4
sysCONFIG Pins.....	8-4
Dedicated sysCONFIG Pins.....	8-6
Dual-Purpose sysCONFIG Pins.....	8-9
PERSISTENT.....	8-11
JTAG Configuration Port Pins .....	8-11
Configuration Process and Flow .....	8-12
Power-up Sequence.....	8-13
Initialization .....	8-13
Configuration.....	8-14
Wake-up.....	8-14
User Mode.....	8-15
Clearing the Configuration Memory and Re-initialization.....	8-15
Reconfiguration Priority.....	8-15
Configuration Modes .....	8-16
Master SPI Modes.....	8-16
Slave SPI Mode (SSPI).....	8-20
Slave Parallel Mode (SPCM) .....	8-30
Slave Serial Configuration Mode (SCM) .....	8-32
JTAG Mode .....	8-32
TransFR Operation .....	8-33
Software Selectable Options.....	8-33
Device Wake-up Sequence.....	8-36
Wake-up Signals .....	8-37
Wake-up Clock Selection.....	8-37
Daisy Chaining .....	8-38
Bypass Option.....	8-40
Flowthrough Option.....	8-41
Technical Support Assistance.....	8-42
Revision History .....	8-42
Appendix A. ECP5 Slave SPI Programming Guide .....	8-43
Appendix B. ECP5 Bitstream File Format .....	8-44
Configuration Bitstream Format .....	8-44
Read Back Bitstream Format .....	8-49
ECP5 Device Specific .....	8-50
Appendix C. Advanced Applications – The Slave SPI Port and SPI Flash Interface .....	8-51
Appendix D. Advanced Applications – Master SPI sysCONFIG Daisy Chaining.....	8-52
Appendix E. Advanced Applications – Slave SPI sysCONFIG Daisy Chaining .....	8-53

**ECP5 SERDES/PCS Usage Guide**

Introduction .....	9-1
Features .....	9-1
New Features Over LatticeECP3™ SERDES/PCS .....	9-2
Using This Technical Note .....	9-2
Standards Supported .....	9-2
Architecture Overview .....	9-3
Multi-Protocol Design Consideration.....	9-5
Detailed Channel Block Diagram .....	9-6
Clocks and Resets .....	9-6
Transmit Data Bus .....	9-7
Receive Data Bus .....	9-7
Control and Status Signals.....	9-9
SERDES/PCS .....	9-10
I/O Descriptions.....	9-10
SERDES/PCS Functional Description .....	9-13
SERDES Buffers.....	9-13
SERDES .....	9-14
Reference Clock from FPGA Core.....	9-16
Full Data, Div 2 and Div 11 Data Rates .....	9-16
Reference Clock Sources .....	9-16
Clock Distribution in Complementary DCUs .....	9-17
Spread Spectrum Clocking (SSC) Support.....	9-17
Loss of Signal.....	9-18
Loss Of Lock .....	9-18
TX Lane-to-Lane Skew .....	9-19
Transmit Data.....	9-19
8b10b Encoder.....	9-19
Serializer .....	9-19
Receive Data.....	9-19
Deserializer .....	9-19
Word Alignment (Byte Boundary Detect) .....	9-19
8b10b Decoder.....	9-20
External Link State Machine Option.....	9-20
Idle Insert for Gigabit Ethernet Mode .....	9-20
Clock Tolerance Compensation (CTC) .....	9-20
Calculating Minimum Interpacket Gap .....	9-23
Protocol Modes .....	9-23
Generic 8b10b Mode.....	9-23
Gigabit Ethernet and SGMII Modes .....	9-24
XAUI Mode .....	9-26
PCI Express Revision 1.1 (2.5 Gpbs) Mode .....	9-27
PCI Express Termination .....	9-27
PCI Express Electrical Idle Transmission .....	9-28
PCI Express Electrical Idle Detection.....	9-29
PCI Express Receiver Detection.....	9-29
PCI Express Power Down Mode.....	9-30
PCI Express Beacon Support .....	9-30
Serial RapidIO (SRIo) Mode.....	9-31
Serial Digital Video and Out-Of-Band Low Speed SERDES Operation.....	9-32
Common Public Radio Interface (CPRI) .....	9-33
SDI (SMPTE) Mode .....	9-35
Example of Dual-based Channel Arrangements.....	9-36
Tx Case 1:.....	9-36

---

Tx Case 2:	9-37
Tx Case 3:	9-38
Rx Case1:	9-39
Rx Case 2:	9-40
FPGA Interface Clocks	9-40
2:1 Gearing	9-42
Case I_a: 8/10bit, CTC FIFO NOT Bypassed	9-43
Case I_b: 8 /10bit, CTC FIFO Bypassed	9-44
Case I_c: 8/10bit, FPGA Bridge FIFOs Bypassed	9-45
Case I_d: 8/10bit, CTC FIFO and FPGA Bridge FIFOs Bypassed	9-45
Case II_a: 16/20bit, CTC FIFO NOT Bypassed	9-46
Case II_b: 16/20bit, CTC FIFO Bypassed	9-47
SERDES/PCS Block Latency	9-48
SERDES Client Interface	9-49
Introduction	9-49
Interrupts and Status	9-52
Dynamic Configuration of the SERDES/PCS Dual	9-52
SERDES Debug Capabilities	9-52
PCS Loopback Modes	9-52
ECO Editor	9-53
Other Design Considerations	9-53
Simulation of the SERDES/PCS	9-53
16/20-Bit Word Alignment	9-53
SERDES/PCS Reset	9-55
Reset and Power-Down Control	9-55
Reset Generation	9-55
Reset Sequence	9-57
Clarity Designer (System Builder/Planner) Overview	9-57
Top Level Block Diagram	9-58
SERDES/PCS Generation in Clarity Designer (System Builder/Planner)	9-58
EXTREF Block	9-71
EXTREF I/O Port Description	9-71
References	9-72
Technical Support Assistance	9-72
Revision History	9-72
Appendix A. Configuration Registers	9-73
Dual Registers Overview	9-73
Per Dual PCS Control Register Details	9-74
Per Dual SerDes Control Register Details	9-75
Per Dual Reset and Clock Control Register Details	9-77
Per Dual PCS Status Register Details	9-77
Per Dual SerDes Status Register Details	9-78
Per Channel Register Overview	9-79
Per Channel PCS Register Details	9-80
Per Channel SerDes Control Register Details	9-84
Per Channel Reset and Clock Control Register Details	9-90
Per Channel PCS Status Register Details	9-90
Per Channel SerDes Status Register Details	9-92
Appendix B. Register Settings for Various Standards	9-93
<b>ECP5 sysIO Usage Guide</b>	
Introduction	10-1
sysIO Buffer Overview	10-1
Supported sysIO Standards	10-1
sysIO Banking Scheme	10-2

---

V <sub>CC</sub> (1.1 V) .....	10-3
VCCIO (1.2 V/1.35 V/1.5 V/1.8 V/2.5 V/3.3 V) .....	10-3
VCCAUX (2.5V) .....	10-3
V <sub>CCIO8</sub> (1.2 V/1.5 V/1.8 V/2.5 V/3.3 V) .....	10-4
VREF1.....	10-4
Hot Socketing Support .....	10-4
Standby .....	10-4
LVDS sysIO Buffer Pairs (A/B and C/D on Left and Right Sides) .....	10-4
sysIO Buffer Pair (A/B Pair on Top and Bottom Sides).....	10-5
Mixed Voltage Support in a Bank.....	10-6
sysIO Buffer Configurations .....	10-7
Software sysIO Attributes.....	10-11
IO_TYPE .....	10-11
OPENDRAIN.....	10-12
DRIVE .....	10-12
DIFFDRIVE .....	10-13
TERMINATION .....	10-13
DIFFRESISTOR.....	10-13
CLAMP .....	10-13
PULLMODE .....	10-13
SLEWRATE .....	10-13
HYSTERESIS .....	10-14
VREF.....	10-14
DIN/DOUT.....	10-14
LOC .....	10-14
Technical Support Assistance .....	10-15
Revision History .....	10-15
Appendix A. sysIO Primitive Symbols and Instance Examples.....	10-16
Primitive Symbols.....	10-16
Instance Examples.....	10-17
Appendix B. sysIO Attribute Examples .....	10-18
Appendix C. sysIO Buffer Design Rules .....	10-20
Appendix D. sysIO Attributes Using the Diamond Spreadsheet View User Interface .....	10-21
<b>ECP5 sysCLOCK PLL/DLL</b>	
<b>Design and Usage Guide</b>	
Introduction .....	11-1
Clock/Control Distribution Network .....	11-1
ECP5 Top-Level View .....	11-2
Clocking Architecture Overview .....	11-2
Primary Clock Network.....	11-2
Edge Clock Network.....	11-2
Overview of Other Clocking Elements .....	11-3
Edge Clock Dividers (CLKDIVF) .....	11-3
PCS Clock Dividers (PCSCLKDIV) .....	11-3
Dynamic Clock Select (DCSC).....	11-3
Edge Clock Bridge with Clock Select (ECLKBRIDGECS) .....	11-3
Edge Clock Stop (ECLKSYNCB) .....	11-3
Oscillator (OSCG) .....	11-3
sysCLOCK PLL Overview .....	11-4
PLL Features.....	11-5
Dedicated PLL Inputs.....	11-5
Input PLL Clock Selection (PLLREFCS) .....	11-5
Clock Injection Delay Removal .....	11-5
Clock Phase Adjustment.....	11-5

---

Frequency Synthesis.....	11-6
Additional Features .....	11-6
Primary Clocks .....	11-6
Primary Clock Sources.....	11-6
Primary Clock Routing .....	11-6
Dedicated Clock Inputs .....	11-7
PCS Clock Dividers (PCSCLKDIV) .....	11-7
PCSCLKDIV Component Definition .....	11-9
PCSCLKDIV Usage in VHDL.....	11-10
PCSCLKDIV Usage in Verilog .....	11-10
Dynamic Clock Select (DCSC).....	11-11
DCS Timing Diagrams .....	11-12
DCSC Component Definition .....	11-13
DCSMODE Attribute .....	11-14
DCSC Usage in VHDL .....	11-14
DCSC Usage in Verilog.....	11-15
Dynamic Clock Control (DCCA) .....	11-15
DCCA Component Definition .....	11-16
DCCA Usage in VHDL .....	11-16
Internal Oscillator (OSCG) .....	11-17
OSCG Component Definition .....	11-17
OSCG Usage in VHDL.....	11-17
OSCG Usage in Verilog .....	11-18
Edge Clocks.....	11-18
Edge Clock Dividers (CLKDIVF) .....	11-19
CLKDIVF Component Definition .....	11-19
CLKDIVF Usage in VHDL .....	11-19
CLKDIVF Usage in Verilog.....	11-20
Edge Clock Bridge (ECLKBRIDGECS).....	11-20
ECLKBRIDGECS Component Definition .....	11-21
ECLKBRIDGECS Usage in VHDL .....	11-21
ECLKBRIDGECS Usage in Verilog.....	11-21
Edge Clock Synchronization (ECLKSYNCB).....	11-22
ECLKSYNCB Component Definition.....	11-22
ECLKSYNCB Usage in VHDL.....	11-23
General Routing for Clocks .....	11-23
General routing PCLK pins .....	11-24
sysCLOCK PLL .....	11-24
Functional Description.....	11-26
PLL Features.....	11-26
PLL Inputs and Outputs .....	11-27
Dynamic Phase Adjustment.....	11-30
Low Power Features .....	11-32
PLL Usage in Clarity Designer .....	11-32
PLL Reference Clock Switch Primitive (PLLREFCS).....	11-38
PLLREFCS Usage in VHDL.....	11-38
Technical Support Assistance .....	11-39
Revision History .....	11-39
Appendix A. Primary Clock Sources and Distribution .....	11-40
Appendix B. Pinout Rules for Clocking in ECP5 Devices .....	11-43

**ECP5 Memory Usage Guide**

Introduction .....	12-1
Memory Generation .....	12-1

---

Clarity Designer Flow .....	12-2
Utilizing PMI .....	12-4
Utilizing Direct instantiation of Memory Primitives .....	12-4
Memory Features .....	12-5
Byte Enable.....	12-5
Memory Modules.....	12-5
Memory Cascading .....	12-6
Single Port RAM (RAM_DQ) – EBR Based .....	12-6
True Dual-Port RAM (RAM_DP_TRUE) – EBR Based.....	12-13
Pseudo Dual-Port RAM (RAM_DP) – EBR Based.....	12-22
Read Only Memory (ROM) – EBR Based.....	12-26
First In First Out (FIFO) Memory.....	12-29
Dual Clock First-In-First-Out (FIFO_DC) – EBR or LUT Based .....	12-36
FIFO_DC Flags.....	12-36
Distributed Single-Port RAM (Distributed_SPRAM) – PFU-Based .....	12-45
Distributed Dual-Port RAM (Distributed_DPRAM) – PFU-Based.....	12-48
Distributed ROM (Distributed_ROM) – PFU-Based .....	12-51
Initializing Memory .....	12-54
Initialization File Formats .....	12-54
Technical Support Assistance.....	12-57
Revision History .....	12-57
Appendix A. Attribute Definitions.....	12-58
DATA_WIDTH.....	12-58
REGMODE.....	12-58
CSDECODE.....	12-58
WRITEMODE.....	12-58

## **ECP5 High-Speed I/O Interface**

Introduction .....	13-1
External Interface Description .....	13-1
High-Speed I/O Interface Building Blocks .....	13-1
Edge Clocks .....	13-2
Primary Clocks .....	13-2
DQS Lane .....	13-2
PLL.....	13-2
DDRDLL .....	13-3
DQSBUF .....	13-3
DLLDEL .....	13-3
Input DDR (IDDR) .....	13-3
Output DDR (ODDR).....	13-4
Edge Clock Dividers (CLKDIV) .....	13-4
Input/Output DELAY .....	13-4
Building Generic High Speed Interfaces .....	13-4
Types of High-Speed DDR Interfaces.....	13-4
High-Speed DDR Interface Details .....	13-6
GDDRX1_RX.SCLK.Centered.....	13-6
GDDRX1_RX.SCLK.Aligned.....	13-7
GDDRX2_RX.ECLK.Centered .....	13-8
GDDRX2_RX.ECLK.Aligned.....	13-9
GDDRX2_RX.MIPI.....	13-11
GDDRX71_RX.ECLK.....	13-11
GDDRX1_TX.SCLK.Aligned .....	13-13
GDDRX1_TX.SCLK.Centered .....	13-13
GDDRX2_TX.ECLK.Aligned .....	13-14

GDDRX2_TX.ECLK.Centered .....	13-15
GDDRX71_TX.ECLK .....	13-16
Generic DDR Design Guidelines.....	13-16
Using the High Speed Edge Clock Bridge .....	13-16
Receive Interface Guidelines .....	13-17
Transmit interface Guidelines .....	13-17
Clocking Guidelines for Generic DDR Interface.....	13-17
Timing Analysis for High Speed DDR Interfaces .....	13-18
ECP5 Memory Interfaces .....	13-22
DDR Memory Interface Requirements .....	13-25
Features for Memory Interface Implementation .....	13-26
Memory Interface Implementation.....	13-31
DDR Memory Interface Design Rules and Guidelines .....	13-37
DDR2/DDR3 Memory Interface Termination Guidelines.....	13-38
DDR Memory Interface Pinout Guidelines .....	13-39
Pin Placement Considerations for Improved Noise Immunity.....	13-40
Using Clarity Designer to Build and Plan High Speed DDR Interfaces.....	13-41
Configuring DDR Modules in Clarity Designer.....	13-41
Configuring SDR Modules.....	13-42
Configuring DDR Generic modules.....	13-44
Configuring 7:1 LVDS Interface Modules.....	13-48
Configuring DDR Memory Interfaces .....	13-50
Building DDR Interfaces in Clarity Designer .....	13-55
Planning DDR Interfaces in Clarity Designer .....	13-55
DDR Software Primitives and Attributes .....	13-56
Input/Output DELAY.....	13-57
DELAYF .....	13-57
DELAYG.....	13-58
DELAY Attribute Description .....	13-58
DDRDLL (Master DLL).....	13-59
DLL Delay (DLLDEL) .....	13-60
Generic DDR Input and Output Primitives .....	13-61
Input DDR Primitives.....	13-61
Output DDR Primitives .....	13-62
Memory DDR Primitives.....	13-64
Input and Output Memory DDR Primitives .....	13-66
Memory Input DDR Primitives.....	13-67
Memory Output DDR Primitives for DQ Outputs.....	13-68
Memory Output DDR Primitives for DQS Output .....	13-68
Memory Output DDR Primitives for Tristate Output Control .....	13-69
Memory Output DDR Primitives for Address and Command .....	13-70
Soft IP Modules.....	13-71
Detailed Description of Each Soft IP .....	13-72
Technical Support Assistance .....	13-76
Revision History .....	13-76
<b>Power Consumption and Management for ECP5 Devices</b>	
Introduction .....	14-1
Power Supply Sequencing and Hot Socketing.....	14-1
Recommended Power-up Sequence .....	14-1
Power Standby Mode.....	14-1
Normal operation.....	14-1
Low power operation (Standby) .....	14-1
Digital Temperature Readout .....	14-4

---

DTR Timing .....	14-5
Equivalent Junction Temperature for DTROUT values .....	14-6
Power Calculator .....	14-8
Power Calculator and Power Equations .....	14-8
Thermal Impedance and Airflow .....	14-11
DELPHI Models .....	14-11
Reducing Power Consumption .....	14-12
Power Calculator Assumptions .....	14-13
Technical Support Assistance .....	14-14
Revision History .....	14-14

## **ECP5 sysDSP Usage Guide**

Introduction .....	15-1
sysDSP Overview .....	15-1
Operating modes and features .....	15-3
Using sysDSP .....	15-4
Primitive Instantiation sysDSP .....	15-4
Using Clarity Designer to Configure and Generate DSP Modules .....	15-4
Inferencing sysDSP slice .....	15-7
Targeting the sysDSP Slice by Instantiating Primitives .....	15-7
MULT9X9C – Advanced 9X9 DSP Multiplier .....	15-7
MULT9X9D – Advanced 9X9 DSP Multiplier for Highspeed .....	15-9
MULT18X18C – Basic 18X18 DSP Multiplier .....	15-11
MULT18X18D – Advanced 18X18 DSP Multiplier for High Speed .....	15-13
ALU24A – 24-bit Ternary Adder/ Subtractor .....	15-15
ALU54A – 54-bit Ternary Adder/ Subtractor .....	15-17
ALU24B – 24-bit Ternary Adder/ Subtractor for 9X9 Mode .....	15-20
ALU54B – 54-bit Ternary Adder/ Subtractor for High Speed .....	15-22
PRADD9A – 9-bit Pre-Adder/Shift .....	15-26
PRADD18A – 18-bit Pre-Adder/Shift .....	15-29
Technical Support Assistance .....	15-32
Revision History .....	15-32
Appendix A: Instantiating DSP Primitives in HDL .....	15-33
Verilog Example Showing Snippet of the MULT18X18D Instantiation .....	15-33
Appendix B: HDL Inference for DSP .....	15-38
VHDL Example to Infer Fully Pipelined Multiplier .....	15-38
Verilog Example to Infer Fully Pipelined Multiplier .....	15-39

## **ECP5 Hardware Checklist**

Introduction .....	16-1
Power Supplies .....	16-2
ECP5 SERDES/PCS Power Supplies .....	16-2
Power Estimation .....	16-3
Configuration Considerations .....	16-3
I/O Pin Assignments .....	16-4
Clock Inputs .....	16-4
Pinout Considerations .....	16-4
LVDS Pin Assignments .....	16-4
HSUL and SSTL Pin Assignments .....	16-5
SERDES Pin Considerations .....	16-5
ECP5U to ECP5UM Migration .....	16-5
Technical Support Assistance .....	16-7
Revision History .....	16-7

**Section III. LatticeECP5 Family Handbook Revision History**

Revision History .....	17-1
------------------------	------



## **Section I. LatticeECP5 Family Data Sheet**

---

DS1044 Version 01.0, March 2014

March 2014

Advance Data Sheet DS1044

## Features

- **Higher Logic Density for Increased System Integration**
  - 24K to 84K LUTs
  - 197 to 365 user programmable I/Os
- **Embedded SERDES**
  - 270 Mbps to 3.2 Gbps for Generic 8b10b, 10-bit SERDES, and 8-bit SERDES modes
  - Data Rates 270 Mbps to 3.2 Gbps per channel for all other protocols
  - Up to 4 channels per device: PCI Express, Ethernet (1GbE, SGMII, XAUI), CPRI, SMPTE 3G and Serial RapidIO
- **sysDSP™**
  - Fully cascadable slice architecture
  - 12 to 160 slices for high performance multiply and accumulate
  - Powerful 54-bit ALU operations
  - Time Division Multiplexing MAC Sharing
  - Rounding and truncation
  - Each slice supports
    - Half 36 x 36, two 18 x 18 or four 9 x 9 multipliers
    - Advanced 18 x 36 MAC and 18 x 18 Multiply-Multiply-Accumulate (MMAC) operations
- **Flexible Memory Resources**
  - Up to 3.744 Mbits sysMEM™ Embedded Block RAM (EBR)
  - 194K to 669K bits distributed RAM
- **sysCLOCK Analog PLLs and DLLs**
  - Four DLLs and four PLLs in LFE5-45 and LFE5-85; two DLLs and two PLLs in LFE5-25
- **Pre-Engineered Source Synchronous I/O**
  - DDR registers in I/O cells
  - Dedicated read/write levelling functionality
  - Dedicated gearing logic
  - Source synchronous standards support
    - ADC/DAC, 7:1 LVDS, XGMII
    - High Speed ADC/DAC devices
  - Dedicated DDR2/DDR3 & LPDDR2/LPDDR3 memory support with DQS logic, up to 800 B00Mbps data-rate
- **Programmable sysI/O™ Buffer Supports Wide Range of Interfaces**
  - On-chip termination
  - LVTTL and LVCMSOS 33/25/18/15/12
  - SSTL 18/15 I, II
  - HSUL12
  - LVDS, Bus-LVDS, LVPECL, RSRS, MLVDS
  - subLVDS & SLVS, MIPI D-PHY input interfaces
- **Flexible Device Configuration**
  - Shared bank for configuration I/Os
  - SPI boot flash interface
  - Dual-boot images supported
  - Slave SPI
  - TransFR™ I/O for simple field updates
  - Soft Error Detect embedded macro
- **System Level Support**
  - IEEE 1149.1 and IEEE 1532 compliant
  - Reveal Logic Analyzer
  - On-chip oscillator for initialization and general use
  - 1.1 V core power supply

**Table 1-1. ECP5 Family Selection Guide**

Device	LFE5UM-25	LFE5UM-45	LFE5UM-85	LFE5U-25	LFE5U-45	LFE5U-85
LUTs (K)	24	44	84	24	44	84
sysMEM Blocks (18 Kbits)	56	108	208	56	108	208
Embedded Memory (Kbits)	1,008	1944	3744	1,008	1944	3744
Distributed RAM Bits (Kbits)	194	351	669	194	351	669
18 X 18 Multipliers	28	72	156	28	72	156
SERDES (Dual/Channels)	1/2	2/4	2/4	0	0	0
PLLs/DLLs	2/2	4/4	4/4	2/2	4/4	4/4
<b>Packages and SERDES Channels / I/O Combinations</b>						
285 csFBGA (10 x 10 mm <sup>2</sup> , 0.5 mm)	2/118	2/118	2/118	0/118	0/118	0/118
381 caBGA (17 x 17 mm <sup>2</sup> )	2/197	4/203	4/205	0/197	0/203	0/205
554 caBGA (23 x 23 mm <sup>2</sup> )		4/245	4/259		0/245	0/259
756 caBGA (27 x 27 mm <sup>2</sup> )			4/365			0/365

© 2014 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at [www.latticesemi.com/legal](http://www.latticesemi.com/legal). All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

## Introduction

The ECP5 family of FPGA devices is optimized to deliver high performance features such as an enhanced DSP architecture, high speed SERDES and high speed source synchronous interfaces in an economical FPGA fabric. This combination is achieved through advances in device architecture and the use of 40nm technology making the devices suitable for high-volume, high-speed, low-cost applications.

The ECP5 device family covers look-up-table (LUT) capacity to 84K logic elements and supports up to 365 user I/Os. The ECP5 device family also offers up to 156 18 x 18 multipliers and a wide range of parallel I/O standards.

The ECP5 FPGA fabric is optimized high performance with low power and low cost in mind. The ECP5 devices utilize reconfigurable SRAM logic technology and provide popular building blocks such as LUT-based logic, distributed and embedded memory, Phase Locked Loops (PLLs), Delay Locked Loops (DLLs), pre-engineered source synchronous I/O support, enhanced sysDSP slices and advanced configuration support, including encryption and dual-boot capabilities.

The pre-engineered source synchronous logic implemented in the ECP5 device family supports a broad range of interface standards, including DDR3, LPDDR3, XGMII and 7:1 LVDS.

The ECP5 device family also features high speed SERDES with dedicated PCS functions. High jitter tolerance and low transmit jitter allow the SERDES plus PCS blocks to be configured to support an array of popular data protocols including PCI Express, SMPTE, Ethernet (XAUI, GbE, and SGMII) and CPRI. Transmit De-emphasis and Receive Equalization settings make the SERDES suitable for transmission and reception over various forms of media.

The ECP5 devices also provide flexible, reliable and secure configuration options, such as dual-boot capability, bit-stream encryption, and TransFR field upgrade features.

The Lattice Diamond™ design software allows large complex designs to be efficiently implemented using the ECP5 FPGA family. Synthesis library support for ECP5 devices is available for popular logic synthesis tools. The Diamond tools use the synthesis tool output along with the constraints from its floor planning tools to place and route the design in the ECP5 device. The tools extract the timing from the routing and back-annotate it into the design for timing verification.

Lattice provides many pre-engineered IP (Intellectual Property) modules for the ECP5 family. By using these configurable soft core IPs as standardized blocks, designers are free to concentrate on the unique aspects of their design, increasing their productivity.

### Architecture Overview

Each ECP5 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 sysDSP™ Digital Signal Processing slices, as shown in Figure 2-1. The LFE5-85 devices have three rows of DSP slices, the LFE5-45 devices have two rows and LFE5-25 devices have one. In addition, the LFE5U devices contain SERDES Duals on the bottom of the device.

The Programmable Functional Unit (PFU) contains the building blocks for logic, arithmetic, RAM and ROM functions. The PFU block is 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 ECP5 devices contain one or more rows of sysMEM EBR blocks. sysMEM EBRs are large, dedicated 18Kbit fast memory blocks. Each sysMEM block can be configured in a variety of depths and widths as RAM or ROM. In addition, ECP5 devices contain up to three rows of DSP slices. Each DSP slice has multipliers and adder/accumulators, which are the building blocks for complex signal processing capabilities.

The ECP5 devices feature up to 4 embedded 3.2 Gbps SERDES (Serializer / Deserializer) channels. Each SERDES channel contains independent 8b/10b encoding / decoding, polarity adjust and elastic buffer logic. Each group of two SERDES channels, along with its Physical Coding Sub-layer (PCS) block, creates a dual. The functionality of the SERDES/PCS duals can be controlled by memory cells set during device configuration or by registers that are addressable during device operation. The registers in every dual can be programmed via the SERDES Client Interface (SCI). These duals (up to two) are located at the bottom of the devices.

Each PIC block encompasses two PIOs (PIO pairs) with their respective sysI/O buffers. The sysI/O buffers of the ECP5 devices are arranged in seven banks (eight banks for LFE5UM-85 device), allowing the implementation of a wide variety of I/O standards. One of these banks (Bank 8) is shared with the programming interfaces. 50% of the PIO pairs on the left and right edges of the device can be configured as LVDS transmit pairs, and all pairs on left and right can be configured as LVDS receive pairs. The PIC logic in the left and right banks also includes pre-engineered support to aid in the implementation of high speed source synchronous standards such as XGMII, 7:1 LVDS, along with memory interfaces including DDR3 and LPDDR3.

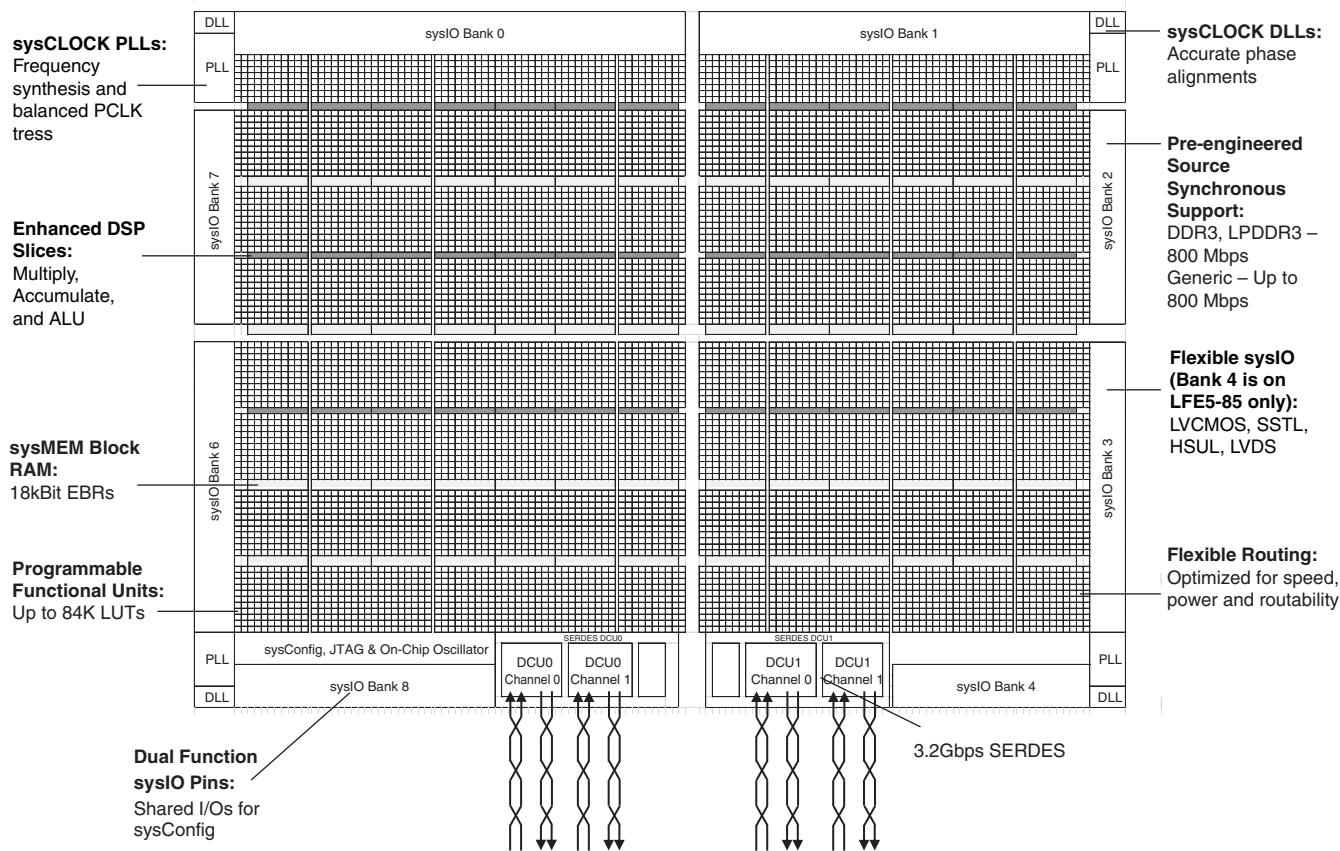
The ECP5 registers in PFU and sysI/O can be configured to be SET or RESET. After power up and the device is configured, it enters into user mode with these registers SET/RESET according to the configuration setting, allowing the device entering to a known state for predictable system function.

Other blocks provided include PLLs, DLLs and configuration functions. The ECP5 architecture provides up to four Delay Locked Loops (DLLs) and up to four Phase Locked Loops (PLLs). The PLL and DLL blocks are located at the corners of each device.

The configuration block that supports features such as configuration bit-stream decryption, transparent updates and dual-boot support is located at the bottom of each device, to the left of the SERDES blocks. Every device in the ECP5 family supports a sysCONFIG™ ports located in that same corner, powered by Vccio8, allowing 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 ECP5 devices use 1.1 V as their core voltage.

**Figure 2-1. Simplified Block Diagram, LFE5UM-85 Device (Top Level)**

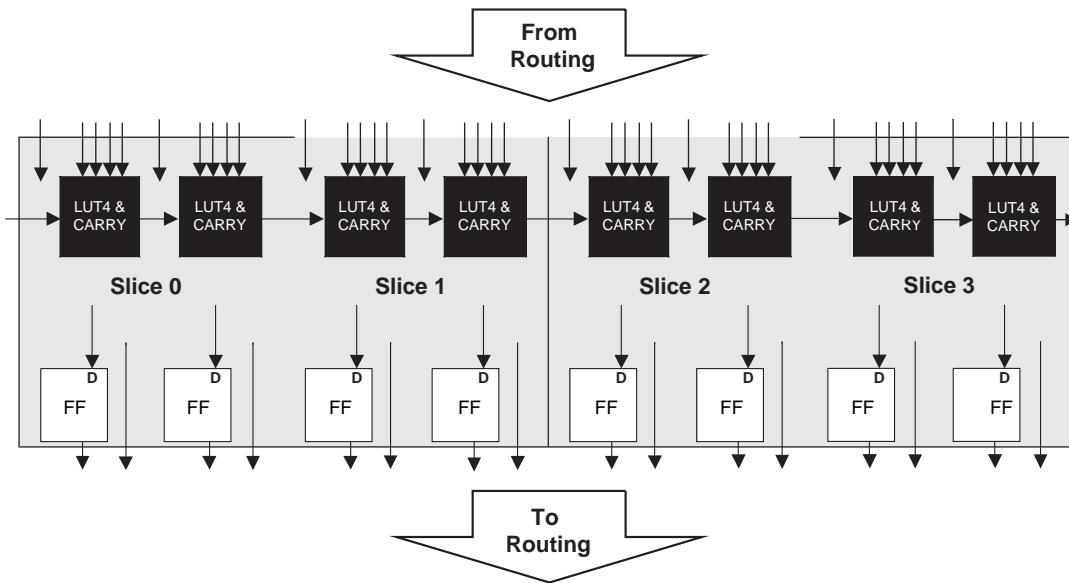


Note: There is no Bank 4 in LFE5-25 and LFE5-45.  
 There are no PLL and DLL on the top corners in LFE5-25.

## PFU Blocks

The core of the ECP5 device consists of PFU blocks. Each PFU block consists of four interconnected slices numbered 0-3 as shown in Figure 2-2. Each slice contains two LUTs. All the interconnections to and from PFU blocks are from routing. There are 50 inputs and 23 outputs associated with each PFU block.

The PFU block can be used in Distributed RAM or ROM function, or used to perform Logic, Arithmetic, or ROM functions. Table 2-1 shows the functions each slice can perform in either mode.

**Figure 2-2. PFU Diagram**


## Slice

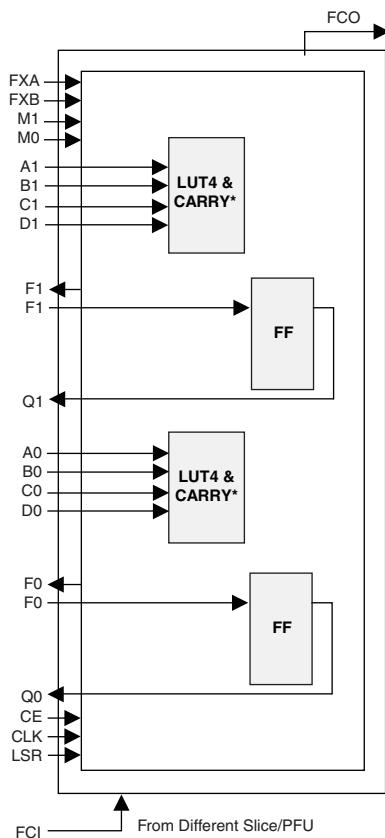
Each slice contains two LUT4s feeding two registers. In Distributed SRAM mode, Slice 0 through Slice 2 are configured as distributed memory, and Slice 3 is used as Logic or ROM. Table 2-1 shows the capability of the slices along with the operation modes they enable. In addition, each PFU contains logic that allows the LUTs to be combined to perform functions such as LUT5, LUT6, LUT7 and LUT8. There is control logic to perform set/reset functions (programmable as synchronous/ asynchronous), clock select, chip-select and wider RAM/ROM functions.

**Table 2-1. Resources and Modes Available per Slice**

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

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

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

**Figure 2-3. Slice Diagram**


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

WCK is CLK

WRE is from LSR

DI[3:2] for Slice 1 and DI[1:0] for Slice 0 data from Slice 2

WAD [A:D] is a 4-bit address from slice 2 LUT input

**Table 2-2. Slice Signal Descriptions**

Function	Type	Signal Names	Description
Input	Data signal	A0, B0, C0, D0	Inputs to LUT4
Input	Data signal	A1, B1, C1, D1	Inputs to LUT4
Input	Multi-purpose	M0	Multipurpose Input
Input	Multi-purpose	M1	Multipurpose Input
Input	Control signal	CE	Clock Enable
Input	Control signal	LSR	Local Set/Reset
Input	Control signal	CLK	System Clock
Input	Inter-PFU signal	FCI	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	Inter-PFU signal	FCO	Fast carry chain output <sup>1</sup>

1. See Figure 2-3 for connection details.  
 2. Requires two PFUs.

## Modes of Operation

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

### Logic Mode

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

### Ripple Mode

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

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

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

### RAM Mode

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

ECP5 devices support distributed memory initialization.

The Lattice design tools support the creation of a variety of different size memories. Where appropriate, the software will construct these using distributed memory primitives that represent the capabilities of the PFU. Table 2-3 shows the number of slices required to implement different distributed RAM primitives. For more information about using RAM in ECP5 devices, please see TN1264, [ECP5 Memory Usage Guide](#).

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

	SPR 16X4	PDPR 16X4
Number of slices	3	3

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

## ROM Mode

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

For more information, please refer to TN1264, [ECP5 Memory Usage Guide](#).

## Routing

There are many resources provided in the ECP5 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 ECP5 family has an enhanced routing architecture that produces a compact design. The Diamond design software tool suites take the output of the synthesis tool and places and routes the design.

## CLOCKING STRUCTURE

ECP5 clocking structure consists of clock synthesis blocks, sysCLOCK PLL; balanced clock tree networks, PCLK and ECLK trees; and efficient clock logic modules, CLOCK DIVIDER and Dynamic Clock Select (DCS), Dynamic Clock Control (DCC), and DLL. Each of these functions is described as follow.

### sysCLOCK PLL

The sysCLOCK PLLs provide the ability to synthesize clock frequencies. The devices in the ECP5 family support two to four full-featured General Purpose PLLs. The sysCLOCK PLLs provide the ability to synthesize clock frequencies.

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

CLKI is the reference frequency input to the PLL and its source can come from two different external CLK inputs or from internal routing. A non-glitchless 2-to-1 input multiplexor is provided to dynamically select between two different external reference clock sources. The CLKI input feeds into the input Clock Divider block.

CLKFB is the feedback signal to the PLL which can come from internal feedback path, routing or an external I/O pin. The feedback divider is used to multiply the reference frequency and thus synthesize a higher frequency clock output.

The PLL has four clock outputs CLKOP, CLKOS, CLKOS2 and CLKOS3. Each output has its own output divider, thus allowing the PLL to generate different frequencies for each output. The output dividers can have a value from 1 to 128. The CLKOP, CLKOS, CLKOS2, and CLKOS3 outputs can all be used to drive the primary clock network. Only CLKOP and CLKOS outputs can go to the edge clock network.

The setup and hold times of the device can be improved by programming a phase shift into the CLKOS, CLKOS2, and CLKOS3 output clocks which will advance or delay the output clock with reference to the CLKOP output clock. This phase shift can be either programmed during configuration or can be adjusted dynamically using the PHASE-SEL, PHASEDIR, PHASESTEP, and PHASELOADREG ports.

The LOCK signal is asserted when the PLL determines it has achieved lock and de-asserted if a loss of lock is detected.

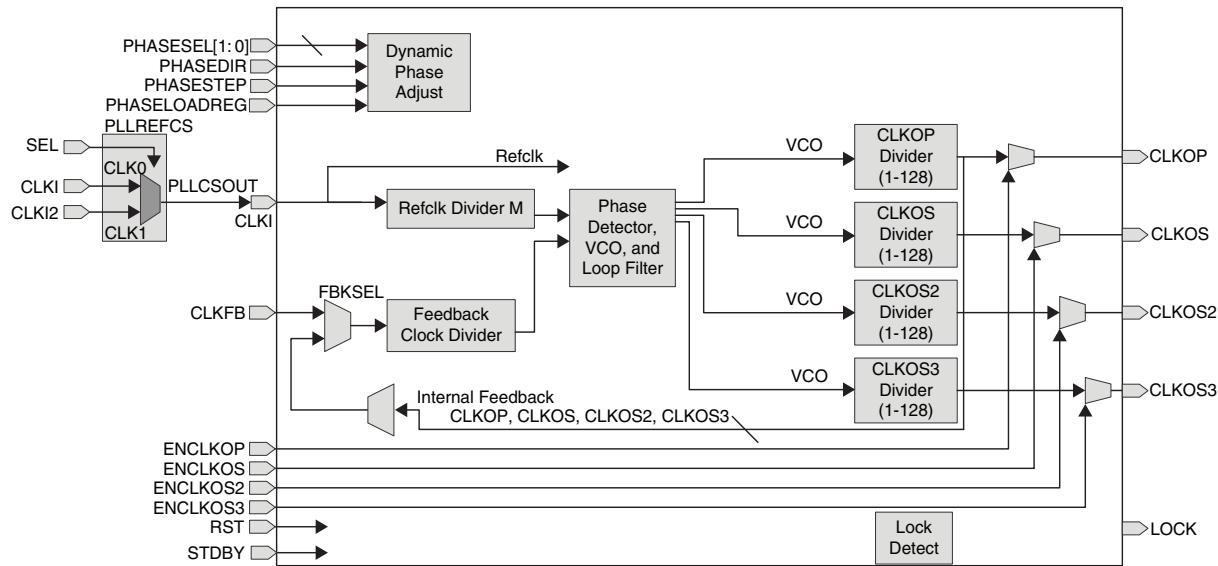
**Figure 2-4. General Purpose PLL Diagram**


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

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

Signal	I/O	Description
CLKI	I	Clock Input Clock to PLL from external pin or routing
CLKI2	I	Muxed clock input to PLL
SEL	I	Input Clock select
CLKFB	I	PLL Feedback Clock
PHASESEL[1:0]	I	Select the output affected by Dynamic Phase adjustment.
PHASEDIR	I	Dynamic Phase adjustment direction.
PHASESTEP	I	Dynamic Phase adjustment step.
PHASELOADREG	I	Load dynamic phase adjustment values into PLL.
CLKOP	O	Primary PLL output clock (with phase shift adjustment)
CLKOS	O	Secondary PLL output clock (with phase shift adjust)
CLKOS2	O	Secondary PLL output clock2 (with phase shift adjust)
CLKOS3	O	Secondary PLL output clock3 (with phase shift adjust)
LOCK	O	PLL LOCK to CLKI, Asynchronous signal. Active high indicates PLL lock
STDBY	I	Standby signal to power down the PLL
RST	I	Resets the PLL
ENCLKOP	I	Enable PLL output CLKOP
ENCLKOS	I	Enable PLL output CLKOS
ENCLKOS2	I	Enable PLL output CLKOS2
ENCLKOS3	I	Enable PLL output CLKOS3

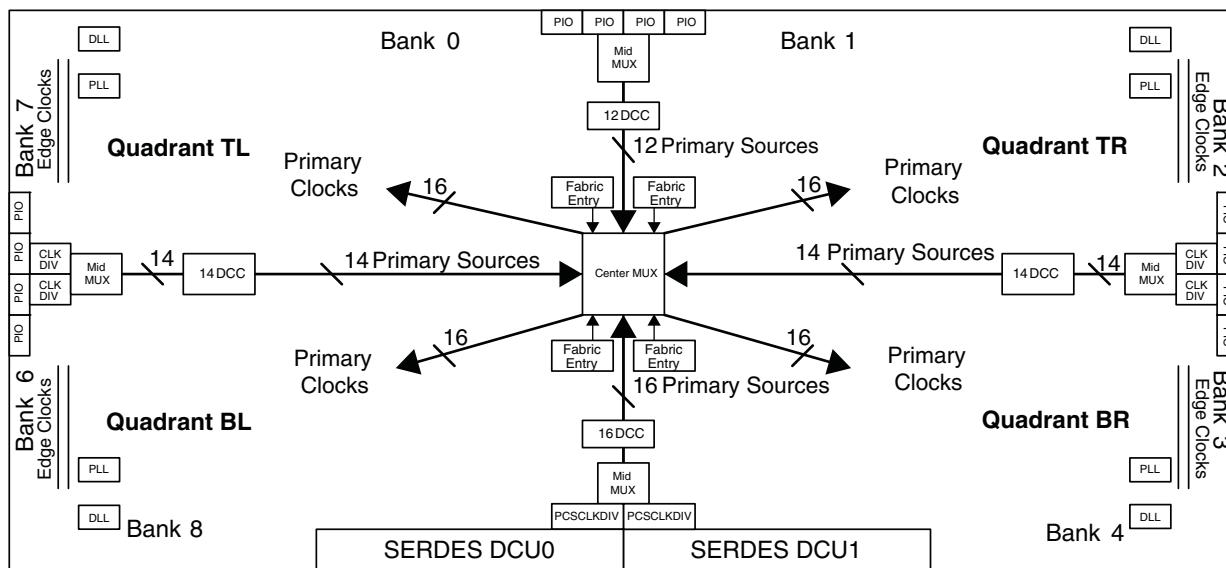
For more details on the PLL you can refer to the TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#).

## Clock Distribution Network

There are two main clock distribution networks for any member of the ECP5 product family, namely Primary Clock (PCLK) and Edge Clock (ECLK). These clock networks have the clock sources come from many different sources, such as Clock Pins, PLL outputs, DLLDEL outputs, Clock divider outputs, SERDES/PCS clocks and some on chip generated clock signal. There are clock dividers (CLKDIV) blocks to provide the slower clock from these clock sources. ECP5 also supports glitch-less dynamic enable function (DCC) for the PCLK Clock to save dynamic power. There are also some logics to allow dynamic glitch-less selection between two clocks for the PCLK network (DCS).

Overview of Clocking Network is shown in Figure 2-5, for LFE5UM-85 device.

**Figure 2-5. LFE5UM-85 Clocking**



## Primary Clocks

The ECP5 device family provides low-skew, high fanout clock distribution to all synchronous elements in the FPGA fabric through the Primary Clock Network.

The primary clock network is divided into four clocking quadrants: Top Left (TL), Bottom Left (BL), Top Right (TR), and Bottom Right (BR). Each of these quadrants has 16 clocks that can be distributed to the fabric in the quadrant.

The Lattice Diamond software can automatically route each clock to one of the 4 quadrants up to a maximum of 16 clocks per quadrant. The user can change how the clocks are routed by specifying a preference in the Lattice Diamond software to locate the clock to specific. The ECP5 device provides the user with a maximum of 64 unique clock input sources that can be routed to the primary Clock network.

Primary clock sources are:

- Dedicated clock input pins
- PLL outputs
- CLKDIV outputs
- Internal FPGA fabric entries (with minimum general routing)
- SERDES/PCS/PCSDIV clocks
- OSC clock

These sources are routed to one of four clock switches called a Mid Mux. The outputs of the Mid MUX are routed to the center of the FPGA where another clock switch, called the Center MUX, is used to route the primary clock sources to primary clock distribution to the ECP5 fabric. Since there is a maximum of 60 unique clock input sources to the clocking quadrants, there are potentially 64 unique clock domains that can be used in the ECP5 Device. For more information about the primary clock tree & connections, please see TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#).

## **Dynamic Clock Control (DCC)**

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

This DCC controls the clock sources from the Primary CLOCK MIDMUX before they are fed to the Primary Center MUXs that drive the quadrant clock network. For more information about the DCC, please see TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#).

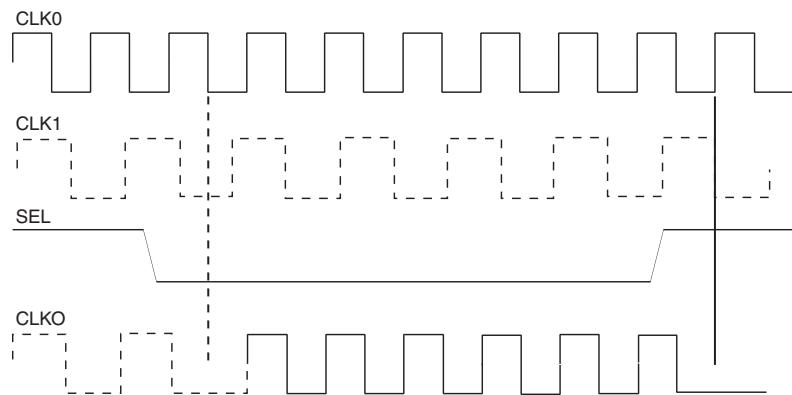
## **Dynamic Clock Select (DCS)**

The DCS is a smart multiplexer function available in the primary clock routing. It switches between two independent input clock sources. Depending on the operation modes, it switches between two (2) independent input clock sources either with or without any glitches. This is achieved regardless of when the select signal is toggled. Both input clocks must be running to achieve functioning glitch-less DCS output clock, but it is not required running clocks when used as non-glitch-less normal clock multiplexer.

There are two DCS blocks per device that are fed to all quadrants. The inputs to the DCS block come from all the output of MIDMUXs and Clock from CIB located at the center of the PLC array core. The output of the DCS is connected to one of the inputs of Primary Clock Center MUX.

See Figure 2-6 shows the timing waveforms of the default DCS operating mode. The DCS block can be programmed to other modes. For more information about the DCS, please see TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#).

**Figure 2-6. DCS Waveforms**

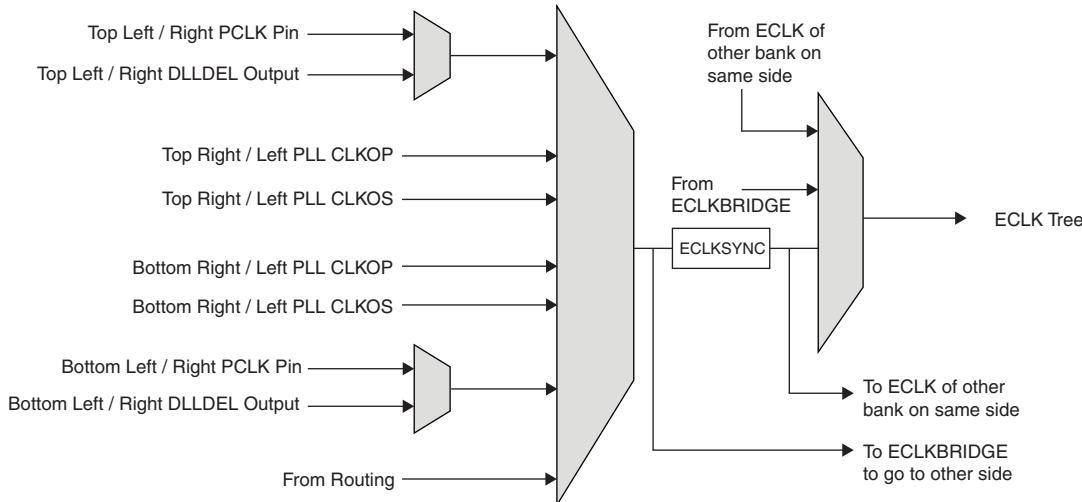


## **Edge Clock**

ECP5 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 two ECLK networks per bank IO on the Left and Right sides of the devices. Edge Clock (ECLK) consists of local DLLDEL delay cell, the ECLK input MUX, ECLK stop logic, ECLK Clock 2nd MUX, and the ECLK tree.

ECLK Input MUX collects all clock sources available shown in Figure 2-7 below. There are two ECLK Input MUXs, one on the left side and one on the right side. These MUXs are located on the edge close by the HIQ region. Each of these MUX will generate total of 4 ECLK Clock sources. Two of them drive the upper IO bank and two of them drive the lower IO bank. Two out of four also drive the ECLK Bridge Switch Block to form an ECLK Bridge high speed clock before drive the ECLK Tree Network.

**Figure 2-7. Edge Clock Sources Per Bank**



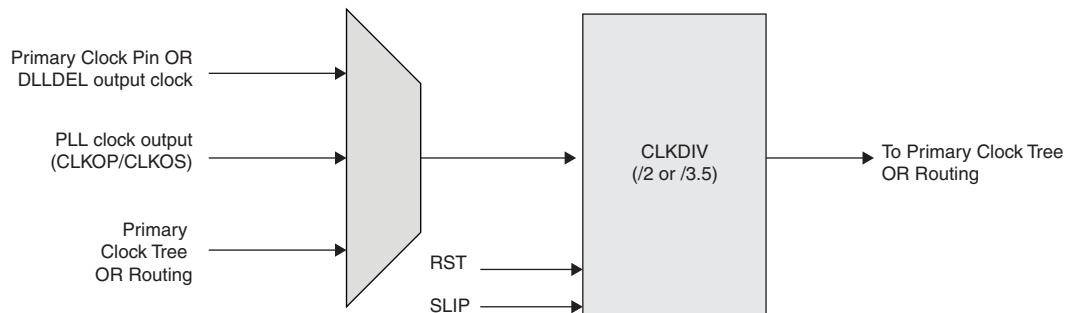
The edge clocks have low injection delay and low skew. They are used for DDR Memory or Generic DDR interfaces. For detailed information on Edge Clock connections, please see TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#).

## Clock Dividers

ECP5 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  $\div 2$ ,  $\div 3.5$  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 outputs, external primary clock pins multiplexed with the DDRDEL Slave Delay or from routing. The clock divider outputs serve as primary clock sources and feed into the clock distribution network. The Reset (RST) control signal resets input and asynchronously forces all outputs to low. The SLIP signal slips the outputs one cycle relative to the input clock. For further information on clock dividers, please see TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#). Figure 2-8 shows the clock divider connections.

**Figure 2-8. ECP5 Clock Divider Sources**



## DDRDLL

Every DDRDLL (master DLL block) can generate phase shift code representing the amount of delay in a delay block that corresponds to 90 degree phase of the reference clock input. The reference clock can be either from PLL, or input pin. This code is used in the DQSBUF block that controls a set of DQS pin groups to interface with DDR memory (slave DLL). There are 2 DDRDLLs that supply two sets of codes (2 different reference clocks) to each side of the I/Os (at each of the corners). The DQSBUF uses this code to control the DQS input of the DDR memory to 90 degree shift to clock DQs at the center of the data eye.

The code is also sent to another slave DLL, DLLDEL, that takes a clock input, and generates a 90 degree shift clock output to drive the clocking structure. This is useful to interface edge-aligned Generic DDR, where 90 degree clocking needs to be created. Figure 2-9 shows DDRDLL functional diagram.

**Figure 2-9. DDRDLL Functional Diagram**

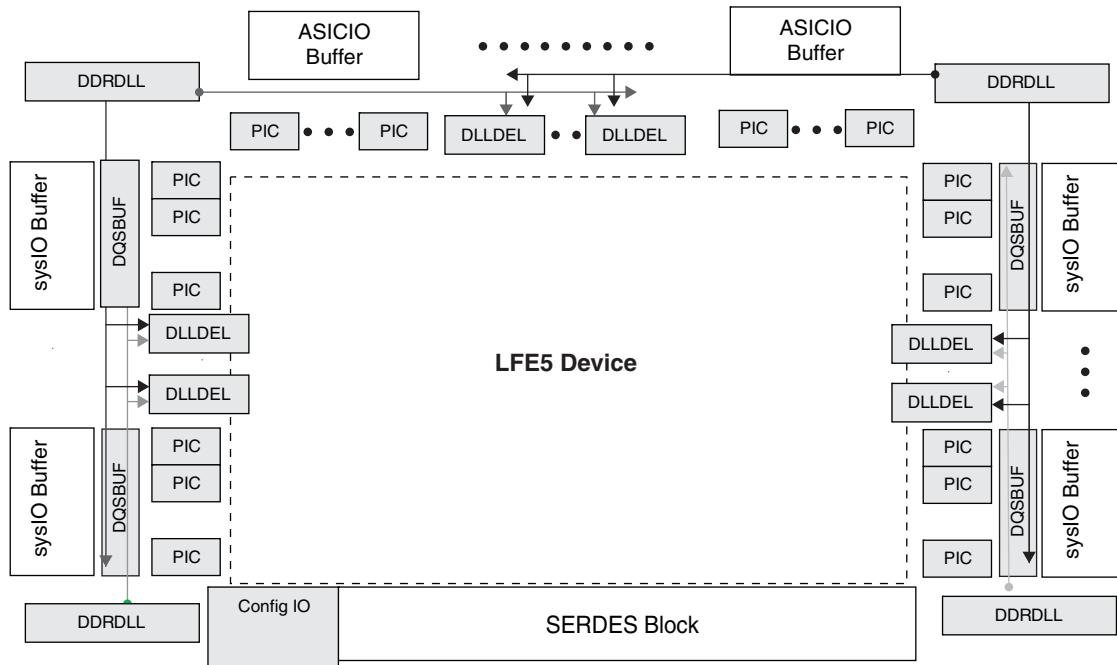


**Table 2-5. DDRDLL Ports List**

Port Name	I/O	Description
CLK	I	Reference clock input to the DDRDLL. Should run at the same frequency as the clock to the delayed.
RST	I	Reset Input to the DDRDLL
UDDCNTLN	I	Update Control to update the delay code. When low the delay code out the DDRDLL is updated. Should not be active during a read or a write cycle
FREEZE	I	FREEZE goes high and, without a glitch, turns off the DLL internal clock and the ring oscillator output clock. When FREEZE goes low, it turns them back on.
DDRDEL	O	The delay codes from the DDRDLL to be used in DQSBUF or DLLDEL
LOCK	O	Lock output to indicate the DDRDLL has valid delay output
DCNTL [7:0]	O	The delay codes from the DDRDLL available for the user IP.

There are 4 identical DDRDLL in the four corners in LFE5-85 and LFE5-45 devices, and two DDRDLLs in LFE5-25 devices. Each DDRDLL can generate delay code based on the reference frequency. The slave DLL (DQSBUF and DLLDEL) use the code to delay the signal, to create the phase shifted signal used for either DDR memory, or creating 90 degree shift clock. Figure 2-10 shows the DDRDLL and the slave DLLs on the top level view.

**Figure 2-10. ECP5 DLL Top Level View**



## sysMEM Memory

ECP5 devices contain a number of sysMEM Embedded Block RAM (EBR). The EBR consists of an 18-Kbit RAM with memory core, dedicated input registers and output registers with separate clock and clock enable. Each EBR includes functionality to support true dual-port, pseudo dual-port, single-port RAM, ROM and FIFO buffers (via external PFUs).

### sysMEM Memory Block

The sysMEM block can implement single port, dual port or pseudo dual port memories. Each block can be used in a variety of depths and widths as shown in Table 2-6. 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. For more information, please see TN1264, [ECP5 Memory Usage Guide](#).

**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 RAM can be created using EBR sysMEM Blocks. Typically, the Lattice design tools cascade memory transparently, based on specific design inputs.

## Single, Dual and Pseudo-Dual Port Modes

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

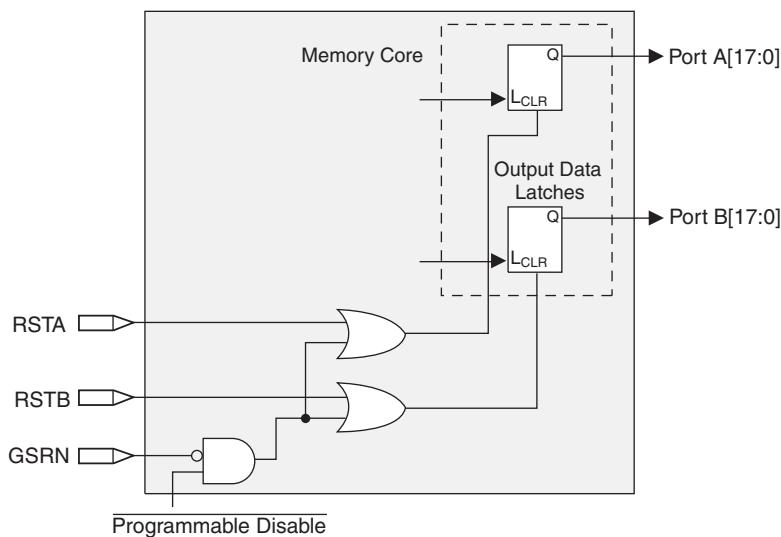
EBR memory supports the following forms of write behavior for single port or dual port operation:

- **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.
- **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.
- **Read-Before-Write** – When new data is written, the old content of the address appears at the output. This mode is supported for x9, x18, and x36 data widths.

## Memory Core Reset

The memory array in the EBR utilizes latches at the A and B output ports. These latches can be reset asynchronously or synchronously. RSTA and RSTB are local signals, which reset the output latches associated with Port A and Port B, respectively. The Global Reset (GSRN) signal can reset both ports. The output data latches and associated resets for both ports are as shown in Figure 2-11.

**Figure 2-11. Memory Core Reset**



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

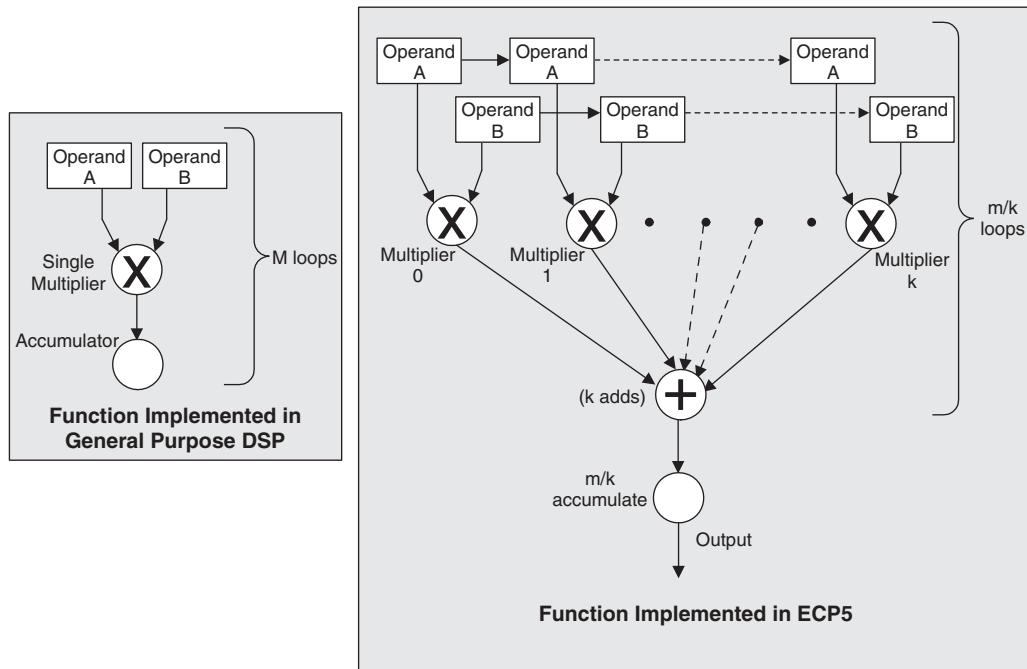
## sysDSP™ Slice

The ECP5 family provides an enhanced sysDSP architecture, making it ideally suited for low-cost, high-performance Digital Signal Processing (DSP) applications. Typical functions used in these applications are Finite Impulse Response (FIR) filters, Fast Fourier Transforms (FFT) functions, Correlators, Reed-Solomon/Turbo/Convolution encoders and decoders. These complex signal processing functions use similar building blocks such as multiply-adders and multiply-accumulators.

### sysDSP Slice Approach Compared to General DSP

Conventional general-purpose DSP chips typically contain one to four (Multiply and Accumulate) MAC units with fixed data-width multipliers; this leads to limited parallelism and limited throughput. Their throughput is increased by higher clock speeds. In the ECP5 device family, there are many DSP slices that can be used to support different data widths. This allows designers to use highly parallel implementations of DSP functions. Designers can optimize DSP performance vs. area by choosing appropriate levels of parallelism. Figure 2-12 compares the fully serial implementation to the mixed parallel and serial implementation.

**Figure 2-12. Comparison of General DSP and ECP5 Approaches**



## ECP5 sysDSP Slice Architecture Features

The ECP5 sysDSP Slice has been significantly enhanced to provide functions needed for advanced processing applications. These enhancements provide improved flexibility and resource utilization.

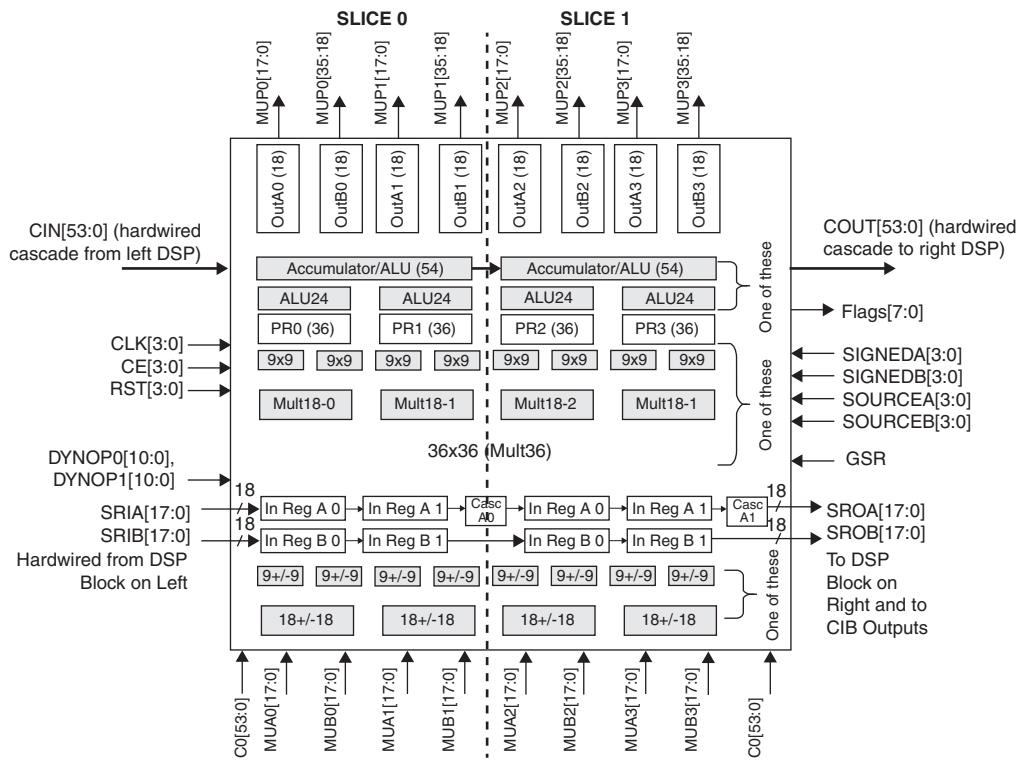
The ECP5 sysDSP Slice supports many functions that include the following:

- Fully double data rate support - Higher operation frequency (throughput of up to 370 Mbps) is achieved by double input and output interfaces that enable twice the fabric operation throughput for most of the operation modes.
- Symmetry support. The primary target application is wireless. 1D Symmetry is useful for many applications that use FIR filters when their coefficients have symmetry or asymmetry characteristics. The main motivation for using 1D symmetry is cost/size optimization. The expected size reduction is up to 2x.
  - Odd mode – Filter with Odd number of taps
  - Even mode – Filter with Even number of taps
  - Two dimensional (2D) symmetry mode – supports 2D filters for mainly video applications
- Dual-multiplier architecture. Lower accumulator overhead to half and the latency to half compared to single multiplier architecture
- Fully cascadable DSP across slices. Support for symmetric, asymmetric and non-symmetric filters.
- Multiply (one 18x36, two 18x18 or four 9x9 Multiplies per Slice)
- Multiply (36x36 by cascading across two sysDSP slices)
- Multiply Accumulate (supports one 18x36 multiplier result accumulation or two 18x18 multiplier result accumulation)
- Two Multiplies feeding one Accumulate per cycle for increased processing with lower latency (two 18x18 Multiplies feed into an accumulator that can accumulate up to 52 bits)
- Pipeline registers

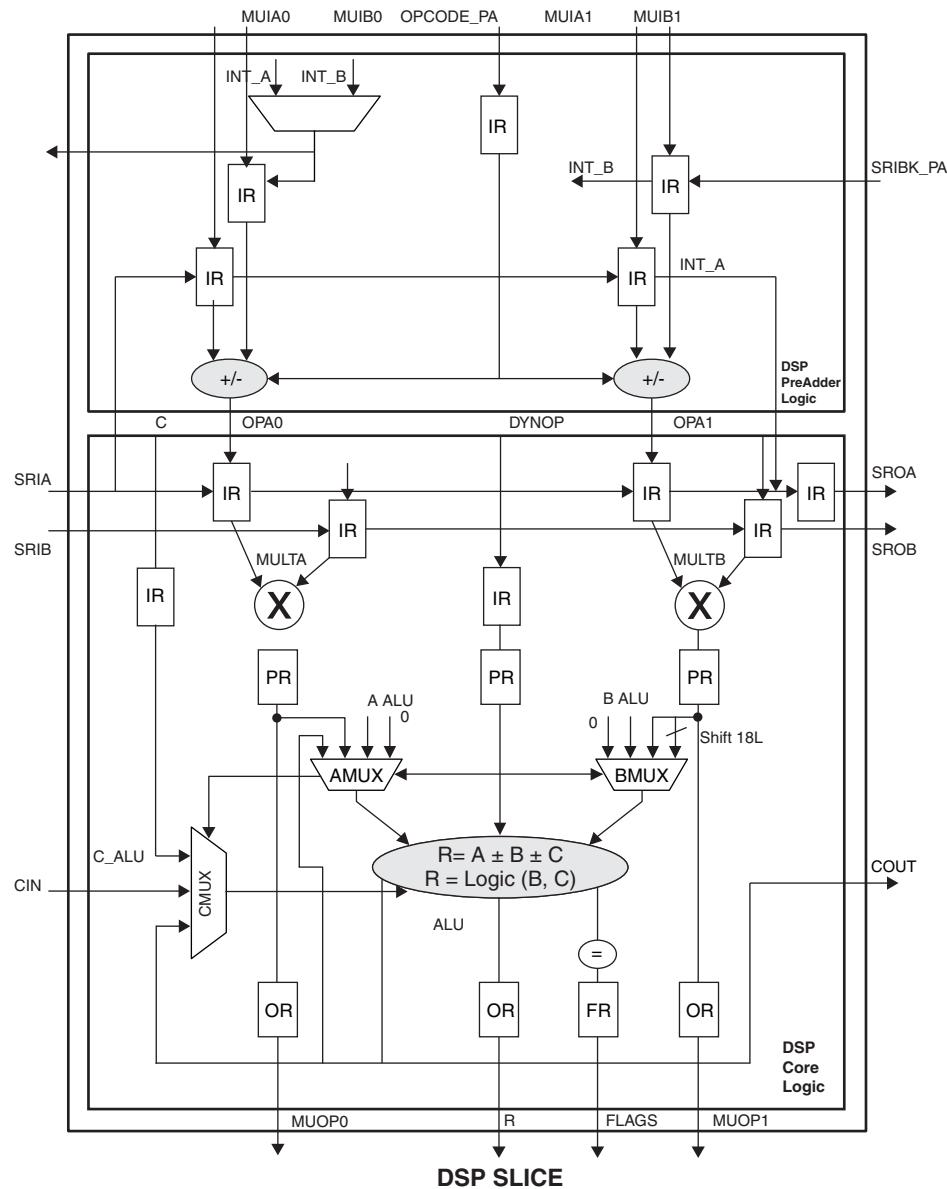
- 1D Symmetry support. The coefficients of FIR filters have symmetry or negative symmetry characteristics.
  - Odd mode – Filter with Odd number of taps
  - Even mode – Filter with Even number of taps
- 2D Symmetry support. The coefficients of 2D FIR filters have symmetry or negative symmetry characteristics.
  - 3\*3 and 3\*5 – Internal DSP Slice support
  - 5\*5 and larger size 2D blocks – Semi internal DSP Slice support
- Flexible saturation and rounding options to satisfy a diverse set of applications situations
- Flexible cascading across DSP slices
  - Minimizes fabric use for common DSP and ALU functions
  - Enables implementation of FIR Filter or similar structures using dedicated sysDSP slice resources only
  - Provides matching pipeline registers
  - Can be configured to continue cascading from one row of sysDSP slices to another for longer cascade chains
- Flexible and Powerful Arithmetic Logic Unit (ALU) Supports:
  - Dynamically selectable ALU OPCODE
  - Ternary arithmetic (addition/subtraction of three inputs)
  - Bit-wise two-input logic operations (AND, OR, NAND, NOR, XOR and XNOR)
  - Eight flexible and programmable ALU flags that can be used for multiple pattern detection scenarios, such as, overflow, underflow and convergent rounding, etc.
  - Flexible cascading across slices to get larger functions
- RTL Synthesis friendly synchronous reset on all registers, while still supporting asynchronous reset for legacy users
- Dynamic MUX selection to allow Time Division Multiplexing (TDM) of resources for applications that require processor-like flexibility that enables different functions for each clock cycle

For most cases, as shown in Figure 2-13, the ECP5 sysDSP slice is backwards-compatible with the LatticeECP2™ and LatticeECP3™ sysDSP block, such that, legacy applications can be targeted to the ECP5 sysDSP slice. Figure 2-13 shows the Diagram of the sysDSP, and Figure 2-14 shows the detailed diagram.

Figure 2-13. Simplified sysDSP Slice Block Diagram



**Figure 2-14. Detailed sysDSP Slice Diagram**



In Figure 2-14, note that A\_ALU, B\_ALU and C\_ALU are internal signals generated by combining bits from AA, AB, BA BB and C inputs. For further information, please refer to TN1267, [ECP5 sysDSP Usage Guide](#).

The ECP5 sysDSP block supports the following basic elements.

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

Table 2-7 shows the capabilities of each of the ECP5 slices versus the above functions.

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

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

1. One slice can implement 1/2 9x9 m9x9addssubsum and two m9x9addssubsum with two slices.

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

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

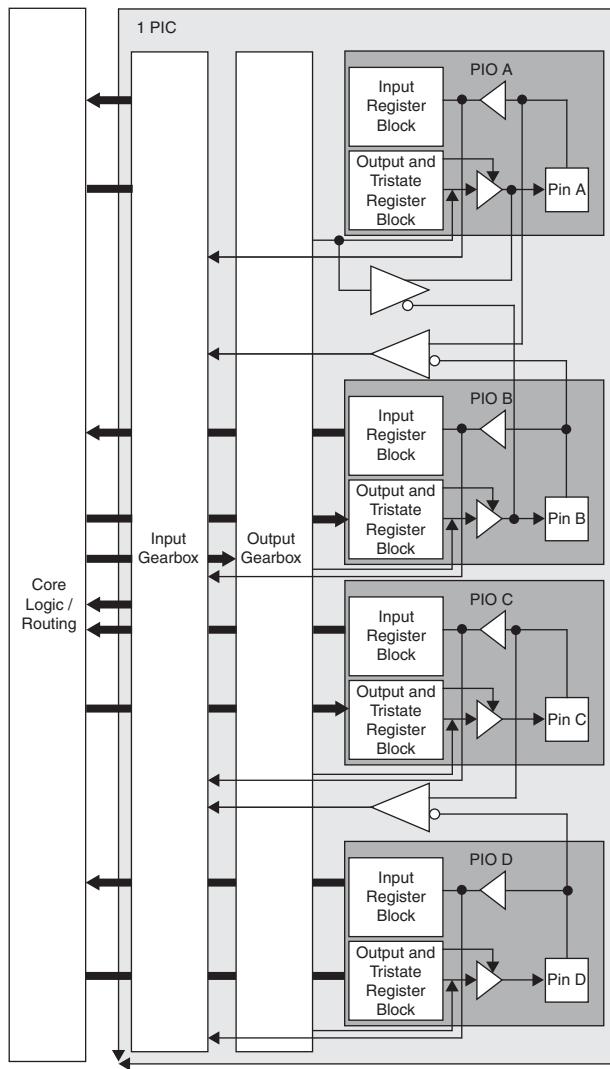
For further information, please refer to TN1267, [ECP5 sysDSP Usage Guide](#).

## Programmable I/O Cells (PIC)

The programmable logic associated with an I/O is called a PIO. The individual PIO are connected to their respective sysIO buffers and pads. On the ECP5 devices, the PIO cells are assembled into groups of four PIO cells called a Programmable I/O Cell or PIC. The PICs are placed on all four sides of the device.

On all the ECP5 devices, two adjacent PIOs can be combined to provide a complementary output driver pair. All PIO pairs can implement differential receivers. Half of the PIO pairs on the left and right edges of these devices can be configured as true LVDS transmit pairs.

**Figure 2-15. Group of Four Programmable I/O Cells on Left/Right Sides**



## PIO

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

### Input Register Block

The input register blocks for the PIOs on all edges contain delay elements and registers that can be used to condition high-speed interface signals before they are passed to the device core. In addition the input register blocks for the PIOs on the left and right edges include built-in FIFO logic to interface to DDR and LPDDR memory.

The Input register block on the right & left sides includes gearing logic and registers to implement IDDRX1, IDDRX2 functions. With two PICs sharing the DDR register path, it can also implement IDDRX71 function used for 7:1 LVDS interfaces. It uses three sets of registers -- shift, update, and transfer to implement gearing and the clock domain transfer. The first stage registers samples the high-speed input data by the high-speed edge clock on its rising and falling edges. The second stage registers perform data alignment based on the control signals. The third stage pipeline registers pass the data to the device core synchronized to the low-speed system clock. The Top side of the device will support IDDRX1 gearing function. For more information on gearing function, refer to TN1265, [ECP5 High-Speed I/O Interface](#).

Figure 2-16 shows the input register block for the PIOs on the top edge.

**Figure 2-16. Input Register Block for PIO on Top Side of the Device**

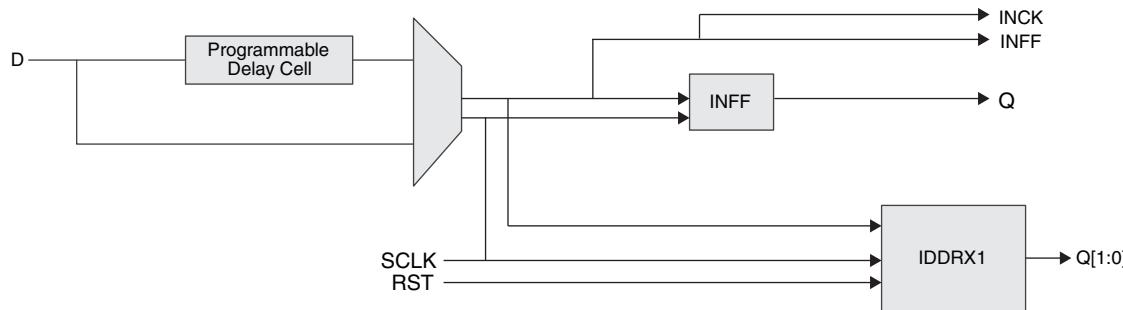
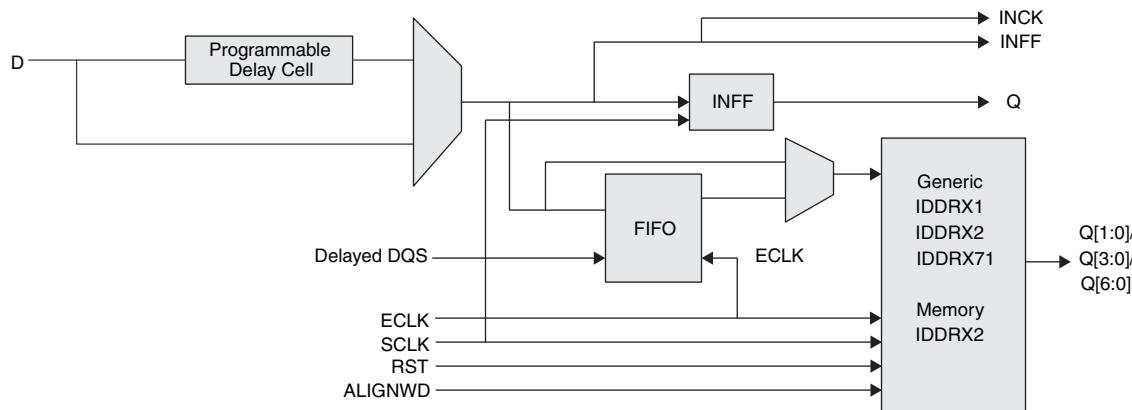


Figure 2-17 shows the input register block for the PIOs located on the left and right edges.

**Figure 2-17. Input Register Block for PIO on Left and Right Side of the Device**



### Input FIFO

The ECP5 PIO has dedicated input FIFO per single-ended pin for input data register for DDR Memory interfaces. The FIFO resides before the gearing logic. It transfers data from DQS domain to continuous ECLK domain. On the Write side of the FIFO, it is clocked by DQS clock which is the delayed version of the DQS Strobe signal from DDR memory. On the Read side of FIFO, it is clocked by ECLK. ECLK may be any high speed clock with identical frequency as DQS (the frequency of the memory chip). Each DQS group has one FIFO control block. It distributes FIFO read/write pointer to every PIC in same DQS group. DQS Grouping and DQS Control Block is described in DDR memory Support section below.

**Table 2-8. Input Block Port Description**

Name	Type	Description
D	Input	High Speed Data Input
Q[1:0]/Q[3:0]/Q[6:0]	Output	Low Speed Data to the device core
RST	Input	Reset to the Output Block
SCLK	Input	Slow Speed System Clock
ECLK	Input	High Speed Edge Clock
DQS	Input	Clock from DQS control Block used to clock DDR memory data
ALIGNWD	Input	Data Alignment signal from device core.

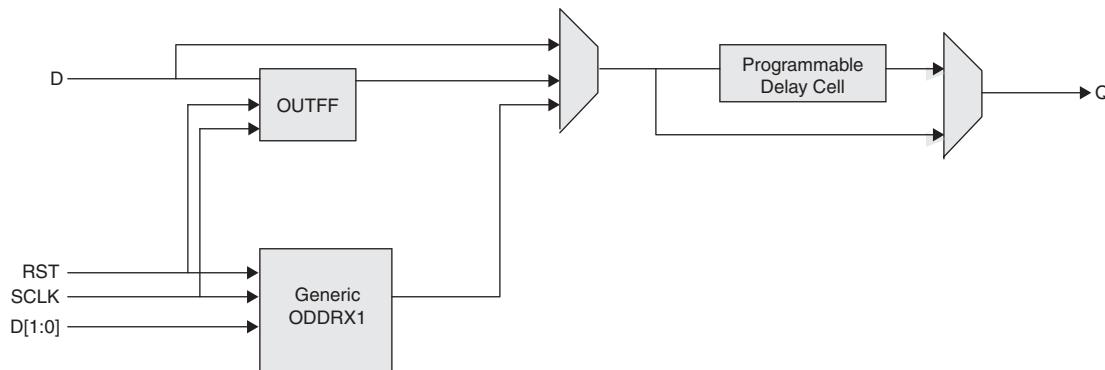
### Output Register Block

The output register block registers signals from the core of the device before they are passed to the sysIO buffers.

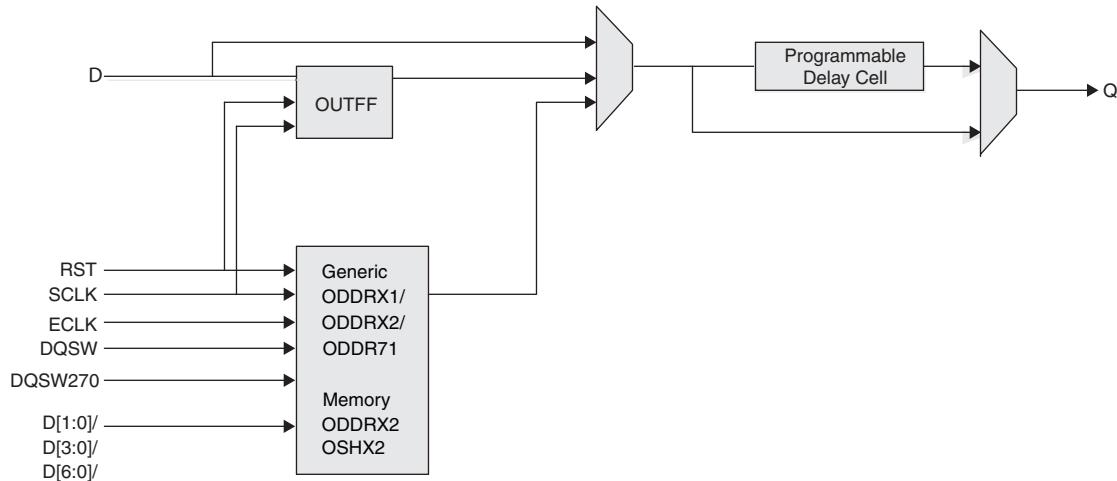
ECP5 output data path has output programmable flip flops and output gearing logic. On the left and right sides the output register block can support 1x, 2x and 7:1 gearing enabling high speed DDR interfaces and DDR memory interfaces. On the top side, the banks will support 1x gearing. ECP5 output data path diagram is shown in Figure 2-18. The programmable delay cells are also available in the output data path.

For detailed description of the output register block modes and usage, you can refer to TN1265, [ECP5 High-Speed I/O Interface](#).

**Figure 2-18. Output Register Block on Top Side**



**Figure 2-19. Output Register Block on Left and Right Sides**



**Table 2-9. Output Block Port Description**

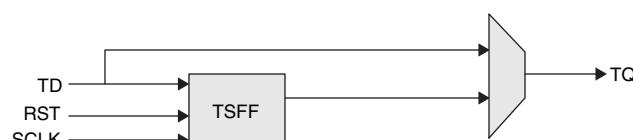
Name	Type	Description
Q	Output	High Speed Data Output
D	Input	Data from core to output SDR register
D[1:0]/D[3:0]/ D[6:0]	Input	Low Speed Data from device core to output DDR register
RST	Input	Reset to the Output Block
SCLK	Input	Slow Speed System Clock
ECLK	Input	High Speed Edge Clock
DQSW	Input	Clock from DQS control Block used to generate DDR memory DQS output
DQSW270	Input	Clock from DQS control Block used to generate DDR memory DQ output

## Tristate Register Block

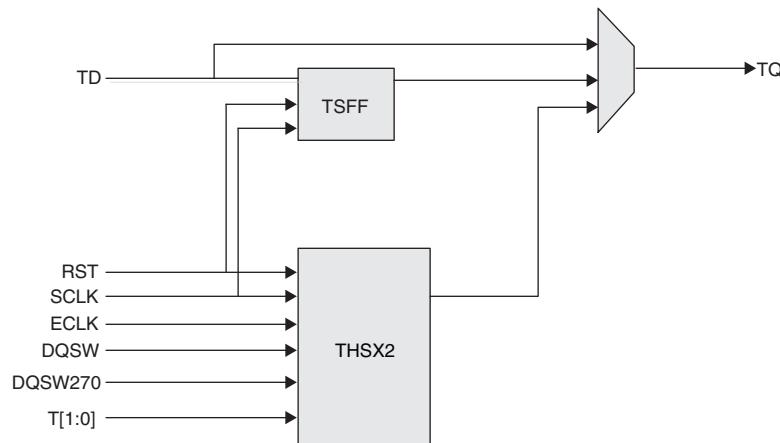
The tristate register block registers tristate control signals from the core of the device before they are passed to the sysIO buffers. The block contains a register for SDR operation. In SDR, TD input feeds one of the flip-flops that then feeds the output. In DDR operation used mainly for DDR memory interface can be implemented on the left and right sides of the device. Here 2 inputs feed the tristate registers clocked by both ECLK and SCLK.

Figure 2-19 and Figure 2-20 show the Tristate Register Block functions on the device. For detailed description of the tristate register block modes and usage, you can refer to TN1265, [ECP5 High-Speed I/O Interface](#).

**Figure 2-20. Tristate Register Block on Top Side**



**Figure 2-21. Tristate Register Block on Left and Right Sides**



**Table 2-10. Tristate Block Port Description**

Name	Type	Description
TD	Input	Tristate Input to Tristate SDR Register
RST	Input	Reset to the Tristate Block
TD[1:0]	Input	Tristate input to TSHX2 function
SCLK	Input	Slow Speed System Clock
ECLK	Input	High Speed Edge Clock
DQSW	Input	Clock from DQS control Block used to generate DDR memory DQS output
DQSW270	Input	Clock from DQS control Block used to generate DDR memory DQ output
TQ	Output	Output of the Tristate block

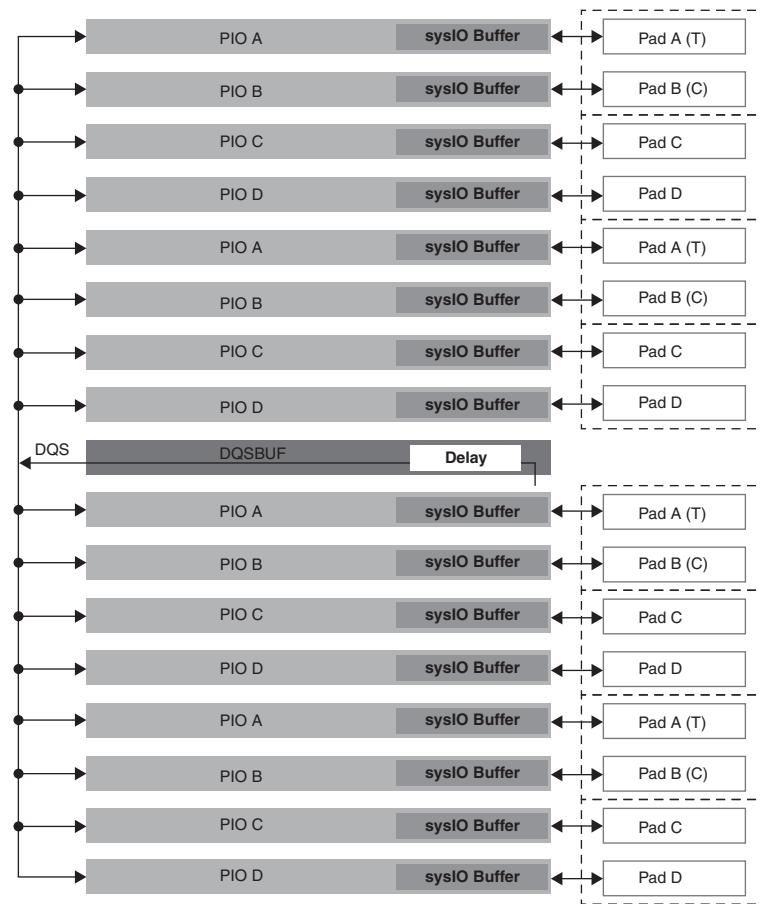
## DDR Memory Support

### DQS Grouping for DDR Memory

Certain PICs have additional circuitry to allow the implementation of high-speed source synchronous and DDR2, DDR3, LPDDR2 or LPDDR3 memory interfaces. The support varies by the edge of the device as detailed below.

The left and right sides of the PIC have fully functional elements supporting DDR2, DDR3, LPDDR2 or LPDDR3 memory interfaces. One of every 12 to 16 IOs supports the dedicated DQS pins with the DQS control logic block. Figure 2-22 shows the DQS bus spanning 16 I/O pins. Two of every 16 IOs support the dedicated DQS and DQS# pins with the DQS control logic block.

**Figure 2-22. DQS Grouping on the Left and Right Edges**



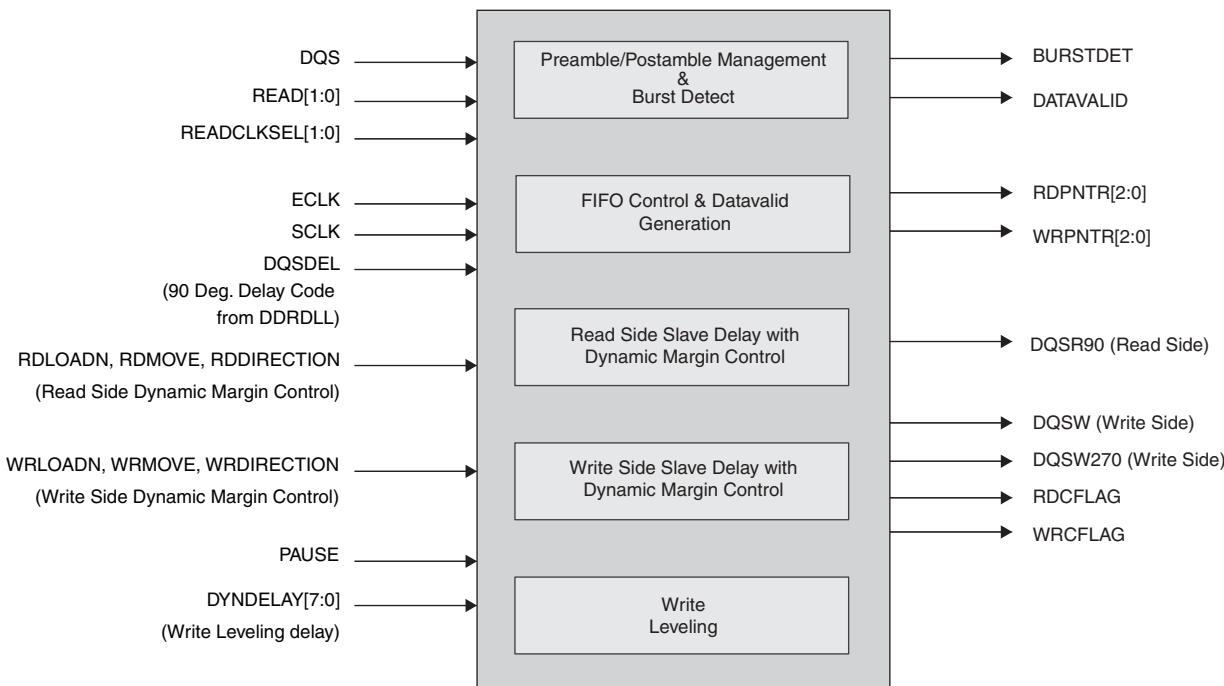
### DLL Calibrated DQS Delay and Control Block (DQSBUF)

To support DDR memory interfaces (DDR2/3, LPDDR2/3), the DQS strobe signal from the memory must be used to capture the data (DQ) in the PIC registers during memory reads. This signal is output from the DDR memory device aligned to data transitions and must be time shifted before it can be used to capture data in the PIC. This time shifted is achieved by using DQSDEL programmable delay line in the DQS Delay Block (DQS read circuit). The DQSDEL is implemented as a slave delay line and works in conjunction with a master DDRDLL.

This block also includes slave delay line to generate delayed clocks used in the write side to generate DQ and DQS with correct phases within one DQS group. There is a third delay line inside this block used to provide write leveling feature for DDR write if needed.

Each of the read and write side delays can be dynamically shifted using margin control signals that can be controlled by the core logic.

FIFO Control Block include here generates the Read and Write Pointers for the FIFO block inside the Input Register Block. These pointers are generated to control the DQS to ECLK domain crossing using the FIFO module.

**Figure 2-23. DQS Control & Delay Block (DQSBUF)**

**Table 2-11. DQSBUF Port list description**

Name	Type	Description
DQS	Input	DDR memory DQS strobe
READ[1:0]	Input	Read Input from DDR Controller
READCLKSEL[1:0]	Input	Read pulse selection
SCLK	Input	Slow System Clock
ECLK	Input	High Speed Edge Clock (same frequency as DDR memory)
DQSDEL	Input	90 Deg Delay Code from DDRDLL
RDLOADN, RDMOVE, RDDIRECTION	Input	Dynamic Margin Control ports for Read delay
WRLOADN, WRMOVE, WRDIRECTION	Input	Dynamic Margin Control ports for Write delay
PAUSE	Input	Used by DDR Controller to Pause write side signals during DDRDLL Code update or Write Leveling
DYNDELAY[7:0]	Input	Dynamic Write Leveling Delay Control
DQSR90	Output	90 delay DQS used for Read
DQSW270	Output	90 delay clock used for DQ Write
DQSW	Output	Clock used for DQS Write
RDPNTR[2:0]	Output	Read Pointer for IFIFO module
WRPNTR[2:0]	Output	Write Pointer for IFIFO module
DATAVALID	Output	Signal indicating start of valid data
BURSTDET	Output	Burst Detect indicator
RDFLAG	Output	Read Dynamic Margin Control output to indicate max value
WRFLAG	Output	Write Dynamic Margin Control output to indicate max value

## sysI/O Buffer

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

## sysI/O Buffer Banks

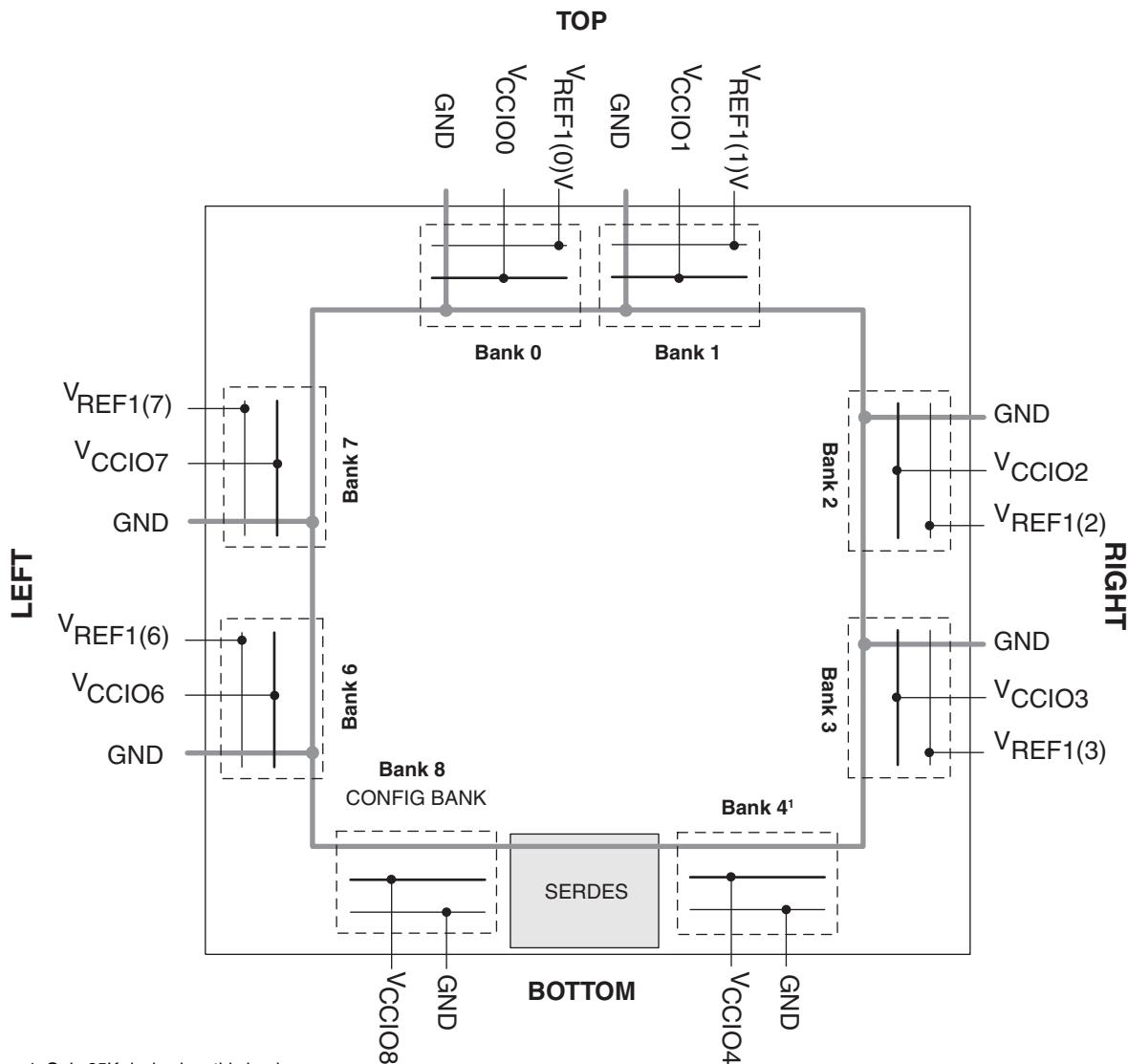
ECP5 devices have seven sysI/O buffer banks, two banks per side at Top, Left and Right, plus one at the bottom left side. The bottom left side bank (Bank 8) is a shared I/O bank. The I/Os in that bank contains both dedicated and shared I/O for sysConfig function. When a shared pin is not used for configuration, It is available as a user I/O. For LFE5-85 devices, there is an additional I/O bank (Bank 4) that is not available in other device in the family.

In ECP5 devices, the Left and Right sides are tailored to support high performance interfaces, such as DDR2, DDR3, LPDDR2, LPDDR3 and other high speed source synchronous standards. The banks on the Left and Right sides of the devices feature LVDS input and output buffers, data-width gearing, and DQSBUF block to support DDR2/3 and LPDDR2/3 interfaces. The I/Os on the top and bottom banks do not have LVDS input and output buffer, and gearing logic, but can use LVCMOS to emulate most of differential output signaling.

Each sysIO bank has its own I/O supply voltage ( $V_{CCIO}$ ). In addition, the banks on the Left and Right sides of the device, have voltage reference input (shared I/O pin), VREF1 per bank, which allow it to be completely independent of each other. The VREF voltage is used to set the threshold for the referenced input buffers, such as SSTL. Figure 2-24 shows the seven banks and their associated supplies.

In ECP5 devices, single-ended output buffers and ratioed input buffers (LVTTL, and LVCMOS) are powered using  $V_{CCIO}$ . LVTTL, LVCMOS33, LVCMOS25 and LVCMOS12 can also be set as fixed threshold inputs independent of  $V_{CCIO}$ .

Figure 2-24. ECP5 Device Family Banks



ECP5 devices contain two types of sysI/O buffer pairs.

- **Top (Bank 0 and Bank 1) and Bottom (Bank 8 and Bank 4) sysI/O Buffer Pairs (Single-Ended Only)**

The sysI/O buffers in the Banks at top and bottom of the device consist of ratioed single-ended output drivers and single-ended input buffers. The I/Os in these banks are not usually used as a pair, except when used as emulated differential output pair. They are used as individual I/Os and be configured as different I/O modes, as long as they are compatible with the Vccio voltage in the bank. When used as emulated differential outputs, the pair can be used together.

The top and bottom side IOs also support hot socketing. They support IO standards from 3.3 V to 1.2 V. They are ideal for general purpose I/Os, or as ADDR/CMD bus for DDR2/DDR3 applications, or for used as emulated differential signaling.

Bank 4 I/O only exists in the LFE5-85 device.

Bank 8 is a bottom bank that shares with sysConfig I/Os. During configuration, these I/Os are used for programming the device. Once the configuration is completed, these I/Os can be released and user can use these I/Os for functional signals in his design.

The top and bottom side pads can be identified by the Lattice Diamond tool.

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

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

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

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

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

The internal power-on-reset (POR) signal is deactivated when  $V_{CC}$ ,  $V_{CCIO8}$  and  $V_{CCAUX}$  have reached satisfactory levels. After the POR signal is deactivated, the FPGA core logic becomes active. It is the user's responsibility to ensure that all other  $V_{CCIO}$  banks are active with valid input logic levels to properly control the output logic states of all the I/O banks that are critical to the application. For more information about controlling the output logic state with valid input logic levels during power-up in ECP5 devices, see the list of technical documentation at the end of this data sheet.

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

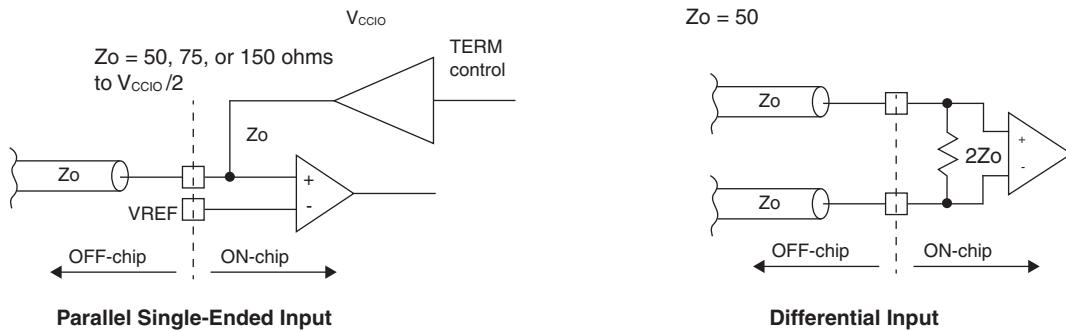
### Supported sysI/O Standards

The ECP5 sysI/O 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, 1.8 V, 2.5 V and 3.3 V standards. In the LVCMOS and LVTTL modes, the buffer has individual configuration options for drive strength, slew rates, bus maintenance (weak pull-up, weak pull-down, or a bus-keeper latch) and open drain. Other single-ended standards supported include SSTL and HSUL. Differential standards supported include LVDS, differential SSTL and differential HSUL. For further information on utilizing the sysI/O buffer to support a variety of standards, please see TN1262, [ECP5 sysIO Usage Guide](#).

### On-Chip Programmable Termination

The ECP5 devices support a variety of programmable on-chip terminations options, including:

- Dynamically switchable Single-Ended Termination with programmable resistor values of 50, 75, or 150 ohms.
- Common mode termination of 100 ohms for differential inputs

**Figure 2-25. On-Chip Termination**


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

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

IO_TYPE	TERMINATE to Vccio/21	DIFFERENTIAL TERMINATION RESISTOR1
LVDS25	-	100
BLVDS25	-	100
MLVDS	-	100
LVPECL33	-	100
subLVDS	-	100
SLVS	-	100
HSUL12	50, 75, 150	-
HSUL12D	-	100
SSTL135_I / II	50, 75, 150	-
SSTL135D_I / II	-	100
SSTL15_I / II	50, 75, 150	-
SSTL15D_I / II	-	100
SSTL18_I / II	50, 75, 150	-
SSTL18D_I / II	-	100

1. TERMINATE to VCCIO/2 (Single-Ended) and DIFFERENTIAL TERMINATION RESISTOR when turned on can only have one setting per bank. Only left and right banks have this feature.

Use of TERMINATE to VCCIO/2 and DIFFERENTIAL TERMINATION RESISTOR are mutually exclusive in an I/O bank.  
On-chip termination tolerance +/- 20%.

Please see TN1262, [ECP5 sysIO Usage Guide](#) for on-chip termination usage and value ranges.

## Hot Socketing

ECP5 devices have been carefully designed to ensure predictable behavior during power-up and power-down. During power-up and power-down sequences, the I/Os remain in tristate until the power supply voltage is high enough to ensure reliable operation. In addition, leakage into I/O pins is controlled within specified limits. Please refer to the Hot Socketing Specifications in the DC and Switching Characteristics in this data sheet.

## SERDES and PCS (Physical Coding Sublayer)

LFE5UM devices feature up to 4 channels of embedded SERDES/PCS arranged in dual-channel blocks at the bottom of the devices supporting up to 3.2 Gbps data rate. Figure 2-26 shows the position of the dual blocks for the LFE5-85. Table 2-13 shows the location of available SERDES Duals for all devices. The LFE5UM SERDES/PCS supports a range of popular serial protocols, including:

- PCI Express 2.0 (2.5 Gbps)

- Ethernet (XAUI, GbE - 1000 Base CS/SX/LX and SGMII)
- Serial RapidIO (SR and LR, Type 1: 1.25 Gbps, Type 2: 2.5 Gbps, Type 3: 3.125 Gbps)
- SMPTE SDI (3G, HD, SD)
- CPRI (E.6.LV: 614.4 Mbps, E.12.LV: 1228.8 Mbps, E.24.V: 2457.6 Mbps, E.30.LV: 3072 Mbps)
- JESD204A/B – ADC and DAC converter interface: 312.5 Mbps to 3.125 Gbps

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

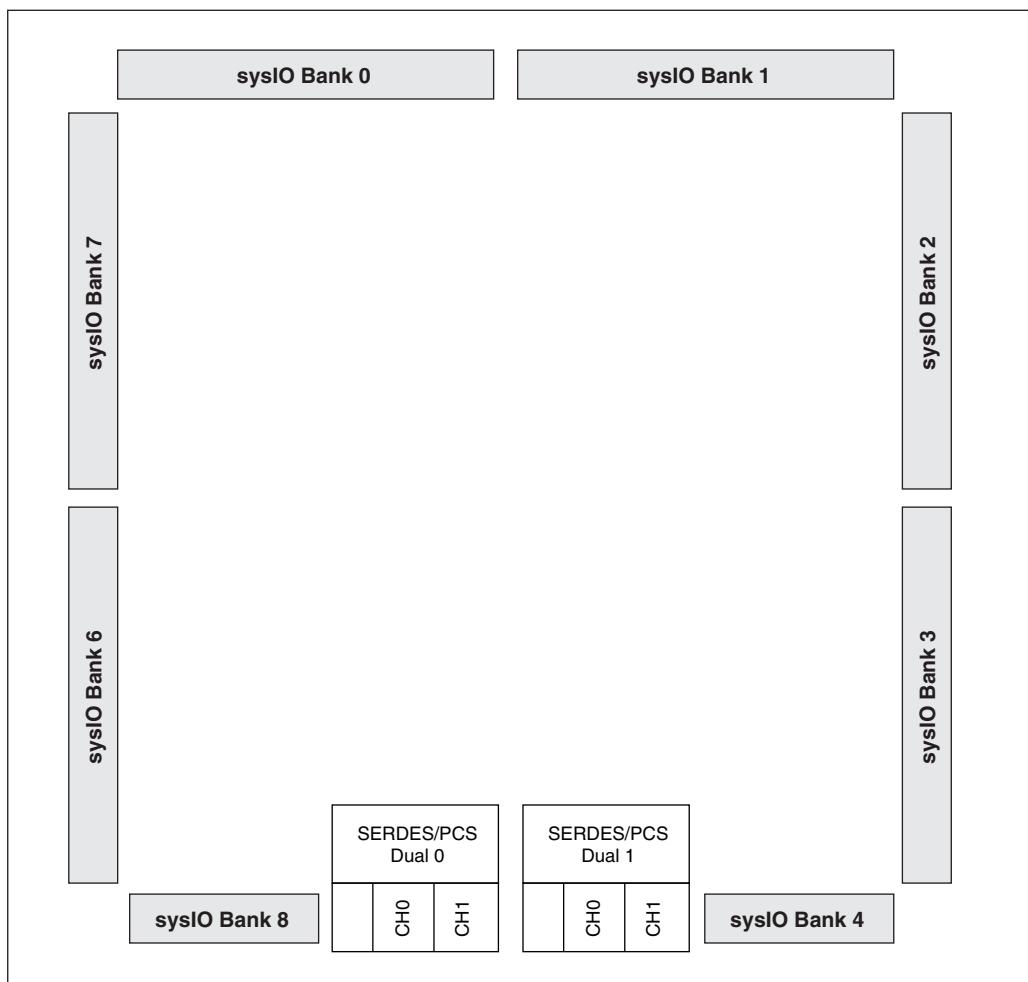
Even though the SERDES/PCS blocks are arranged in duals, multiple baud rates can be supported within a dual with the use of dedicated, per channel /1, /2 and /11 rate dividers. Additionally, two duals can be arranged together to form x4 channel link.

When a SERDES Dual in a 2-Dual device is not used, the power VCCA power supply for that Dual should be connected. It is advised to connect the VCCA of unused channel to core if the user knows he will not use the Dual at all, or it should be connected to a different regulated 1.1 V supply, if that Dual maybe used in the future.

For an unused channel in a Dual, it is advised to connect the VCCHTX to VCCA, and user can leave VCCHRX unconnected.

For information on how to use the SERDES/PCS blocks to support specific protocols, as well on how to combine multiple protocols and baud rates within a device, please refer to TN1261, [ECP5 SERDES/PCS Usage Guide](#).

Figure 2-26. SERDES/PCS Duals (LFE5UM-85)



**Table 2-13. LFE5UM SERDES Standard Support**

Standard	Data Rate (Mbps)	Number of General/Link Width	Encoding Style
PCI Express 1.1	2500	x1, x2, x4	8b10b
Gigabit Ethernet	1250, 2500	x1	8b10b
SGMII	1250	x1	8b10b
XAUI	3125	x4	8b10b
Serial RapidIO Type I	1250	x1, x4	8b10b
Serial RapidIO Type II	2500		
Serial RapidIO Type III	3125		
CPRI-1	614.4	x1	8b10b
CPRI-2	1228.8		
CPRI-3	2457.6		
CPRI-4	3072.0		
SD-SDI (259M, 344M) <sup>1</sup>	270, 360, 540	x1	NRZI/ Scrambled
HD-SDI (292M)	1483.5 1485	x1	NRZI/ Scrambled
3G-SDI (424M)	2967, 2970	x1	NRZI/ Scrambled
JESD204A/B	3125	x1	8b/10b
1. For slower rates, the SERDES are bypassed and CML signals are directly connected to the FPGA routing.			

**Table 2-14. Available SERDES Duals per LFE5UM Devices**

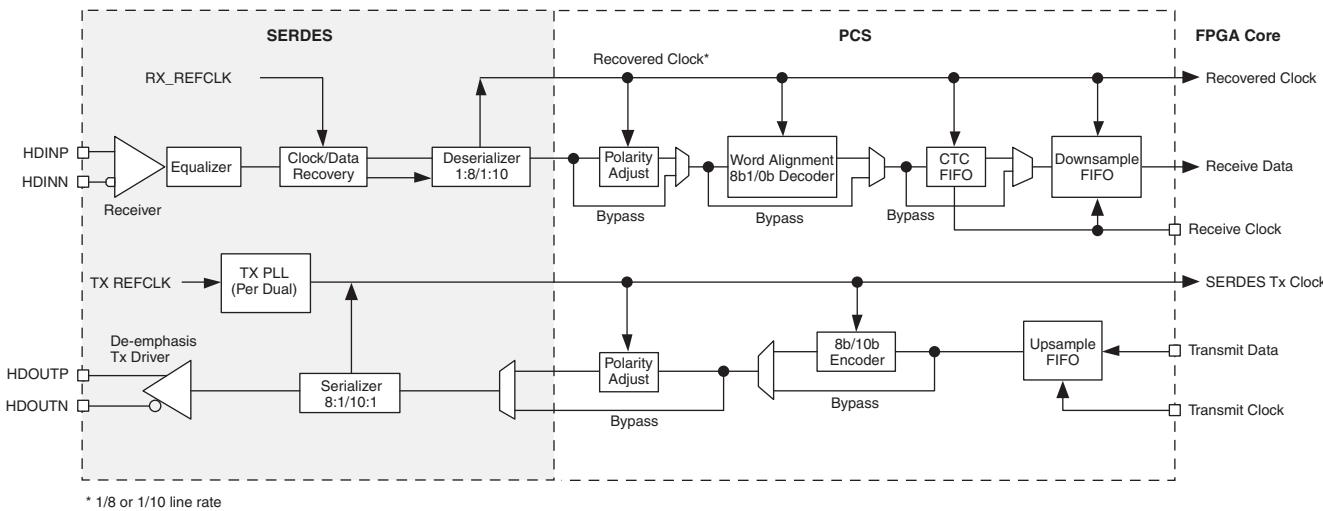
Package	LFE5UM-25	LFE5UM-45	LFE5UM-85
285 csfBGA	1	1	1
381 caBGA	1	2	2
554 caBGA	—	2	2
756 caBGA	—	—	2

## SERDES Block

A SERDES receiver channel may receive the serial differential data stream, equalize the signal, perform Clock and Data Recovery (CDR) and de-serialize the data stream before passing the 8- or 10-bit data to the PCS logic. The SERDES transmitter channel may receive the parallel 8- or 10-bit data, serialize the data and transmit the serial bit stream through the differential drivers. Figure 2-27 shows a single-channel SERDES/PCS block. Each SERDES channel provides a recovered clock and a SERDES transmit clock to the PCS block and to the FPGA core logic.

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

**Figure 2-27. Simplified Channel Block Diagram for SERDES/PCS Block**



## PCS

As shown in Figure 2-27, the PCS receives the parallel digital data from the deserializer and selects the polarity, performs word alignment, decodes (8b/10b), provides Clock Tolerance Compensation and transfers the clock domain from the recovered clock to the FPGA clock via the Down Sample FIFO.

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

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

Some of the enhancements in LFE5UM SERDES/PCS include:

- Higher clock/channel granularity: Dual channel architecture provides more clock resource per channel.
- Enhanced Tx de-emphasis: Programmable pre- and post- cursors improves Tx output signaling
- Bit-slip function in PCS: Improves logic needed to perform Word Alignment function

Please refer to TN1261, [ECP5 SERDES/PCS Usage Guide](#) for more information.

## SCI (SERDES Client Interface) Bus

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

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

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

The LFE5UM devices support a wide range of primary and secondary protocols. Within the same dual, the LFE5UM devices support mixed protocols with semi-independent clocking as long as the required clock frequen-

cies are integer  $x_1$ ,  $x_2$ , or  $x_{11}$  multiples of each other. Table 2-15 lists the allowable combination of primary and secondary protocol combinations.

## Flexible Dual SERDES Architecture

The LFE5UM SERDES architecture is a dual channel-based architecture. For most SERDES settings and standards, the whole dual (consisting of two SERDES channels) is treated as a unit. This helps in silicon area savings, better utilization, higher granularity on clock/SERDES channel and overall lower cost.

However, for some specific standards, the LFE5UM dual-channel architecture provides flexibility; more than one standard can be supported within the same dual.

Table 2-15 shows the standards that can be mixed and matched within the same dual. In general, the SERDES standards whose nominal data rates are either the same or a defined subset of each other, can be supported within the same dual. In Table 2-15, the two Protocol columns define the different combinations of protocols that can be implemented together within a Dual.

**Table 2-15. LFE5UM Mixed Protocol Support**

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

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

While the LFE5UM architecture will allow the mixing of a PCI Express channel and a Gigabit Ethernet, Serial RapidIO or SGMII channel within the same dual, using a PCI Express spread spectrum clocking as the transmit reference clock will cause a violation of the Gigabit Ethernet, Serial RapidIO and SGMII transmit jitter specifications.

For further information on SERDES, please see TN1261, [ECP5 SERDES/PCS Usage Guide](#).

## IEEE 1149.1-Compliant Boundary Scan Testability

All ECP5 devices have boundary scan cells that are accessed through an IEEE 1149.1 compliant Test Access Port (TAP). This allows functional testing of the circuit board on which the device is mounted through a serial scan path that can access all critical logic nodes. Internal registers are linked internally, allowing test data to be shifted in and loaded directly onto test nodes, or test data to be captured and shifted out for verification. The test access port consists of dedicated I/Os: TDI, TDO, TCK and TMS. The test access port has its own supply voltage  $V_{CCJ}$  and can operate with LVCMOS3.3, 2.5, 1.8, 1.5 and 1.2 standards.

For more information, please see TN1260, [ECP5 sysCONFIG Usage Guide](#).

## Device Configuration

All ECP5 devices contain two ports that can be used for device configuration. The Test Access Port (TAP), which supports bit-wide configuration, and the sysCONFIG port, support dual-byte, byte and serial configuration. The TAP supports both the IEEE Standard 1149.1 Boundary Scan specification and the IEEE Standard 1532 In- System Configuration specification. There are 11 dedicated pins for TAP and sysConfig supports (TDI, TDO, TCK, TMS, CFG[2:0], PROGRAMN, DONE, INITN and CCLK). The remaining sysCONFIG pins are used as dual function pins. See TN1260, [ECP5 sysCONFIG Usage Guide](#) for more information about using the dual-use pins as general purpose I/Os.

There are various ways to configure a ECP5 device:

- JTAG
- Standard Serial Peripheral Interface (SPI) - interface to boot PROM Support x1, x2, x4 wide SPI memory interfaces.
- System microprocessor to drive a x8 CPU port SPCM mode
- System microprocessor to drive a serial slave SPI port (SSPI mode)
- Slave Serial model (SCM)

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.

ECP5 devices also support the Slave SPI Interface. In this mode, the FPGA behaves like a SPI Flash device (slave mode) with the SPI port of the FPGA to perform read-write operations.

## Enhanced Configuration Options

ECP5 devices have enhanced configuration features such as: decryption support, decompression support, TransFR™ I/O and dual-boot and multi-boot image support.

### TransFR (Transparent Field Reconfiguration)

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

### Dual-Boot and Multi-Boot Image Support

Dual-boot and multi-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 ECP5 devices can be re-booted from this new configuration file. If there is a problem, such as corrupt data during download or incorrect version number with this new boot image, the ECP5 device can revert back to the original backup golden configuration and try again. This all can be done without power cycling the system. For more information, please see TN1260, [ECP5 sysCONFIG Usage Guide](#).

### Soft Error Detect (SED) Support

ECP5 devices have dedicated logic to perform Cycle Redundancy Code (CRC) checks. During configuration, the configuration data bitstream can be checked with the CRC logic block. In addition, the ECP5 device can also be programmed to utilize a Soft Error Detect (SED) mode that checks for soft errors in configuration SRAM. The SED operation can be run in the background during user mode. If a soft error occurs, during user mode (normal operation) the device can be programmed to generate an error signal.

For further information on SED support, please see TN1268, ECP5 Soft Error Detection (SED) Usage Guide.

## On-Chip Oscillator

Every ECP5 device has an internal CMOS oscillator which is used to derive a Master Clock (MCCLK) for configuration. The oscillator and the MCCLK run continuously and are available to user logic after configuration is completed. The software default value of the MCCLK is nominally 2.5MHz. Table 2-16 lists all the available MCCLK frequencies. When a different Master Clock is selected during the design process, the following sequence takes place:

1. Device powers up with a nominal Master Clock frequency of 2.4 MHz.
2. During configuration, users select a different master clock frequency.
3. The Master Clock frequency changes to the selected frequency once the clock configuration bits are received.
4. If the user does not select a master clock frequency, then the configuration bitstream defaults to the MCCLK frequency of 2.5MHz.

This internal 130 MHz +/- 15% CMOS oscillator is available to the user by routing it as an input clock to the clock tree. For further information on the use of this oscillator for configuration or user mode, please see TN1260, [ECP5 sysCONFIG Usage Guide](#).

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

MCCLK Frequency (MHz)
2.4
4.8
9.7
19.4
38.8
77.5
155

## Density Shifting

The ECP5 family is designed to ensure that different density devices in the same family and in the same package have the same pinout. Furthermore, the architecture ensures a high success rate when performing design migration from lower density devices to higher density devices. In many cases, it is also possible to shift a lower utilization design targeted for a high-density device to a lower density device. However, the exact details of the final resource utilization will impact the likelihood of success in each case. An example is that some user I/Os may become No Connects in smaller devices in the same package. Refer to the ECP5 Pin Migration Tables and Diamond software for specific restrictions and limitations.



# ECP5 Family Data Sheet

## DC and Switching Characteristics

March 2014

Advance Data Sheet DS1044

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

Supply Voltage V <sub>CC</sub> .....	-0.5 to 1.21V
Supply Voltage V <sub>CCAUX</sub> .....	-0.5 to 2.75V
Supply Voltage V <sub>CCIO</sub> .....	-0.5 to 3.63V
Input or I/O Transient Voltage Applied .....	-0.5 to 3.63V
SERDES RX/TX Buffer Supply Voltages V <sub>CCHRX</sub> , V <sub>CCHTX</sub> .....	-0.5 to 1.32V
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.

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

Symbol	Parameter	Min.	Max.	Units
V <sub>CC</sub> <sup>2</sup>	Core Supply Voltage	1.045	1.155	V
V <sub>CCAUX</sub> <sup>2, 4</sup>	Auxiliary Supply Voltage	2.375	2.625	V
V <sub>CCIO</sub> <sup>2, 3</sup>	I/O Driver Supply Voltage	1.14	3.465	V
V <sub>REF</sub> <sup>1</sup>	Input Reference Voltage	0.5	1.0	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 <sup>5</sup>				
V <sub>CCA</sub>	SERDES Analog Power Supply	1.045	1.155	V
V <sub>CCAUXA</sub>	SERDES Auxiliary Supply Voltage	2.374	2.625	V
V <sub>CCHRX</sub>	SERDES Input Buffer Power Supply	0.30 <sup>6</sup>	1.26	V
V <sub>CCHTX</sub>	SERDES Output Buffer Power Supply (1.2V)	1.14	1.26	V

1. For correct operation, all supplies except V<sub>REF</sub> must be held in their valid operation range. This is true independent of feature usage.
2. All supplies with same voltage, except V<sub>CCA</sub>, should be connected together.
3. See recommended voltages by I/O standard in subsequent table.
4. V<sub>CCAUX</sub> ramp rate must not exceed 30mV/μs during power-up when transitioning between 0 V and 3 V.
5. See TN1261, [ECP5 SERDES/PCS Usage Guide](#) for information on board considerations for SERDES power supplies.
6. If V<sub>CCHRX</sub> is used as DC bias for external AC coupling, HDin pins swing about V<sub>CCHRX</sub> must meet the HDin input voltage level requirement specified in the Rx section of the data sheet.

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

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
IDK_HS	Input or I/O Leakage Current for Top and Bottom Banks Only	$0 \leq V_{IN} \leq V_{IH}$ (Max.)	—	—	+/-1	mA
IDK	Input or I/O Leakage Current for Left and Right Banks Only	$0 \leq V_{IN} < V_{CCIO}$	—	—	+/-1	mA
		$V_{CCIO} \leq V_{IN} \leq V_{CCIO} + 0.5V$	—	18	—	mA

1.  $V_{CC}$ ,  $V_{CCAUX}$  and  $V_{CCIO}$  should rise/fall monotonically.

2.  $I_{DK}$  is additive to  $I_{PU}$ ,  $I_{PW}$  or  $I_{BH}$ .

3. LVCMOS and LVTTL only.

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

Description	Min.	Typ.	Max.	Units
Input current per SERDES I/O pin when device is powered down and inputs driven.	—	—	8	mA
Input current per HDIN pin when device power supply is off, inputs driven <sup>1, 2</sup>	-	-	15	mA
Current per HDIN pin when device power ramps up, input driven <sup>3</sup>	-	-	50	mA
Current per HDOUT pin when device power supply is off, outputs pulled up <sup>4</sup>	-	-	30	mA

1. Device is powered down with all supplies grounded, both HDINP and HDINN inputs driven by a CML driver with maximum allowed output ( $V_{CCTx} = 1.26V$ ), 8b/10b data, no external AC coupling.
2. Each P and N input must have less than the specified maximum input current during hot plug. For a device with 2 DCU, the total input current would be  $15\text{mA} * 4 \text{ channels} * 2 \text{ input pins per channel} = 120 \text{ mA}$ .
3. Device power supplies are ramping up ( $V_{CCA}$  and  $V_{CCAUX}$ ), both HDINP and HDINN inputs are driven by a CML driver with maximum allowed output ( $V_{CCHTX}=1.26V$ ), 8b/10b data, internal AC coupling.
4. Device is powered down with all supplies grounded. Both HDOUTP and HDOUN outputs are pulled up to 1.26 V by the far end receiver termination of  $50 \Omega$  single ended.

## ESD Performance

Please refer to the [ECP5 Product Family Qualification Summary](#) for complete qualification data, including ESD performance.

## DC Electrical Characteristics

### Over Recommended Operating Conditions

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
$I_{IL}, I_{IH}^{1,4}$	Input or I/O Low Leakage	$0 \leq V_{IN} \leq (V_{CCIO} - 0.2V)$	—	—	10	$\mu A$
$I_{IH}^{1,3}$	Input or I/O High Leakage	$(V_{CCIO} - 0.2V) < V_{IN} \leq 3.6V$	—	—	100	$\mu A$
$I_{PU}$	I/O Active Pull-up Current	$0 \leq V_{IN} \leq 0.7 V_{CCIO}$	-30	—	-150	$\mu A$
$I_{PD}$	I/O Active Pull-down Current	$V_{IL} (\text{MAX}) \leq V_{IN} \leq V_{CCIO}$	30	—	150	$\mu A$
C1	I/O Capacitance <sup>2</sup>	$V_{CCIO} = 3.3V, 2.5V, 1.8V, 1.5V, 1.2V,$ $V_{CC} = 1.2V, V_{IO} = 0 \text{ to } V_{IH} (\text{MAX})$	—	5	8	pf
C2	Dedicated Input Capacitance <sup>2</sup>	$V_{CCIO} = 3.3V, 2.5V, 1.8V, 1.5V, 1.2V,$ $V_{CC} = 1.2V, V_{IO} = 0 \text{ to } V_{IH} (\text{MAX})$	—	5	7	pf

1. Input or I/O leakage current is measured with the pin configured as an input or as an I/O with the output driver tristated. It is not measured with the output driver active. Bus maintenance circuits are disabled.
2.  $T_A = 25^\circ C$ ,  $f = 1.0\text{MHz}$ .
3. Applicable to general purpose I/Os in top and bottom banks.
4. When used as  $V_{REF}$  maximum leakage= 25 $\mu A$ .

## ECP5 Supply Current (Standby)<sup>1, 2, 3, 4, 5, 6</sup>

**Over Recommended Operating Conditions**

Symbol	Parameter	Device	Typical	Units
$I_{CC}$	Core Power Supply Current	LFE5-25	17.07	mA
		LFE5-45	25.59	mA
		LFE5-85	42.26	mA
$I_{CCAUX}$	Auxiliary Power Supply Current	LFE5-25	12.7	mA
		LFE5-45	17.72	mA
		LFE5-85	23.71	mA
$I_{CCIO}$	Bank Power Supply Current (Per Bank)	LFE5-25	0.48	mA
		LFE5-45	0.48	mA
		LFE5-85	0.48	mA
$I_{CCA}$	SERDES Power Supply Current (Per Dual)	LFE5-25	3.62	mA
		LFE5-45	3.62	mA
		LFE5-85	3.62	mA

1. For further information on supply current, please see the list of technical documentation at the end of this data sheet.
2. Assumes all outputs are tristated, all inputs are configured as LVCMOS and held at the  $V_{CCIO}$  or GND.
3. Frequency 0 Hz.
4. Pattern represents a “blank” configuration data file.
5.  $T_J = 85^\circ\text{C}$ , power supplies at nominal voltage.
6. To determine the ECP5 peak start-up current, please use the Power Calculator tool in the Lattice Diamond Design Software.

## SERDES Power Supply Requirements<sup>1, 2, 3</sup>

### Over Recommended Operating Conditions

Symbol	Description	Typ.	Max.	Units
<b>Standby (Power Down)</b>				
I <sub>CCA-SB</sub>	V <sub>CCA</sub> Power Supply Current (Per Channel)			mA
I <sub>CCHRX-SB</sub>	V <sub>CCHRX</sub> , Input Buffer Current (Per Channel)			mA
I <sub>CCHTX-SB</sub>	V <sub>CCHTX</sub> , Output Buffer Current (Per Channel)			mA
<b>Operating (Data Rate = 3.2 Gbps)</b>				
I <sub>CCA-OP</sub>	V <sub>CCA</sub> Power Supply Current (Per Channel)	63.47	-	mA
I <sub>CCHRX-OP</sub>	V <sub>CCHRX</sub> , Input Buffer Current (Per Channel)	0.4	-	mA
I <sub>CCHTX-OP</sub>	V <sub>CCHTX</sub> , Output Buffer Current (Per Channel)	17.75	-	mA
<b>Operating (Data Rate = 2.5 Gbps)</b>				
I <sub>CCA-OP</sub>	V <sub>CCA</sub> Power Supply Current (Per Channel)	53.71	-	mA
I <sub>CCHRX-OP</sub>	V <sub>CCHRX</sub> , Input Buffer Current (Per Channel)	0.45	-	mA
I <sub>CCHTX-OP</sub>	V <sub>CCHTX</sub> , Output Buffer Current (Per Channel)	15.715	-	mA
<b>Operating (Data Rate = 1.25 Gbps)</b>				
I <sub>CCA-OP</sub>	V <sub>CCA</sub> Power Supply Current (Per Channel)	34.185	-	mA
I <sub>CCHRX-OP</sub>	V <sub>CCHRX</sub> , Input Buffer Current (Per Channel)	0.4	-	mA
I <sub>CCHTX-OP</sub>	V <sub>CCHTX</sub> , Output Buffer Current (Per Channel)	11.595	-	mA
<b>Operating (Data Rate = 270 Mbps)</b>				
I <sub>CCA-OP</sub>	V <sub>CCA</sub> Power Supply Current (Per Channel)	18.875	-	mA
I <sub>CCHRX-OP</sub>	V <sub>CCHRX</sub> , Input Buffer Current (Per Channel)	0.4	-	mA
I <sub>CCHTX-OP</sub>	V <sub>CCHTX</sub> , Output Buffer Current (Per Channel)	8.36	-	mA

1. Rx Equalization enabled, Tx De-emphasis (pre-cursor and post-cursor) disabled
2. Per Channel current is calculated with both channels on in a Dual, and divide current by two. If only one channel is on current will be higher.
3. To calculate with Tx De-emphasis enabled, please refer to the Diamond Power Calculator tool.

## sysI/O Recommended Operating Conditions

Standard	$V_{CCIO}$			$V_{REF} (V)$		
	Min.	Typ.	Max.	Min.	Typ.	Max.
LVCMOS33 <sup>2</sup>	3.135	3.3	3.465	—	—	—
LVCMOS33D <sup>4</sup> Output	3.135	3.3	3.465	—	—	—
LVCMOS25 <sup>2</sup>	2.375	2.5	2.625	—	—	—
LVCMOS18	1.71	1.8	1.89	—	—	—
LVCMOS15	1.425	1.5	1.575	—	—	—
LVCMOS12 <sup>2</sup>	1.14	1.2	1.26	—	—	—
LVTTL33 <sup>2</sup>	3.135	3.3	3.465	—	—	—
SSTL15_I, II <sup>3</sup>	1.43	1.5	1.57	0.68	0.75	0.9
SSTL18_I, II <sup>2</sup>	1.71	1.8	1.89	0.833	0.9	0.969
SSTL135_1, II	1.28	1.35	1.42	0.6	0.675	0.75
HSUL12	1.14	1.2	1.26	0.588	0.6	0.612
MIPI D-PHY LP Input <sup>4</sup>	1.425	1.5	1.575	—	—	—
LVDS25 <sup>2,4</sup> Output	2.375	2.5	2.625	—	—	—
subLVS <sup>4</sup> (Input only)	—	—	—	—	—	—
SLVS <sup>4</sup> (Input only)	—	—	—	—	—	—
LVDS25E <sup>4</sup> Output	2.375	2.5	2.625	—	—	—
MLVDS <sup>1,4</sup> Output	2.375	2.5	2.625	—	—	—
LVPECL33 <sup>1,2,4</sup> Output	3.135	3.3	3.465	—	—	—
BLVDS25 <sup>1,2,4</sup> Output	2.375	2.5	2.625	—	—	—
HSULD12D <sup>3,4</sup>	1.43	1.5	1.57	—	—	—
SSTL135D_I, II <sup>3,4</sup>	1.28	1.35	1.42	—	—	—
SSTL15D_I, II <sup>3,M4</sup>	1.43	1.5	1.57	—	—	—
SSTL18D_I <sup>2,3,4</sup> , II <sup>2,3,4</sup>	1.71	1.8	1.89	—	—	—

1. Inputs on chip. Outputs are implemented with the addition of external resistors.

2. For input voltage compatibility, see TN1262, [ECP5 sysIO Usage Guide](#).

3. VREF is required when using Differential SSTL and HSUL to interface to DDR/LPDDR memories.

4. These differential inputs use LVDS input comparator, which uses VCCAUX power

5. All differential inputs and LVDS25 output are supported in the Left and Right banks only. Please refer to TN1262, [ECP5 sysIO Usage Guide](#) for details.

## sysI/O Single-Ended DC Electrical Characteristics<sup>1, 2</sup>

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)				
LVCMOS33	-0.3	0.8	2.0	3.465	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
LVCMOS25	-0.3	0.7	1.7	3.465	0.4	V <sub>CCIO</sub> - 0.4	12, 8, 4	-12, -8, -4
					0.2	V <sub>CCIO</sub> - 0.2	0.1	-0.1
LVCMOS18	-0.3	0.35 V <sub>CCIO</sub>	0.65 V <sub>CCIO</sub>	3.465	0.4	V <sub>CCIO</sub> - 0.4	12, 8, 4	-12, -8, -4
					0.2	V <sub>CCIO</sub> - 0.2	0.1	-0.1
LVCMOS15	-0.3	0.35 V <sub>CCIO</sub>	0.65 V <sub>CCIO</sub>	3.465	0.4	V <sub>CCIO</sub> - 0.4	8, 4	-8, -4
					0.2	V <sub>CCIO</sub> - 0.2	0.1	-0.1
LVCMOS12	-0.3	0.35 V <sub>CCIO</sub>	0.65 V <sub>CCIO</sub>	3.465	0.4	V <sub>CCIO</sub> - 0.4	8, 4	-8, -4
					0.2	V <sub>CCIO</sub> - 0.2	0.1	-0.1
LVTTL33	-0.3	0.8	2.0	3.465	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
SSTL18_I (DDR2 Memory)	-0.3	V <sub>REF</sub> - 0.125	V <sub>REF</sub> + 0.125	3.465	0.4	V <sub>CCIO</sub> - 0.4	6.7	-6.7
SSTL18_II	-0.3	V <sub>REF</sub> - 0.125	V <sub>REF</sub> + 0.125	3.465	0.28	V <sub>CCIO</sub> - 0.28	13.4	-13.4
SSTL15_I (DDR3 Memory)	-0.3	V <sub>REF</sub> - 0.1	V <sub>REF</sub> + 0.1	3.465	0.3	V <sub>CCIO</sub> - 0.31	7.5	-7.5
SSTL15_II (DDR3L Memory)	-0.3	V <sub>REF</sub> - 0.1	V <sub>REF</sub> + 0.1	3.465	0.3	V <sub>CCIO</sub> - 0.31	8.8	-8.8
SSTL135_I	-0.3	V <sub>REF</sub> - 0.1	V <sub>REF</sub> + 0.1	3.465	0.4	V <sub>CCIO</sub> - 0.27	7	-7
SSTL15_II	-0.3	V <sub>REF</sub> - 0.09	V <sub>REF</sub> + 0.09	3.465	0.4	V <sub>CCIO</sub> - 0.27	8	-8
MIPI D-PHY (LP)	-0.3	V <sub>REF</sub> - 0.55	V <sub>REF</sub> + 0.98	3.465	—	—	—	—
HSUL12 (LPDDR2/3 Memory)	-0.3	V <sub>REF</sub> - 0.2	V <sub>REF</sub> + 0.2	3.465	0.4	V <sub>CCIO</sub> - 0.3	18, 4	-8, -4

1. For electromigration, the average DC current drawn by the I/O pads within a bank of I/Os shall not exceed 10mA per I/O (All I/Os used in the same V<sub>CCIO</sub>).

2. Not all IO types are supported in all banks. Please refer to TN1262, [ECP5 sysIO Usage Guide](#) for details.

## sysI/O Differential Electrical Characteristics

### LVDS

#### Over Recommended Operating Conditions

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

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

### SSTLD

All differential 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.

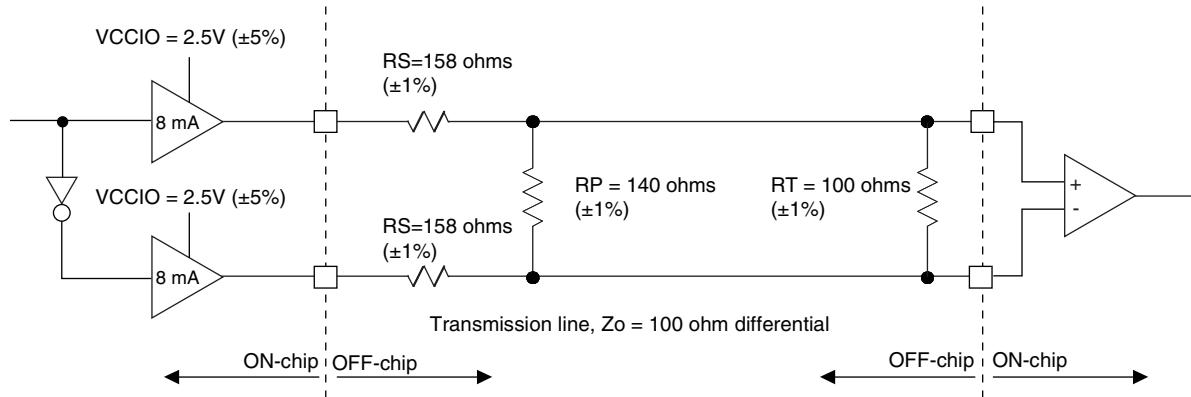
### LVC MOS33D

All I/O banks support emulated differential I/O using the LVC MOS33D I/O type. This option, along with the external resistor network, provides the system designer the flexibility to place differential outputs on an I/O bank with 3.3 V  $V_{CCIO}$ . The default drive current for LVC MOS33D output is 12mA with the option to change the device strength to 4mA, 8mA, 12mA or 16mA. Follow the LVC MOS33 specifications for the DC characteristics of the LVC MOS33D.

## LVDS25E

The top and bottom sides of ECP5 devices support LVDS outputs via emulated complementary LVCMOS outputs in conjunction with a parallel resistor across the driver outputs. The scheme shown in Figure 3-1 is one possible solution for point-to-point signals.

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



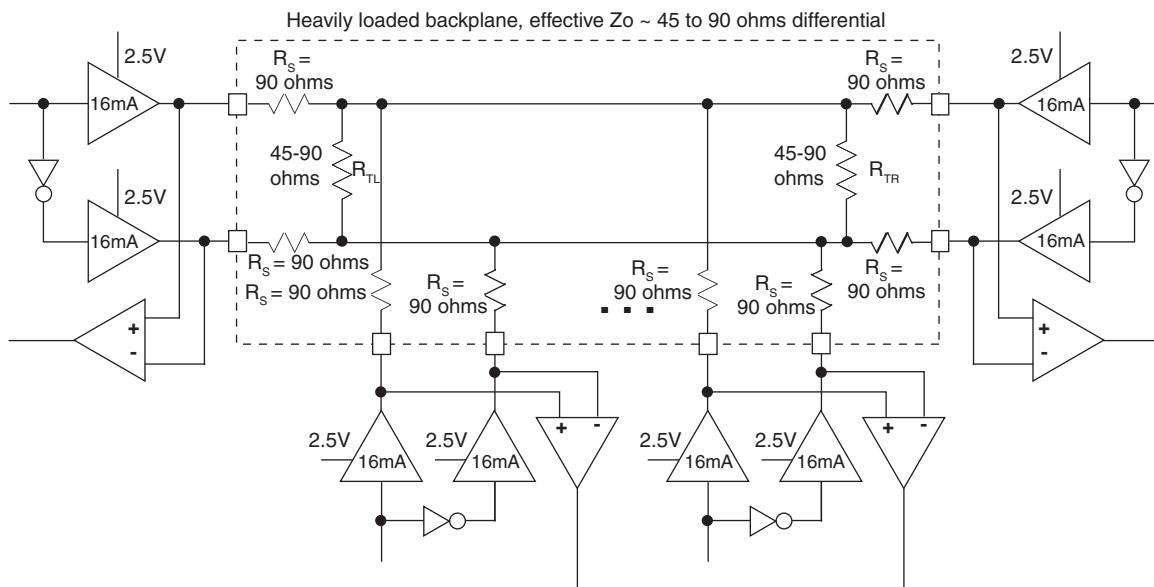
**Table 3-1. LVDS25E DC Conditions**

Parameter	Description	Typical	Units
$V_{CCIO}$	Output Driver Supply (+/-5%)	2.50	V
$Z_{OUT}$	Driver Impedance	20	$\Omega$
$R_S$	Driver Series Resistor (+/-1%)	158	$\Omega$
$R_P$	Driver Parallel Resistor (+/-1%)	140	$\Omega$
$R_T$	Receiver Termination (+/-1%)	100	$\Omega$
$V_{OH}$	Output High Voltage	1.43	V
$V_{OL}$	Output Low Voltage	1.07	V
$V_{OD}$	Output Differential Voltage	0.35	V
$V_{CM}$	Output Common Mode Voltage	1.25	V
$Z_{BACK}$	Back Impedance	100.5	$\Omega$
$I_{DC}$	DC Output Current	6.03	mA

## BLVDS25

The ECP5 devices support the BLVDS standard. This standard is emulated using complementary LVCMOS outputs in conjunction with a parallel external resistor across the driver outputs. BLVDS is intended for use when multi-drop and bi-directional multi-point differential signaling is required. The scheme shown in Figure 3-2 is one possible solution for bi-directional multi-point differential signals.

**Figure 3-2. BLVDS25 Multi-point Output Example**



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

### Over Recommended Operating Conditions

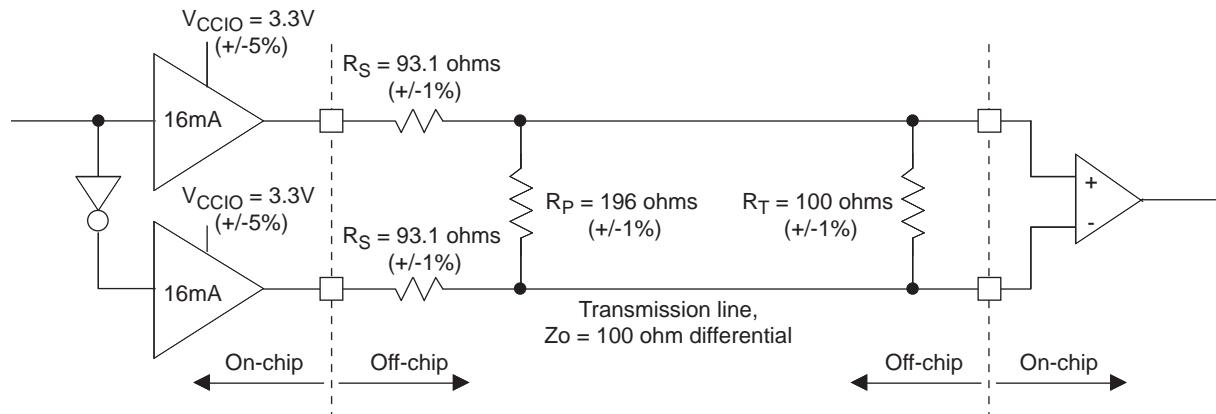
Parameter	Description	Typical		Units
		Zo = 45Ω	Zo = 90Ω	
V <sub>CCIO</sub>	Output Driver Supply (+/- 5%)	2.50	2.50	V
Z <sub>OUT</sub>	Driver Impedance	10.00	10.00	Ω
R <sub>S</sub>	Driver Series Resistor (+/- 1%)	90.00	90.00	Ω
R <sub>TL</sub>	Driver Parallel Resistor (+/- 1%)	45.00	90.00	Ω
R <sub>TR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

1. For input buffer, see LVDS table.

### LVPECL33

The ECP5 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 LVPECL33**



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

Over Recommended Operating Conditions

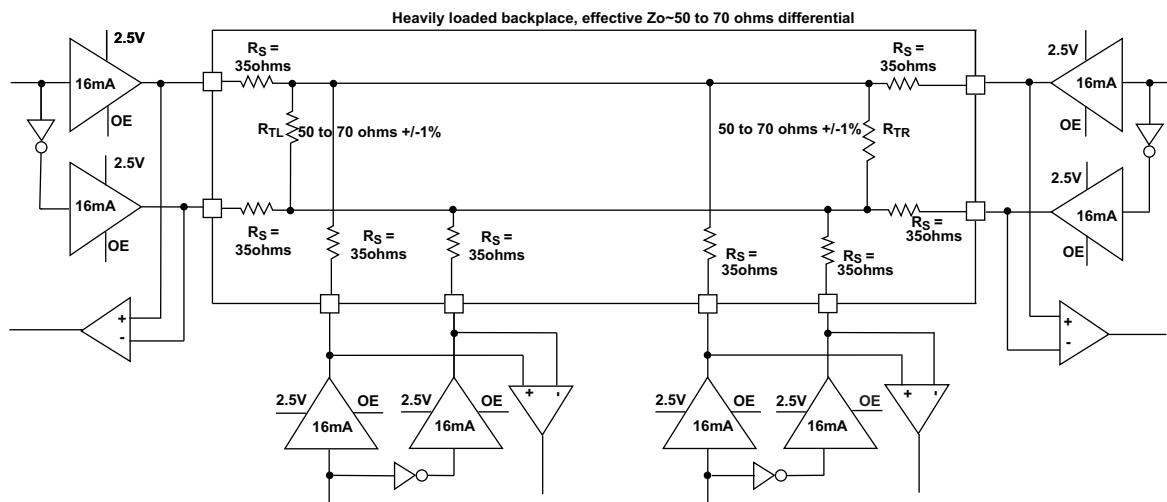
Parameter	Description	Typical	Units
V <sub>CCIO</sub>	Output Driver Supply (+/-5%)	3.30	V
Z <sub>OUT</sub>	Driver Impedance	10	Ω
R <sub>S</sub>	Driver Series Resistor (+/-1%)	93	Ω
R <sub>P</sub>	Driver Parallel Resistor (+/-1%)	196	Ω
R <sub>T</sub>	Receiver Termination (+/-1%)	100	Ω
V <sub>OH</sub>	Output High Voltage	2.05	V
V <sub>OL</sub>	Output Low Voltage	1.25	V
V <sub>OD</sub>	Output Differential Voltage	0.80	V
V <sub>CM</sub>	Output Common Mode Voltage	1.65	V
Z <sub>BACK</sub>	Back Impedance	100.5	Ω
I <sub>DC</sub>	DC Output Current	12.11	mA

1. For input buffer, see LVDS table.

## MLVDS25

The ECP5 devices support the differential MLVDS standard. This standard is emulated using complementary LVC-MOS 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 3-4 is one possible solution for MLVDS standard implementation. Resistor values in Figure 3-4 are industry standard values for 1% resistors.

**Figure 3-4. MLVDS25 (Multipoint Low Voltage Differential Signaling)**



**Table 3-4. MLVDS25 DC Conditions<sup>1</sup>**

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

1. For input buffer, see LVDS table.

## SLVS

Scalable Low-Voltage Signaling (SLVS) is based on a point-to-point signaling method defined in the JEDEC JESD8-13 (SLVS-400) standard. This standard evolved from the traditional LVDS standard and relies on the advantage of its use of smaller voltage swings and a lower common-mode voltage. The 200 mV (400 mV p-p) SLVS swing contributes to a reduction in power.

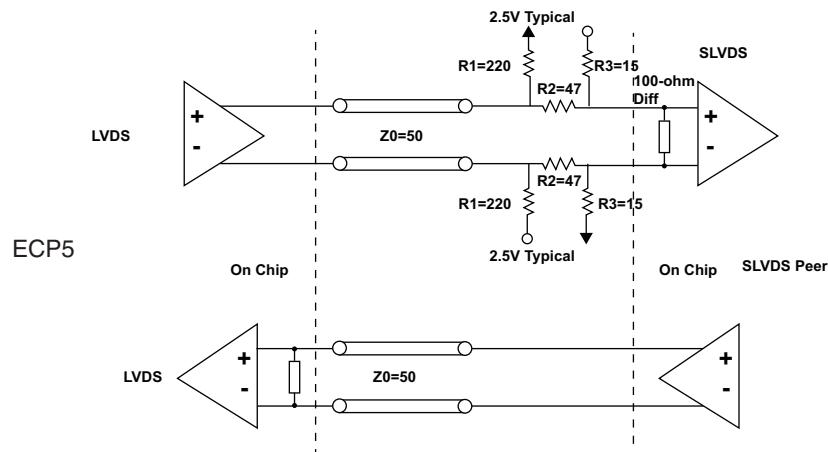
The ECP5 devices can receive differential input up to 800 Mbps with its LVDS input buffer. This LVDS input buffer is used to meet the SLVS input standard specified by the JEDEC standard. The SLVS output parameters are compared to ECP5 LVDS input parameters, as listed in Table 3-5:

**Table 3-5. Input to SLVS**

Parameter	ECP5 LVDS Input	SLVS Output	Units
Vcm (min)	50	150	mV
Vcm (max)	2350	250	mV
Differential Voltage (min)	100	140	mV
Differential Voltage (max)	-	270	mV

ECP5 does not support SLVS output. However, SLVS output can be created using ECP5 LVDS outputs by level shift to meet the low Vcm/Vod levels required by SLVS. Figure 3-5 shows how the LVDS output can be shifted external to meet SLVS levels.

**Figure 3-5. SLVS Interface**



## ECP5 External Switching Characteristics<sup>1,2</sup>

Over Recommended Commercial Operating Conditions

Parameter	Description	Device	-8		-7		-6		Units			
			Min.	Max.	Min.	Max.	Min.	Max.				
<b>Clocks</b>												
<b>Primary Clock<sup>6</sup></b>												
$f_{MAX\_PRI}$	Frequency for Primary Clock Tree		-	370	-	303	-	257	MHz			
$t_{W\_PRI}$	Clock Pulse Width for Primary Clock		0.8	-	0.8	-	0.8	-	ns			
$t_{ISKEW\_PRI}$	Primary Clock Skew Within a Device		-	420	-	462	-	505	ps			
<b>Edge Clock<sup>6</sup></b>												
$f_{MAX\_EDGE}$	Frequency for Edge Clock Tree		-	400	-	327.9	-	277.8	MHz			
$t_{W\_EDGE}$	Clock Pulse Width for Edge Clock		1.175	-	1.434	-	1.76	-	ns			
$t_{ISKEW\_EDGE}$	Edge Clock Skew Within a Bank		-	160	-	180	-	200	ps			
<b>Generic SDR Input</b>												
<b>General I/O Pin Parameters Using Dedicated Clock Input Primary Clock Without PLL<sup>2</sup></b>												
$t_{CO}$	Clock to Output - PIO Output Register	LFE5-85	-	5.4	-	6.1	-	6.8	ns			
$t_{SU}$	Clock to Data Setup - PIO Input Register	LFE5-85	0	-	0	-	0	-	ns			
$t_H$	Clock to Data Hold - PIO Input Register	LFE5-85	2.7	-	3	-	3.3	-	ns			
$t_{SU\_DEL}$	Clock to Data Setup - PIO Input Register with Data Input Delay	LFE5-85	1.2	-	1.33	-	1.46	-	ns			
$t_{H\_DEL}$	Clock to Data Hold - PIO Input Register with Data Input Delay	LFE5-85	0	-	0	-	0	-	ns			
$f_{MAX\_IO}$	Clock Frequency of I/O and PFU Register	LFE5-85	-	400	-	327.9	-	277.8	MHz			
$t_{CO}$	Clock to Output - PIO Output Register	Other Devices	-	-	-	-	-	-	ns			
$t_{SU}$	Clock to Data Setup - PIO Input Register	Other Devices		-		-		-	ns			
$t_H$	Clock to Data Hold - PIO Input Register	Other Devices		-		-		-	ns			
$t_{SU\_DEL}$	Clock to Data Setup - PIO Input Register with Data Input Delay	Other Devices		-		-		-	ns			
$t_{H\_DEL}$	Clock to Data Hold - PIO Input Register with Data Input Delay	Other Devices		-		-		-	ns			
$f_{MAX\_IO}$	Clock Frequency of I/O and PFU Register	Other Devices	-	-	-	-	-	-	MHz			
<b>General I/O Pin Parameters Using Dedicated Clock Input Primary Clock With PLL with Clock Injection Removal Setting<sup>2</sup></b>												
$t_{COPLL}$	Clock to Output - PIO Output Register	LFE5-85	-	3.1	-	3.44	-	3.78	ns			
$t_{SUPLL}$	Clock to Data Setup - PIO Input Register	LFE5-85	0.7	-	0.78	-	0.85	-	ns			
$t_{HPLL}$	Clock to Data Hold - PIO Input Register	LFE5-85	0.8	-	0.89	-	0.98	-	ns			
$t_{SU\_DELPLL}$	Clock to Data Setup - PIO Input Register with Data Input Delay	LFE5-85	1.6	-	1.78	-	1.95	-	ns			
$t_{H\_DELPLL}$	Clock to Data Hold - PIO Input Register with Data Input Delay	LFE5-85	0	-	0	-	0	-	ns			
$t_{COPLL}$	Clock to Output - PIO Output Register	Other Devices	-	-	-	-	-	-	ns			
$t_{SUPLL}$	Clock to Data Setup - PIO Input Register	Other Devices		-		-		-	ns			
$t_{HPLL}$	Clock to Data Hold - PIO Input Register	Other Devices		-		-		-	ns			
$t_{SU\_DELPLL}$	Clock to Data Setup - PIO Input Register with Data Input Delay	Other Devices		-		-		-	ns			
$t_{H\_DELPLL}$	Clock to Data Hold - PIO Input Register with Data Input Delay	Other Devices		-		-		-	ns			

## ECP5 External Switching Characteristics (Continued)<sup>1,2</sup>

Over Recommended Commercial Operating Conditions

Parameter	Description	Device	-8		-7		-6		Units			
			Min.	Max.	Min.	Max.	Min.	Max.				
<b>Generic DDR<sup>12</sup> Input</b>												
<b>Generic DDRX1 Inputs with Clock and Data Centered at Pin (GDDRX1_RX.SCLK.Centered) Using PCLK Pin for Clock Input</b>												
t <sub>SU_GDDR1</sub>	Data Setup Before CLK	All Devices	0.52	-	0.52	-	0.52	-	ns			
t <sub>HO_GDDR1</sub>	Data Hold After CLK	All Devices	0.52	-	0.52	-	0.52	-	ns			
f <sub>MAX_GDDR1</sub>	DDRX1 Clock Frequency	All Devices	-	250	-	250	-	250	MHz			
<b>Generic DDRX1 Inputs with Clock and Data Aligned at Pin (GDDRX1_RX.SCLK.Aligned) Using PCLK Pin for Clock Input</b>												
t <sub>DVA_GDDR1</sub>	Input Data Valid After CLK	All Devices	-	-0.55	-	-0.55	-	-0.55	ns + 1/2 UI			
t <sub>DVE_GDDR1</sub>	Input Data Hold After CLK	All Devices	0.55	-	0.55	-	0.55	-	ns + 1/2 UI			
t <sub>DW_GDDR1</sub>	Input Data Setup-Hold Window	All Devices	1.1	-	1.1	-	1.1	-	ns			
f <sub>MAX_GDDR1</sub>	DDRX1 Clock Frequency	All Devices	-	250	-	250	-	250	MHz			
<b>Generic DDRX2 Inputs with Clock and Data Centered at Pin (GDDRX2_RX.ECLK.Centered, GDDRX2_RX.MIPI) Using PCLK Pin for Clock Input, Using Left and Right Sides Only</b>												
t <sub>SU_GDDR2</sub>	Data Setup Before CLK	All Devices	0.321	-	0.403	-	0.471	-	ns			
t <sub>HO_GDDR2</sub>	Data Hold After CLK	All Devices	0.321	-	0.403	-	0.471	-	ns			
f <sub>MAX_GDDR2</sub>	DDRX2 Clock Frequency (ECLK)	All Devices	-	400	-	327.9	-	277.8	MHz			
<b>Generic DDRX2 Inputs with Clock and Data Aligned at Pin (GDDRX2_RX.ECLK.Aligned) Using PCLK Pin for Clock Input, Using Left and Right Sides Only</b>												
t <sub>DVA_GDDR2</sub>	Input Data Valid After CLK	All Devices	-	-0.344	-	-0.42	-	-0.495	ns + 1/2 UI			
t <sub>DVE_GDDR2</sub>	Input Data Hold After CLK	All Devices	0.344	-	0.42	-	0.495	-	ns + 1/2 UI			
t <sub>DW_GDDR2</sub>	Input Data Setup-Hold Window	All Devices	0.688	-	0.839	-	0.99	-	ns			
f <sub>MAX_GDDR2</sub>	DDRX2 Clock Frequency (ECLK)	All Devices	-	400	-	327.9	-	277.8	MHz			
<b>Video DDRX71 Inputs with Clock and Data Aligned at Pin (GDDRX71_RX.ECLK) Using PCLK Pin for Clock Input, Using Left and Right Sides Only</b>												
t <sub>RPBi_DVA</sub>	Input Valid Bit i switching from CLK Rising Edge (i = 0 to 6, 0 aligns with CLK)	All Devices	-0.271	-	-0.39	-	-0.41	-	ns + (i + 1/2) * UI			
t <sub>RPBi_DVE</sub>	Input Hold Bit i switching from CLK Rising Edge (i = 0 to 6, 0 aligns with CLK)	All Devices	-	0.271	-	0.39	-	0.41	ns + (i + 1/2) * UI			
f <sub>MAX_GDDR71</sub>	DDR71 Clock Frequency (ECLK)	All Devices	-	378	-	310	-	262.5	MHz			
<b>Generic DDR<sup>12</sup> Output</b>												
<b>Generic DDRX1 Outputs with Clock and Data (&gt;10 Bits Wide) Centered at Pin (GDDRX1_TX.SCLK.Centered) Using PCLK Pin for Clock Input</b>												
t <sub>DVB_GDDR1</sub>	Data Output Valid Before CLK Output	All Devices	0.67	-	0.67	-	0.67	-	ns			
t <sub>DVA_GDDR1</sub>	Data Output Valid After CLK Output	All Devices	0.67	-	0.67	-	0.67	-	ns			
f <sub>MAX_GDDR1</sub>	DDRX1 Clock Frequency	All Devices	-	250	-	250	-	250	MHz			
<b>Generic DDRX1 Outputs with Clock and Data Aligned at Pin (GDDRX1_TX.SCLK.Aligned) Using PCLK Pin for Clock Input</b>												
t <sub>DIA_GDDR1</sub>	Data Output Invalid After CLK Output	All Devices	-	0.3	-	0.3	-	0.3	ns			
t <sub>DIB_GDDR1</sub>	Data Output Invalid Before CLK Output	All Devices	-	0.3	-	0.3	-	0.3	ns			
f <sub>MAX_GDDR1</sub>	DDRX1 Clock Frequency	All Devices	-	250	-	250	-	250	MHz			
<b>Generic DDRX2 Outputs with Clock and Data Centered at Pin (GDDRX2_TX.ECLK.Centered) Using PCLK Pin for Clock Input, Using Left and Right Sides Only</b>												
t <sub>DVB_GDDR2</sub>	Data Output Valid Before CLK Output	All Devices	0.442	-	0.56	-	0.676	-	ns			
t <sub>DVA_GDDR2</sub>	Data Output Valid After CLK Output	All Devices	0.442	-	0.56	-	0.676	-	ns			
f <sub>MAX_GDDR2</sub>	DDRX2 Clock Frequency (ECLK)	All Devices	-	400	-	327.9	-	277.8	MHz			

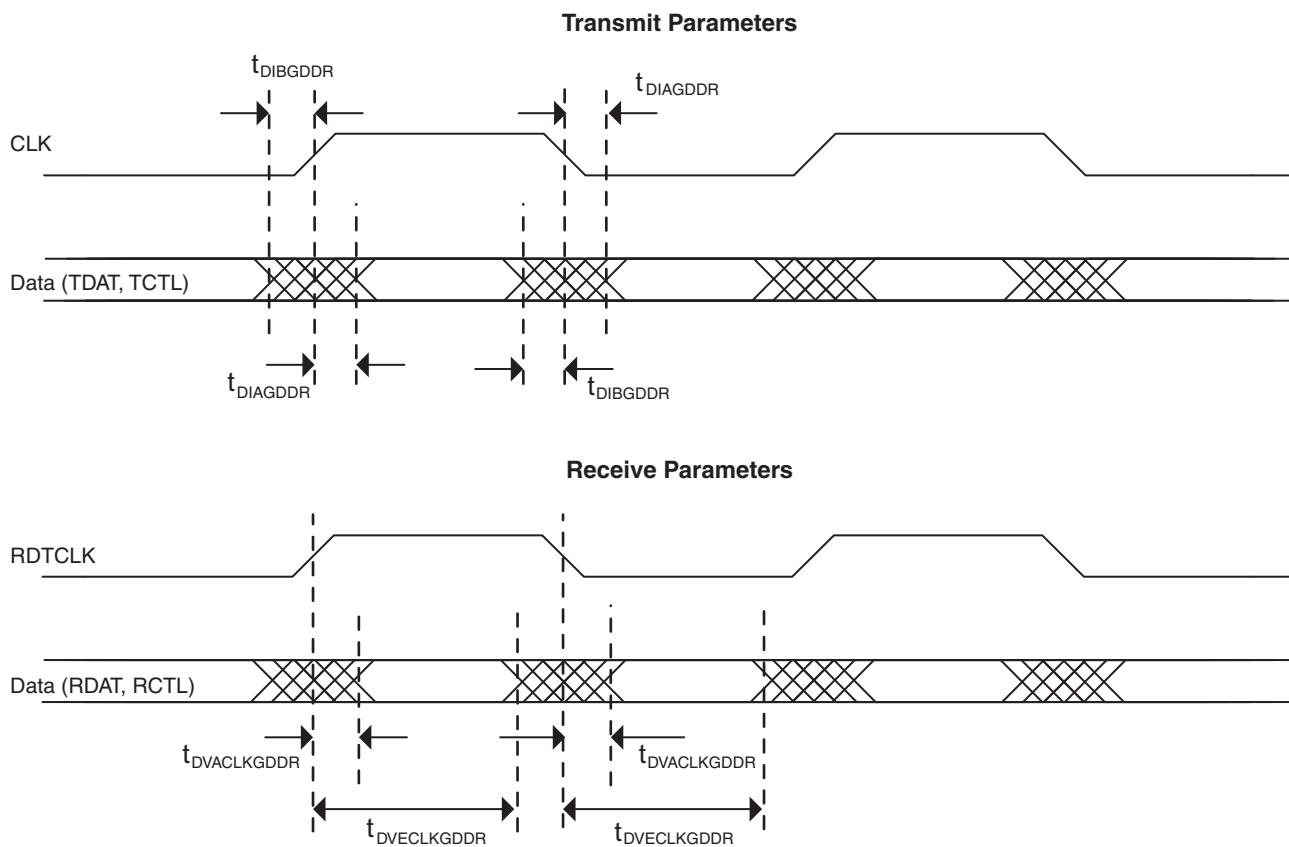
## ECP5 External Switching Characteristics (Continued)<sup>1,2</sup>

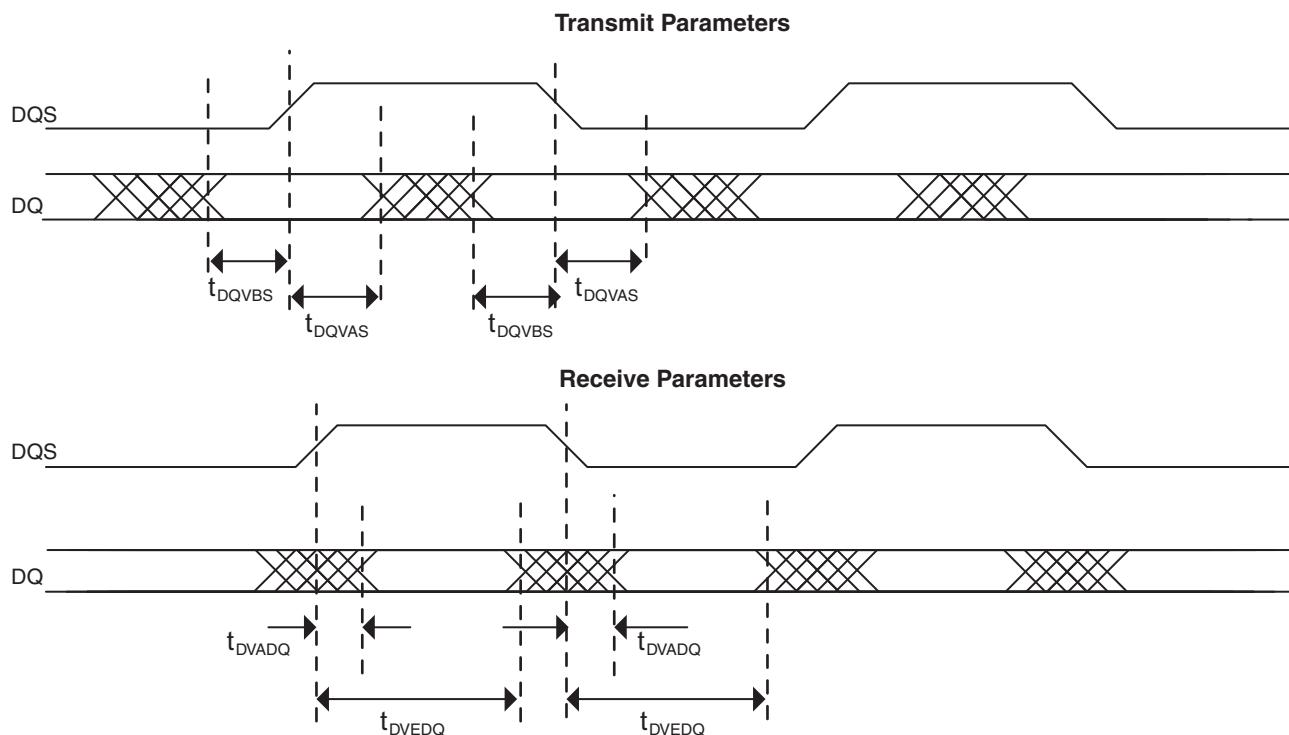
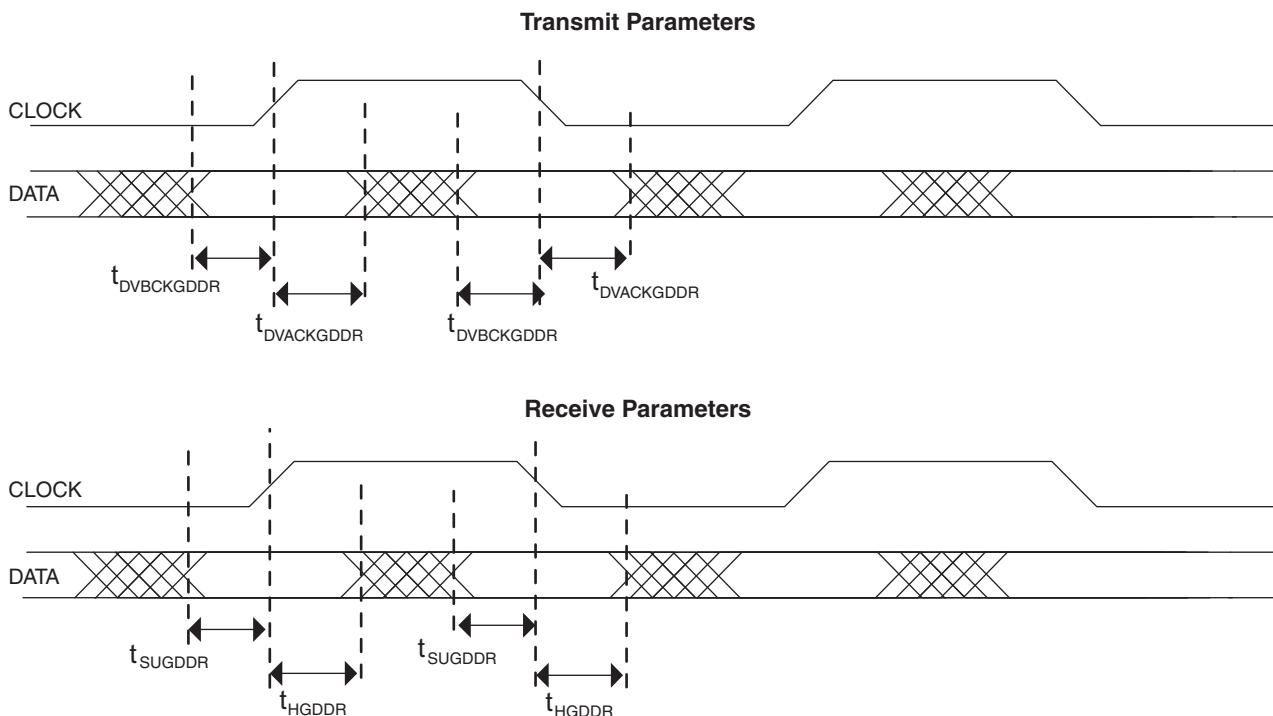
Over Recommended Commercial Operating Conditions

Parameter	Description	Device	-8		-7		-6		Units
			Min.	Max.	Min.	Max.	Min.	Max.	
<b>Generic DDRX2 Outputs with Clock and Data Aligned at Pin (GDDRX2_TX.ECLK.Aligned) Using PCLK Pin for Clock Input, Using Left and Right Sides Only</b>									
t <sub>DIA_GDDR2</sub>	Data Output Invalid After CLK Output	All Devices	-	0.16	-	0.18	-	0.2	ns
t <sub>DIB_GDDR2</sub>	Data Output Invalid Before CLK Output	All Devices	-	0.16	-	0.18	-	-	ns
f <sub>MAX_GDDR2</sub>	DDRX2 Clock Frequency (ECLK)	All Devices	-	400	-	327.9	-	277.8	MHz
<b>Video DDRX71 Outputs with Clock and Data Aligned at Pin (GDDRX71_TX.ECLK) Using PCLK Pin for Clock Input, Using Left and Right Sides Only</b>									
t <sub>TPBi_DOV</sub>	Data Output Valid for Bit i from CLK Rising Edge (i = 0 to 6, 0 aligns with CLK)	All Devices	-	0.16	-	0.18	-	0.2	ns + i * UI
t <sub>TPBi_DOI</sub>	Data Output Invalid for Bit i from CLK Rising Edge (i = 0 to 6, 0 aligns with CLK)	All Devices	-0.16	-	-0.18	-	-0.2	-	ns + (i + 1) * UI
f <sub>MAX_GDDRX71</sub>	DDR71 Clock Frequency (ECLK)	All Devices	-	378	-	310	-	262.5	MHz
<b>Memory Interface</b>									
<b>DDR2/DDR3/LPDDR2/LPDDR3 READ (Input Data are Strobed Aligned by DQS)</b>									
t <sub>DVADQ</sub>	DQ Input Valid After DQS Input	All Devices	-	-0.26	-	-0.317	-	-0.374	ns + 1/2 UI
t <sub>DVBDQ</sub>	DQ Input Valid Before DQS Input	All Devices	0.26	-	0.317	-	0.374	-	ns + 1/2 UI
t <sub>DWDQ</sub>	DQ Input Valid Window	All Devices	0.519	-	0.633	-	0.747	-	MHz
<b>DDR2/DDR3/LPDDR2/LPDDR3 WRITE (DQS Output is Centered Aligned to DQ Data Outputs)</b>									
t <sub>DQVBS</sub>	DQ Output Valid Before DQS	All Devices	0.25	-	0.25	-	0.25	-	UI
t <sub>DQVAS</sub>	DQ Output Valid After DQS	All Devices	0.25	-	0.25	-	0.25	-	UI
f <sub>MAX_DDR2</sub>	DDR2 Memory Max ECLK Frequency	All Devices	-	400	-	327.9	-	277.8	MHz
f <sub>MAX_DDR3</sub>	DDR3 Memory Max ECLK Frequency	All Devices	-	400	-	327.9	-	277.8	MHz
f <sub>MAX_LPDDR2</sub>	LPDDR2 Memory Max ECLK Frequency	All Devices	-	400	-	327.9	-	277.8	MHz
f <sub>MAX_LPDDR3</sub>	LPDDR3 Memory Max ECLK Frequency	All Devices	-	400	-	327.9	-	277.8	MHz

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

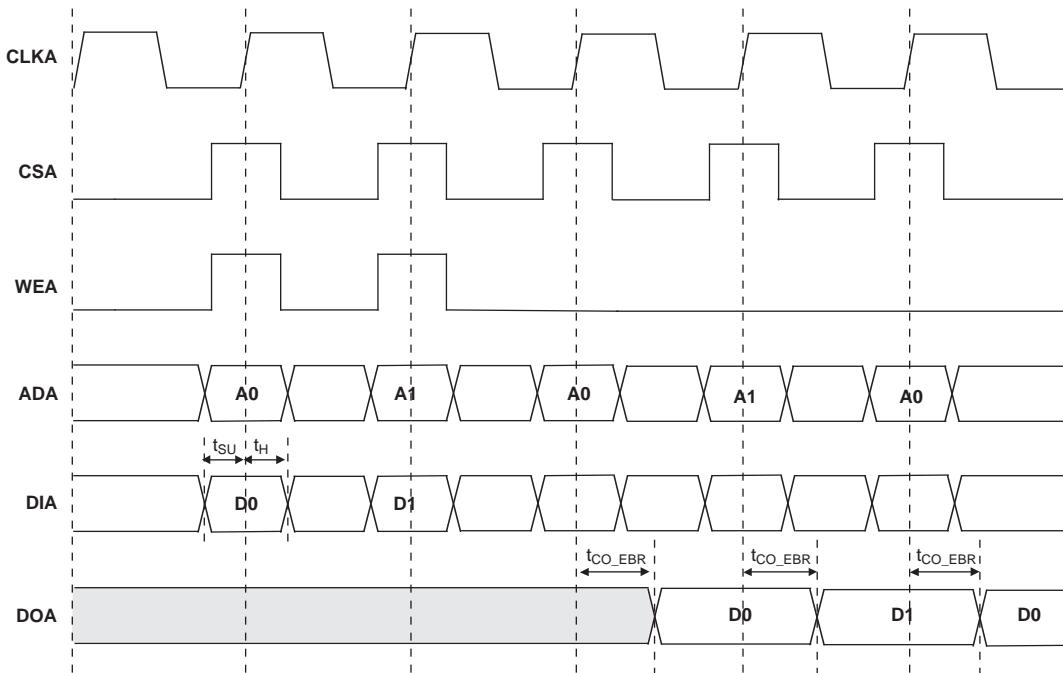
**Figure 3-6. Generic DDRX1/DDRX2 (With Clock and Data Edges Aligned)**



**Figure 3-7. DDR/DDR2/DDR3 Parameters**

**Figure 3-8. Generic DDRX1/DDRX2 (With Clock Center on Data Window)**


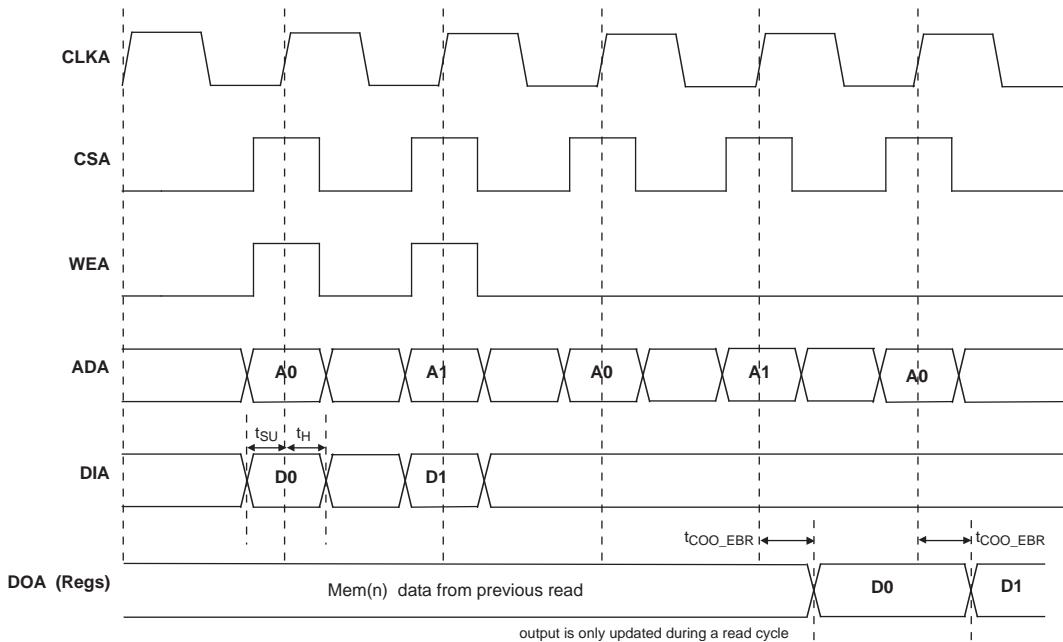
## Timing Diagrams

**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**



**ECP5 Family Timing Adders<sup>1, 2, 3, 4, 5</sup>**

Over Recommended Commercial Operating Conditions

Buffer Type	Description	-8	-7	-6	Units
<b>Input Adjusters</b>					
LVDS25E	LVDS, Emulated, VCCIO = 2.5V	0.03	-0.01	-0.03	ns
LVDS25	LVDS, VCCIO = 2.5V	0.03	0.00	-0.04	ns
BLVDS25	BLVDS, Emulated, VCCIO = 2.5V	0.03	0.00	-0.04	ns
MLVDS25	MLVDS, Emulated, VCCIO = 2.5V	0.03	0.00	-0.04	ns
LVPECL33	LVPECL	0.17	0.23	0.28	ns
MIPI D-PHY	MIPI Video				ns
SLVS	SLVS, similar to MIPI				ns
SSTL18_I	SSTL_18 class I, VCCIO = 1.8V	0.08	0.06	0.04	ns
SSTL18_II	SSTL_18 class II, VCCIO = 1.8V	0.08	0.06	0.04	ns
SSTL18D_I	Differential SSTL_18 class I	0.08	0.06	0.04	ns
SSTL18D_II	Differential SSTL_18 class II	0.08	0.06	0.04	ns
SSTL15_I	SSTL_15, VCCIO = 1.5 V class I	0.09	0.06	0.03	ns
SSTL15_II	SSTL_15, VCCIO = 1.5 V class II	0.09	0.06	0.03	ns
SSTL15D_I	Differential SSTL_15 class I	0.09	0.06	0.03	ns
SSTL15D_II	Differential SSTL_15 class II	0.09	0.06	0.03	ns
SSTL135_I	SSTL_135, VCCIO = 1.35 V class I				ns
SSTL135_II	SSTL_135, VCCIO = 1.35 V class II				ns
SSTL135D_I	Differential SSTL_135 class I				ns
SSTL135D_II	Differential SSTL_135 class II				ns
HSUL12	HSUL, VCCIO=1.2 V				ns
LVTTL33	LVTTL, VCCIO = 3.3 V	0.07	0.07	0.07	ns
LVTTL33D	LVTTL, VCCIO = 3.3 V	0.07	0.07	0.07	ns
LVCMOS33	LVCMOS, VCCIO = 3.3 V	0.07	0.07	0.07	ns
LVCMOS33D	LVCMOS, VCCIO = 3.3 V	0.07	0.07	0.07	ns
LVCMOS25	LVCMOS, VCCIO = 2.5 V	0.00	0.00	0.00	ns
LVCMOS25D	LVCMOS, VCCIO = 2.5 V	0.00	0.00	0.00	ns
LVCMOS18	LVCMOS, VCCIO = 1.8 V	-0.13	-0.13	-0.13	ns
LVCMOS15	LVCMOS, VCCIO = 1.5 V	-0.07	-0.07	-0.07	ns
LVCMOS12	LVCMOS, VCCIO = 1.2 V	-0.20	-0.19	-0.19	ns
<b>Output Adjusters</b>					
LVDS25E	LVDS, Emulated, VCCIO = 2.5 V	1.02	1.14	1.26	ns
LVDS25	LVDS, VCCIO = 2.5 V	-0.11	-0.07	-0.03	ns
BLVDS25	BLVDS, Emulated, VCCIO = 2.5 V	1.01	1.13	1.25	ns
MLVDS25	MLVDS, Emulated, VCCIO = 2.5 V	1.01	1.13	1.25	ns
LVPECL33	LVPECL, Emulated, VCCIO = 3.3 V	0.67	0.76	0.86	ns
HSUL12_4mA	HSUL, VCCIO=1.2, 4 mA Drive				ns
HSUL12_8mA	HSUL, VCCIO=1.2, 8 mA Drive				ns
SSTL18_I	SSTL_1.8 class I, VCCIO = 1.8 V	0.70	0.84	0.97	ns
SSTL18D_I	Differential SSTL_1.8 class I	0.70	0.84	0.97	ns
SSTL15_I	SSTL_1.5, VCCIO = 1.5 V	1.22	1.35	1.48	ns
SSTL15_II	SSTL_1.5, VCCIO = 1.5 V	1.22	1.35	1.48	ns

**ECP5 Family Timing Adders<sup>1, 2, 3, 4, 5</sup> (Continued)**

Over Recommended Commercial Operating Conditions

Buffer Type	Description	-8	-7	-6	Units
SSTL15D_I	Differential SSTL_15	1.22	1.35	1.48	ns
SSTL15D_II	Differential SSTL_15	1.22	1.35	1.48	ns
SSTL135_I	SSTL_135, VCCIO=1.35V				ns
SSTL135D_I	Differential SSTL_135, VCCIO=1.35V				ns
SSTL135D_II	Differential SSTL_135, VCCIO=1.35V				ns
LVTTL33_4mA	LVTTL 4mA drive, VCCIO = 3.3V	0.04	0.02	0.01	ns
LVTTL33_8mA	LVTTL 8mA drive, VCCIO = 3.3V	-0.02	-0.03	-0.05	ns
LVTTL33_12mA	LVTTL 12mA drive, VCCIO = 3.3V	-0.08	-0.08	-0.09	ns
LVTTL33_16mA	LVTTL 16mA drive, VCCIO = 3.3V	-0.10	-0.10	-0.11	ns
LVTTL33D_4mA	LVTTL 4mA drive, VCCIO = 3.3V	0.04	0.02	0.01	ns
LVTTL33D_8mA	LVTTL 8mA drive, VCCIO = 3.3V	-0.02	-0.03	-0.05	ns
LVTTL33D_12mA	LVTTL 12mA drive, VCCIO = 3.3V	-0.08	-0.08	-0.09	ns
LVTTL33D_16mA	LVTTL 16mA drive, VCCIO = 3.3V	-0.10	-0.10	-0.11	ns
LVCMOS33_4mA	LVCMOS 3.3 4mA drive, fast slew rate	0.04	0.02	0.01	ns
LVCMOS33_8mA	LVCMOS 3.3 8mA drive, fast slew rate	-0.02	-0.03	-0.05	ns
LVCMOS33_12mA	LVCMOS 3.3 12mA drive, fast slew rate	-0.08	-0.08	-0.09	ns
LVCMOS33_16mA	LVCMOS 3.3 16mA drive, fast slew rate	-0.10	-0.10	-0.11	ns
LVCMOS33D_4mA	LVCMOS 3.3 4mA drive, fast slew rate	0.04	0.02	0.01	ns
LVCMOS33D_8mA	LVCMOS 3.3 8mA drive, fast slew rate	-0.02	-0.03	-0.05	ns
LVCMOS33D_12mA	LVCMOS 3.3 12mA drive, fast slew rate	-0.08	-0.08	-0.09	ns
LVCMOS33D_16mA	LVCMOS 3.3 16mA drive, fast slew rate	-0.10	-0.10	-0.11	ns
LVCMOS25_4mA	LVCMOS 2.5 4mA drive, fast slew rate	0.12	0.10	0.08	ns
LVCMOS25_8mA	LVCMOS 2.5 8mA drive, fast slew rate	0.06	0.05	0.04	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	-0.03	-0.03	ns
LVCMOS25D_4mA	LVCMOS 2.5 4mA drive, fast slew rate	0.12	0.10	0.08	ns
LVCMOS25D_8mA	LVCMOS 2.5 8mA drive, fast slew rate	0.06	0.05	0.04	ns
LVCMOS25D_12mA	LVCMOS 2.5 12mA drive, fast slew rate	-0.04	-0.03	-0.03	ns
LVCMOS25D_16mA	LVCMOS 2.5 16mA drive, fast slew rate	-0.04	-0.03	-0.03	ns
LVCMOS18_4mA	LVCMOS 1.8 4mA drive, fast slew rate	0.11	0.12	0.14	ns
LVCMOS18_8mA	LVCMOS 1.8 8mA drive, fast slew rate	0.11	0.12	0.14	ns
LVCMOS18_12mA	LVCMOS 1.8 12mA drive, fast slew rate	-0.04	-0.03	-0.03	ns
LVCMOS18_16mA	LVCMOS 1.8 16mA drive, fast slew rate	-0.04	-0.03	-0.03	ns
LVCMOS15_4mA	LVCMOS 1.5 4mA drive, fast slew rate	0.21	0.25	0.29	ns
LVCMOS15_8mA	LVCMOS 1.5 8mA drive, fast slew rate	0.05	0.07	0.09	ns
LVCMOS12_2mA	LVCMOS 1.2 2mA drive, fast slew rate	0.43	0.51	0.59	ns
LVCMOS12_6mA	LVCMOS 1.2 6mA drive, fast slew rate	0.23	0.28	0.33	ns
LVCMOS33_4mA	LVCMOS 3.3 4mA drive, slow slew rate	1.44	1.58	1.72	ns
LVCMOS33_8mA	LVCMOS 3.3 8mA drive, slow slew rate	0.98	1.10	1.22	ns
LVCMOS33_12mA	LVCMOS 3.3 12mA drive, slow slew rate	0.67	0.77	0.86	ns
LVCMOS33_16mA	LVCMOS 3.3 16mA drive, slow slew rate	0.97	1.09	1.21	ns
LVCMOS33_20mA	LVCMOS 3.3 20mA drive, slow slew rate	0.67	0.76	0.85	ns
LVCMOS25_4mA	LVCMOS 2.5 4mA drive, slow slew rate	1.48	1.63	1.78	ns

## ECP5 Family Timing Adders<sup>1, 2, 3, 4, 5</sup> (Continued)

Over Recommended Commercial Operating Conditions

Buffer Type	Description	-8	-7	-6	Units
LVCMS25_8mA	LVCMS 2.5 8mA drive, slow slew rate	1.02	1.14	1.27	ns
LVCMS25_12mA	LVCMS 2.5 12mA drive, slow slew rate	0.74	0.84	0.94	ns
LVCMS25_16mA	LVCMS 2.5 16mA drive, slow slew rate	1.02	1.14	1.26	ns
LVCMS25_20mA	LVCMS 2.5 20mA drive, slow slew rate	0.74	0.83	0.93	ns
LVCMS18_4mA	LVCMS 1.8 4mA drive, slow slew rate	1.60	1.77	1.93	ns
LVCMS18_8mA	LVCMS 1.8 8mA drive, slow slew rate	1.11	1.25	1.38	ns
LVCMS18_12mA	LVCMS 1.8 12mA drive, slow slew rate	0.87	0.98	1.09	ns
LVCMS18_16mA	LVCMS 1.8 16mA drive, slow slew rate	0.86	0.97	1.07	ns
LVCMS15_4mA	LVCMS 1.5 4mA drive, slow slew rate	1.71	1.89	2.08	ns
LVCMS15_8mA	LVCMS 1.5 8mA drive, slow slew rate	1.20	1.34	1.48	ns
LVCMS12_2mA	LVCMS 1.2 2mA drive, slow slew rate	1.37	1.56	1.74	ns
LVCMS12_6mA	LVCMS 1.2 6mA drive, slow slew rate	1.11	1.27	1.43	ns

1. Timing adders are characterized but not tested on every device.
2. LVCMS timing measured with the load specified in Switching Test Condition table.
3. All other standards tested according to the appropriate specifications.
4. Not all I/O standards and drive strengths are supported for all banks. See the Architecture section of this data sheet for details.
5. Commercial timing numbers are shown. Industrial numbers are typically slower and can be extracted from the Diamond or isp-LEVER software.

## ECP5 Maximum I/O Buffer Speed<sup>1, 2, 3, 4, 5, 6</sup>

### Over Recommended Operating Conditions

Buffer	Description	Max.	Units
<b>Maximum Input Frequency</b>			
LVDS25	LVDS, $V_{CCIO} = 2.5$ V	400	MHz
MLVDS25	MLVDS, Emulated, $V_{CCIO} = 2.5$ V	400	MHz
BLVDS25	BLVDS, Emulated, $V_{CCIO} = 2.5$ V	400	MHz
MIPI D-PHY (HS Mode)	MIPI Video	400	MHz
SLVS	SLVS similar to MIPI	400	MHz
Mini LVDS	Mini LVDS	400	MHz
LVPECL33	LVPECL, Emulated, $V_{CCIO} = 3.3$ V	400	MHz
SSTL18 (all supported classes)	SSTL_18 class I, II, $V_{CCIO} = 1.8$ V	400	MHz
SSTL15 (all supported classes)	SSTL_15 class I, II, $V_{CCIO} = 1.5$ V	400	MHz
SSTL135 (all supported classes)	SSTL_135 class I, II, $V_{CCIO} = 1.35$ V	400	MHz
HSUL12 (all supported classes)	HSUL_12 class I, II, $V_{CCIO} = 1.2$ V	400	MHz
LVTTL33	LVTTL, $V_{CCIO} = 3.3$ V	150	MHz
LVCMOS33	LVCMOS, $V_{CCIO} = 3.3$ V	150	MHz
LVCMOS25	LVCMOS, $V_{CCIO} = 2.5$ V	150	MHz
LVCMOS18	LVCMOS, $V_{CCIO} = 1.8$ V	150	MHz
LVCMOS15	LVCMOS 1.5, $V_{CCIO} = 1.5$ V	150	MHz
LVCMOS12	LVCMOS 1.2, $V_{CCIO} = 1.2$ V	150	MHz
<b>Maximum Output Frequency</b>			
LVDS25E	LVDS, Emulated, $V_{CCIO} = 2.5$ V	300	MHz
LVDS25	LVDS, $V_{CCIO} = 2.5$ V	400	MHz
MLVDS25	MLVDS, Emulated, $V_{CCIO} = 2.5$ V	300	MHz
BLVDS25	BLVDS, Emulated, $V_{CCIO} = 2.5$ V	300	MHz
LVPECL33	LVPECL, Emulated, $V_{CCIO} = 3.3$ V	300	MHz
SSTL18 (all supported classes)	SSTL_18 class I, II, $V_{CCIO} = 1.8$ V	400	MHz
SSTL15 (all supported classes)	SSTL_15 class I, II, $V_{CCIO} = 1.5$ V	400	MHz
SSTL135 (all supported classes)	SSTL_135 class I, II, $V_{CCIO} = 1.35$ V	400	MHz
HSUL12 (all supported classes)	HSUL12 class I, II, $V_{CCIO} = 1.2$ V	400	MHz
LVTTL33	LVTTL, $V_{CCIO} = 3.3$ V	150	MHz
LVCMOS33 (For all drives)	LVCMOS, 3.3 V	150	MHz
LVCMOS25 (For all drives)	LVCMOS, 2.5 V	150	MHz
LVCMOS18 (For all drives)	LVCMOS, 1.8 V	150	MHz
LVCMOS15 (For all drives)	LVCMOS, 1.5 V	150	MHz
LVCMOS12 (For all drives except 2mA)	LVCMOS, $V_{CCIO} = 1.2$ V	150	MHz
LVCMOS12 (2mA drive)	LVCMOS, $V_{CCIO} = 1.2$ V	150	MHz

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

## sysCLOCK PLL Timing

### Over Recommended Operating Conditions

Parameter	Descriptions	Conditions	Min.	Max.	Units
$f_{IN}$	Input Clock Frequency (CLKI, CLKFB)		10	400	MHz
$f_{OUT}$	Output Clock Frequency (CLKOP, CLKOS,		3.125	400	MHz
$f_{VCO}$	PLL VCO Frequency		400	800	MHz
$f_{PFD}^3$	Phase Detector Input Frequency		10	400	MHz
<b>AC Characteristics</b>					
$t_{DT}$	Output Clock Duty Cycle		47	53	%
$t_{PH4}$	Output Phase Accuracy		-5	5	%
$t_{OPJIT}^1$	Output Clock Period Jitter	$f_{OUT} \geq 100\text{MHz}$	—	100	ps p-p
		$f_{OUT} < 100\text{MHz}$	—	0.025	UIPP
	Output Clock Cycle-to-Cycle Jitter	$f_{OUT} \geq 100\text{MHz}$	—	200	ps p-p
		$f_{OUT} < 100\text{MHz}$	—	0.050	UIPP
	Output Clock Phase Jitter	$f_{PFD} > 100\text{MHz}$	—	160	ps p-p
		$f_{PFD} < 100\text{MHz}$	—	0.011	UIPP
$t_{SPO}$	Static Phase Offset	Divider ratio = integer	—	400	ps p-p
$t_w$	Output Clock Pulse Width	At 90% or 10% <sup>3</sup>	0.9	—	ns
$t_{LOCK}^2$	PLL Lock-in Time		—	15	ms
$t_{UNLOCK}$	PLL Unlock Time		—	50	ns
$t_{IPJIT}^6$	Input Clock Period Jitter	$f_{PFD} \geq 20\text{ MHz}$	—	1,000	ps p-p
		$f_{PFD} < 20\text{ MHz}$	—	0.02	UIPP
$t_{HI}$	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		1	—	ms
$t_{RSTREC}$	RST Recovery Time		1	—	ns
$t_{LOAD\_REG}$	Min Pulse for CIB_LOAD_REG		10	—	ns
$t_{ROTATE-SETUP}$	Min time for CIB dynamic phase controls to be stable fore CIB_ROTATE		5	—	ns
$t_{ROTATE-WD}$	Min pulse width for CIB_ROTATE to maintain "0" or "1"		4	—	VCO cycles

1. Jitter sample is taken over 10,000 samples of the primary PLL output with clean reference clock with no additional I/O toggling.

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

3. Period jitter and cycle-to-cycle jitter numbers are guaranteed for  $f_{PFD} > 4\text{MHz}$ . For  $f_{PFD} < 4\text{MHz}$ , the jitter numbers may not be met in certain conditions. Please contact the factory for  $f_{PFD} < 4\text{MHz}$ .

## DLL Timing

### Over Recommended Operating Conditions

Parameter	Description	Condition	Min.	Typ.	Max.	Units
$f_{REF}$	Input reference clock frequency (on-chip or off-chip)		100	—	400	MHz
$t_{PWH}$	Input clock minimum pulse width high (at 80% level)		550	—	—	ps
$t_{PWL}$	Input clock minimum pulse width low (at 20% level)		550	—	—	ps
$t_{INSTB}$	Input clock period jitter		—	—	500	ps
$t_{LOCK}$	DLL lock time		8	—	8200	cycles
$t_{RSWD}$	Digital reset minimum pulse width (at 80% level)		3	—	—	ns

1. CLKOP runs at the same frequency as the input clock.
2. CLKOS minimum frequency is obtained with divide by 4.
3. This is intended to be a “path-matching” design guideline and is not a measurable specification.

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

**Table 3-6. Serial Output Timing and Levels**

Symbol	Description	Min.	Typ.	Max.	Units
$V_{TX-DIFF-PP-1.2}$	Differential Swing, 1.2 V setting, only at $V_{CCHTX}=1.2V\pm5\%$ <sup>1,2</sup>	-20%	1200	20%	mV, pp
$V_{TX-DIFF-PP-1.1}$	Differential Swing, 1.1 V setting <sup>1,2</sup>	-25%	1100	25%	mV, pp
$V_{TX-DIFF-PP-1.0}$	Differential Swing, 1.0 V setting <sup>1,2</sup>	-25%	1000	25%	mV, pp
$V_{TX-DIFF-PP-0.9}$	Differential Swing, 0.9 V setting <sup>1,2</sup>	-25%	900	25%	mV, pp
$V_{TX-DIFF-PP-0.85}$	Differential Swing, 0.85 V setting <sup>1,2</sup>	-25%	850	25%	mV, pp
$V_{TX-DIFF-PP-0.75}$	Differential Swing, 0.75 V setting <sup>1,2</sup>	-25%	750	25%	mV, pp
$V_{TX-DIFF-PP-0.7}$	Differential Swing, 0.7 V setting <sup>1,2</sup>	-25%	700	25%	mV, pp
$V_{TX\_COM}$	Output common mode voltage	-10%	$V_{CCHTX}/2$	10%	mV, pp
$T_{TX-R}$	Rise time (20% to 80%)	50	-	-	ps
$T_{TX-F}$	Fall time (80% to 20%)	50	-	-	ps
$T_{TX-SKEW}$	Transmit output skew of P/N	-	-	15	ps
$Z_{TX\_SE}$	Single ended output impedance	-20%	50/75/HiZ	20%	Ohms
$R_{LTX\_DIFF}$	Differential return loss (with package included) <sup>3</sup>	-	-	-10	dB
$R_{LTX\_COM}$	Common mode return loss (with package included) <sup>3</sup>			-6	dB

1. All measurements are with 50 ohm impedance.
2. See TN1261, [ECP5 SERDES/PCS Usage Guide](#) for actual binary settings and the min-max range.
3. Return los = -10dB (differential), -6dB (common mode) for 50KHz  $\leq f \leq 1.6\text{GHz}$  with 50-ohm output impedance configuration. This includes degradation due to package effects.

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

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

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

## SERDES/PCS Block Latency

Table 3-8 describes the latency of each functional block in the transmitter and receiver. Latency is given in parallel clock cycles. Figure 3-11 shows the location of each block.

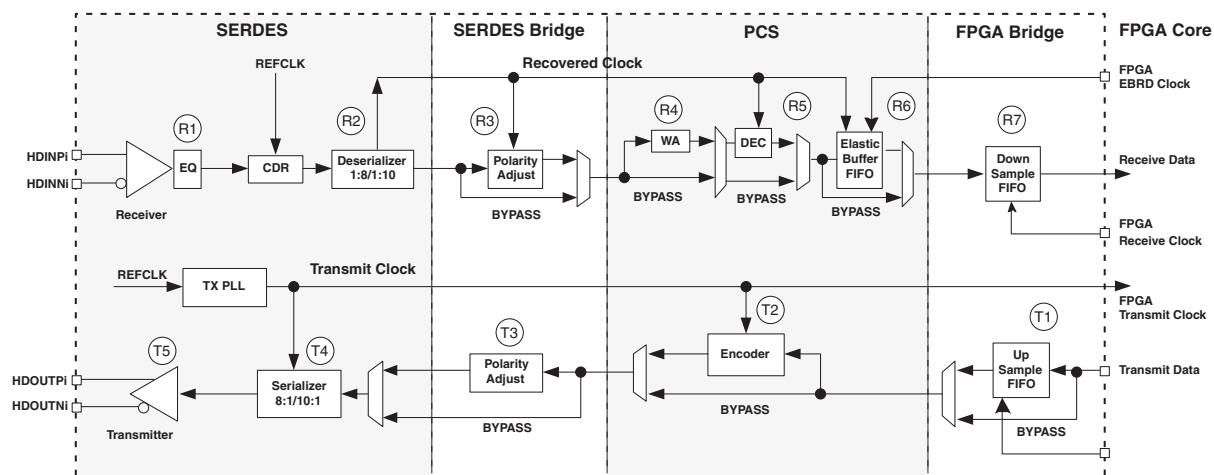
**Table 3-8. SERDES/PCS Latency Breakdown**

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

1.  $\Delta 1 = -245\text{ps}$ ,  $\Delta 2 = +88\text{ps}$ ,  $\Delta 3 = +112\text{ps}$ .

2.  $\Delta 1 = +118\text{ps}$ ,  $\Delta 2 = +132\text{ps}$ ,  $\Delta 3 = +700\text{ps}$ .

**Figure 3-11. Transmitter and Receiver Latency Block Diagram**



## SERDES High Speed Data Receiver

**Table 3-9. Serial Input Data Specifications**

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

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

## Input Data Jitter Tolerance

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

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

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

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

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

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

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

## SERDES External Reference Clock

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

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

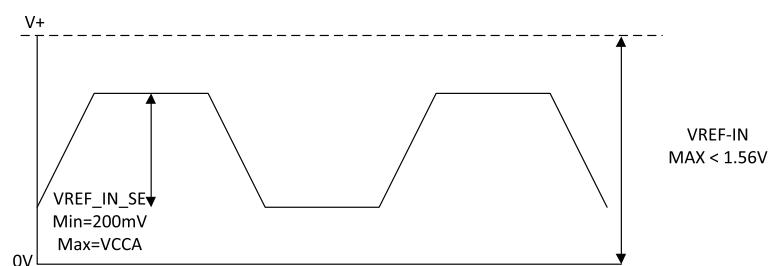
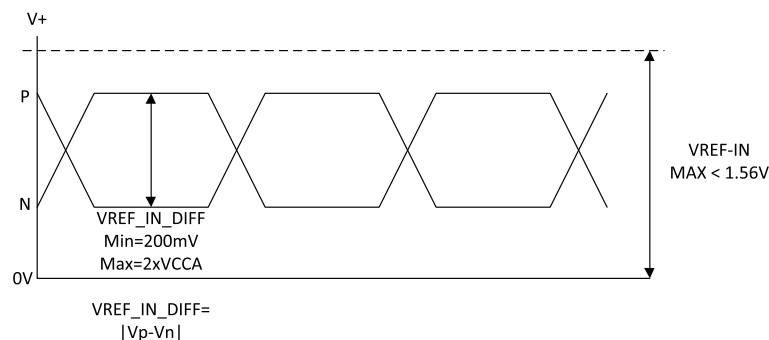
Symbol	Description	Min.	Typ.	Max.	Units
$F_{REF}$	Frequency range	50	—	320	MHz
$F_{REF-PPM}$	Frequency tolerance <sup>1</sup>	-1000	—	1000	ppm
$V_{REF-IN-SE}$	Input swing, single-ended clock <sup>2</sup>	200	—	$V_{CCA}$	mV, p-p
$V_{REF-IN-DIFF}$	Input swing, differential clock	200	—	$2 \times V_{CCA}$	mV, p-p differential
$V_{REF-IN}$	Input levels	0	—	$V_{CCA} + 0.4$	V
$D_{REF}$	Duty cycle <sup>3</sup>	40	—	60	%
$T_{REF-R}$	Rise time (20% to 80%)	200	500	1000	ps
$T_{REF-F}$	Fall time (80% to 20%)	200	500	1000	ps
$Z_{REF-IN-TERM-DIFF}$	Differential input termination	-20%	100/2K	+20%	Ohms
$C_{REF-IN-CAP}$	Input capacitance	—	—	7	pF

1. Depending on the application, the PLL\_LOL\_SET and CDR\_LOL\_SET control registers may be adjusted for other tolerance values as described in TN1261, [ECP5 SERDES/PCS Usage Guide](#).

2. The signal swing for a single-ended input clock must be as large as the p-p differential swing of a differential input clock to get the same gain at the input receiver. Lower swings for the clock may be possible, but will tend to increase jitter.

3. Measured at 50% amplitude.

**Figure 3-12. SERDES External Reference Clock Waveforms**



## PCI Express Electrical and Timing Characteristics

### AC and DC Characteristics

#### Over Recommended Operating Conditions

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

1. Values are measured at 2.5 Gbps.

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

3. Not in compliance with PCI Express 1.1 standard.

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

### AC and DC Characteristics

**Table 3-13. Transmit**

#### Over Recommended Operating Conditions

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
T <sub>RF</sub>	Differential rise/fall time	20%-80%	—	80	—	ps
Z <sub>TX_DIFF_DC</sub>	Differential impedance		80	100	120	Ohms
J <sub>TX_DDJ</sub> <sup>2, 3, 4</sup>	Output data deterministic jitter		—	—	0.17	UI
J <sub>TX_TJ</sub> <sup>1, 2, 3, 4</sup>	Total output data jitter		—	—	0.35	UI

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

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

#### Over Recommended Operating Conditions

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
RL <sub>RX_DIFF</sub>	Differential return loss	From 100 MHz to 3.125 GHz	10	—	—	dB
RL <sub>RX_CM</sub>	Common mode return loss	From 100 MHz to 3.125 GHz	6	—	—	dB
Z <sub>RX_DIFF</sub>	Differential termination resistance		80	100	120	Ohms
J <sub>RX_DJ</sub> <sup>1, 2, 3</sup>	Deterministic jitter tolerance (peak-to-peak)		—	—	0.37	UI
J <sub>RX_RJ</sub> <sup>1, 2, 3</sup>	Random jitter tolerance (peak-to-peak)		—	—	0.18	UI
J <sub>RX_SJ</sub> <sup>1, 2, 3</sup>	Sinusoidal jitter tolerance (peak-to-peak)		—	—	0.10	UI
J <sub>RX_TJ</sub> <sup>1, 2, 3</sup>	Total jitter tolerance (peak-to-peak)		—	—	0.65	UI
T <sub>RX_EYE</sub>	Receiver eye opening		0.35	—	—	UI

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

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

### AC and DC Characteristics

**Table 3-15. Transmit**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
T <sub>RF</sub> <sup>1</sup>	Differential rise/fall time	20%-80%	—	80	—	ps
Z <sub>TX_DIFF_DC</sub>	Differential impedance		80	100	120	Ohms
J <sub>TX_DDJ</sub> <sup>3, 4, 5</sup>	Output data deterministic jitter		—	—	0.17	UI
J <sub>TX_TJ</sub> <sup>2, 3, 4, 5</sup>	Total output data jitter		—	—	0.35	UI

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

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

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
RL <sub>RX_DIFF</sub>	Differential return loss	From 100 MHz to 2.5 GHz	10	—	—	dB
RL <sub>RX_CM</sub>	Common mode return loss	From 100 MHz to 2.5 GHz	6	—	—	dB
Z <sub>RX_DIFF</sub>	Differential termination resistance		80	100	120	Ohms
J <sub>RX_DJ</sub> <sup>2, 3, 4, 5</sup>	Deterministic jitter tolerance (peak-to-peak)		—	—	0.37	UI
J <sub>RX_RJ</sub> <sup>2, 3, 4, 5</sup>	Random jitter tolerance (peak-to-peak)		—	—	0.18	UI
J <sub>RX_SJ</sub> <sup>2, 3, 4, 5</sup>	Sinusoidal jitter tolerance (peak-to-peak)		—	—	0.10	UI
J <sub>RX_TJ</sub> <sup>1, 2, 3, 4, 5</sup>	Total jitter tolerance (peak-to-peak)		—	—	0.65	UI
T <sub>RX_EYE</sub>	Receiver eye opening		0.35	—	—	UI

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

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

### AC and DC Characteristics

**Table 3-17. Transmit**

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
T <sub>RF</sub>	Differential rise/fall time	20%-80%	—	80	—	ps
Z <sub>TX_DIFF_DC</sub>	Differential impedance		80	100	120	Ohms
J <sub>TX_DDJ</sub> <sup>3, 4, 5</sup>	Output data deterministic jitter		—	—	0.10	UI
J <sub>TX_TJ</sub> <sup>2, 3, 4, 5</sup>	Total output data jitter		—	—	0.24	UI

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

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

Symbol	Description	Test Conditions	Min.	Typ.	Max.	Units
RL <sub>RX_DIFF</sub>	Differential return loss	From 100 MHz to 1.25 GHz	10	—	—	dB
RL <sub>RX_CM</sub>	Common mode return loss	From 100 MHz to 1.25 GHz	6	—	—	dB
Z <sub>RX_DIFF</sub>	Differential termination resistance		80	100	120	Ohms
J <sub>RX_DJ</sub> <sup>1, 2, 3, 4, 5</sup>	Deterministic jitter tolerance (peak-to-peak)		—	—	0.34	UI
J <sub>RX_RJ</sub> <sup>1, 2, 3, 4, 5</sup>	Random jitter tolerance (peak-to-peak)		—	—	0.26	UI
J <sub>RX_SJ</sub> <sup>1, 2, 3, 4, 5</sup>	Sinusoidal jitter tolerance (peak-to-peak)		—	—	0.11	UI
J <sub>RX_TJ</sub> <sup>1, 2, 3, 4, 5</sup>	Total jitter tolerance (peak-to-peak)		—	—	0.71	UI
T <sub>RX_EYE</sub>	Receiver eye opening		0.29	—	—	UI

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

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

### AC and DC Characteristics

**Table 3-19. Transmit**

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

Notes:

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

**Table 3-20. Receive**

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

**Table 3-21. Reference Clock**

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

## ECP5 sysCONFIG Port Timing Specifications

Over Recommended Operating Conditions

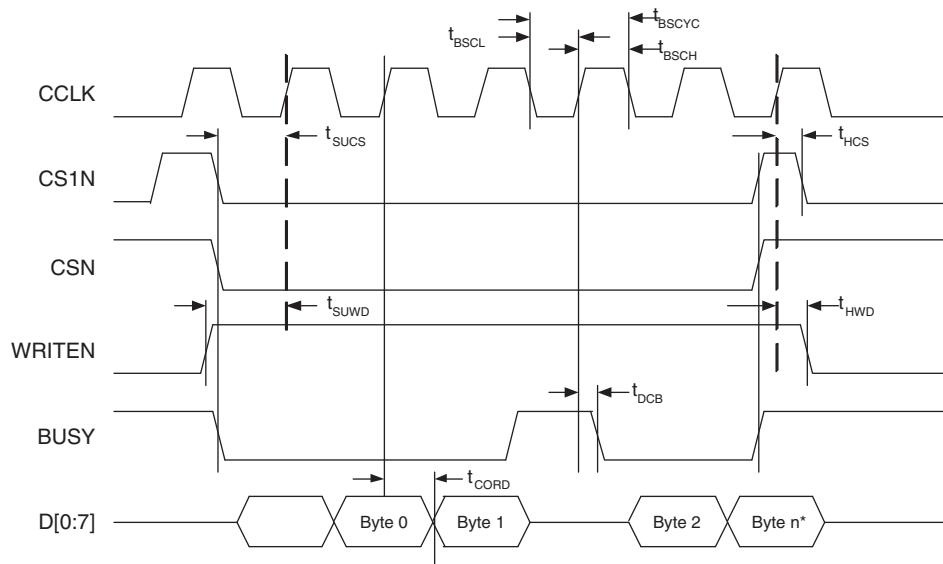
Symbol	Parameter		Min.	Max.	Units
<b>POR, Configuration Initialization, and Wakeup</b>					
$t_{ICFG}$	Time from the Application of $V_{CC}$ , $V_{CCAUX}$ or $V_{CCIO8}$ (whichever is the last) to the rising edge of INITN	Master mode	-	33	ms
		Slave mode	-	16	ms
$t_{VMC}$	Time from $t_{ICFG}$ to the valid Master CCLK		-	5	us
$t_{MWC}$	Additional wake master clock cycles after DONE goes HIGH		100	500	cycles
$t_{cz}$	CCLK from Active to High-Z		-	300	ns
<b>Master CCLK</b>					
	Frequency	Select from OSC	-20	20	%
	Duty Cycle	Select from OSC	40	60	%
<b>All Configuration Modes</b>					
$t_{PRGM}$	PROGRAMN LOW pulse accepted		55	-	ns
$t_{PRGMRJ}$	PROGRAMN LOW pulse rejected		-	25	ns
$t_{INITL}$	INITN LOW time		-	55	ns
$t_{DPPINT}$	PROGRAMN LOW to INITN LOW		-	70	ns
$t_{DPPDONE}$	PROGRAMN LOW to DONE LOW		-	80	ns
$t_{IODISS}$	PROGRAMN LOW to I/O Disabled		-	120	ns
<b>Slave SPI</b>					
$f_{CCLK}$	CCLK input clock frequency		-	66	MHz
$t_{CCLKH}$	CCLK input clock pulsewidth HIGH		6	-	ns
$t_{CCLKL}$	CCLK input clock pulsewidth LOW		6	-	ns
$t_{STSU}$	CCLK setup time		2	-	ns
$t_{STH}$	CCLK hold time		0	-	ns
$t_{STCO}$	CCLK falling edge to valid output		-	10	ns
$t_{STOZ}$	CCLK falling edge to valid disable		-	10	ns
$t_{STOV}$	CCLK falling edge to valid enable		-	10	ns
$t_{SCS}$	Chip Select HIGH time		25	-	ns
$t_{SCSS}$	Chip Select setup time		3	-	ns
$t_{SCSH}$	Chip Select hold time		3	-	ns
<b>Master SPI</b>					
$f_{CCLK}$	CCLK output clock frequency		-	133	MHz
$t_{CCLKH}$	CCLK output clock pulse width HIGH		3.5	-	ns
$t_{CCLKL}$	CCLK output clock pulse width LOW		3.5	-	ns
$t_{STSU}$	CCLK setup time		5	-	ns
$t_{STH}$	CCLK hold time		1	-	ns
$t_{CSSPI}$	INITN HIGH to Chip Select LOW		100	200	ns
$t_{CFGX}$	INITN HIGH to first CCLK edge		0.75	1	ns
<b>Slave Serial</b>					
$f_{CCLK}$	CCLK input clock frequency		-	66	MHz
$t_{SSCH}$	CCLK input clock pulse width HIGH		5	-	ns
$t_{SSCL}$	CCLK input clock pulse width LOW		5	-	ns
$t_{SUSCDI}$	CCLK setup time		2	-	ns
$t_{HSCDI}$	CCLK hold time		0	-	ns

## ECP5 sysCONFIG Port Timing Specifications (Continued)

Over Recommended Operating Conditions

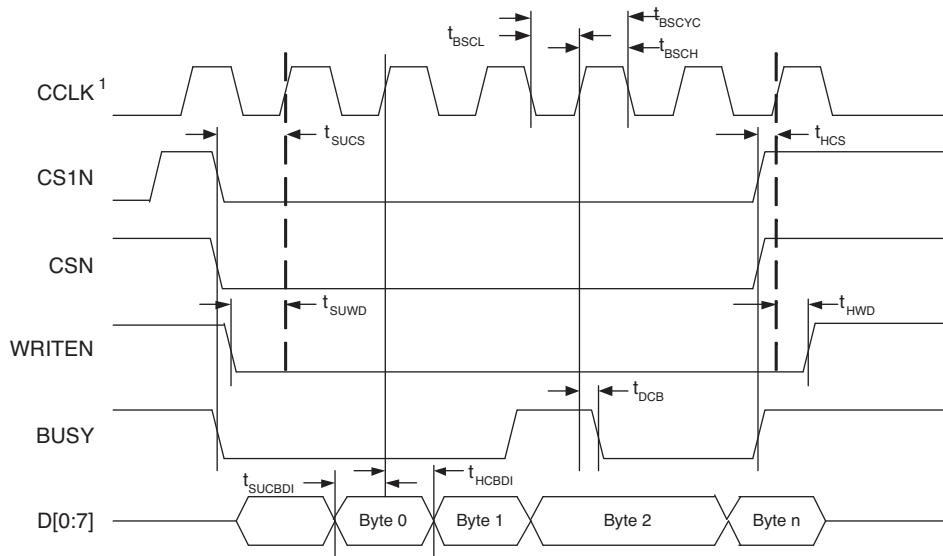
Slave Parallel					
$f_{CCLK}$	CCLK input clock frequency		-	66	MHz
$t_{BSCH}$	CCLK input clock pulsewidth HIGH		6	-	ns
$t_{BSCL}$	CCLK input clock pulsewidth LOW		6	-	ns
$t_{CORD}$	CCLK to DOUT for Read Data		-	12	ns
$t_{SUCBDI}$	Data Setup Time to CCLK		5	-	ns
$t_{HCBDI}$	Data Hold Time to CCLK		1	-	ns
$t_{SUCS}$	CSN, CSN1 Setup Time to CCLK		7	-	ns
$t_{HCS}$	CSN, CSN1 Hold Time to CCLK		1	-	ns
$t_{SUWD}$	WRITEN Setup Time to CCLK		7	-	ns
$t_{HCWD}$	WRITEN Hold Time to CCLK		1	-	ns
$t_{DCB}$	CCLK to BUSY Delay Time		-	12	ns

Figure 3-13. sysCONFIG Parallel Port Read Cycle



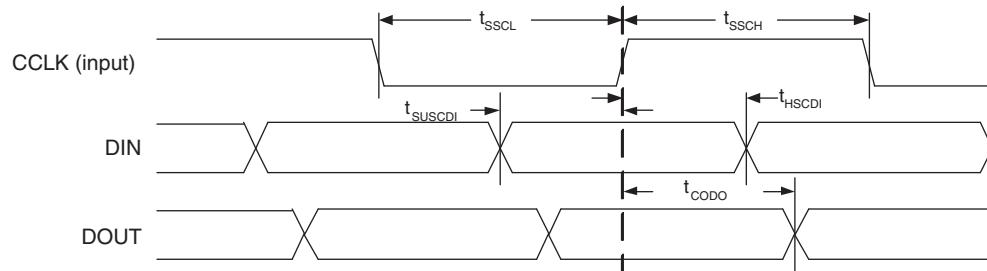
\*n = last byte of read cycle.

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

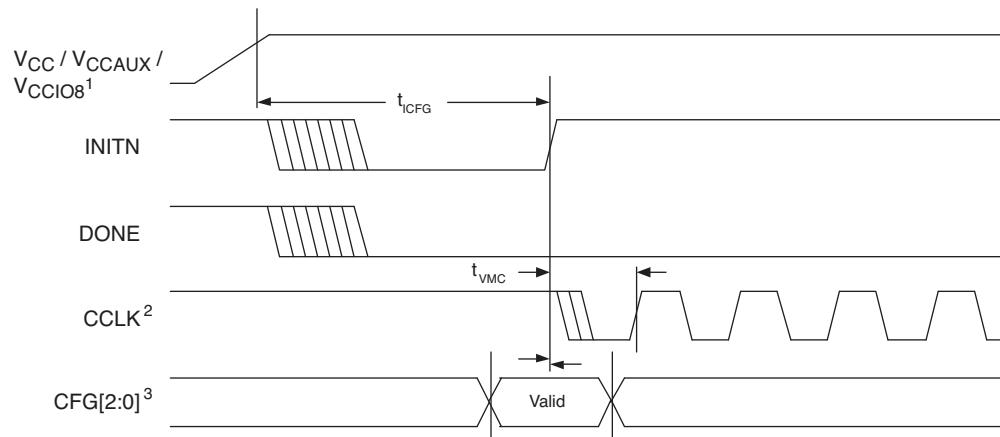


1. In Master Parallel Mode the FPGA provides CCLK (MCLK). In Slave Parallel Mode the external device provides CCLK.

**Figure 3-15. sysCONFIG Slave Serial Port Timing**



**Figure 3-16. Power-On-Reset (POR) Timing**

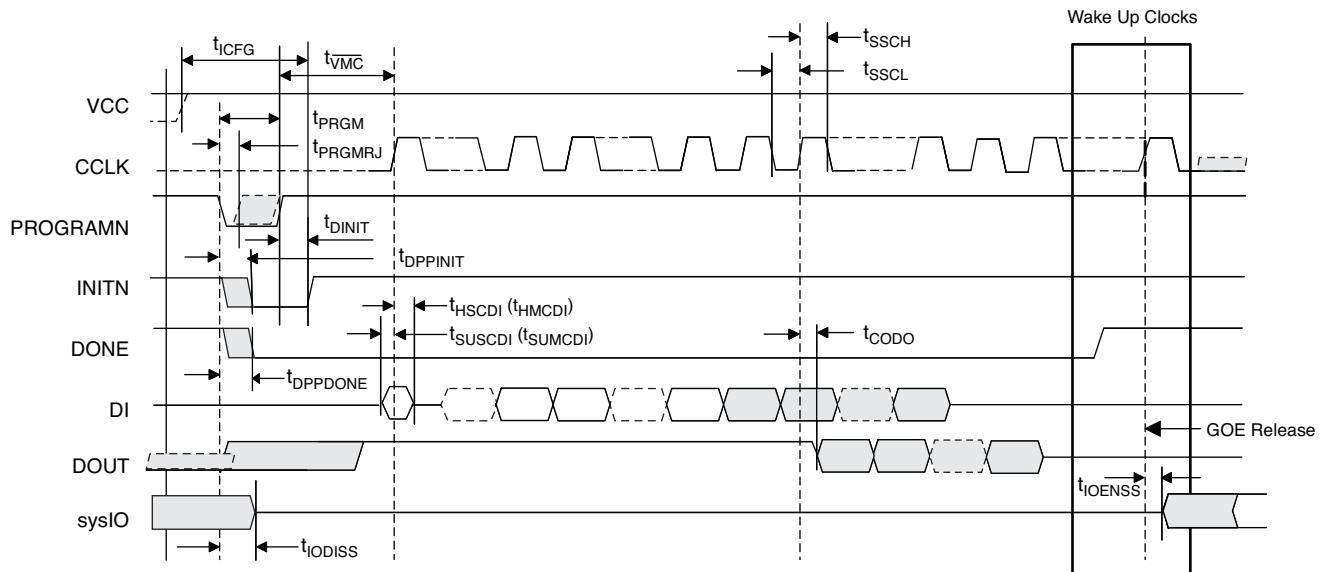


1. Time taken from V<sub>CC</sub>, V<sub>CCHAUX</sub> or V<sub>CCIO8</sub>, whichever is the last to cross the POR trip point.

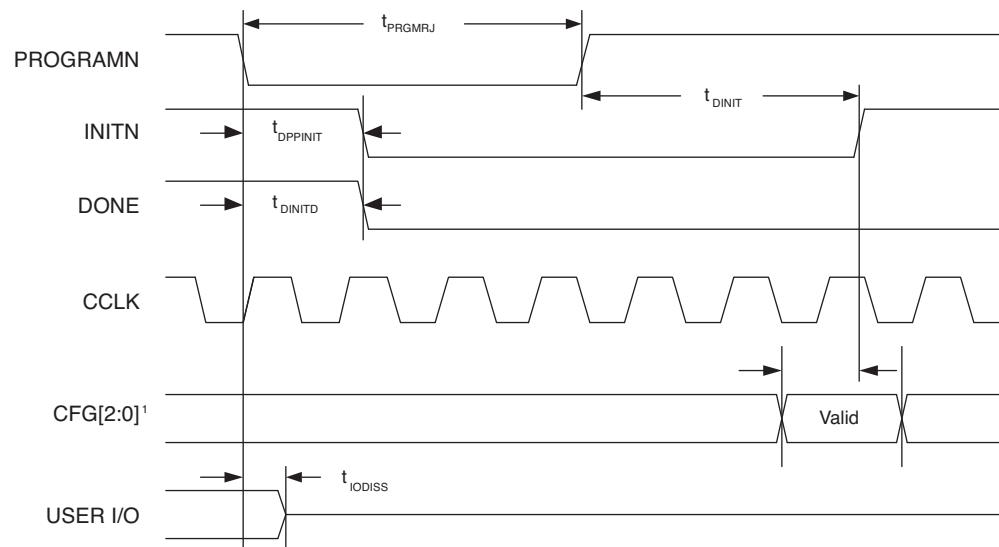
2. Device is in a Master Mode (SPI, SPI<sub>M</sub>).

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

**Figure 3-17. sysCONFIG Port Timing**

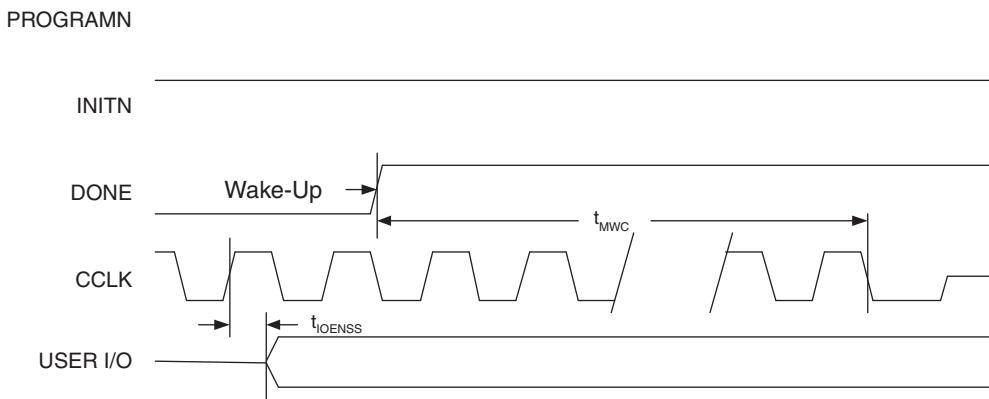


**Figure 3-18. Configuration from PROGRAMN Timing**

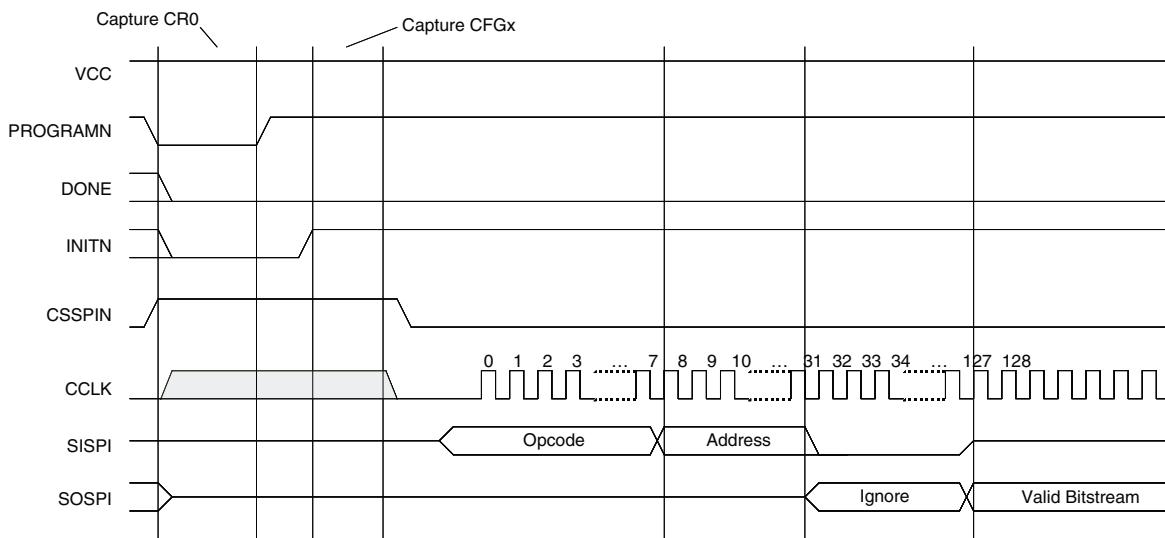


1. The CFG pins are normally static (hard wired)

**Figure 3-19. Wake-Up Timing**



**Figure 3-20. Master SPI Configuration Waveforms**

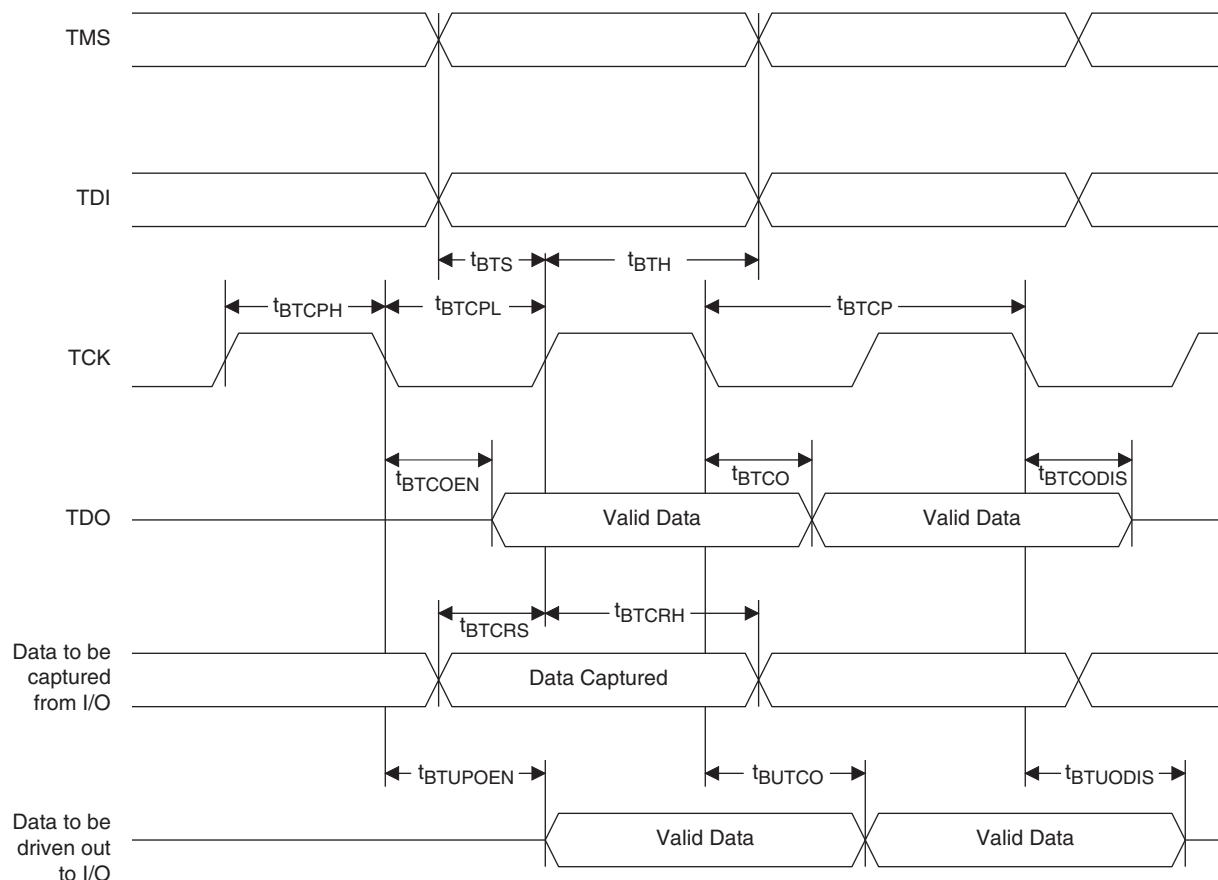


## JTAG Port Timing Specifications

Over Recommended Operating Conditions

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

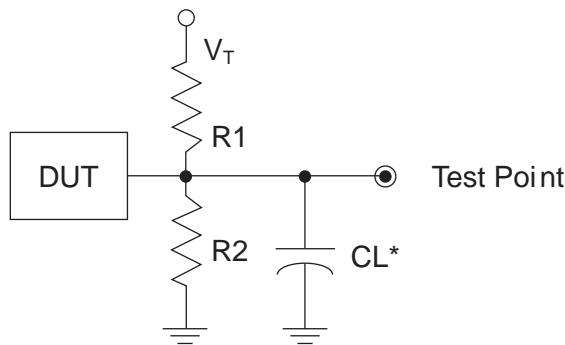
Figure 3-21. JTAG Port Timing Waveforms



## 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 3-22.

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



\*CL Includes Test Fixture and Probe Capacitance

**Table 3-22. 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 LVC MOS settings (L → H, H → L)	$\infty$	$\infty$	0pF	LVC MOS 3.3 = V <sub>CCIO</sub> /2	—
				LVC MOS 2.5 = V <sub>CCIO</sub> /2	—
				LVC MOS 1.8 = V <sub>CCIO</sub> /2	—
				LVC MOS 1.5 = V <sub>CCIO</sub> /2	—
				LVC MOS 1.2 = V <sub>CCIO</sub> /2	—
LVC MOS 2.5 I/O (Z → H)	$\infty$	1MΩ	0pF	V <sub>CCIO</sub> /2	—
LVC MOS 2.5 I/O (Z → L)	1MΩ	$\infty$	0pF	V <sub>CCIO</sub> /2	V <sub>CCIO</sub>
LVC MOS 2.5 I/O (H → Z)	$\infty$	100	0pF	V <sub>OH</sub> - 0.10	—
LVC MOS 2.5 I/O (L → Z)	100	$\infty$	0pF	V <sub>OL</sub> + 0.10	V <sub>CCIO</sub>

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



# ECP5 Family Data Sheet

## Pinout Information

March 2014

Advance Data Sheet DS1044

### Signal Descriptions

Signal Name	I/O	Description
<b>General Purpose</b>		
P[L/R][Group Number]_[A/B/C/D]	I/O	<p>[L/R] indicates the L (Left), or R (Right) edge of the device.</p> <p>[Group Number] indicates the PIO [A/B/C/D] group.</p> <p>[A/B/C/D] indicates the PIO within the PIC to which the pad is connected.</p> <p>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 tristated with an internal pull-up resistor enabled. If any pin is not used (or not bonded to a package pin), it is also tristated with an internal pull-up resistor enabled after configuration.</p> <p>PIO A and B are grouped as a pair, and PIO C and D are group as a pair. Each pair supports true LVDS differential input buffer. Only PIO A and B pair supports true LVDS differential output buffer.</p> <p>Each A/B and C/D pair supports programmable on/off differential input termination of 100 ohms.</p>
P[T/B][Group Number]_[A/B]	I/O	<p>[T/B] indicates the T (top) or B (bottom) edge of the device.</p> <p>[Group Number] indicates the PIO [A/B] group.</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 sysConfig pins. These pins, when not used as configuration pins, can be programmed as I/Os for user logic. During configuration, the pins not used in configuration are tristated with an internal pull-up resistor enabled. If any pin is not used (or not bonded to a package pin), They are also tristated with an internal pull-up resistor enabled after configuration.</p> <p>PIOS on top and bottom do not support differential input signaling or true LVDS output signaling, but it can support emulated differential output buffer. PIO A/B forms a pair of emulated differential output buffer.</p>
GSRN	I	Global RESET signal (active low). Any I/O pin can be GSRN.
NC	—	No connect.
RESERVED	—	This pin is reserved and should not be connected to anything on the board.
GND	—	Ground. Dedicated pins.
VCC	—	Power supply pins for core logic. Dedicated pins. VCC = 1.1V
VCCAUX	—	Auxiliary power supply pin. This dedicated pin powers all the differential and referenced input buffers. VCCAUX = 2.5 V.
VCCIOx	—	Dedicated power supply pins for I/O bank x. VCCIO8 is used for configuration and JTAG.
VREF1_x	—	Reference supply pins for I/O bank x. Pre-determined shared pin in each bank are assigned as VREF1 input. When not used, they may be used as I/O pins.
<b>PLL, DLL and Clock Functions</b>		
[LOC]_[GPLL][T, C]_IN	I	General Purpose PLL (GPLL) input pads: [LOC] = ULC, LLC, URC and LRC, T = true and C = complement. These pins are shared I/O pins. When not configured as GPLL input pads, they can be used as general purpose I/O pins.

<b>Signal Name</b>	<b>I/O</b>	<b>Description</b>
GR_PCLK[Bank][num]	I	General Routing Signals in Banks 0, 1, 2, 3, 4, 6 and 7. There are 2 in each bank ([num] = 0, 1). Please refer to ECP5 sysCLOCK Usage Guide. These pins are shared I/O pins. When not configured as GR pins, they can be used as general purpose I/O pins.
PCLK[T/C][Bank]_[num]	I/O	General Purpose Primary CLK pads: [T/C] = True/Complement, [Bank] = (0, 1, 2, 3, 6 and 7). There are 2 in each bank ([num] = 0, 1). These are shared I/O pins. When not configured as PCLK pins, they can be used as general purpose I/O pins.
[L/R]DQS[group_num]	I/O	DQS input/output pads: T (top), R (right), group_num = ball number associated with DQS[T] pin.
[T/R]DQ[group_num]	I/O	DQ input/output pads: T (top), R (right), group_num = ball number associated with DQS[T] pin.
<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. This is a dedicated input pin.
TCK	I	Test Clock input pin, used to clock the 1149.1 state machine. No pull-up enabled. This is a dedicated input pin.
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. This is a dedicated input pin.
TDO	O	Output pin. Test Data Out pin used to shift data out of a device using 1149.1. This is a dedicated output pin.
<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. This 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	Input Configuration Clock for configuring an FPGA in Slave SPI, Serial, and CPU modes. Output Configuration Clock for configuring an FPGA in Master configuration modes (Master SPI, Master Serial). This is a dedicated pin.
HOLDN/DI/BUSY/CSSPIN/CEN	I/O	Parallel configuration mode busy indicator. SPI/SPIIm mode data output. This is a shared I/O pin. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
CSN/SN	I/O	Parallel configuration mode active-low chip select. Slave SPI chip select. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
CS1N	I	Parallel configuration mode active-low chip select. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
WRITEN	I	Write enable for parallel configuration modes. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.

Signal Name	I/O	Description
DOUT/CS0N	O	Serial data output. Chip select output. SPI/SPIIm mode chip select. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
D0/MOSI/IO0	I/O	Parallel configuration I/O. Open drain during configuration. When in SPI modes, it is an output in Master mode, and input in Slave mode. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
D1/MISO/IO1	I/O	Parallel configuration I/O. Open drain during configuration. When in SPI modes, it is an input in Master mode, and output in Slave mode. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
D2/IO2	I/O	Parallel configuration I/O. Open drain during configuration. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
D3/IO3	I/O	Parallel configuration I/O. Open drain during configuration. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
D4/MOSI2/IO4	I/O	Parallel configuration I/O. Open drain during configuration. When in SPI modes, it is an output in Master mode, and input in Slave mode. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
D5/MISO/IO5	I/O	Parallel configuration I/O. Open drain during configuration. When in SPI modes, it is an output in Master mode, and input in Slave mode. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
D6/IO6	I/O	Parallel configuration I/O. Open drain during configuration. When in SPI modes, it is an output in Master mode, and input in Slave mode. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin.
D7/IO7	I/O	Parallel configuration I/O. Open drain during configuration. When in SPI modes, it is an output in Master mode, and input in Slave mode. This is a shared I/O pin. When not in configuration, it can be used as general purpose I/O pin
<b>SERDES Function</b>		
VCCA <sub>x</sub>	—	SERDES, transmit, receive, PLL and reference clock buffer power supply for SERDES Dual x. All VCCA supply pins must always be powered to the recommended operating voltage range. If no SERDES channels are used, connect VCCA to VCC. VCCA <sub>x</sub> = 1.1 V.
VCCAUXA <sub>x</sub>	—	SERDES Aux Power Supply pin for SERDES Dual x. VCCAUXA <sub>x</sub> = 2.5 V.
HDRX[P/N]_D[dual_num]CH[chan_num]	I	High-speed SERDES inputs, P = Positive, N = Negative, dual_num = [0, 1], chan_num = [0, 1]. These are dedicated SERDES input pins.
HDTX[P/N]_D[dual_num]CH[chan_num]	O	High-speed SERDES outputs, P = Positive, N = Negative, dual_num = [0, 1], chan_num = [0, 1]. These are dedicated SERDES output pins.
REFCLK[P/N]_D[dual_num]	I	SERDES Reference Clock inputs, P = Positive, N = Negative, dual_num = [0, 1]. These are dedicated SERDES input pins.
VCCHRX_D[dual_num]CH[chan_num]	—	SERDES High-Speed Inputs Termination Voltage Supplies, dual_num = [0, 1], chan_num = [0, 1]. These can be powered to 1.1 V or 1.2 V. If powered to 1.1V, it is recommended to connect it to VCCA <sub>x</sub> .
VCCHTX_D[dual_num]CH[chan_num]	—	SERDES High-Speed Outputs Buffer Voltage Supplies, dual_num = [0, 1], chan_num = [0, 1]. These can be powered to 1.1 V or 1.2 V.

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

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

PICs Associated with DQS Strobe	PIO Within PIC	DDR Strobe (DQS) and Data (DQ) Pins
<b>For Left and Right Edges of the Device Only</b>		
P[L/R] [n-6]	A	DQ
	B	DQ
	C	DQ
	D	DQ
P[L/R] [n-3]	A	DQ
	B	DQ
	C	DQ
	D	DQ
P[L/R] [n]	A	DQS (P)
	B	DQS (N)
	C	DQ
	D	DQ
P[L/R] [n+3]	A	DQ
	B	DQ
	C	DQ
	D	DQ

Note: "n" is a row PIC number.

## Pin Information Summary

### LFE5UM

Pin Information Summary		LFE5UM-25		LFE5UM-45			LFE5UM-85			
Pin Type		285 csfBGA	381 caBGA	285 csf- BGA	381 caBGA	554 caBGA	285 csfBGA	381 caBGA	554 caBGA	756 caBGA
General Purpose Inputs/Outputs per Bank	Bank 0	6	24	6	27	32	6	27	32	56
	Bank 1	6	32	6	33	40	6	33	40	48
	Bank 2	21	32	21	32	32	21	34	32	48
	Bank 3	28	32	28	33	48	28	33	48	64
	Bank 4	0	0	0	0	0	0	0	14	24
	Bank 6	26	32	26	33	48	26	33	48	64
	Bank 7	18	32	18	32	32	18	32	32	48
	Bank 8	13	13	13	13	13	13	13	13	13
Total Single-Ended User I/O		118	197	118	203	245	118	205	259	365
VCC		13	20	13	20	24	13	20	24	36
VCCAUX (Core)		3	4	3	4	9	3	4	9	8
VCCIO	Bank 0	1	2	1	2	3	1	2	3	4
	Bank 1	1	2	1	2	3	1	2	3	4
	Bank 2	2	3	2	3	4	2	3	4	4
	Bank 3	2	3	2	3	3	2	3	3	4
	Bank 4	0	0	0	0	0	0	0	2	2
	Bank 6	2	3	2	3	4	2	3	4	4
	Bank 7	2	3	2	3	3	2	3	3	4
	Bank 8	2	2	2	2	2	2	2	2	2
TAP		4	4	4	4	4	4	4	4	4
Miscellaneous Dedicated Pins		7	7	7	7	7	7	7	7	7
GND		83	59	83	59	113	83	59	113	166
NC		1	8	1	2	33	1	0	17	29
Reserved		0	2	0	2	4	0	2	4	4
SERDES		14	28	14	28	28	14	28	28	28
VCCA (SERDES)	VCCA0	2	2	2	2	6	2	2	6	8
	VCCA1	0	2	0	2	6	0	2	6	9
VCCAUX (SERDES)	VCCAUXA0	2	2	2	2	2	2	2	2	2
	VCCAUXA1	0	2	0	2	2	0	2	2	2
GNDA (SERDES)			26	26	26	26	49	26	26	49
Total Balls			285	381	285	381	554	285	381	554
										756

Pin Information Summary		LFE5UM-25		LFE5UM-45			LFE5UM-85			
Pin Type		285 csfBGA	381 caBGA	285 csfBGA	381 caBGA	554 caBGA	285 csfBGA	381 caBGA	554 caBGA	756 caBGA
High Speed Differential Input / Output Pairs	Bank 0	0	0	0	0	0	0	0	0	0
	Bank 1	0	0	0	0	0	0	0	0	0
	Bank 2	10/8	16/8	10/8	16/8	16/8	10/8	17/9	16/8	24/12
	Bank 3	14/7	16/8	14/7	16/8	24/12	14/7	16/8	24/12	32/16
	Bank 4	0	0	0	0	0	0	0	0	0
	Bank 6	13/6	16/8	13/6	16/8	24/12	13/6	16/8	24/12	32/16
	Bank 7	8/6	16/8	8/6	16/8	16/8	8/6	16/8	16/8	24/12
	Bank 8	0	0	0	0	0	0	0	0	0
Total High Speed Differential I/O Pairs		45/27	64/32	45/27	64/32	80/40	45/27	65/33	80/40	112/56
DQS Groups (> 11 pins in group)	Bank 0	0	0	0	0	0	0	0	0	0
	Bank 1	0	0	0	0	0	0	0	0	0
	Bank 2	1	2	1	2	2	1	2	2	3
	Bank 3	2	2	2	2	3	2	2	3	4
	Bank 4	0	0	0	0	0	0	0	0	0
	Bank 6	2	2	2	2	3	2	2	3	4
	Bank 7	1	2	1	2	2	1	2	2	3
	Bank 8	0	0	0	0	0	0	0	0	0
Total DQS Groups		6	8	6	8	10	6	8	10	14

**LFE5U**

Pin Information Summary		LFE5U-25		LFE5U-45			LFE5U-85			
Pin Type		285 csfBGA	381 caBGA	285 csfBGA	381 caBGA	554 caBGA	285 csfBGA	381 caBGA	554 caBGA	756 caBGA
General Purpose Inputs/Outputs per Bank	Bank 0	6	24	6	27	32	6	27	32	56
	Bank 1	6	32	6	33	40	6	33	40	48
	Bank 2	21	32	21	32	32	21	34	32	48
	Bank 3	28	32	28	33	48	28	33	48	64
	Bank 4	0	0	0	0	0	0	0	14	24
	Bank 6	26	32	26	33	48	26	33	48	64
	Bank 7	18	32	18	32	32	18	32	32	48
	Bank 8	13	13	13	13	13	13	13	13	13
Total Single-Ended User I/O		118	197	118	203	245	118	205	259	365
VCC		13	20	13	20	24	13	20	24	36
VCCAUX (Core)		3	4	3	4	9	3	4	9	8
VCCIO	Bank 0	1	2	1	2	3	1	2	3	4
	Bank 1	1	2	1	2	3	1	2	3	4
	Bank 2	2	3	2	3	4	2	3	4	4
	Bank 3	2	3	2	3	3	2	3	3	4
	Bank 4	0	0	0	0	0	0	0	2	2
	Bank 6	2	3	2	3	4	2	3	4	4
	Bank 7	2	3	2	3	3	2	3	3	4
	Bank 8	2	2	2	2	2	2	2	2	2
TAP		4	4	4	4	4	4	4	4	4
Miscellaneous Dedicated Pins		7	7	7	7	7	7	7	7	7
GND		123	59	123	59	113	123	59	113	166
NC		1	8	1	2	33	1	0	17	29
Reserved		4	2	4	2	4	4	2	4	4
SERDES		0	28	0	28	28	0	28	28	28
VCCA (SERDES)	VCCA0	0	2	0	2	6	0	2	6	8
	VCCA1	0	2	0	2	6	0	2	6	9
VCCAUX (SERDES)	VCCAUXA0	0	2	0	2	2	0	2	2	2
	VCCAUXA1	0	2	0	2	2	0	2	2	2
GNDA (SERDES)		0	26	0	26	49	0	26	49	60
Total Balls		285	381	285	381	554	285	381	554	756

Pin Information Summary		LFE5U-25		LFE5U-45			LFE5U-85			
Pin Type		285 csfBGA	381 caBGA	285 csfBGA	381 caBGA	554 caBGA	285 csfBGA	381 caBGA	554 caBGA	756 caBGA
High Speed Differential Input / Output Pairs	Bank 0	0	0	0	0	0	0	0	0	0
	Bank 1	0	0	0	0	0	0	0	0	0
	Bank 2	10/8	16/8	10/8	16/8	16/8	10/8	17/9	16/8	24/12
	Bank 3	14/7	16/8	14/7	16/8	24/12	14/7	16/8	24/12	32/16
	Bank 4	0	0	0	0	0	0	0	0	0
	Bank 6	13/6	16/8	13/6	16/8	24/12	13/6	16/8	24/12	32/16
	Bank 7	8/6	16/8	8/6	16/8	16/8	8/6	16/8	16/8	24/12
	Bank 8	0	0	0	0	0	0	0	0	0
Total High Speed Differential I/O Pairs		45/27	64/32	45/27	64/32	80/40	45/27	65/33	80/40	112/56
DQS Groups (> 11 pins in group)	Bank 0	0	0	0	0	0	0	0	0	0
	Bank 1	0	0	0	0	0	0	0	0	0
	Bank 2	1	2	1	2	2	1	2	2	3
	Bank 3	2	2	2	2	3	2	2	3	4
	Bank 4	0	0	0	0	0	0	0	0	0
	Bank 6	2	2	2	2	3	2	2	3	4
	Bank 7	1	2	1	2	2	1	2	2	3
	Bank 8	0	0	0	0	0	0	0	0	0
Total DQS Groups		6	8	6	8	10	6	8	10	14



# ECP5 Family Data Sheet

## Ordering Information

March 2014

Advance Data Sheet DS1044

### ECP5 Part Number Description

LFE5U - XX - X XXXXX X

**Device Family**

LFE5U (LFE5 FPGA)

**Logic Capacity**

25F = 25K LUTs  
45F = 45K LUTs  
85F = 85K LUTs

**Speed**

6 = Slowest  
7  
8 = Fastest

**Grade**

C = Commercial  
I = Industrial

**Package**

MG285 = 285-ball csfBGA  
MG381 = 381-ball caBGA  
MG554 = 554-ball caBGA  
MG756 = 756-ball caBGA

LFE5UM - XX - X XXXXX X

**Device Family**

LFE5UM (LFE5 FPGA  
with SERDES)

**Logic Capacity**

25F = 25K LUTs  
45F = 45K LUTs  
85F = 85K LUTs

**Speed**

6 = Slowest  
7  
8 = Fastest

**Grade**

C = Commercial  
I = Industrial

**Package**

MG285 = 285-ball csfBGA  
MG381 = 381-ball caBGA  
MG554 = 554-ball caBGA  
MG756 = 756-ball caBGA

## Ordering Part Numbers

### Commercial

Part number	Grade	Package	Pins	Temp.	LUTs (K)	SERDES
LFE5U-25F-6MG285C	-6	Lead free csfBGA	285	Commercial	24	No
LFE5U-25F-7MG285C	-7	Lead free csfBGA	285	Commercial	24	No
LFE5U-25F-8MG285C	-8	Lead free csfBGA	285	Commercial	24	No
LFE5U-25F-6MG381C	-6	Lead free caBGA	381	Commercial	24	No
LFE5U-25F-7MG381C	-7	Lead free caBGA	381	Commercial	24	No
LFE5U-25F-8MG381C	-8	Lead free caBGA	381	Commercial	24	No
LFE5U-45F-6MG285C	-6	Lead free csfBGA	285	Commercial	44	No
LFE5U-45F-7MG285C	-7	Lead free csfBGA	285	Commercial	44	No
LFE5U-45F-8MG285C	-8	Lead free csfBGA	285	Commercial	44	No
LFE5U-45F-6MG381C	-6	Lead free caBGA	381	Commercial	44	No
LFE5U-45F-7MG381C	-7	Lead free caBGA	381	Commercial	44	No
LFE5U-45F-8MG381C	-8	Lead free caBGA	381	Commercial	44	No
LFE5U-45F-6MG554C	-6	Lead free caBGA	554	Commercial	44	No
LFE5U-45F-7MG554C	-7	Lead free caBGA	554	Commercial	44	No
LFE5U-45F-8MG554C	-8	Lead free caBGA	554	Commercial	44	No
LFE5U-85F-6MG285C	-6	Lead free csfBGA	285	Commercial	84	No
LFE5U-85F-7MG285C	-7	Lead free csfBGA	285	Commercial	84	No
LFE5U-85F-8MG285C	-8	Lead free csfBGA	285	Commercial	84	No
LFE5U-85F-6MG381C	-6	Lead free caBGA	381	Commercial	84	No
LFE5U-85F-7MG381C	-7	Lead free caBGA	381	Commercial	84	No
LFE5U-85F-8MG381C	-8	Lead free caBGA	381	Commercial	84	No
LFE5U-85F-6MG554C	-6	Lead free caBGA	554	Commercial	84	No
LFE5U-85F-7MG554C	-7	Lead free caBGA	554	Commercial	84	No
LFE5U-85F-8MG554C	-8	Lead free caBGA	554	Commercial	84	No
LFE5U-85F-6MG756C	-6	Lead free caBGA	756	Commercial	84	No
LFE5U-85F-7MG756C	-7	Lead free caBGA	756	Commercial	84	No
LFE5U-85F-8MG756C	-8	Lead free caBGA	756	Commercial	84	No
LFE5UM-25F-6MG285C	-6	Lead free csfBGA	285	Commercial	24	Yes
LFE5UM-25F-7MG285C	-7	Lead free csfBGA	285	Commercial	24	Yes
LFE5UM-25F-8MG285C	-8	Lead free csfBGA	285	Commercial	24	Yes
LFE5UM-25F-6MG381C	-6	Lead free caBGA	381	Commercial	24	Yes
LFE5UM-25F-7MG381C	-7	Lead free caBGA	381	Commercial	24	Yes
LFE5UM-25F-8MG381C	-8	Lead free caBGA	381	Commercial	24	Yes
LFE5UM-45F-6MG285C	-6	Lead free csfBGA	285	Commercial	44	Yes
LFE5UM-45F-7MG285C	-7	Lead free csfBGA	285	Commercial	44	Yes
LFE5UM-45F-8MG285C	-8	Lead free csfBGA	285	Commercial	44	Yes
LFE5UM-45F-6MG381C	-6	Lead free caBGA	381	Commercial	44	Yes
LFE5UM-45F-7MG381C	-7	Lead free caBGA	381	Commercial	44	Yes
LFE5UM-45F-8MG381C	-8	Lead free caBGA	381	Commercial	44	Yes
LFE5UM-45F-6MG554C	-6	Lead free caBGA	554	Commercial	44	Yes
LFE5UM-45F-7MG554C	-7	Lead free caBGA	554	Commercial	44	Yes
LFE5UM-45F-8MG554C	-8	Lead free caBGA	554	Commercial	44	Yes



**Ordering Information**  
**ECP5 Family Data Sheet**

Part number	Grade	Package	Pins	Temp.	LUTs (K)	SERDES
LFE5UM-85F-6MG285C	-6	Lead free csfBGA	285	Commercial	84	Yes
LFE5UM-85F-7MG285C	-7	Lead free csfBGA	285	Commercial	84	Yes
LFE5UM-85F-8MG285C	-8	Lead free csfBGA	285	Commercial	84	Yes
LFE5UM-85F-6MG381C	-6	Lead free caBGA	381	Commercial	84	Yes
LFE5UM-85F-7MG381C	-7	Lead free caBGA	381	Commercial	84	Yes
LFE5UM-85F-8MG381C	-8	Lead free caBGA	381	Commercial	84	Yes
LFE5UM-85F-6MG554C	-6	Lead free caBGA	554	Commercial	84	Yes
LFE5UM-85F-7MG554C	-7	Lead free caBGA	554	Commercial	84	Yes
LFE5UM-85F-8MG554C	-8	Lead free caBGA	554	Commercial	84	Yes
LFE5UM-85F-6MG756C	-6	Lead free caBGA	756	Commercial	84	Yes
LFE5UM-85F-7MG756C	-7	Lead free caBGA	756	Commercial	84	Yes
LFE5UM-85F-8MG756C	-8	Lead free caBGA	756	Commercial	84	Yes

**Industrial**

Part number	Grade	Package	Pins	Temp.	LUTs (K)	SERDES
LFE5U-25F-6MG285I	-6	Lead free csfBGA	285	Industrial	24	No
LFE5U-25F-7MG285I	-7	Lead free csfBGA	285	Industrial	24	No
LFE5U-25F-8MG285I	-8	Lead free csfBGA	285	Industrial	24	No
LFE5U-25F-6MG381I	-6	Lead free caBGA	381	Industrial	24	No
LFE5U-25F-7MG381I	-7	Lead free caBGA	381	Industrial	24	No
LFE5U-25F-8MG381I	-8	Lead free caBGA	381	Industrial	24	No
LFE5U-45F-6MG285I	-6	Lead free csfBGA	285	Industrial	44	No
LFE5U-45F-7MG285I	-7	Lead free csfBGA	285	Industrial	44	No
LFE5U-45F-8MG285I	-8	Lead free csfBGA	285	Industrial	44	No
LFE5U-45F-6MG381I	-6	Lead free caBGA	381	Industrial	44	No
LFE5U-45F-7MG381I	-7	Lead free caBGA	381	Industrial	44	No
LFE5U-45F-8MG381I	-8	Lead free caBGA	381	Industrial	44	No
LFE5U-45F-6MG554I	-6	Lead free caBGA	554	Industrial	44	No
LFE5U-45F-7MG554I	-7	Lead free caBGA	554	Industrial	44	No
LFE5U-45F-8MG554I	-8	Lead free caBGA	554	Industrial	44	No
LFE5U-85F-6MG285I	-6	Lead free csfBGA	285	Industrial	84	No
LFE5U-85F-7MG285I	-7	Lead free csfBGA	285	Industrial	84	No
LFE5U-85F-8MG285I	-8	Lead free csfBGA	285	Industrial	84	No
LFE5U-85F-6MG381I	-6	Lead free caBGA	381	Industrial	84	No
LFE5U-85F-7MG381I	-7	Lead free caBGA	381	Industrial	84	No
LFE5U-85F-8MG381I	-8	Lead free caBGA	381	Industrial	84	No
LFE5U-85F-6MG554I	-6	Lead free caBGA	554	Industrial	84	No
LFE5U-85F-7MG554I	-7	Lead free caBGA	554	Industrial	84	No
LFE5U-85F-8MG554I	-8	Lead free caBGA	554	Industrial	84	No
LFE5U-85F-6MG756I	-6	Lead free caBGA	756	Industrial	84	No
LFE5U-85F-7MG756I	-7	Lead free caBGA	756	Industrial	84	No
LFE5U-85F-8MG756I	-8	Lead free caBGA	756	Industrial	84	No
LFE5UM-25F-6MG285I	-6	Lead free csfBGA	285	Industrial	24	Yes
LFE5UM-25F-7MG285I	-7	Lead free csfBGA	285	Industrial	24	Yes
LFE5UM-25F-8MG285I	-8	Lead free csfBGA	285	Industrial	24	Yes
LFE5UM-25F-6MG381I	-6	Lead free caBGA	381	Industrial	24	Yes
LFE5UM-25F-7MG381I	-7	Lead free caBGA	381	Industrial	24	Yes
LFE5UM-25F-8MG381I	-8	Lead free caBGA	381	Industrial	24	Yes
LFE5UM-45F-6MG285I	-6	Lead free csfBGA	285	Industrial	44	Yes
LFE5UM-45F-7MG285I	-7	Lead free csfBGA	285	Industrial	44	Yes
LFE5UM-45F-8MG285I	-8	Lead free csfBGA	285	Industrial	44	Yes
LFE5UM-45F-6MG381I	-6	Lead free caBGA	381	Industrial	44	Yes
LFE5UM-45F-7MG381I	-7	Lead free caBGA	381	Industrial	44	Yes
LFE5UM-45F-8MG381I	-8	Lead free caBGA	381	Industrial	44	Yes
LFE5UM-45F-6MG554I	-6	Lead free caBGA	554	Industrial	44	Yes
LFE5UM-45F-7MG554I	-7	Lead free caBGA	554	Industrial	44	Yes
LFE5UM-45F-8MG554I	-8	Lead free caBGA	554	Industrial	44	Yes
LFE5UM-85F-6MG285I	-6	Lead free csfBGA	285	Industrial	84	Yes
LFE5UM-85F-7MG285I	-7	Lead free csfBGA	285	Industrial	84	Yes



**Ordering Information**  
**ECP5 Family Data Sheet**

Part number	Grade	Package	Pins	Temp.	LUTs (K)	SERDES
LFE5UM-85F-8MG285I	-8	Lead free csfBGA	285	Industrial	84	Yes
LFE5UM-85F-6MG381I	-6	Lead free caBGA	381	Industrial	84	Yes
LFE5UM-85F-7MG381I	-7	Lead free caBGA	381	Industrial	84	Yes
LFE5UM-85F-8MG381I	-8	Lead free caBGA	381	Industrial	84	Yes
LFE5UM-85F-6MG554I	-6	Lead free caBGA	554	Industrial	84	Yes
LFE5UM-85F-7MG554I	-7	Lead free caBGA	554	Industrial	84	Yes
LFE5UM-85F-8MG554I	-8	Lead free caBGA	554	Industrial	84	Yes
LFE5UM-85F-6MG756I	-6	Lead free caBGA	756	Industrial	84	Yes
LFE5UM-85F-7MG756I	-7	Lead free caBGA	756	Industrial	84	Yes
LFE5UM-85F-8MG756I	-8	Lead free caBGA	756	Industrial	84	Yes



# ECP5 Family Data Sheet

## Supplemental Information

---

March 2014

Advance Data Sheet DS1044

---

### For Further Information

A variety of technical notes for the ECP5 family are available.

- TN1260, ECP5 sysCONFIG Usage Guide
- TN1261, ECP5 SERDES/PCS Usage Guide
- TN1262, ECP5 sysIO Usage Guide
- TN1263, ECP5 sysClock PLL/DLL Design and Usage Guide
- TN1264, ECP5 Memory Usage Guide
- TN1265, ECP5 High-Speed I/O Interface
- TN1266, Power Consumption and Management for ECP5 Devices
- TN1267, ECP5 sysDSP Usage Guide

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

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



# ECP5 Family Data Sheet

## Revision History

---

March 2014

Advance Data Sheet DS1044

---

Date	Version	Section	Change Summary
March 2014	01.0	All	Initial release.



## **Section II. LatticeECP5 Family Technical Notes**

---

## Introduction

The configuration memory in ECP5™ 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 ECP5 device when it powers up or anytime the device needs to be updated. The ECP5 device provides a rich set of features for programming and configuration of the FPGA. You have many options available to you for building the programming solution that fits your needs. Each of the options available will be described in detail so that you can put together the programming and configuration solution that meets your needs. Waveforms presented in this document are for reference only and all detailed timing recommendations should reference DS1044, [ECP5 Family Data Sheet](#).

## Features

Key programming and configuration features of ECP5 devices are:

- Multiple programming and configuration interfaces:
  - 1149.1 JTAG
  - Slave Parallel
  - Slave Serial
  - Slave SPI
  - Master SPI, includes SERIAL, DUAL and QUAD read mode
  - Dual Boot and Multi Boot support
  - Daisy chaining
- Transparent programming of external memory
- Readback security and encryption for design protection
- Compression of bitstreams

## Definition of Terms

This document uses the following terms to describe common functions:

- **Programming:** Programming refers to the process used to alter the contents of the external configuration memory.
- **Configuration:** Configuration refers to a change in the state of the SRAM memory cells.
- **Configuration Mode:** The configuration mode defines the method the device uses to acquire the configuration data from the volatile memory.
- **Configuration Data** – This is the data read from the non-volatile memory and loaded into the FPGA's SRAM configuration memory. This is also referred to as a bitstream, or device bitstream.
- **BIT** – The BIT file is the configuration data for the device that is stored in an external SPI Flash or other memory device. It is a binary file and is programmed unmodified into the SPI Flash by Lattice programming tool.
- **HEX** - The HEX file is also a configuration data file for the device. It is in HEX format and is normally requested by third party programming vendors.
- **Port** – A port refers to the physical connection used to perform programming and some configuration operations. Ports on the ECP5 include JTAG and SPI physical connections.

- **User Mode** – The ECP5 device is in user mode when configuration is complete, and the FPGA is performing the logic functions you have programmed it to perform.
- **Offline Mode** – Offline mode is a term that is applied to SRAM configuration or external memory programming. When using offline mode, the FPGA no longer operates in user mode. The contents of the SRAM configuration memory or external memory device are updated. The FPGA does not perform your logic operations until offline mode programming/configuration is complete.
- **Direct Mode (Foreground Mode)** – The device is in a configuration mode and all the I/O pins are kept tri-stated.
- **Background Mode** – The device is in a configuration mode where all the I/O pins remain operational.
- **Dual Boot** – Supports two configuration patterns that reside in a SPI Flash device. Whenever loading failure occurs with the primary pattern, the FPGA device searches for and loads a so-called ‘golden’ or fail-safe pattern. Both patterns come from an off-chip non-volatile SPI memory.
- **Multi Boot** – The FPGA device determines and triggers the loading of the next pattern after a prior successful configuration. Multiple patterns (i.e., more than two patterns) are available for the FPGA device to choose to load on demand. All the patterns are stored in an external SPI Flash memory.
- **Transparent Mode** – Transparent mode is used to update the SRAM configuration ‘while leaving the ECP5 device in User Mode.
- **Number Formats** – The following nomenclature is used to denote the radix of numbers
  - 0x: Numbers preceded by ‘0x’ are hexadecimal
  - b (suffix): Numbers suffixed with ‘b’ are binary
  - All other numbers are decimal
- **SPI** – Serial Peripheral Interface Bus. An industry standard, full duplex, synchronous serial data link that uses a four wire interface. The interface supports a single master and single or multiple slaves.
- **Master SPI** – A configuration mode where the FPGA drives the master clock and issues commands to read the bitstream from an external SPI Flash device.
- **Slave SPI (SSPI)** – A configuration mode where the CPU drives the clock and issues commands to the FPGA for writing the bitstream into the SRAM cells.
- **Refresh** – The process of re-triggering a bitstream write operation. It is activated by toggling of the PROGRAMN pin or issuing a REFRESH command, which emulates the PROGRAMN pin toggling. Only the JTAG port and the Slave SPI port support the REFRESH command.

## Configuration Details

The ECP5 SRAM memory contains the active configuration, essentially the “fuses” that define the behavior of the FPGA. The active configuration is, in most cases, retrieved from an external memory. The non-volatile memory holds the configuration data that is loaded into the FPGAs SRAM.

## Bitstream/PROM Sizes

The ECP5 is an SRAM based FPGA. The SRAM configuration memory must be loaded from an external non-volatile memory that can store all of the configuration data. The size of the configuration data is variable. It is based on the amount of logic available in the FPGA, and the number of pre-initialized Embedded Block RAM (EBR) components. An ECP5 design using the largest device, with every EBR pre-initialized with unique data values, and generated without compression turned on requires the largest amount of storage.

**Table 8-1. Maximum Configuration Bits<sup>1</sup>**

Device	All Uncompressed	SPI Mode	
	Unencrypted/Encrypted Bitstream Size (Mb)	Recommended SPI Flash Size (Mb)	Dual Boot Recommended SPI Flash Size (Mb)
LFE5-25 No EBR	4.45	8	16
LFE5-25 Max EBR	5.42	8	16
LFE5-45 No EBR	7.86	8	16
LFE5-45 Max EBR	9.74	16	32
LFE5-85 No EBR	14.71	16	32
LFE5-85 Max EBR	18.35	32	64

1. Both unencrypted and encrypted bitstreams are the same size. Compression ratio depends on bitstream, so we only provide uncompressed bitstream data.

## Device Status Register

The ECP5 device has a 32-bit device status register, which indicates the status of the device. The READ\_STATUS command shifts out this 32-bit internal status of the device. The first bit shifted out on the SO pin is bit 0, and the last bit is bit 31. The device status register can only be read by JTAG, Slave SPI and Slave Parallel port.

**Table 8-2. 32-Bit Device Status Register**

Bit	Function	Status Value			Comments
		Reset Value	0	1	
0	Transparent Mode	0	No	Yes	Device is currently in Transparent mode
[3:1]	Config Target Selection	000			000 SRAM array
					001 Efuse
4	JTAG Active	0	No	Yes	JTAG State Machine is currently active
5	PWD Protection	0	No	Yes	Configuration logic is password protected
6	Internal use	0			Not used
7	Decrypt Enable	0	No	Yes	Encrypted bitstreams will be accepted
8	DONE	0	No	Yes	Done bit has been set
9	ISC Enable	0	No	Yes	JTAG instructions are being executed with ISC Enabled
10	Write Enable	0	Not Writable	Writable	Selected configuration target is write-protected from at least one of the following setup: 1. Selected configuration target security bit is set 2. Password protection is enabled and password is mismatch
11	Read Enable	0	Not Readable	Readable	Read-protected from at least one of the following setup: 1. Security bit is set 2. Password protection is enabled and password is mismatch
12	Busy Flag	0	No	Yes	Configuration logic is busy
13	Fail Flag	0	No	Yes	Last instruction/command execution failed
14	FEA OTP	0			Feature row is set to be One-Time Programmable
15	Decrypt Only	0	No	Yes	Only encrypted data will be accepted

Bit	Function	Status Value			Comments
		Reset Value	0	1	
16	PWD Enable	0	No	Yes	Password protection is enabled
17	Internal use	0			Not used
18	Internal use	0			Not used
19	Internal use	0			Not used
20	Encrypt Preamble	0	No	Yes	Encrypted Preamble is detected
21	Std Preamble	0	No	Yes	Standard Preamble is detected
22	SPIm Fail 1	0	No	Yes	Failed to configure from the primary pattern
[25:23]	BSE Error Code	000			000 No error 001 ID error 010 CMD error - illegal command detected 011 CRC error 100 PRMB error - preamble error 101 ABRT Error - configuration aborted by the user 110 OVFL Error - data overflow error 111 SDM Error - bitstream pass the size of the SRAM array
26	Execution Error	0	No	Yes	Error occur during execution.
27	ID Error	0	No	Yes	ID mismatch with Verify_ID command
28	Invalid Command	0	No	Yes	Invalid command received
29	SED Error	0	No	Yes	SED error detected
30	Bypass Mode	0	No	Yes	Device currently in Bypass mode
31	Flow Through Mode	0	No	Yes	Device currently in Flow Through Mode

## sysCONFIG™ Ports

**Table 8-3. ECP5 Programming and Configuration Ports**

Interface	Port	Description
JTAG	JTAG (IEEE 1149.1 and IEEE 1532 compliant)	4-wire or 5-wire JTAG Interface
sysCONFIG	SSPI	Slave Serial Peripheral Interface (SPI)
	MSPI (SERIAL, DUAL, QUAD)	Master Serial Peripheral Interface (SPI)
	SPCM	Slave-Parallel (CPU) Interface
	SCM	Slave Serial Interface for daisy chain configuration

## sysCONFIG Pins

The ECP5 device provides a set of sysCONFIG I/O pins that you use to program and configure the FPGA. The sysCONFIG pins are grouped together to create ports (i.e. JTAG, SSPI, MSPI) that are used to interact with the FPGA for programming, configuration, and access of resources inside the FPGA. The sysCONFIG pins in a configuration port group may be active, and used for programming the FPGA, or they can be reconfigured to act as general purpose I/O excluding the dedicated JTAG pins.

Recovering the configuration port pins for use as general purpose I/O requires you to adhere to the following guidelines:

- You must DISABLE the unused port. You can accomplish this by using the Lattice Diamond® Spreadsheet View's Global Preferences tab. Each configuration port is listed in the sysCONFIG options tree.
- You must prevent external logic from interfering with device programming.

Table 8-4 lists the shared sysCONFIG pins of the device, and the default state of these pins in user mode. After programming the ECP5 device, the default state of the SSPI sysCONFIG pins become general purpose I/O. This means you lose the ability to program the ECP5 device using SSPI or SPCM when using the default sysCONFIG port settings. To retain the SSPI or SPCM sysCONFIG pins in user mode, be sure to ENABLE them using the Diamond Spreadsheet View editor.

Unless specified otherwise, the sysCONFIG pins are powered by the VCCIO8 voltage. It is crucial for this to be taken into consideration when provisioning other logic attached to Bank 8.

The function of each sysCONFIG pin is described in detail.

**Table 8-4. Default State of the sysCONFIG Pins**

sysCONFIG Pins		Pull During Configuration	Configuration Modes			
Name	Type		Slave SPI	Slave Serial	Slave Parallel	Master SPI SERIAL(S), DUAL(D), QUAD(Q),
CFGMDN[2:0]	Dedicated	UP	=001	=101	=111	=010
PROGRAMN	Dedicated	UP				PROGRAMN
INITN	Dedicated	UP				INITN
DONE	Dedicated	UP				DONE
MCLK/CCLK	Dedicated	UP	CCLK	CCLK	CCLK	MCLK
D7	Shared	NO	—	—	D7 <sup>3</sup>	—
D6	Shared	NO	—	—	D6 <sup>3</sup>	—
D5	Shared	NO	—	—	D5 <sup>3</sup>	—
D4	Shared	NO	—	—	D4 <sup>3</sup>	—
D3/IO3	Shared	NO	—	—	D3 <sup>3</sup>	(S)N/C, (D)N/C, (Q)IO3
D2/IO2	Shared	NO	—	—	D2 <sup>3</sup>	(S)N/C, (D)N/C, (Q)IO2
D1/MISO	Shared	NO	MISO	—	D1 <sup>3</sup>	MISO
D0/MOSI	Shared	NO	MOSI	—	D0 <sup>3</sup>	MOSI
CSN/SN	Shared	UP	SN <sup>1</sup>	—	CSN	—
CS1N	Shared	UP	—	—	CS1N	—
WRITEN	Shared	UP	—	—	WRITEN	—
HOLDN/DI/BUSY/ CSSPIN/ CEN	Shared	UP	HOLDN	DI	BUSY	(S)(D)(Q)CSSPIN <sup>2</sup>
DOUT/CSON	Shared	UP	DOUT	DOUT	CSON	DOUT

1. SN should have 4.7k-ohm pullup resistor on-board for SSPI.

2. CSSPIN should have 4.7k-ohm pullup on-board resistor for MSPI.

3. D[7:0] should have 10k-ohm pullup resistor on-board for SPCM.

4. MOSI and MISO should have 10k-ohm pullup resistor on-board for MSPI.

5. IO[3:2] should have 10k-ohm pullup resistor on-board for QUAD MSPI.

## Dedicated sysCONFIG Pins

CFGMDN[2:0]: The Configuration Mode pins, CFGMDN[2:0], are dedicated inputs with weak pull-ups. The CFGMDN pins are sampled on the rising edge of INITN and are used to select the configuration mode (i.e. what type of device the ECP5 device will configure from). External <500-ohm pull-down resistors ensure that the CFGMDN pin senses a low (or logic '0'). 4.7k-ohm pullup resistors are recommended to assure proper operation. The JTAG TAP port remains operative at all times, independent of the CFGMDN[2:0] setting. As a consequence the CFGMDN pins determine which groups of dual-purpose pins will be used for device configuration.

**Table 8-5. CFGMDN Mode Settings**

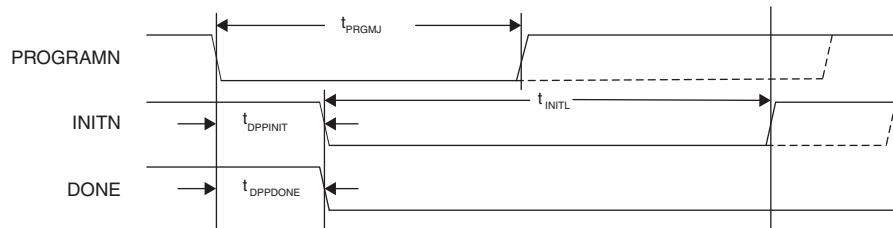
Configuration Mode	Bus Size	Options	Clock	CFGMDN2	CFGMDN1	CFGMDN0
SSPI	1		CCLK	0	0	1
MSPI	1	Serial (SPI_Serial)	MCLK	0	1	0
	2	Dual (SPI_Dual)				
	4	Quad (SPI_Quad)				
	1, 2, 4	Dual-Boot				
	1, 2, 4	Multi-Boot				
SCM (Slave_Serial)	1		CCLK	1	0	1
SPCM (Slave_Parallel)	8		CCLK	1	1	1

PROGRAMN: PROGRAMN is an input used to configure the FPGA. The PROGRAMN pin is low level sensitive, and has an internal weak pull-up. When PROGRAMN is asserted low, the FPGA exits user mode and starts a device configuration sequence at the Initialization phase, as described earlier. Holding the PROGRAMN pin low prevents the ECP5 device from leaving the Initialization phase. The PROGRAMN has a minimum pulse width assertion period in order for it to be recognized by the FPGA. You can find this minimum time in the AC timing section of DS1044, [ECP5 Family Data Sheet](#).

Be aware of the following special cases when the PROGRAMN pin is active:

- If the device is currently being programmed via JTAG then PROGRAMN will be ignored until the JTAG mode programming sequence is complete.
- Toggling the PROGRAMN pin during device configuration will interrupt the process and restart the configuration cycle.
- PROGRAMN must not be asserted low until after all power rails have reached stable operation. This can be achieved by either placing an external pullup resistor and tying it to the VCCIO8 voltage, or permitting the FPGA's internal pull-up resistor to pull the input high.
- PROGRAMN must not make a falling edge transition during the time the FPGA is in the Initialization state. PROGRAMN must be asserted for a minimum low period of tPRGMRJ in order for it to be recognized by the FPGA. Failure to meet this requirement can cause the device to become non-operational, requiring power to be cycled.

**Figure 8-1. Configuration from PROGRAMN Timing**



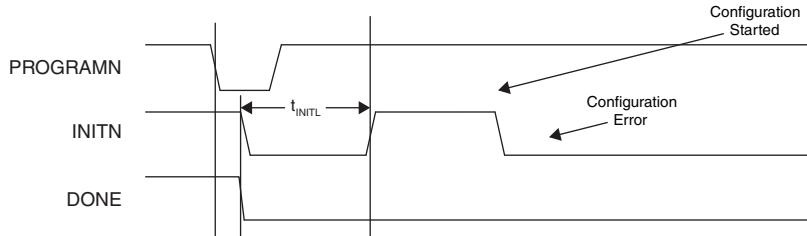
INITN: The INITn pin is a bidirectional open-drain control pin. It has the following functions:

- After power is applied, after a PROGRAMN assertion, or a REFRESH command it goes low to indicate the SRAM configuration memory is being erased. The low time assertion is specified with the  $t_{INTL}$  parameter.
- After the  $t_{INTL}$  time period has elapsed the INITn pin is deasserted (i.e. is active high) to indicate the ECP5 device is ready for its configuration bits. The ECP5 device begins loading configuration data from an external SPI Flash.
- INITn can be asserted low by an external agent before the  $t_{INTL}$  time period has elapsed in order to prevent the FPGA from reading configuration bits. This is useful when there are multiple programmable devices chained together. The programmable device with the longest  $t_{INTL}$  time can hold all other devices in the chain from starting to get data until it is ready itself.
- The last function provided by INITn is to signal an error during the time configuration data is being read. Once  $t_{INTL}$  has elapsed and the INITn pin has gone high, any subsequent INITn assertion signals the ECP5 device has detected an error during configuration.

The following conditions will cause INITN to become active, indicating the Initialization state is active:

- Power has just been applied
- PROGRAMN falling edge occurred
- The IEEE 1532 REFRESH command has been sent using a slave configuration port (JTAG or SSPI). If the INITN pin is asserted due to an error condition, the error can be cleared by correcting the configuration bitstream and forcing the FPGA into the Initialization state.

**Figure 8-2. Configuration Error Notification**



If an error is detected when reading the bitstream, INITN will go low, the internal DONE bit will not be set, the DONE pin will stay low, and the device will not wake up. The device will fail configuration when the following happens:

- The bitstream CRC error is detected
- The invalid command error detected
- A time out error is encountered when loading from the external Flash. This can occur when the device is in MSPI configuration mode and the SPI Flash device is not programmed.
- The program done command is not received when the end of on-chip SRAM configuration or external Flash memory is reached

DONE: The DONE pin is a bi-directional open drain with a weak pull-up that signals the FPGA is in User mode. DONE is first able to indicate entry into User mode only after an internal DONE bit is asserted. The internal DONE bit defines the beginning of the FPGA Wake-Up state.

The FPGA can be held from entering User mode indefinitely by having an external agent keep the DONE pin asserted low. A common reason for keeping DONE driven low is to allow multiple FPGAs to be completely configured. As each FPGA reaches the DONE state, it is ready to begin operation. The last FPGA to configure can cause all FPGAs to start in unison.

The DONE pin drives low in tandem with the INITN pin when the FPGA enters Initialization mode. As described earlier, this condition happens when power is applied, PROGRAMN is asserted, or an IEEE 1532 Refresh command is received via an active configuration port.

Sampling the DONE pin is a 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.

**MCLK/CCLK:** The MCLK/CCLK, when active, are clocks used to sequentially load the configuration data for the FPGA. As an input, CCLK is a dedicated input pin that supports all slave configuration modes. The CCLK pin is defined as an input configuration clock only to support all slave mode configuration modes.

When Master Configuration mode is used, MCLK will be an output clock with the frequency set by the user. The output is used to drive to external memory device.

The MCLK/CCLK pin's default state is determined by the state of the CFGMDN[2:0] settings. The maximum CCLK frequency and the data setup/hold parameters can be found in the AC timing section of DS1044, [ECP5 Family Data Sheet](#).

The MCLK/CCLK pin functions as a Master Clock (MCLK) when the CFGMDN[2:0] is set to MSPI mode. The MCLK becomes an output and provides a reference clock for a SPI Flash attached to the ECP5 device's Master SPI Configuration port. MCLK actively drives until all of the configuration data has been received. When the ECP5 device enters user mode the MCLK output tri-states. The MCLK is always reserved for use in MSPI mode, in most post-configuration applications, as the reference clock for performing memory transactions with the external SPI PROM. Please see [Master SPI Modes](#) section for details.

The ECP5 device generates MCLK from an internal oscillator. The initial frequency of the MCLK is nominally 2.4 MHz. The MCLK frequency can be altered using the MCCLK\_FREQ parameter. You can select the MCCLK\_FREQ using the Diamond Spreadsheet View. For a complete list of the supported MCLK frequencies, see Table 8-6.

**Table 8-6. ECP5 MCLK Valid Frequencies**

MCLK Frequency (MHz)
2.4
4.8
9.7
19.4
38.8

At startup, the lowest frequency MCLK is used by the FPGA. During the initial stages of device configuration the frequency value specified using MCCLK\_FREQ is loaded into the FPGA. Once the ECP5 device accepts the new MCCLK\_FREQ value the MCLK output begins driving the selected frequency. Make certain when selecting the MCCLK\_FREQ that you do not exceed the frequency specification of your configuration memory, or of your PCB. When making MCCLK\_FREQ decisions, review the ECP5 device AC specifications in DS1044, [ECP5 Family Data Sheet](#).

## 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 VCCIO8.

### **HOLDN/DI/BUSY/CSSPIN**

**HOLDN** – When Slave SPI mode is used, the HOLDN pin is an asynchronous active low input that tri-states the serial read out data of the SPI port and sets the device to the suspend state by ignoring the clock. HOLDN is only available in SSPI mode during configuration.

**DI** – Data Input for Serial Configuration Mode (SCM). DI has an internal weak pull-up and captures the data on the rising edge of CCLK.

**BUSY** – In Slave Parallel configuration mode, the BUSY pin is a tri-stated output. The BUSY pin will be driven low by the device only when a byte of data is ready for reading. Set the SLAVE\_PARALLEL\_PORT to ENABLE to retain the pin as the BUSY pin to access the Slave Parallel port in user mode.

**CSSPIN** – In MSPI mode, the CSSPIN becomes a low true Chip Select output that drives the SPI Serial Flash chip select. If SPI memory needs to be accessed using the SPI port while the part is in user mode (the DONE pin is high) then the MASTER\_SPI\_PORT= ENABLE, must be used to preserve this pin as CSSPIN. When the ECP5 device CFGMDN[2:0] pins are not in MSPI mode, the CSSPIN is a general purpose I/O with a weak pulldown. Adding a 4.7K to 10K ohm pull-up resistor to CSSPIN pin on the ECP5 device is recommended. CSSPIN must ramp in tandem with the SPI PROM VCC input. It remains a general purpose I/O when the FPGA enters user mode.

### **DOUT/CSON**

This is an output pin that has the following purposes.

**DOUT** – For both serial and parallel configuration modes. When the Bypass mode is selected, this pin becomes a data output pin for serial daisy chaining. When the device is fully configured, the Bypass instruction contained within 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.

**CSON** – For parallel daisy chaining implemented with the Flow-through attribute, this attribute allows the CSON pin to be driven when the done bit is set and configuration of the first device is complete. The CSON of the first device will drive the CSN of the second part.

You will find this attribute in the Diamond Generate Bitstream Data property under Chain Mode or in the Diamond Bitstream Strategy options window.

The DOUT/CSON 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/SN**

CSN/SN is a bidirectional pin with following functions.

**CSN** – This is an active low input pin with a weak pull-up used in parallel mode only. See the CS1N pins details for more information.

**SN** – When Slave SPI mode is used, this pin is an active low chip select input. Set the SLAVE\_SPI\_PORT to ENABLE to retain the pin as the SN pin to access the Slave SPI port in user mode. Lattice recommends that the SN pin be pulled high externally to augment the weak internal pull-up.

## CS1N

CS1N – CS1N is an active low input pin. Both CSN and CS1N are OR'ed and used to enable the D[0:7] data pins to receive or output a byte of data in parallel mode.

When CSN or CS1N is high, the D[0:7], and BUSY pins are tri-stated. CSN and CS1N are interchangeable when controlling the D[0:7] and BUSY pins. Driving both CSN and CS1N high causes the ECP5 device to exit the Bypass or Flow-through modes and resets the Bypass register. If the Bypass or Flow-through modes 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). Set SLAVE\_PARALLEL\_PORT to ENABLE to retain the pin as CS1N pin to access the SLAVE PARALLEL port in user mode.

## WRITEN

This 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 SLAVE\_PARALLEL\_PORT must be set to ENABLE to preserve this pin as WRITEN.

## D0/MOSI

D0 – In parallel mode this pin is D[0] and operates in the same way as D[7:0] below.

MOSI – In slave SPI mode, the MOSI pin is the serial input for SPI command and data.

MOSI/IO0 – In MSPI configuration mode, the MOSI pin becomes an output that drives control and data to an SPI Flash. Control and data are output on the falling edge of MCLK.

Set the SLAVE\_SPI\_PORT to ENABLE or set the MASTER\_SPI\_PORT to ENABLE to retain the pin as the MOSI pin to access the SPI port in user mode.

## D1/MISO

D1 – In parallel mode this pin is D[1] and operates in the same way as D[7:0] below.

MISO – In slave SPI mode, the MISO pin is the serial output data from FPGA.

MISO/IO1 – In MSPI configuration mode, the MISO pin is serial data input.

Set the SLAVE\_SPI\_PORT to ENABLE or set the MASTER\_SPI\_PORT to ENABLE to retain the pin as the MISO pin to access the SPI port in user mode.

## D[7:0]

These are tri-stated bi-directional I/O pins. A byte of data is driven into or read from these pins. Taken together D[7:0] form the parallel data bus. 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 SLAVE\_PARALLEL\_PORT=ENABLE must be set to preserve these pins.

When the WRITEN signal is low and CSN is low, these pins are inputs. When the WRITEN signal is driven high and CSN is low, these pins are output pins. These pins are enabled by the CSN signal. D[0:7] is always power-up tristated with no pull-up.

## IO[3:0]

These pins are used in conjunction with advanced DUAL or Quad SPI Flash memory interfaces. Dependent on the targeted read mode, these pins provide data to and from the external memory.

## PERSISTENT

The internal PERSISTENT control bits are used to determine whether the dual-purpose sysCONFIG pins remain sysCONFIG pins during normal operation. The ECP5 device has several PERSISTENT physical SRAM cells that determine the existence of either the Slave parallel port, the Master SPI port or a Slave SPI port when entering the user mode.

**Table 8-7. sysCONFIG Pins Restricted by Port Setting**

Port Setting	Pins Affected	Details
SLAVE_PARALLEL_PORT	D[0:7],CSN, CS1N, WRITEN, BUSY	If enabled, restricted from recovering as user I/Os.
SLAVE_SPI_PORT	MOSI, MISO, SN	If enabled, restricted from recovering as user I/Os.
MASTER_SPI_PORT	MOSI, MISO, CSSPIN	If enabled, restricted from recovering as user I/Os.

## JTAG Configuration Port Pins

The JTAG pins provide a standard IEEE 1149.1 Test Access Port (TAP). Programming and configuration over the JTAG port uses IEEE 1532 compliant commands. In addition to the IEEE 1532 capabilities, the ECP5 device provides all of the mandatory IEEE 1149.1 Test Access Port commands allowing printed circuit board assembly verification.

The JTAG port is always an available port in the ECP5 device.

When the device is programmed through IEEE 1149.1 control, the sysCONFIG programming pins, such as DONE, cannot be used to determine programming progress. This is because the state of the boundary scan cell will drive the pin, per the IEEE JTAG standard, rather than normal internal logic.

**Table 8-8. JTAG Port Pins**

Pin Name	Pin Function	Pin Direction
TDI	TDI	Input with weak pull-up
TDO	TDO	Output with weak pull-up
TCK	TCK	Input
TMS	TMS	Input with weak pull-up
VCCIO8	Power Supply	N/A

TDO: The Test Data Output (TDO) 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. The only time TDO is not in a high impedance state is when the JTAG state machine is in the Shift IR or Shift DR 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 VCCJ.

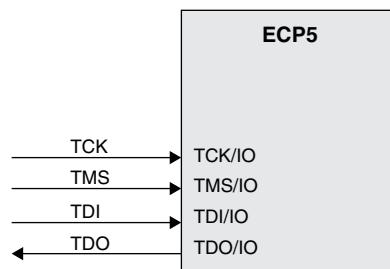
TDI: The Test Data Input (TDI) pin is used to shift in serial test instructions and data. This pin should be wired to TDI of the JTAG connector, or to TDO of an upstream device in a JTAG chain. An internal pull-up resistor on the TDI pin is provided. The internal resistor is pulled up to VCCJ.

TMS: The Test Mode Select (TMS) pin is an input pin that controls the progression through the 1149.1 compliant state machine states. The TMS pin is sampled on the rising edge of TCK. The JTAG state machine remains in or transitions to a new TAP state depending on the current state of the TAP, and the present state of the TMS input. An internal pull-up resistor is present on TMS per the JTAG specification. The internal resistor is pulled to the VCCJ.

TCK: The test clock pin (TCK) provides the clock used to time the other JTAG port pins. Data is shifted into the instruction or data registers on the rising edge of TCK and shifted out on the falling edge of TCK. The TAP is a static design permitting TCK to be stopped in either the high or low state. The maximum input frequency for TCK is specified in the DC and Switching Characteristics section of DS1044, [ECP5 Family Data Sheet](#). The TCK pin does

not have a pull-up. An external pull-down resistor of 4.7 K ohms is recommended to avoid inadvertently clocking the TAP controller as power is applied to the ECP5 device.

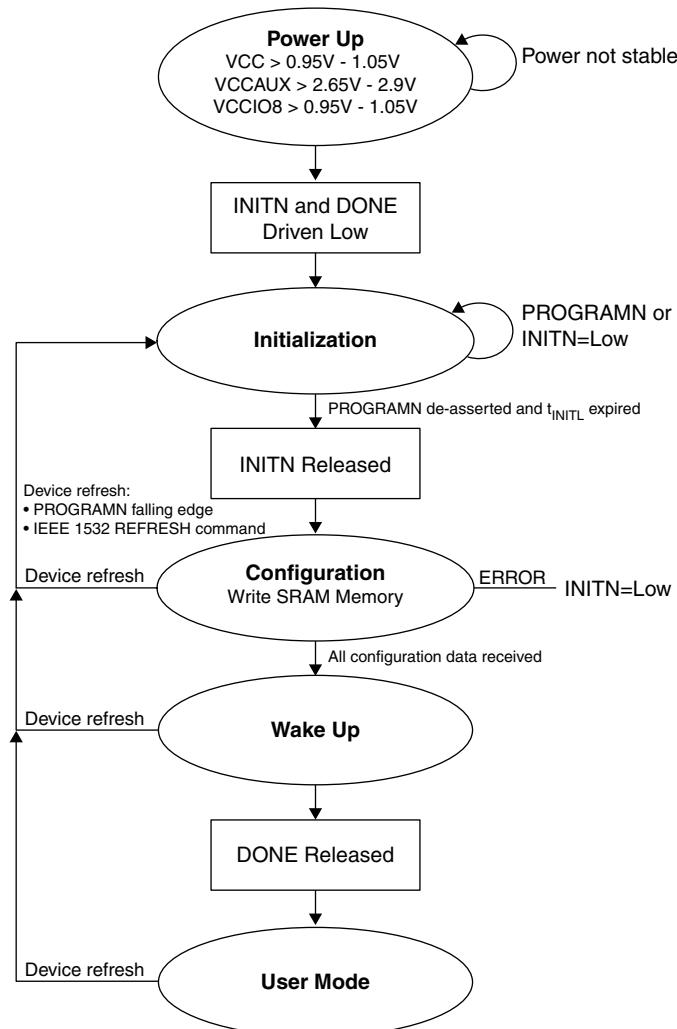
**Figure 8-3. JTAG Port**



## Configuration Process and Flow

Prior to becoming operational, the FPGA goes through a sequence of states, including initialization, configuration and wake-up.

**Figure 8-4. Configuration Flow**



The ECP5 device sysCONFIG ports provide industry standard communication protocols for programming and configuring the FPGA. Each of the protocols shown in Table 8-3 provides a way to access the ECP5 device's configuration SRAM. The Reconfiguration Priority section provides information about the capabilities of each sysCONFIG port.

## Power-up Sequence

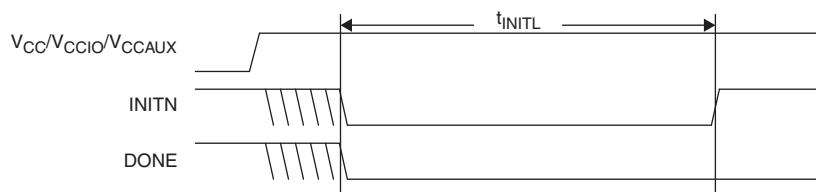
In order for the ECP5 device to operate, power must be applied to the device. During a short period of time, as the voltages applied to the system rise, the FPGA will have an indeterminate state.

As power continues to ramp, a Power On Reset (POR) circuit inside the FPGA becomes active. The POR circuit, once active, makes sure the external I/O pins are in a high-impedance state. It also monitors the  $V_{CC}$  and  $V_{CCIO8}$  input rails. The POR circuit waits for the following conditions:

- $V_{CC} > 0.95V - 1.05V$
- $V_{CCIO8} > 0.95V - 1.05V$
- $V_{CCAUX} > 2.65V - 2.9V$

When these conditions are met the POR circuit releases an internal reset strobe, allowing the device to begin its initialization process. The ECP5 device asserts INITN active low, and drives DONE low. When INITN and DONE are asserted low the device moves to the initialization state, as shown in Figure 8-4.

**Figure 8-5. Configuration from Power-On-Reset Timing**



## Initialization

The ECP5 device enters the memory initialization phase immediately after the Power On Reset circuit drives the INITN and DONE status pins low. The purpose of the initialization state is to clear all of the SRAM memory inside the FPGA.

The FPGA remains in the initialization state until all of the following conditions are met:

- The  $t_{INITL}$  time period has elapsed
- The PROGRAMN pin is deasserted
- The INITN pin is no longer asserted low by an external master

The dedicated INITN pin provides two functions during the initialization phase. The first is to indicate the FPGA is currently clearing its configuration SRAM. The second is to act as an input preventing the transition from the initialization state to the configuration state.

During the  $t_{INITL}$  time period the FPGA is clearing the configuration SRAM. When the ECP5 device is part of a chain of devices each device will have different  $t_{INITL}$  initialization times. The FPGA with the slowest  $t_{INITL}$  parameter can prevent other devices in the chain from starting to configure. Premature release of the INITN in a multi-device chain may cause configuration of one or more chained devices to fail to configure intermittently.

The active-low, open-drain initialization signal INITN must be pulled high by an external resistor when initialization is complete. To synchronize the configuration of multiple FPGAs, one or more INITN pins should be wire-ANDed. If one or more FPGAs or an external device holds INITN low, the FPGA remains in the initialization state.

The GPIO of the device at power-up will default to tri-stated outputs with active weak input pull-downs. After configuration, all GPIO included in the user design will wake up in the user-defined condition. GPIO not defined in the user design will remain output tri-stated and the input will have a weak pull-down enabled. This default avoids inadvertent effects of the inputs rising while powering up. In some cases, this can cause a problem if other connected devices on the board reset or trigger from an active high signal.

Before/during configuration, the D[7:0] pins have no pull-up/down. Other dedicated and dual-purpose I/O with pull-up exceptions, such as PROGRAMN, DONE, etc., are excluded from the GPIO default setting.

## Configuration

The rising edge of the INITN pin causes the FPGA to enter the configuration state. The FPGA is able to accept the configuration bitstream created by the Diamond development tools.

The ECP5 device begins fetching configuration data from non-volatile memory. The ECP5 device does not leave the Configuration state if there is no valid configuration data.

INITN is used to indicate an error exists in the configuration data. When INITN is active high configuration is proceeding without issue. If INITN is asserted low, an error has occurred and the FPGA will not operate.

## Wake-up

Wake-up is the transition from configuration mode to user mode. The ECP5 device's fixed four-phase wake-up sequence starts when the device has correctly received all of its configuration data. The order of these four phases can be sequenced by the user to meet specific implementation requirements. When all configuration data is received, the FPGA asserts an internal DONE status bit. The assertion of the internal DONE causes a Wake Up state machine to run that sequences four controls. The four control strobes are:

- Global Set/Reset (GSR)
- Global Output Enable (GOE)
- External DONE
- Global Write Disable (GWDISn)

One phase of the Wake-Up process is for the FPGA to release the Global Output Enable. When it is asserted, permits the FPGA's I/O to exit a high-impedance state and take on their programmed output function. The FPGA inputs are always active. The input signals are prevented from performing any action on the FPGA flip-flops by the assertion of the Global Set/Reset (GSR).

Other phases of the Wake-Up process release the Global Set/Reset and the Global Write Disable controls.

The Global Set/Reset is an internal strobe that, when asserted, causes all I/O flip-flops, Look Up Table (LUT) flip-flops, distributed RAM output flip-flops, and Embedded Block RAM output flip-flops that have the **GSR enabled** attribute to be set/cleared per their hardware description language definition.

The Global Write Disable is a control that overrides the write enable strobe for all RAM logic inside the FPGA. The inputs on the FPGA are always active, as mentioned in the Global Output Enable section. Keeping GWDIS asserted prevents accidental corruption of the instantiated RAM resources inside the FPGA.

Another phase of the Wake-Up process asserts the external DONE pin. The external DONE is a bi-directional, open-drain I/O only when it is enabled. An external agent that holds the external DONE pin low prevents the wake-up process of the FPGA from proceeding. Only after the external DONE, if enabled, is active high does the final wake-up phase complete. Wake-Up completes uninterrupted when the external DONE pin is not enabled.

Once the final wake-up phase is complete, the FPGA enters user mode. This process is detailed later in the document.

## User Mode

The ECP5 device enters User Mode immediately following the Wake-Up sequence has completed. User Mode is the point in time when the ECP5 device begins performing the logic operations you designed. The device remains in this state until one of three events occurs:

- The PROGRAMN input pin is asserted
- A REFRESH command is received via one of the configuration ports
- Power is cycled or power supply levels drop below their specified trigger levels

## Clearing the Configuration Memory and Re-initialization

The current user mode configuration of the ECP5 device remains in operation until it is actively cleared, or power is lost. Several methods are available to clear the internal configuration memory of the ECP5 device. The first is to remove power and reapply power. Another method is to toggle the PROGRAMN pin. Lastly you can reinitialize the memory through a Refresh command. Any active configuration port can be used to send a Refresh command.

Invoking one of these methods causes the ECP5 device to drive INITN and DONE low. The ECP5 device enters the initialization state as described earlier.

## Reconfiguration Priority

There are many sources that can initiate a reconfiguration while a configuration is already in process. There is a priority as to which of these sources can interrupt the configuration process depending on which of the sources initiated the original configuration. Note that if an interruption occurs, the reconfiguration will occur without informing the original configuration source that the configuration did not complete. JTAG always has the highest priority and any JTAG initiated configuration event will cause a reconfiguration to occur. Toggling the PROGRAMN pin has the next highest priority. It will interrupt any current configuration other than a JTAG configuration.

8-bit slave-parallel mode (SPCM) will not interrupt JTAG or PROGRAMN pin initiated configurations.

## Configuration Modes

The ECP5 device provides multiple options for loading the configuration SRAM from a non-volatile memory. The previous section described the physical interface necessary to interact with the ECP5 device configuration logic. This section focuses on describing the functionality of each of the different configuration modes. Descriptions of important settings required in the Diamond Spreadsheet View are also discussed.

### Master SPI Modes

The Master SPI port is considered an intelligent port. It is capable of performing read and write actions based on the command shifted into the device. All commands are built inside the bitstreams. In Master SPI mode (MSPI), the ECP5 device begins retrieving configuration data from the SPI Flash when power is applied, a REFRESH command is received, or the PROGRAMN pin is asserted and released. This occurs when the CFGMDN[2:0] pins are set to [010] respectively. One MSPI mode CFGMDN setting supports several different sub-modes such as SLOW or FAST SPI speed, DUAL-boot and MULTI-boot, as well as supporting DUAL and QUAD read SPI Flash devices. All the different sub-modes are supported in Lattice Diamond Deployment Tool. The MCLK/CCLK I/O takes on the Master Clock (MCLK) function, and begins driving a nominal 2.4 MHz clock to the SPI Flash's SCLK input. CSSPIN is asserted low, commands are transmitted to the PROM over the SI/SISPI output, and data is read from the PROM on the SO/SPISO input pin. When all of the configuration data is retrieved from the PROM, the CSSPIN pin is deasserted, and the MSPI output pins are tri-stated.

**Table 8-9. Master SPI Configuration Port Pins**

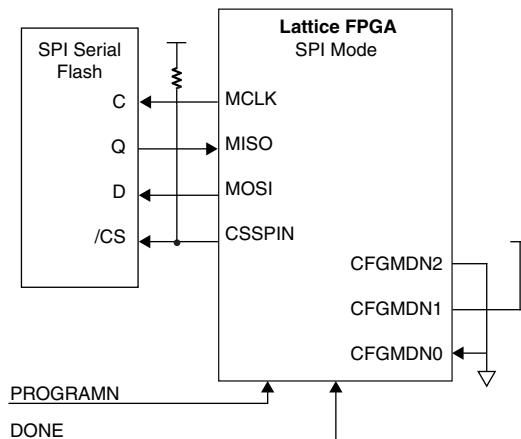
Pin Name	Function	Direction	Description
MCLK/CCLK	MCLK	Output with weak pullup	Master clock used to time data transmission/reception from the ECP5 device Configuration Logic to a slave SPI PROM.
CSSPIN <sup>1</sup>	CSSPIN	Output	Chip select used to enable an external SPI PROM containing configuration data.
MOSI/IO0 <sup>2</sup>	MOSI	Output	Carries output data from the ECP5 device Configuration Logic to the slave SPI PROM
	IO0	Input	Input pin for bitstream data as DUAL and QUAD read mode
MISO/IO1 <sup>2</sup>	MISO	Input	Carries input data from the slave SPI PROM to the ECP5 device Configuration Logic
	IO1	Input	Input pin for bitstream data as DUAL and QUAD read mode
IO[3:2] <sup>2</sup>	IO[3:2]	Input	This is the input pins for bitstream data from SPI Flash, used only on QUAD read mode.

1. Use 4.7k-ohm pullup resistor

2. Use 10k-ohm pullup resistor

The MCLK frequency always starts downloading the configuration data at the nominal 2.4 MHz frequency. The MCCLK\_FREQ parameter, accessed using Spreadsheet View, can be used to increase the configuration frequency. The configuration data in the PROM has some padding bits, and then the data altering the MCLK base frequency is read. The ECP5 device reads the remaining configuration data bytes using the new MCLK frequency.

**Figure 8-6. ECP5 Master SPI Port with SPI Flash**



Once the SPI Flash contains your configuration data, you can test the configuration. Assert the PROGRAMN, transmit a REFRESH command, or cycle power to the board, and the ECP5 device will configure from the external SPI Flash.

#### Method to Enable the Master SPI Port

Similar to all configuration ports, the Master SPI port is enabled by two standard methods.

1. Setting the CFGMDN[2:0] pin to [0,1, 0].

When the device is powered up, or when the PROGRAMN pin is toggled, the device checks the state of the CFGMDN pins. If they are set to [0,1, 0], then the device will choose the Master SPI port as the configuration port. This is the only method to enable the Master SPI port as the configuration port. A port is said to be a configuration port when it is capable of executing both bitstream write and read commands. And this is the only method that user can perform DUAL read and QUAD read from SPI Flash.

2. Enabling Master SPI port persistence.

The configuration bitstream contains optional Master SPI persistence bits. When the device completes configuration and wakes up, it checks the persistent bits to determine if the Master SPI port is to remain operational once in User Mode. This selection is independent of the CFG pins setting. A port enabled by persistence is a Background Mode port. It mainly used for background self re-configuration upon SED failure when user set SED auto correction.

Note that both the DONE pin and the INITN pin must be high. If not, then the device is not in user mode. The persistent bits have no affect when the device is not in user mode.

#### Customized SPI Port in User Mode

After the ECP5 device enters user mode the Master SPI configuration port pins are tri-stated. This allows data transfers across the SPI. User need to create their own Master SPI Controller to perform the data transfers. The ECP5 device provides a solution for users to choose any user clock as MCLK under this scenario by instantiating USRMCLK macro in your Verilog or VHDL.

#### Verilog

```

module USRMCLK (USRMCLKI, USRMCLKTS);
  input USRMCLKI, USRMCLKTS;
  endmodule

  USRMCLK u1 (.USRMCKI(<clock_name>), .USRMCLKTS(<tristate_name>) /* synthesis
syn_noprune=1 */;
```

## VHDL

```

COMPONENT USRMCLK
  PORT (
    USRMCLKI : IN STD_ULONGIC;
    USRMCLKTS : IN STD_ULONGIC
  );
END COMPONENT;
attribute syn_noprune: boolean ;
attribute syn_noprune of USRMCLK: component is true;

begin
  u1: USRMCLK port map (
    USRMCLKI => <clock name>,
    USRMCLKTS => ,tristate_name>);

```

### Dual-Boot and Multi-Boot Configuration Modes

Both the primary and the golden (fail-safe) configuration data is stored in external SPI memory. The fail-safe pattern is available in case the primary pattern would fail to load. The primary image can fail in one of two ways:

- A bitstream CRC error is detected
- A time-out error is encountered when loading from the primary pattern stored in the external memory

A CRC error is caused by incorrect data being written into the SRAM configuration memory. Data is read from the external Flash memory. As data enters the Configuration Engine the data is checked for CRC consistency. Before the data enters the Configuration SRAM the CRC must be correct. Any incorrect CRC causes the device to erase the Configuration SRAM and retrieve configuration data from the external golden pattern.

It is possible for the data to be correct from a CRC calculation perspective, but not be functionally correct. In this instance the internal DONE bit will never become active. The ECP5 device counts the number of master clock pulses it has provided after the Power On Reset signal was released. When the count expires without DONE becoming active the FPGA attempts to get its configuration data from the external golden pattern.

Dual boot configuration mode typically requires two configuration data files. One of the two configuration data files is a fail-safe image that is rarely, if ever, updated. The second configuration data file is a working image that is routinely updated. Both the working (primary) image and the fail-safe image are stored in the external SPI memory. One Diamond project can be used to create both the working and the fail-safe configuration data files. Configure the Diamond project with an implementation named **working**, and an implementation named **failsafe**. Read the Diamond Online Help for more information about using Diamond implementations.

The ECP5 device supports dual-boot with the MSPI mode CFGMDN[2:0] setting. If the primary pattern fails to load correctly, the ECP5 device will start loading data from the golden sector in the SPI Flash device. In cases where the CFGMDN[2:0] setting is MSPI, a blank external Flash device will cause a dual-boot event failure indicated by INITn going low. This is due to the absence of a primary or golden boot image.

The dual boot feature allows you to split a SPI Flash device into two sections, the first containing a sacred “golden boot” file, and a second updatable “primary boot” file, which can be erased and reprogrammed. By default, the FPGA loads the primary boot file in block 0 (0x000000). If the FPGA fails in configuration, it will automatically load the golden boot file in the last block (0xFFFFF00). This allows your system to boot to a known operable state, so that it continues to operate if for some reason (such as a power failure) the SPI Flash fails to program correctly.

Users can dynamically switch between up to five different design revisions stored in external Flash. Multi-boot, or the ability to dynamically reconfigure from multiple design bitstreams, is similar to the dual-boot where there is one primary (working) bitstream and up to four alternate bitstreams.

For multi-boot operation, the next target address is set in memory that was loaded during the current configuration memory load. Initiating reprogramming by toggling the PROGRAMN pin or issuing a REFRESH via any sysCONFIG port will cause the device to load from the defined SPI Flash address. Dual-boot can also be deployed with multi-boot allowing a golden (fail-safe) design (or sixth design) to also be available in the external Flash.

Diamond Deployment Tool allows users to assemble SPI Flash images formatted to correctly match the hardware sector mapping.

### **Dual and Quad Master SPI Read Mode**

The master SPI configuration mode in the ECP5 device is expanded to support new industry standard Quad I/O SPI Flash memory. The support of (Serial Multi I/O) Flash memory enables fast parallel read.

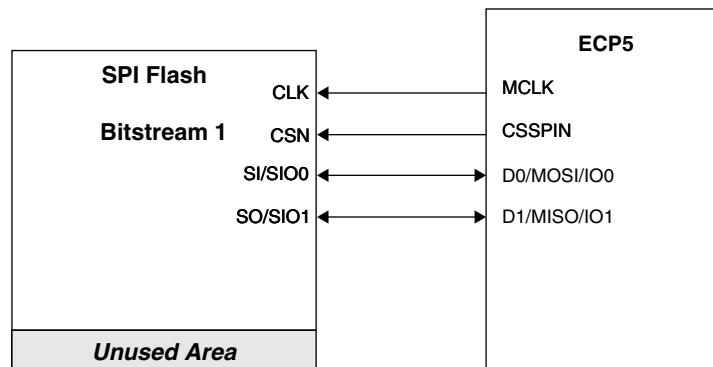
A typical SPI Flash interface uses either 4 or 6 interface signals to the FPGA. The Standard SPI flash uses CLK, CS, SI and SO while Quad SPI Flash uses CLK, CS, I/O0, I/O1, I/O2 and I/O3, maintaining function and pin-out compatibility with the single SPI Flash devices, while adding Dual-I/O and Quad-I/O SPI capability. All SPI modes are sub-modes of MASTER SPI, therefore there is no longer a separate CFGMDN pin decode for each SPI mode. MASTER SPI supports all bus widths and sub-modes are controlled by control register settings.

In Dual mode, the Fast-Read Dual Output (BBh) instruction is issued and is similar to the standard Fast Read (0Bh) instruction except that data is output on two pins, SO and SIO0, instead of just SO. This allows data to be transferred from the dual output at twice the rate of standard SPI devices. In QUAD mode, the Fast-Read Quad Output (EBh) instruction is issued and is similar to the standard Fast Read (0Bh) instruction except that data is output on four data pins, instead of just SO. This allows data to be transferred from the quad output at four times the rate of standard SPI devices.

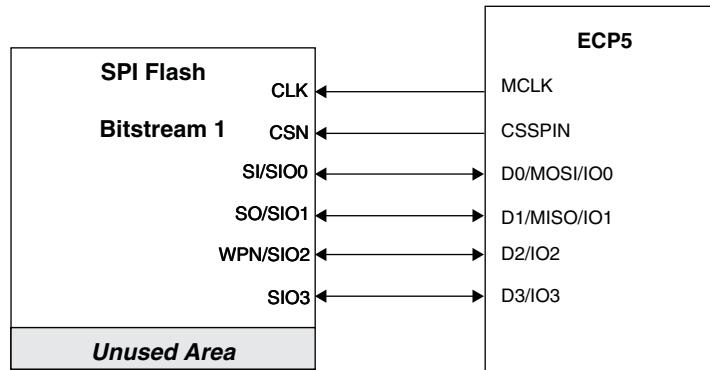
**Table 8-10. Dual/Quad SPI Port Names**

<b>ECP5 Device Pin Name</b>	<b>Flash Device Pin</b>	<b>MSPI Function DUAL(D), QUAD(Q)</b>
MCLK	SCK	Clock output from the ECP5 device Configuration Logic and Master SPI controller. Connect MCLK to the SCLK input of the Slave SPI device. (D)(Q)
IO[3:2]	SIO[3:2]	Data I/O between the ECP5 device and SPI device. (Q)
IO[1:0]	SIO[1:0]	Data I/O between the ECP5 device and SPI device. (D)(Q)
CSSPIN	CSN	Chip select output from the ECP5 device configuration logic to the slave SPI Flash holding configuration data for the ECP5 device. (D)(Q)

**Figure 8-7. One Dual SPI Flash Interface (Dual)**



**Figure 8-8. One Quad SPI Flash Interface (Quad)**



### Slave SPI Mode (SSPI)

The ECP5 device provides a Slave SPI configuration port that allows you to access features provided by the Configuration Logic. You can reprogram the Configuration SRAM and access status/control registers within the Configuration Logic block. The external Flash memory updates are done using either offline or transparent operations. It is necessary to send a REFRESH command to load a new external Flash image into the configuration SRAM. When the ECP5 device is in Background Mode, only read type commands are supported, allowing for device verification or debugging. The command set consists of 8-bit opcodes. Some of the commands have a 24-bit operand following the 8-bit opcode. The Slave SPI port is a byte bounded port; all input and output data must be byte bounded.

In the Slave SPI mode, the MCLK/CCLK pin becomes CCLK (i.e. Configuration clock). Input data is read into the ECP5 device on the MOSI pin at the rising edge of CCLK. Output data is valid on the MISO pin at the falling edge of CCLK. The SN acts as the chip select signal. When SN is high, the SSPI interface is deselected and the MISO pin is tri-stated. Commands can be written into and data read from the ECP5 device when SN is asserted.

The SSPI port is active when the CFGMDN[2:0] is set to [001]. Diamond's default preference for the SLAVE\_SPI\_PORT is to DISABLE the port. Use the Spreadsheet View to ENABLE the SLAVE\_SPI\_PORT preference in your design to keep the SSPI port active for configuration after the device enters user mode.

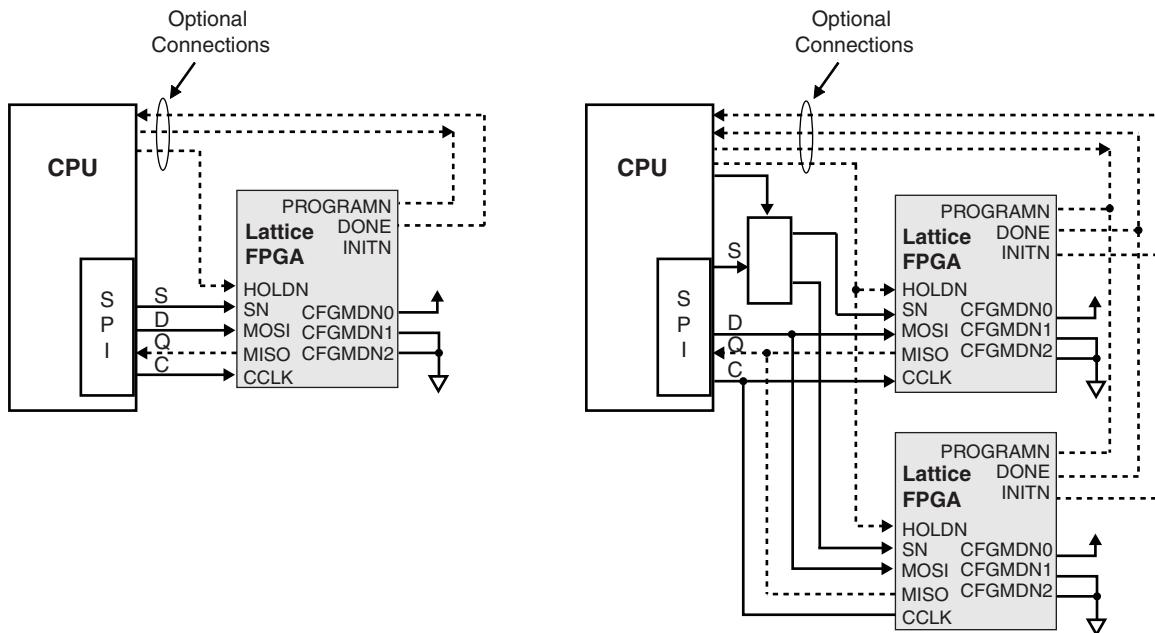
Using the SSPI port simplifies the ECP5 device configuration process. Lattice provides 'C' source code called SSPIEmbedded to insulate you from the complexity of programming the ECP5 device. Please check Diamond online help about SSPIEmbedded.

**Table 8-11. Slave SPI Configuration Port Pins**

Pin name	Function	Direction	Description
MCLK/CCLK	CCLK	Input with weak pullup	Clock used to time data transmission/reception from an external SPI master device to the ECP5 device Configuration Logic.
MOSI	MOSI	Input	Carries output data from the external SPI master to the ECP5 device Configuration Logic
MISO	MISO	Output	Carries output data from the ECP5 device Configuration Logic to the external SPI master. It is normally tri-stated with an internal pull-up. It becomes active only when the command is a read type command.
SN <sup>1</sup>	SN	Input with weak pullup	ECP5 device Configuration Logic slave SPI chip select input. SN is an active low input. High to Low transition: reset the device, prepare it to receive commands. Low to High transition: Completes or terminates the current command.
HOLDN <sup>2</sup>	HOLDN	Input	Although not a standard SPI pin, this is an industry standard pin that allows the CPU to suspend data transmission. This pin can be asserted while shifting the bitstream into the device. Do not assert the HOLDN pin while shifting commands or operand into the device.

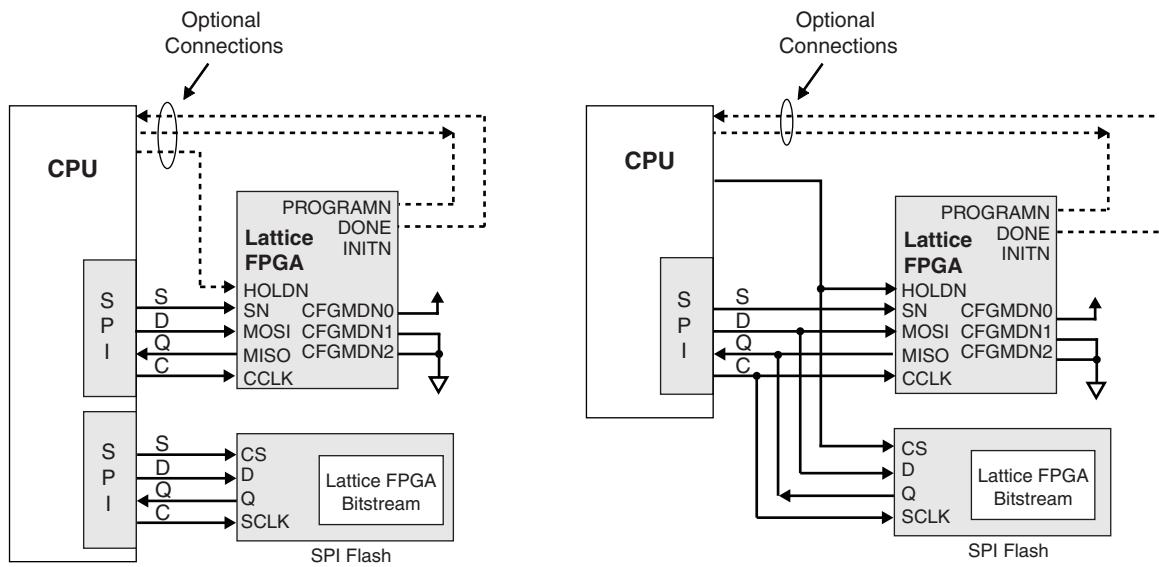
1. Use external 4.7K-ohm pullup resistor.

2. By definition, the Slave SPI port is a four (4) wire port. The HOLDN pin was added by SPI Flash vendors to provide the CPU a method to support suspension. The ECP5 devices also support the HOLDN pin to provide the CPU with a method to suspend data transmission when it cannot process a large bitstream in one single burst and needs time to fetch the bitstream in fragments.

**Figure 8-9. ECP5 Slave SPI Port with CPU and Single or Multiple Devices**

**Notes:**

- The dotted lines indicate optional connections.
- The wake up time of the device does vary with the bitstream size and the speed of the SPI port. Lattice recommends connecting the DONE pin to the CPU to monitor when the configuration is complete.
- If the bitstream for the two ECP5 devices is the same, the chip select logic (S/SN) is not required.
- The MISO to Q connection is optional if read back is not needed and the DONE pin is connected.

**Figure 9. ECP5 Slave SPI Port with SPI Flash**



Notes:

- The dotted lines indicate the connection is optional.
- The ECP5 device bitstream can reside in the SPI Flash device instead of the system Flash memory. The advantage of this is that the bitstream can be easily updated without changing the system software.

### Method to Enable the Slave SPI Port

Similar to all configuration ports, the Slave SPI port is enabled by the two standard methods.

1. Setting the CFGMDN[2:0] pin to [0,0,1].

When the device is powered up, or when the PROGRAMN pin is toggled, the device checks the state of the CFGMDN pins. If they are set to [0,0,1], then the device will choose the Slave SPI port as the configuration port. This is the only method to enable the Slave SPI port as the configuration port. The CONFIG\_MODE option in Diamond Global Preference is for our software to preserve appropriated pins to help customer design flow, it does not serve the purpose to enable SPI port. A port is said to be a configuration port when it is capable of executing both bitstream write and read commands.

2. Enabling Slave SPI port persistence.

The configuration bitstream contains optional Slave SPI persistence bits. When the device completes configuration and wakes up, it checks the persistent bits to determine if the Slave SPI port is to remain operational once in User Mode. This selection is independent of the CFGMDN pins setting. A port enabled by persistence is capable of read commands only. Thus, the port is a Background Mode port and is not a configuration port, and can only be used for read back operations.

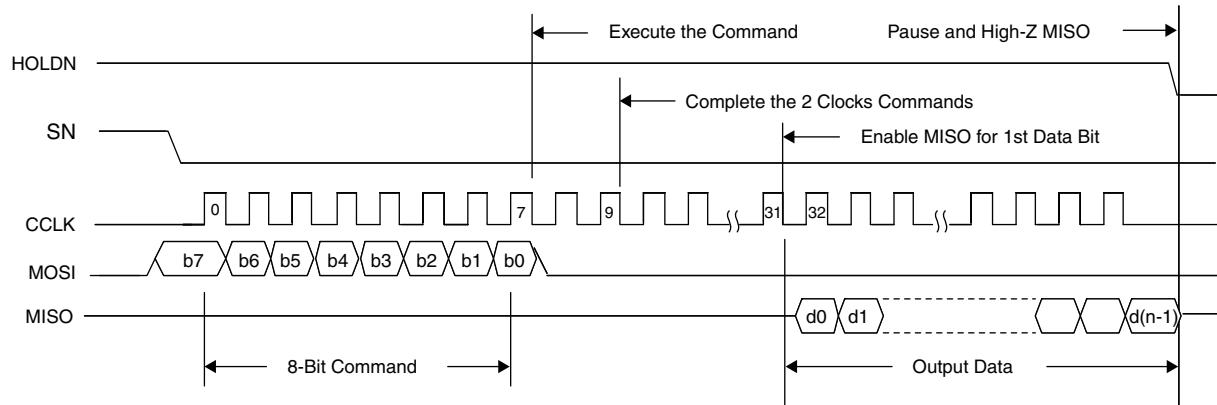
Note that both the DONE pin and the INITN pin must be high to qualify the Slave SPI port as a read back port. If not, then the device is not in user mode. The persistent bits have no affect when the device is not in user mode.

### Specifications and Timing Diagrams - Slave SPI Port Waveforms

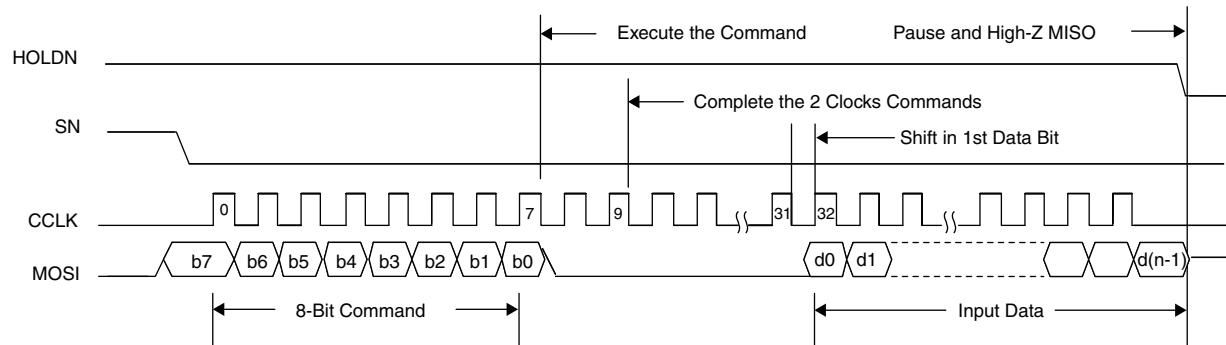
Data and commands shift into the MOSI pin on the rising edge of clock. Data is shifted out of the MISO pin on the falling edge of clock. Only a read command will cause the MISO pin to be enabled for data read out.

The Slave SPI read and write waveforms are shown in Figure 8-1 and Figure 8-2. The Slave SPI HOLDN pin waveform is shown in Figure 8-3.

**Figure 8-1. Slave SPI Read Waveforms**

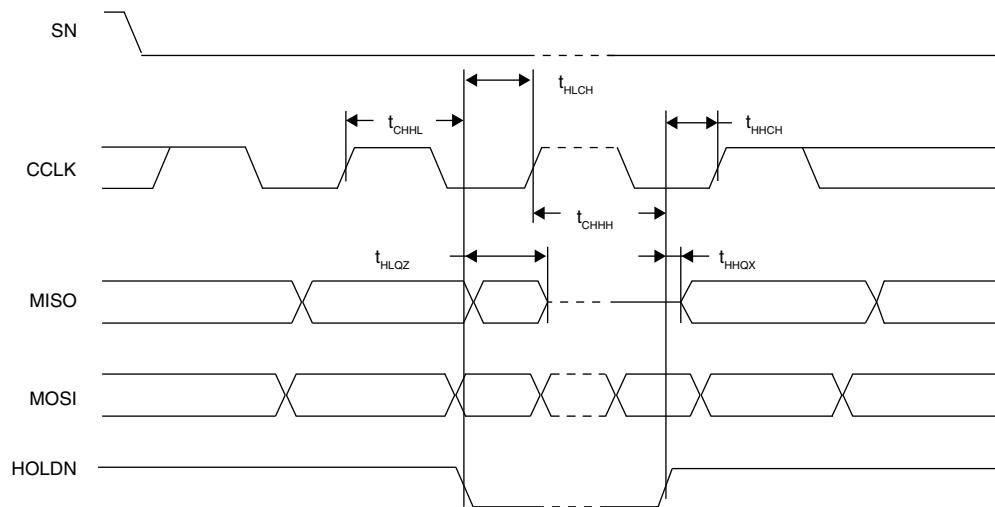


**Figure 8-2. Slave SPI Write Waveforms**



Note: The bitstream is transferred starting with the first byte of the data file, starting with the MSB of the byte.

**Figure 8-3. Slave SPI HOLDN Waveforms**



### Slave SPI Port AC Timing Requirements

The Slave SPI port AC timing requirements are listed in Table 9.

**Table 9. AC Timing Requirements**

Description	Parameter	Min.	Max.	Units
CCLK Frequency	$f_{CCLK}$		33	MHz
CCLK Minimum High Pulse	$t_{SSCH}$	5		ns
CCLK Minimum Low Pulse	$t_{SSCL}$	5		ns
HOLDN Low Setup Time (relative to CCLK)	$t_{HLCH}$	5		ns
HOLDN Low Hold Time (relative to CCLK)	$t_{CHHH}$	5		ns
HOLDN High Hold Time (relative to CCLK)	$t_{CHHL}$	5		ns
HOLDN High Setup Time (relative to CCLK)	$t_{HHCH}$	5		ns
HOLDN to Output High-Z	$t_{HLQZ}$		9	ns
HOLDN to Output Low-Z	$t_{HHQX}$		9	ns

**Table 10. Slave SPI Command Table**

Command	Binary	Hex	Operand	Description
ISC_NOOP	11111111	FF	0 bits	non-operation
READ_ID	11100000	E0	24 bits	Read out the 32-bit IDCODE of the device
USERCODE	11000000	C0	24 bits	Read 32-bit usercode
LSC_READ_STATUS	00111100	3C	24 bits	Read out internal status
LSC_CHECK_BUSY	11110000	F0	24 bits	Read 1 bit busy flag to check the command execution status
LSC_REFRESH	01111001	79	24 bits	Equivalent to toggle PROGRAMN pin
ISC_ENABLE	11000110	C3	24 bits	Enable the Offline configuration mode
ISC_ENABLE_X	01110100	74	24 bits	Enable the Transparent configuration mode
ISC_DISABLE	00100110	26	24 bits	Disable the configuration operation
ISC_PROGRAM_USERCODE	11000010	C2	24 bits	write the 32-bit new USERCODE data to USERCODE register
ISC_ERASE	00001110	0E	24 bits	Bulk erase the memory array base on the access mode and array selection
ISC_PROGRAM_DONE	01011110	5E	24 bits	Program the DONE bit if the device is in Configuration state.
ISC_PROGRAM_SECURITY	11001110	CE	24 bits	Program the Security bit if the device is in Configuration state
LSC_INIT_ADDRESS	01000110	46	24 bits	Initialize the Address Shift Register
LSC_WRITE_ADDRESS	10110100	B4	24 bits	Write the 16 bit Address Register to move the address quickly
LSC_BITSTREAM_BURST	01111010	7A	24 bits	Program the device the whole bitstream sent in as the command operand
LSC_PROG_INCR_RTI	10000010	82	24 bits	Write configuration data to the configuration memory frame at current address and post increment the address, Byte 2~0 of the opcode indicate number of the frames included in the operand field
LSC_PROG_INCR_ENC	10110110	B6	24 bits	Encrypt the configuration data then write
LSC_PROG_INCR_CMP	10111000	B8	24 bits	Decompress the configuration data, then write
LSC_PROG_INCR_CNE	10111010	BA	24 bits	Decompress and Encrypt the configuration data, then write

**Table 10. Slave SPI Command Table (Continued)**

Command	Binary	Hex	Operand	Description
LSC_VERIFY_INCR_RTI	01101010	6A	24 bits	Read back the configuration memory frame selected by the address register and post increment the address
LSC_PROG_CTRL0	00100010	22	24 bits	Modify the Control Register 0
LSC_READ_CTRL0	00100000	20	24 bits	Read the Control Register 0
LSC_RESET_CRC	00111011	3B	24 bits	Reset 16-bit frame CRC register to 0x0000
LSC_READ_CRC	01100000	60	24 bits	Read 16-bit frame CRC register content
LSC_PROG_SED_CRC	10100010	A2	24 bits	Program the calculated 32-bit CRC based on configuration bit values only into overall CRC register
LSC_READ_SED_CRC	10100100	A4	24 bits	Read the 32-bit SED CRC
LSC_PROG_PASSWORD	11110001	F1	24 bits	Program 64-bit password into the non-volatile memory (Efuse)
LSC_READ_PASSWORD	11110010	F2	24 bits	Read out the 64-bit password before activated for verification
LSC_SHIFT_PASSWORD	10111100	BC	24 bits	Shift in the password to unlock for re-configuration of the part is locked by the password
LSC_PROG_CIPHER_KEY	11110011	F3	24 bits	Program the 128-bit cipher key into Efuse
LSC_READ_CIPHER_KEY	11110100	F4	24 bits	Read out the 128-bit cipher key before activated for verification

**Table 11. Slave SPI Command Usage Table**

Command	OPCODE	CLASS	Flow Operation Number	Delay Time
ISC_NOOP	FF	C		None
READ_ID	E0	A	2,3	None
USERCODE	C0	A		None
LSC_READ_STATUS	3C	A	8	None
LSC_CHECK_BUSY	F0	A		None
LSC_REFRESH	79	D		<sup>1</sup>
ISC_ENABLE	C3	C	5	None
ISC_ENABLE_X	74	C		None
ISC_DISABLE	26	C	9	None
ISC_PROGRAM_USERCODE	C2	B		None
ISC_ERASE	0E	D		<sup>1</sup>
ISC_PROGRAM_DONE	5E	D		<sup>1</sup>
ISC_PROGRAM_SECURITY	CE	C		None
LSC_INIT_ADDRESS	46	C		None
LSC_WRITE_ADDRESS	B4	B		None
LSC_BITSTREAM_BURST	7A	C		None
LSC_PROG_INCR_RTI	82	B	6,7	<sup>1</sup>
LSC_PROG_INCR_ENC	B6	B		<sup>1</sup>
LSC_PROG_INCR_CMP	B8	B		<sup>1</sup>
LSC_PROG_INCR_CNE	BA	B		<sup>1</sup>
LSC_VERIFY_INCR_RTI	6A	A		None
LSC_PROG_CTRL0	22	B		None
LSC_READ_CTRL0	20	A		None
LSC_RESET_CRC	3B	C		None

**Table 11. Slave SPI Command Usage Table**

Command	OPCODE	CLASS	Flow Operation Number	Delay Time
LSC_READ_CRC	60	A		None
LSC_PROG_SED_CRC	A2	B		1
LSC_READ_SED_CRC	A4	A		None
LSC_PROG_PASSWORD	F1	B		1
LSC_READ_PASSWORD	F2	A		None
LSC_SHIFT_PASSWORD	BC	B		None
LSC_PROG_CIPHER_KEY	F3	B		1
LSC_READ_CIPHER_KEY	F4	A		None

1. Delay time depends on bitstream size and clock frequency. Duration could be in seconds. Please use LSC\_CHECK\_BUSY to check the status.

### Slave SPI Configuration Flow Diagrams

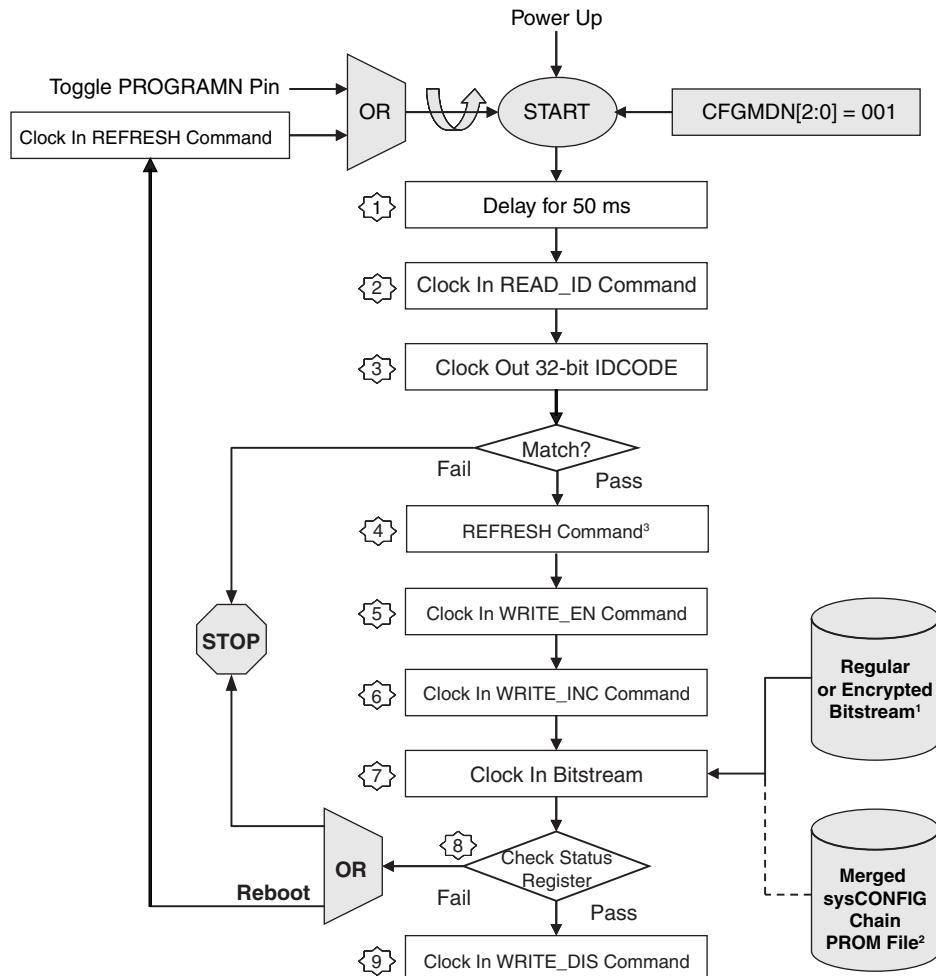
The Slave SPI port supports the regular ECP5 device bitstream and the encrypted bitstream (SSPI Mode).

The regular bitstream format is the same for all configuration modes. However, the encrypted bitstream format is configuration mode dependent. When generating an encrypted bitstream, the user must select the Slave SPI configuration mode (SSPI mode) for use with the Slave SPI port.

The Slave SPI configuration flow diagram is shown in Figure 8-4. Highlights are listed below.

- The bitstream file is a stand-alone file. It is not part of the driver or system code. This provides seamless system integration and flexible file management. For example, the bitstream can be switched on the fly without changing a single line of system code.
- The ECP5 device will wake up and enter User Mode once it reads in the entire bitstream. If it is necessary to delay the wake-up, the simplest method is by using the DONE pin. Wake-up can be delayed by holding the open-drain DONE pin low until the wake-up is desired.

**Figure 8-4. Slave SPI Configuration Flow Diagram**



Notes:

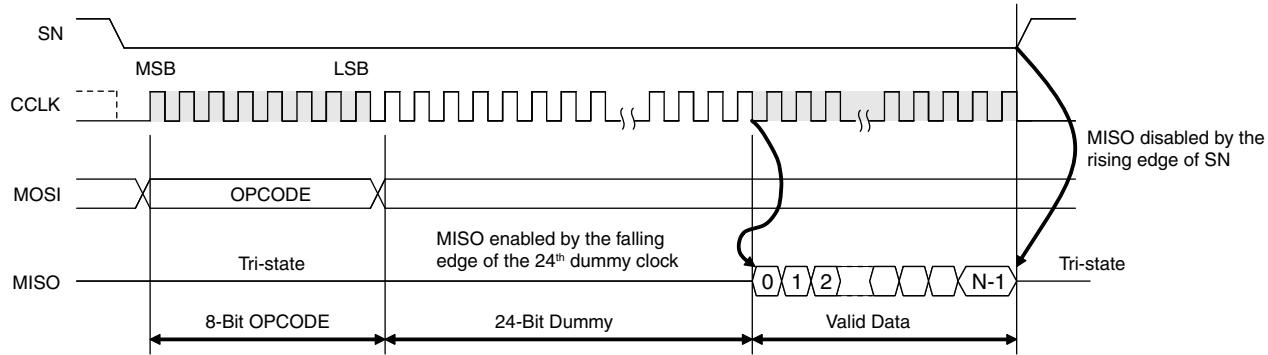
1. For a single ECP5 device, the input file is a bitstream, which may be a standard or encrypted bitstream.
2. For a sysCONFIG chain of devices, the input file can be a merged PROM file.  
Refer to the "Appendix E. Advanced Applications – Slave SPI sysCONFIG Daisy Chaining" on page 53 for more details.
3. Use REFRESH command with No Delay as a Class C command instead of a Class D command.  
This REFRESH command is not for clearing all the SRAM, therefore no delay is needed.

## Command Waveforms

### Class A Command Waveforms

The Class A commands are ones that read data out from the ECP5 devices. Bit 0 of the data or bitstream will be read out first. The twenty four (24) dummy clocks provide the device the necessary delay for the proper execution of the command.

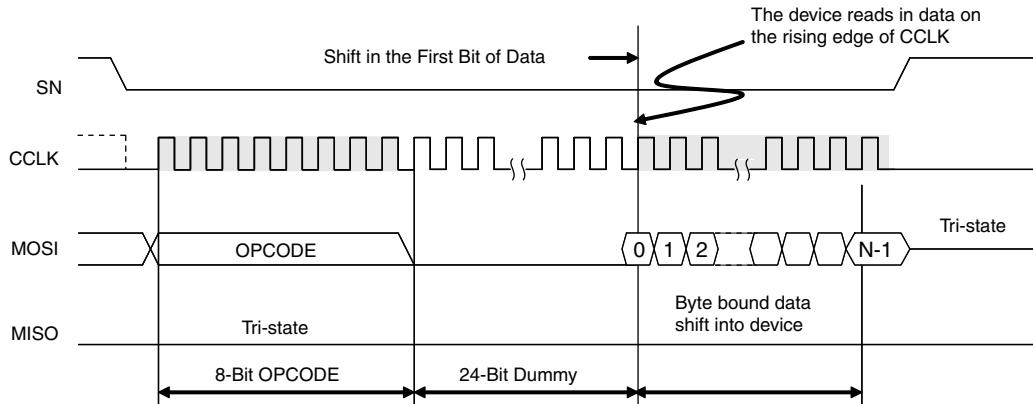
**Figure 8-5. Class A Command Waveforms**



### Class B Command Waveforms

The Class B commands are used to shift data into the port. Bit 0 of the data or bitstream is shifted in first. The twenty four (24) dummy clocks provide the device the necessary delay time to execute the command properly.

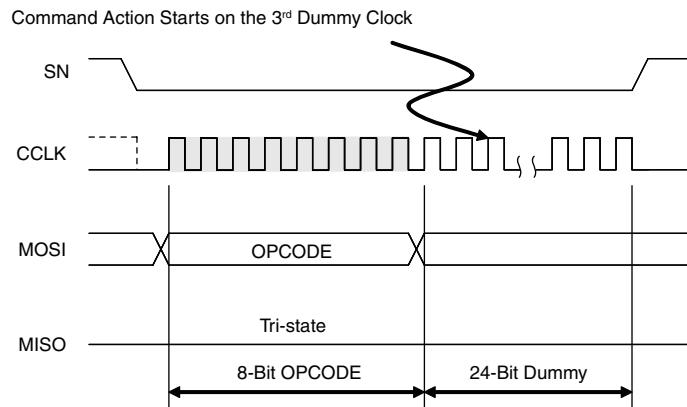
**Figure 8-6. Class B Command Waveforms**



### Class C Command Waveforms

The Class C commands do not require any data to be shifted in or out. The twenty four (24) dummy clocks provide the device the necessary delay for the proper execution of the command. Even if extra dummy clocks are presented, the device ignores them.

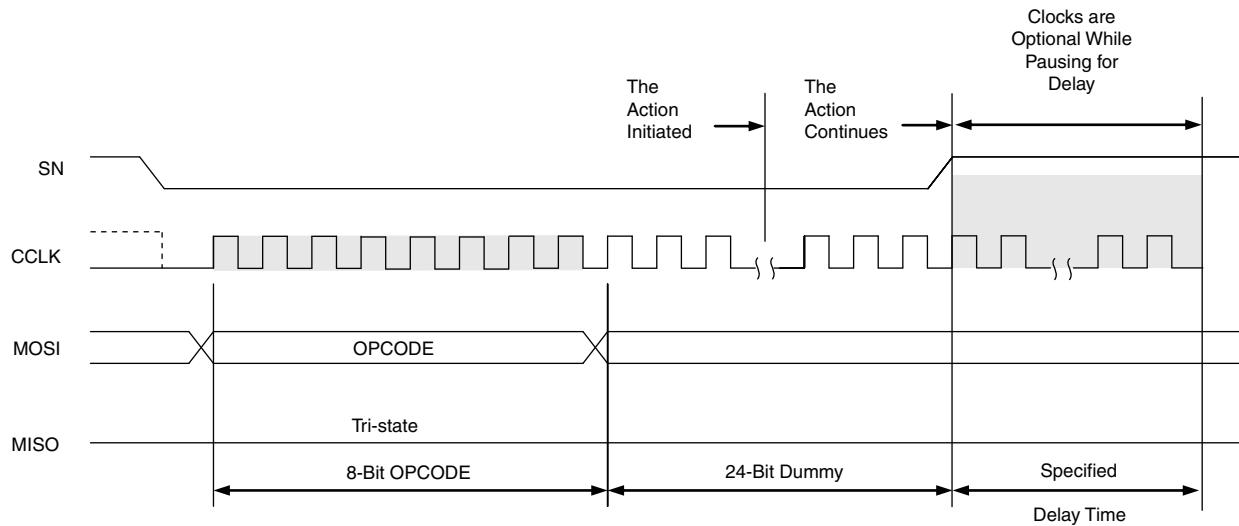
**Figure 8-7. Class C Command Waveforms**



### Class D Commands Waveforms

The Class D commands do not need to shift data in or out but still require a delay to execute the action associated with the command. This type of command cannot terminate the action of any commands including itself. After the 24th dummy clock, continuing to clock or suspending the clock or driving the SN pin high will not terminate the action. The action will end when it is complete. This class of commands is defined particularly for the benefit of the two unique commands: CLEAR and REFRESH.

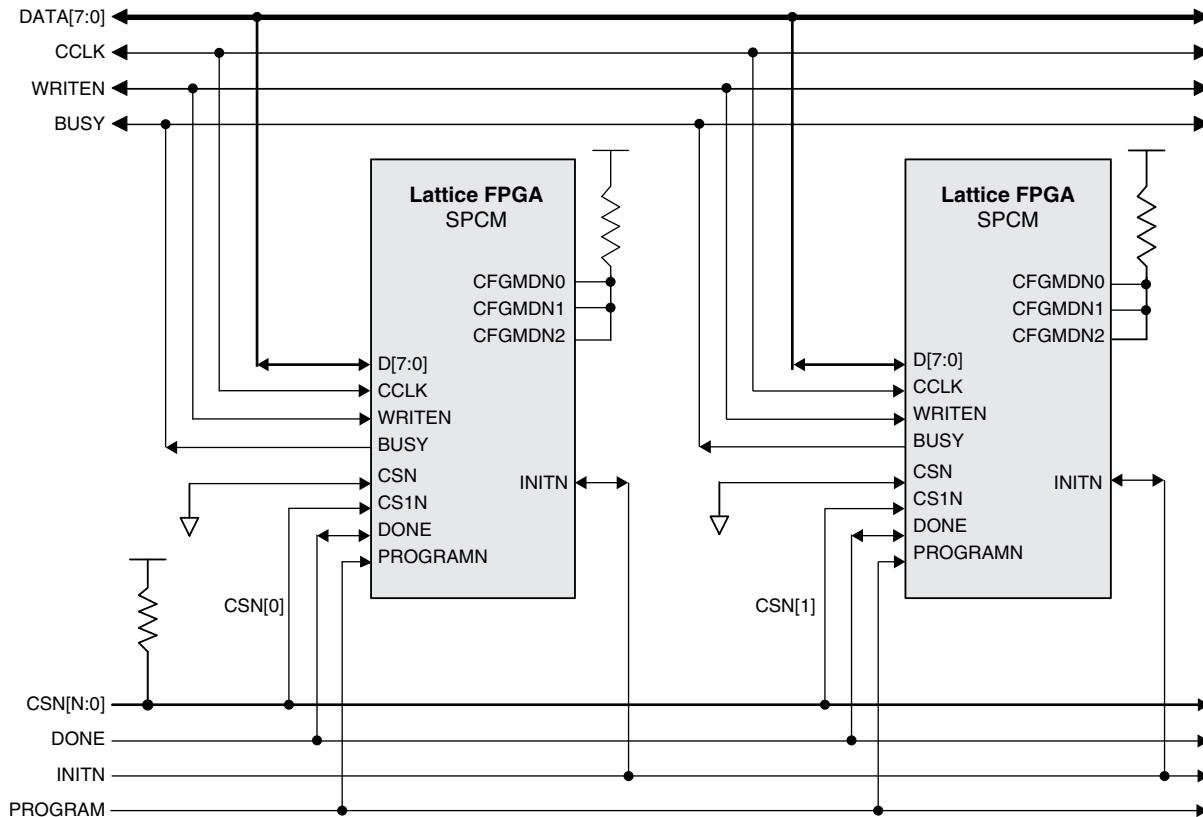
**Figure 8-8. Class D Command Waveforms**



## Slave Parallel Mode (SPCM)

The Slave Parallel interface supports 8-bit wide buses to configure or read back the SRAM. In Slave Parallel mode, a host system sends the configuration data in a byte-wide stream to the ECP5 device. 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[7:0]. D7 lines up with the very first bit (MSB of the first byte) in the bitstream. No internal pull-up or pull-down resistors are on the Data[7:0] pins during configuration, therefore the PCB design should include them.

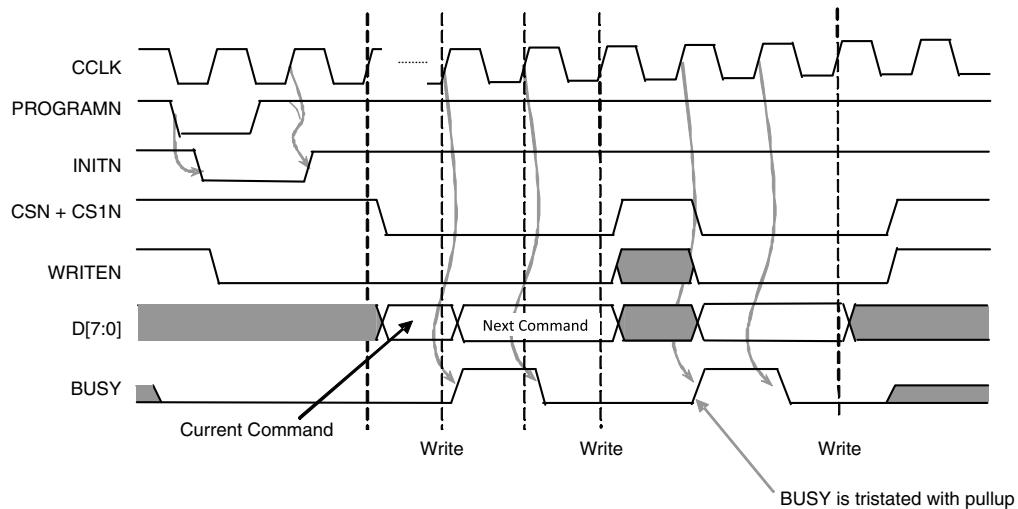
**Figure 8-9. Slave Parallel Interface**



In slave-parallel mode, multiple devices can be chained in parallel. The D[7:0], CCLK, WRITEN, BUSY, DONE, PROGRAMN, and INITN may be connected in parallel between devices. CSN is connected separately to allow individual devices to be selected. For example to select the first device in the chain, CSN[0] is set low while CSN[1] to CSN[n] is set high. After configuring, the first device, CSN[0] is set to high and CSN[1] is set to low to select the second device. CSN acts as a clock enable signal and CS1N starts and executes individual commands.

Figure 8-10 shows a slave-parallel write sequence. To send configuration data to a device, the WRITEN signal has to be asserted. During the write cycle, the BUSY signal provides handshaking between the host system and the device. When BUSY signal is low, the device is ready to read a byte of data at the next rising edge of CCLK. The BUSY signal is set high when the device reads the data and the device requires extra clock cycles to process the data. The CSN signal is used to temporary stop the write process by setting it to high, if the host system is busy. The device will resume the configuration when the CSN signal is set to low again. After the last byte of configuration data is sent, the WRITEN signal is set to high.

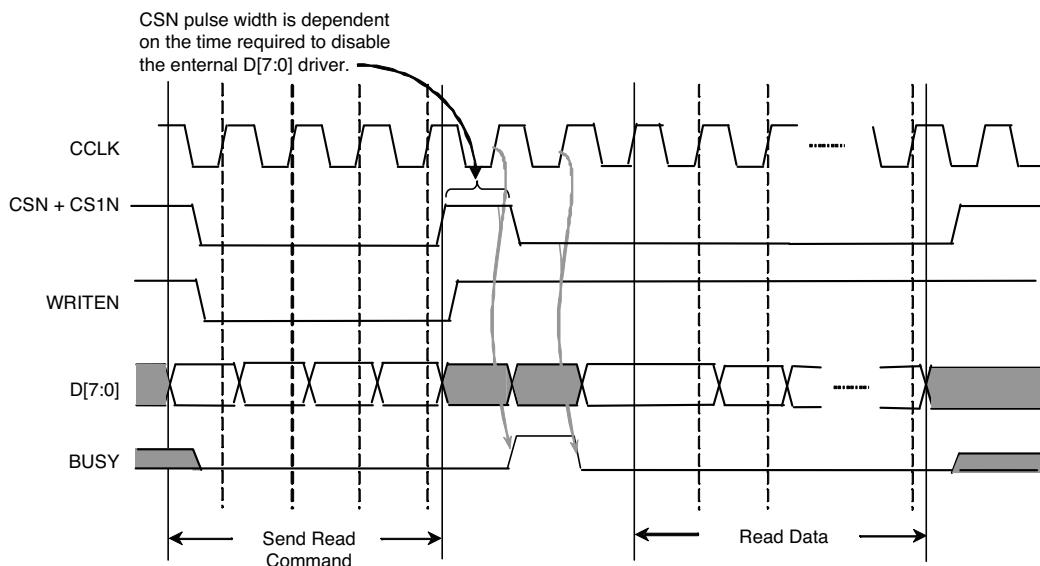
**Figure 8-10. Slave Parallel Write**



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[7:0] pins; WRITEN is then driven high and read data is output on D[0:7]. In order to support readback, the SLAVE PARALLEL PORT in the Diamond Spreadsheet View must be set to ENABLE.

To read back the configuration data or register contents, WRITEN is first set low to send the read instruction into the device. The device will read in the command from the CPU and execute the command. If the device cannot have the data ready by the next clock cycle, it will pull the busy signal up. When BUSY is high, the device will continue to execute the command regardless of the CSN pin. The device will pull the BUSY pin low when the data is ready but will not drive the D[7:0] until the CSN pin is pulled low by the CPU. The WRITEN pin is pulled high after sending in the command. Both the CSN and WRITEN signals are latched in and will switch the read-write mode on the rising edge of CCLK. If the device needs more than one clock cycle to switch the bus around, BUSY will be kept high until the D[7:0] is ready for read by CPU. As in the Write sequence, CSN signal is also used to temporary pulse read sequence in case the host system is busy. The data is read at the next rising CCLK edge, after CSN is set to low and BUSY is low. Slave-parallel reading is not available once configuration begins via the SPCM port. Reading is only available once the configuration is completed.

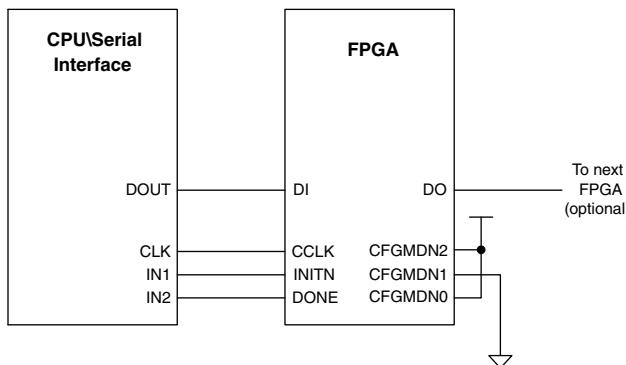
**Figure 8-11. Slave Parallel Read**



## Slave Serial Configuration Mode (SCM)

Slave Serial Configuration mode provides a simple, low pin count method for configuring one or more FPGAs. Data is presented to the FPGA on the Data Input pin DI at every CCLK rising edge.

**Figure 8-12. Slave Serial Block Diagram**



The bitstream data generated by Lattice Diamond is formatted so that it is ready to shift into the FPGA. Left shift the data out of the file in order for it to be correctly received by the FPGA.

The FPGA synchronizes itself on either a 0XBDB3 or 0BAB3 code word. It is critical that any data presented on DIN not be recognized as one of these two synchronization words early. To guarantee proper recognition of the synchronization word it is recommended that the synchronization word always be preceded by a minimum of 128 '1' bits. Presenting any other bitstream data, Programmer generated header information for example, risks the being misinterpreted due to bit slippage.

Slave Serial Configuration Mode can be used to configure a chain of FPGAs. Details about configuring a chain of devices is discussed in [Daisy Chaining](#).

## JTAG Mode

The JTAG port provides:

- Offline external Flash memory programming
- Background external Flash memory programming
- Direct SRAM configuration
- Full access to the ECP5 device Configuration Logic
- Device chaining
- IEEE 1149.1 testability
- IEEE 1532 Compliant programming

As a dedicated port on the ECP5 device, the JTAG port is always available. The JTAG port pins are dedicated to performing the IEEE 1149.1 TAP function.

The advantages of keeping the JTAG port include:

- **Multi-chain Architectures:** The JTAG port is the only configuration and programming port that permits the ECP5 device to be combined in a chain of other programmable logic.
- **Reveal Debug:** The Lattice Reveal debug tool is an embed-able logic analyzer tool. It allows you to analyze the logic inside the ECP5 device in the same fashion as an external logic analyzer permits analysis of board level logic. Reveal access is only available via the JTAG port.

- SRAM Readback:** The JTAG port is able to directly access the configuration SRAM. It is occasionally necessary to perform failure analysis for SRAM based FPGAs. A key component to failure analysis can involve reading the configuration SRAM.
- Boundary Scan Testability:** Board level connectivity testing performed using IEEE 1149.1 JTAG is a key capability for assuring the quality of assembled printed-circuit-boards. Lattice provides Boundary Scan Description Language files for the ECP5 device on the Lattice website.

## TransFR Operation

The ECP5, like other Lattice FPGAs, provides for the TransFR™ capability. TransFR is described in TN1087 [Minimizing System Interruption During Configuration Using TransFR Technology](#).

## Software Selectable Options

The operation of the ECP5 device configuration logic is managed by options selected in the Diamond design software. The ECP5 device uses dedicated I/O pins to select the configuration mode. You use the Diamond Spreadsheet View to make the changes to the operation of the ECP5 device programming which alters the operation of the configuration logic.

The configuration logic preferences are accessed using Spreadsheet View. Click on the Global Preferences tab, and look for the sysCONFIG tree. The sysCONFIG section is shown in Figure 8-13.

**Figure 8-13. sysCONFIG Preferences in Global Preferences Tab, Diamond Spreadsheet View**

Preference Name	Preference Value
Junction Temperature (Tj)(C)	125.000
Voltage (V)	1.045
SYSTEM_JITTER(ns)	Default
<b>Block Path</b>	
Block Asynchnpaths	ON
Block Resetpaths	ON
Block RD During WR Paths	OFF
Block InterClock Domain Paths	OFF
Block Jitter	OFF
<b>sysConfig</b>	
SLAVE_SPI_PORT	DISABLE
MASTER_SPI_PORT	DISABLE
SLAVE_PARALLEL_PORT	DISABLE
DONE_EX	OFF
DONE_OD	ON
DONE_PULL	ON
MCLK_FREQ	2.4
TRANSFR	OFF
CONFIG_IOVOLTAGE	2.5
CONFIG_SECURE	OFF
WAKE_UP	21
COMPRESS_CONFIG	OFF
CONFIG_MODE	JTAG
<b>User Code</b>	
UserCode Format	Binary
UserCode	0000000000000000000000000000000000000000000000000000000000000000
<b>Unique ID</b>	
UniqueId	0000
Global Set/Reset Net	
<b>Bank VCCIO</b>	
Bank0 (V)	Auto
Bank1 (V)	Auto
Bank2 (V)	Auto
Bank3 (V)	Auto
Bank6 (V)	Auto
Bank7 (V)	Auto
Bank8 (V)	Auto

**Table 8-1. sysCONFIG Options**

Option Name	Default Setting	All Settings
SLAVE_SPI_PORT	DISABLE	DISABLE, ENABLE
MASTER_SPI_PORT	DISABLE	DISABLE, ENABLE
SLAVE_PARALLEL_PORT	DISABLE	DISABLE, ENABLE
CONFIG_MODE	JTAG	JTAG, SSPI, SPI_SERIAL, SPI_DUAL, SPI_QUAD, SLAVE_PARALLEL, SLAVE_SERIAL
DONE_EX	OFF	OFF, ON
DONE_OD	ON	OFF, ON
DONE_PULL	ON	OFF, ON
MCCLK_FREQ	2.4	See MCCLK section below
TRANSFR	OFF	OFF, ON
CONFIG_IOVOLTAGE	2.5	1.2, 1.5, 1.8, 2.5, 3.3
CONFIG_SECURE	OFF	OFF, ON
WAKE_UP	21 [DONE_EX = Off] 4 [DONE_EX = On]	[4, 21]
COMPRESS_CONFIG	OFF	OFF ON
USERCODE Format	Binary	Binary, Hex, ASCII, Auto
USERCODE	0 (32 binary zeros)	32-bit
UNIQUE_ID	0	Upper 16-bit user defined code for above Auto USERCODE.

### Slave SPI Port

The SLAVE\_SPI\_PORT allows you to preserve the Slave SPI configuration port after the ECP5 device enters user mode. There are two states to which the SLAVE\_SPI\_PORT preference can be set:

- **ENABLE** – This setting preserves the SPI port I/O when the ECP5 device is in user mode. When the pins are preserved, an external SPI master controller can interact with the configuration logic. The preference also prevents you from over-assigning I/O to the port pins.
- **DISABLE** – This setting disconnects the SPI port pins from the configuration logic. By itself it does not make the port pins general purpose I/O. Both the SLAVE\_SPI\_PORT and MASTER\_SPI\_PORT must be in the DISABLE state for the SPI port pins to be general purpose I/O.

The SLAVE\_SPI\_PORT cannot be enabled at the same time as the MASTER\_SPI\_PORT. It is necessary to guarantee that the internal Master SPI controller, if the customer has created one, does not perform SPI transactions at the same time as an external SPI master. It is your responsibility to prevent two SPI masters from operating simultaneously.

### Master SPI Port

The MASTER\_SPI\_PORT allows you to preserve the SPI configuration port after the ECP5 device enters user mode. There are two states to which the MASTER\_SPI\_PORT preference can be set:

- **ENABLE** – This setting preserves the SPI port I/O when the ECP5 device is in user mode. The preference also prevents you from over-assigning I/O to the port pins.
- **DISABLE** – This setting disconnects the SPI port pins from the configuration logic. By itself it does not make the port pins general purpose I/O. Both the SLAVE\_SPI\_PORT and MASTER\_SPI\_PORT must be in the DISABLE state for the SPI port pins to be general purpose I/O. Select this setting if you prefer to generate your own Master SPI Controller in user mode.

### Slave Parallel Port

The SLAVE\_PARALLEL\_PORT allows you to preserve the SPCM configuration port after the ECP5 device enters user mode. There are two states to which SLAVE\_PARALLEL\_PORT preference can be set:

- **ENABLE** – This setting preserve the SPCM port when the ECP5 device is in user mode. It allows user to read all of the con-figuration data, except the EBR and the distributed RAM contents in user mode.
- **DISABLE** – This setting disconnects the SPCM port pins from the configuration logic. It allow SPCM ports pins to be general purpose I/O.

### MCCLK Frequency

The MCLK\_FREQ preference allows you to alter the MCLK frequency used to retrieve data from an external SPI Flash when using EXTERNAL or Dual Boot configuration modes. The ECP5 device uses a nominal 2.4MHz (+/- 15%) clock frequency to begin retrieving data from the external SPI Flash. The MCLK\_FREQ value is stored in the incoming configuration data. The ECP5 device reads a series of padding bits, a “start of data” word (0xBDB3) and a control register value. The control register contains the new MCLK\_FREQ value. The ECP5 device switches to the new clock frequency shortly after receiving the MCLK\_FREQ value. The MCLK\_FREQ has a range of possible frequencies available from 2.4 MHz up to 38.8 MHz (see Table 8-6). Take care not to exceed the maximum clock rate of your SPI Flash, or of your printed circuit board.

### TRANSFR

The TransFR function used by the ECP5 device requires the configuration data loaded into the configuration SRAM, and any future configuration data file loaded into the external Flash memory have the TRANSFR set to the ENABLE state. See the [TransFR Operation](#) section and TN1087 [Minimizing System Interruption During Configuration Using TransFR Technology](#) for more information about using TransFR.

### **COMPRESS\_CONFIG**

The COMPRESS\_CONFIG preference alters the way files are generated. The COMPRESS\_CONFIG default setting is to be ON.

### **UNIQUE\_ID**

The ECP5 device contains a 16-bit register for storing a user-defined value. The default value stored in the register is 0x0000. UniqueID can only be set when USERCODE\_FORMAT is Auto.

### **USERCODE**

The ECP5 device contains a 32-bit register for storing a user-defined value. The default value stored in the register is 0x00000000. Using the USERCODE preference you can assign any value to the register you desire. Suggested uses include the configuration data version number, a manufacturing ID code, date of assembly, or the JEDEC file checksum.

The format of the USERCODE field is controlled using the USERCODE\_FORMAT preference. Data entry can be performed in either Binary, Hex, ASCII or Auto formats.

### **USERCODE\_FORMAT**

The USERCODE\_FORMAT preference selects the format for the data field used to assign a value in the USERCODE preference. The USERCODE\_FORMAT has three options:

- **Binary** – USERCODE is set using 32 ‘1’ or ‘0’ characters.
- **Hex** – USERCODE is set using eight hexadecimal digits (i.e., 0-9A-F)
- **ASCII** – USERCODE is set using up to four ASCII characters
- **Auto** – USERCODE is automatically created by the software. The upper 16 bits is UniqueID and the lower 16 bits are sequentially increased automatically for every bitstream generation.

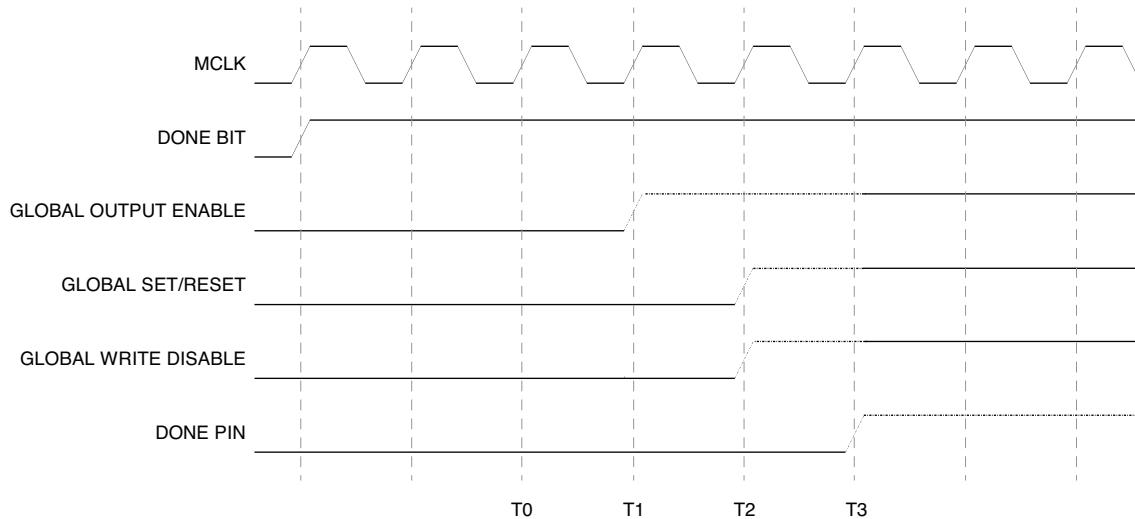
### **CONFIG\_SECURE**

When this preference set to ON, the read-back of the SRAM memory is blocked. The device must be reprogrammed in order to reset the security setting. Once the security fuses are reset, the device can be programmed again.

## **Device Wake-up Sequence**

When configuration is complete (the SRAM has been loaded), the device will wake up in a predictable fashion. If the ECP5 device is the only device in the chain, or the last device in a chain, the wake-up process should be initiated by the completion of configuration. Once configuration is complete, the internal DONE bit will be set and then the wake-up process will begin. Figure 8-14 shows the wake-up sequence 21 using the internal clock.

**Figure 8-14. Wake-up Sequence Using Internal Clock**



## Wake-up Signals

Three internal signals, GSR, GWDIS, and GOE, determine the wake-up sequence.

- 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. This control signal does not control the primary input pin to the device but controls specific control ports of EBR and LUTs.
- When low, GOE prevents the device's I/O buffers from driving the pins. The GOE only controls **output** pins. Once the internal DONE is asserted the ECP5 device will respond to input data.
- When high, the DONE pin indicates that configuration is complete and that no errors were detected.

The wake-up sequences available to users are shown in Table 8-2. A wake-up sequence is the order in which the signals change. The phases transition based on a wakeup clock, as discussed below. The exact timing relationship between the internal signals and the wakeup clock varies and is not specified.

**Table 8-2. Wake-up Sequences**

Sequence	Phase T0	Phase T1	Phase T2	Phase T3
4 (Default if DONE_EX = ON)	DONE	GOE	GWDIS, GSR	
21 (Default if DONE_EX = OFF)		GOE	GWDIS, GSR	DONE

## Wake-up Clock Selection

The clock source used to complete the four state transitions in the wake-up sequence is user-selectable. Once the ECP5 device is configured, it enters the wake-up state, which is the transition between the configuration mode and user mode. This sequence is synchronized to a clock source, which defaults to internal clock. The start-up clock can be user-defined and brought into the device. This user clock cannot exceed 100MHz and instantiated as shown below in the user design.

You can change the clock used by instantiating the START macro in your Verilog or VHDL. The clock must be supplied on an external input pin, because the ECP5 device does not begin internal operations until the Wake-up sequence is complete. There is no external indication the device is ready to perform the last four state transitions. You must either provide a free running clock frequency, or you must wait until the device is guaranteed to be ready

---

to wake up. Using the START macro provides another mechanism for holding off configuring one or more programmable devices and then starting them in lock step.

**Verilog**

```
module START (STARTCLK);
  input STARTCLK;
endmodule

START u1 (.STARTCLK(<clock_name>)) /* synthesis syn_noprune=1 */;
```

**VHDL**

```
COMPONENT START
  PORT(
    STARTCLK      : IN STD_ULOGIC
  );
END COMPONENT;
attribute syn_noprune: boolean ;
attribute syn_noprune of START: component is true;

begin
  u1: START port map (STARTCLK =><clock name>);
```

## Daisy Chaining

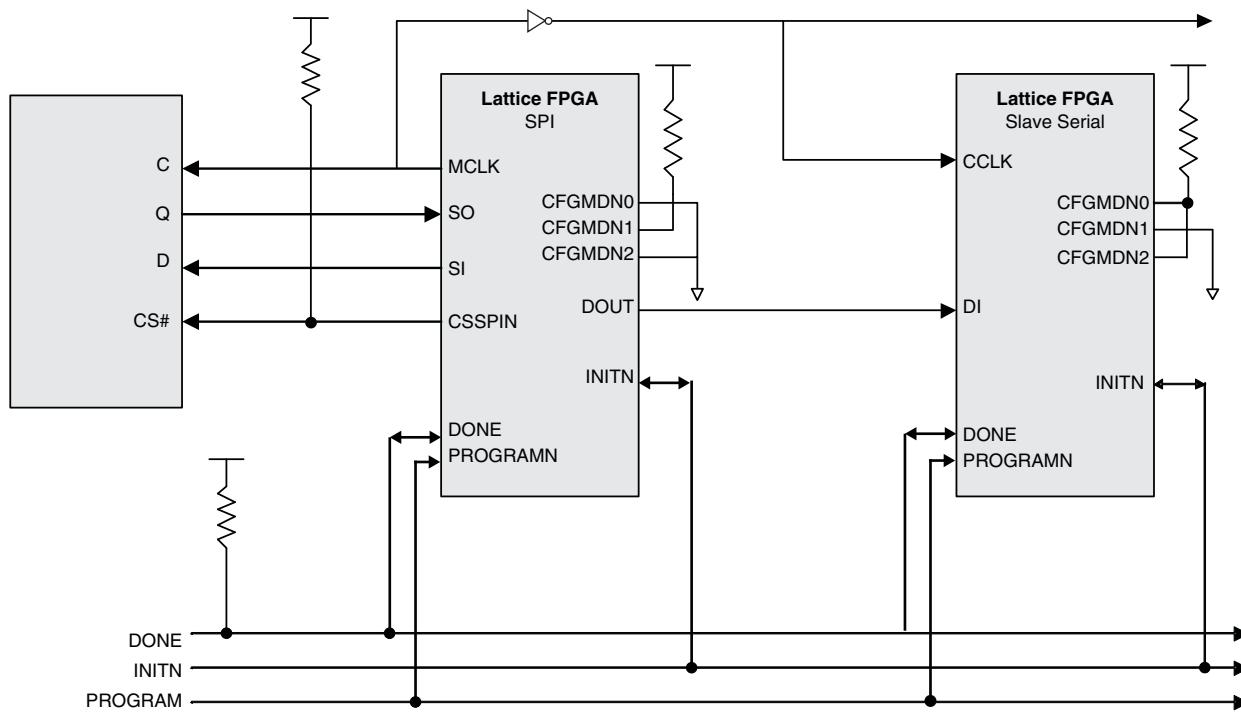
Typically, there is one configuration bitstream per FPGA in a system. Today's systems have several FPGA devices. If all the FPGAs in the application utilize the same device and use the same bitstream, only a single bitstream is required. Using a ganged configuration will load multiple, similar FPGAs with the same bitstream at the same time.

However, to save PCB space and use external storage device more efficiently, several different FPGA bitstreams from various devices and designs can share a single configuration mechanism by using a daisy chain method.

ECP5 devices support two distinct daisy chaining methods. Bypass Option or Flowthrough Option must be set for all Lattice FPGA devices in the sysCONFIG chain when using daisy chaining. The 'Synchronous to External DONE pin' option (DONE\_EX) must be enabled in Lattice Diamond Spreadsheet View.

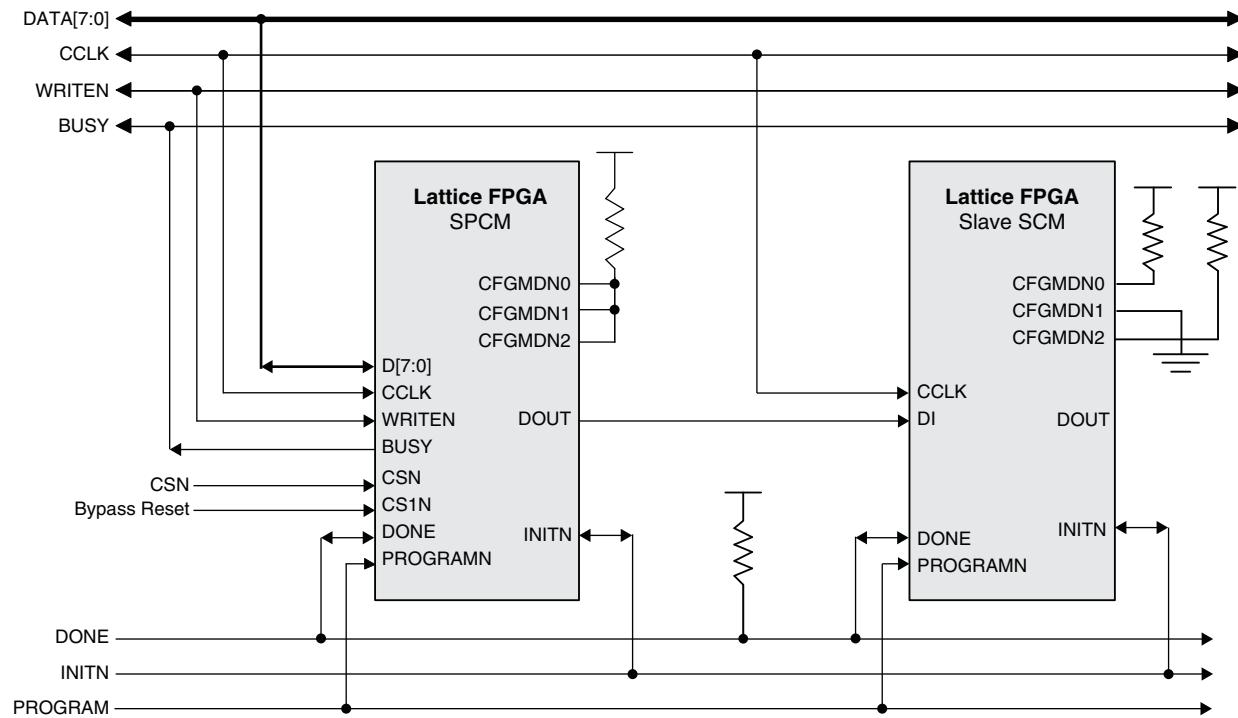
The first method supports multiple FPGAs from a single configuration source such as a SPI Flash device or the serial port of a microprocessor. In this scenario, the storage device contains all the configuration data to program all the FPGAs. The data comes into the first FPGA by its sysCONFIG port programming the first part. After the first part completes its configuration, it serially sends the remaining data from its DOUT output to the daisy chained device which receives the data into the SCM sysCONFIG port. Examples are shown as Figure 8-15 and Figure 8-16. Bypass Option must be set in this scenario.

**Figure 8-15. Serial Daisy Chaining**



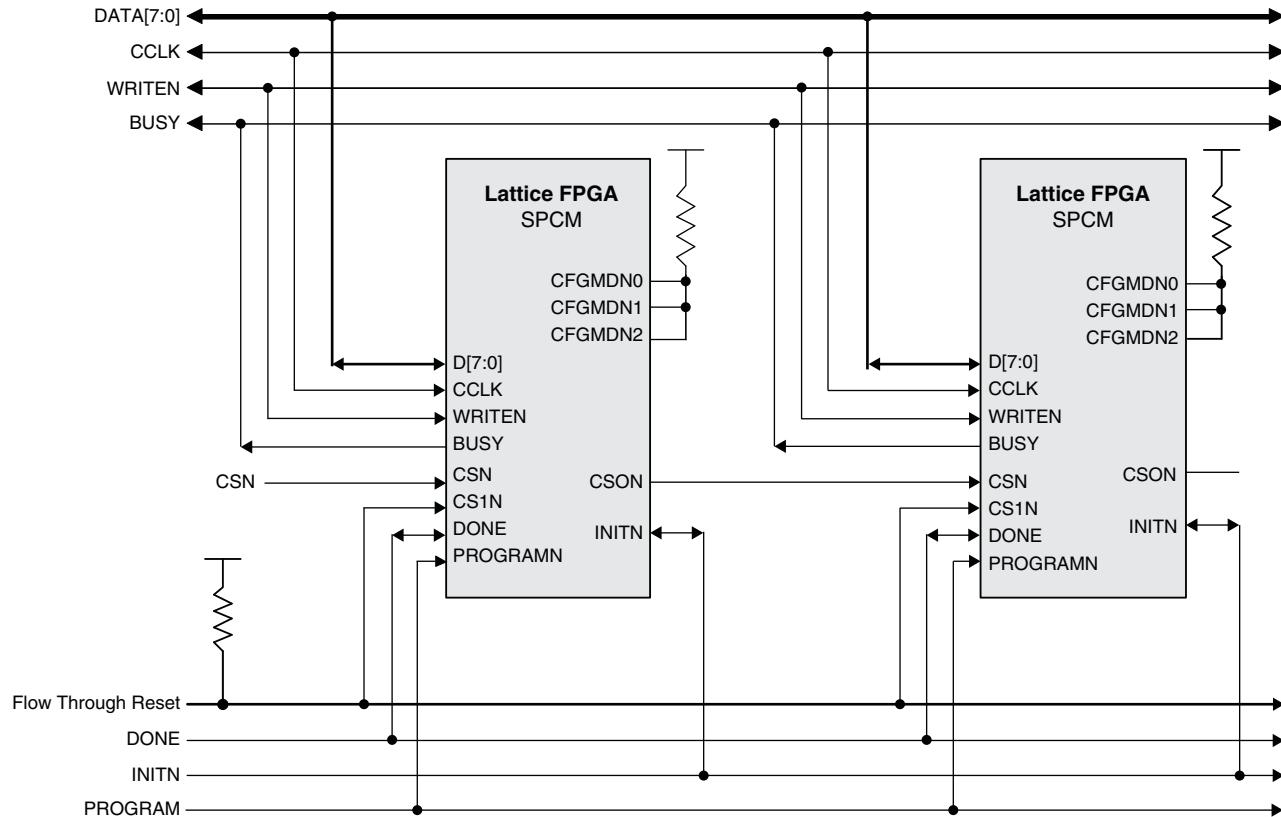
Note: inverter from MCLK is to guarantee DI hold time of the daisy chained FPGA device.

**Figure 8-16. SPCM with Serial Daisy Chain**



The second method permits sharing an 8-bit data bus from a storage device or CPU. The FPGAs use handshaking to signal which FPGA on the bus is reading the configuration data. An example is shown in Figure 8-17. FLowthrough Option must be set in this scenario.

**Figure 8-17. SPCM Configuration with Daisy Chaining**



### Bypass Option

The Bypass option can be set by using the Diamond strategy properties, or a chain of bitstreams can be assembled and Bypass set using the Diamond Deployment Tool. The Bypass option is not supported when using dual-, or multi-boot configurations. The Bypass option is supported when using encrypted bitstream files with ECP5 devices.

When the first device completes configuration, and a Bypass command is included within 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 modes, the Bypass option causes the excess data coming in on D[15:0] 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.

## Flowthrough Option

The Flowthrough option can be used with parallel daisy chains only. The Flowthrough option is not supported when using encrypted bitstream files with ECP5 devices. Flowthrough does not support SCM, SSPI, SPI, and SPI modes.

When the first device completes configuration and a Flowthrough command is included with the bitstream, the CS<sub>ON</sub> pin is driven low. In addition to driving CS<sub>ON</sub> low, Flowthrough also tri-states the device's D[15:0] and BUSY pins in order to avoid contention with the other daisy chained devices. Once the Flowthrough option starts, the device will remain in Flowthrough until the wake-up sequence completes. If Flowthrough needs to be aborted, drive both CS<sub>N</sub> and CS<sub>1N</sub> high. This acts as a Flowthrough reset signal.

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
March 2014	01.0	Initial release,

## Appendix A. ECP5 Slave SPI Programming Guide

The SPI port of the ECP5 device can be used for device programming and user function. When it is set to Slave SPI mode, the port can be used for direct or background Flash programming. The Slave SPI port must be enabled in order to support the device programming using the SPI protocol. This is done by setting the SLAVE\_SPI\_PORT preference to ENABLE in the software. Slave SPI programming supports single device programming.

The Lattice programming software, Diamond Programmer, supports the Slave SPI programming mode as one of the device access options. Selecting this option allows users to perform device erase, program, verify, readback, refresh, and more. The connections of Slave SPI pins to the Lattice programming cable are:

- TDI -> SISPI
- ISPEN -> SN
- TCK -> CCLK
- TDO -> SPISO

The Slave SPI chip select pin (SN) is held low during the command sequences. The SVF file generated by the Deployment Tool software for the Slave SPI programming mode shows the details of the command sequences.

## Appendix B. ECP5 Bitstream File Format

### Configuration Bitstream Format

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 8-3 shows the format of a non-encrypted and compressed bitstream. The bitstream consists of a comment string, a header, the preamble, and the configuration setup and data.

Only the frame data can be compressed. The EBR data cannot be compressed.

The data frame padding for compression is as follows:

1. Before compression, pad the leftmost bits of the frame data with zeroes (0) to make the data frame 64-bit bounded. For example, if the original uncompressed data frame length is 125-bit, such as 01.....10, the data frame has to be padded with all 0 (3-bit 0s) at leftmost to make it 64-bit bounded (128-bit), 00001.....10.
2. After compressing the frame data, pad the rightmost bits of the frame data with zeroes (0) to make the data frame byte bounded. For example, if the 128-bit data frame is compressed to become 101-bit (not byte bounded), such as 10.....01, the compressed data frame has to be padded with all 0 (3-bit 0s) at the rightmost to make it byte bounded (104-bit), 10.....01000.

**Table 8-3. ECP5 Compressed Bitstream Format**

Frame	Contents (D0..D7)	Description	CRC
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator (See Table 10).	None
Header	(LSB) 1111111111111111	First 16 bits of the 32-bit Bitstream Preamble.	
	1011110110110011	Second 16 bits of the 32-bit Bitstream Preamble (0xFFFFBDB3).	
	11111111...11111111	32-bit Dummy	
Frame (Reset CRC)	00111011	LSC_RESET_CRC Command.	Exclude
	0	CRC Comparison Flag (0 = no).	Exclude
	000...0000	23-bit Command Information (23 zeros).	Exclude
Frame (Verify ID) See Note <sup>6</sup>	11100010	VERIFY_ID Command.	Start: Include if <sup>6</sup>
	0	CRC Comparison Flag (0 = no) for Verify ID command.	Include if <sup>6</sup>
	000...0000	23-bit Command Information (23 zeros).	Include if <sup>6</sup>
	(DeviceID[31..0])	32-bit Device ID.	Include if <sup>6</sup>
Frame (Store Compress)	00000010	LSC_WRITE_COMP_DIC Command.	Include
	0	CRC Comparison Flag (0 = no).	Include
	000...0000	23-bit Command Information (23 zeros).	Include
	(Pattern8[7..0])	Next most frequently occurring 8-bit pattern occurring in the Frame Data.	Include
	(Pattern7[7..0])	Next most frequently occurring 8-bit pattern occurring in the Frame Data.	Include
	•	•	•
	•	•	•
	(Pattern1[7..0])	Next most frequently occurring 8-bit pattern occurring in the Frame Data.	Include
Frame (Control Register 0)	00100010	LSC_PROG_CNTRL0 Command.	Include
	0	CRC Comparison Flag (0 = no).	Include
	000...0000	23-bit Command Information (23 zeros).	Include
	(CtlReg0[31..0])	Control Register 0 Data (See Note <sup>8</sup> ).	Include

<b>Frame</b>	<b>Contents (D0..D7)</b>	<b>Description</b>	<b>CRC</b>
Frame (Reset Address)	01000110	LSC_INIT_ADDRESS Command.	Include
	0	CRC Comparison Flag (0 = no).	Include
	000...0000	23-bit Command Information (23 zeros).	Include
Frame (Write Increment)	10111000	LSC_PROG_INCR_CMP Command.	Include
	1	CRC Comparison Flag (1 = yes).	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame).	Include
	0	Include dummy bits in bitstream.	Include
	1	Dummy definition (1 = use Bit[3:0] as dummy byte count).	Include
	0001	Dummy definition.	Include
	(16-bit Frame Size)	Number of configuration data frames included in operand (see Table 103). For example: 2819 Frames = 0x0B03 or b000101100000011.	
Frame (Data 0)	(Frame 0 Data)	x Data bits for Frame 0 (See Table 103 and Notes 1 and 2).	Include: End
	(CRC[15..0])	16-bit CRC (See Note <sup>4</sup> ).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Data 1)	(Frame 1 Data)	x Data bits for Frame 1 (See Table 103 and Notes 1 and 2).	Include: End
	(CRC[15..0])	16-bit CRC (See Note <sup>4</sup> ).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
• • •	• • •	• • •	• • •
Frame (Data n-1)	(Frame n-1 Data)	x Data bits for Frame n-1 (See Table 103 and Notes 1 and 2).	Include: End
	(CRC[15..0])	16-bit CRC (See Note <sup>4</sup> ).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (SED CRC) (Optional) See Note <sup>3</sup>	10100010	LSC_PROG_SED_CRC Command.	Include if <sup>3</sup>
	0	CRC Comparison Flag (0 = no).	Include if <sup>3</sup>
	000...0000	23-bit Command Information (23 zeros).	Include if <sup>3</sup>
	(CRC[31..0])	32-bit SED CRC (See Note <sup>3</sup> ).	Include if <sup>3</sup>
Frame (Program Security) (Optional) See Note <sup>7</sup>	11001110	ISC_PROGRAM_SECURITY Command (See Note <sup>7</sup> ).	Include if <sup>7</sup>
	0	CRC Comparison Flag (0 = no).	Include if <sup>7</sup>
	000...0000	23-bit Command Information (23 zeros).	Include if <sup>7</sup>
Frame (Usercode)	11000010	ISC_PROGRAM_USERCODE Command.	Include
	1	CRC Comparison Flag (1 = yes).	Include
	000...0000	23-bit Command Information (23 zeros).	Include
	(U[31..0])	32-bit Usercode Data.	Include: End
	(CRC[15..0])	16-bit CRC (See Note <sup>4</sup> ).	CRC
Frame (EBR Address) (Optional) See Note <sup>5</sup>	11110110	EBR Address Command.	Start: Include if <sup>5</sup>
	0	CRC Comparison Flag (0 = no).	Include if <sup>5</sup>
	000...0000	23-bit Command Information (23 zeros).	Include if <sup>5</sup>
	000000000000	11-bit dummy (11 zeros).	Include if <sup>5</sup>
	(10-bit EBR ID)	10-bit EBR ID. One = 0000000001.	Include if <sup>5</sup>
	(11-bit Address)	11-bit EBR Beginning address (default address 0).	Include if <sup>5</sup>

<b>Frame</b>	<b>Contents (D0..D7)</b>	<b>Description</b>	<b>CRC</b>
Frame (EBR Write) (Optional) See Note <sup>5</sup>	10110010	LSC_EBR_WRITE Command.	Include if <sup>5</sup>
	1	CRC Comparison Flag (1 = yes).	Include if <sup>5</sup>
	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands).	Include if <sup>5</sup>
	0	Exclude dummy bytes from bitstream.	Include if <sup>5</sup>
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings).	Include if <sup>5</sup>
	0000	Dummy definition.	Include if <sup>5</sup>
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001. For 1024*18/72 = 256 EBR, the size = 100000000.	Include if <sup>5</sup>
	(Data)	Data frame for one EBR. For 256 EBR, the data is treated in 2Kx9 mode. The data order is B0[8:0], B1[8:0], B2[8:0], ...	Include End if <sup>5</sup>
	(CRC[15..0])	16-bit CRC (See Note <sup>4</sup> ).	CRC
• • •	• • •	• • •	• • •
Frame (EBR Address) (Optional) See Note <sup>5</sup>	11110110	EBR Address Command.	Start: Include if <sup>5</sup>
	0	CRC Comparison Flag (0 = no).	Include if <sup>5</sup>
	000...0000	23-bit Command Information (23 zeros).	Include if <sup>5</sup>
	000000000000	11-bit dummy (11 zeros).	Include if <sup>5</sup>
	(10-bit EBR ID)	10-bit EBR ID. One = 0000000001.	Include if <sup>5</sup>
	(11-bit Address)	11-bit EBR Beginning address (default address 0).	Include if <sup>5</sup>
Frame (EBR Write) (Optional) See Note <sup>5</sup>	10110010	LSC_EBR_WRITE Command.	Include if <sup>5</sup>
	1	CRC Comparison Flag (1 = yes).	Include if <sup>5</sup>
	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands).	Include if <sup>5</sup>
	0	Exclude dummy bytes from bitstream.	Include if <sup>5</sup>
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings).	Include if <sup>5</sup>
	0000	Dummy definition.	Include if <sup>5</sup>
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001. For 1024*18/72 = 256 EBR, the size = 100000000.	Include if <sup>5</sup>
	(Data)	Data frame for one EBR. For 128 EBR, the data is treated in 2Kx9 mode. The data order is B0[8:0], B1[8:0], B2[8:0], ...	Include End if <sup>5</sup>
	(CRC[15..0])	16-bit CRC (See Note <sup>4</sup> ).	CRC
Frame (Program Done)	01011110	ISC_PROGRAM_DONE Command.	Exclude
	0	CRC Comparison Flag (0 = no).	Exclude
	000...0000	23-bit Command Information (23 zeros).	Exclude
Frame (DUMMY)	1111...1111	4 bytes DUMMY command (all ones).	Exclude

**Table 8-4. ECP5 Uncompressed Bitstream Format**

Frame	Contents (D0..D7)	Description	CRC
Comments	(Comment String)	ASCII Comment (Argument) String and Terminator (See Table 10).	None
Header	(LSB) 1111111111111111	First 16 bits of the 32-bit Bitstream Preamble.	
	1011110110110011	Second 16 bits of the 32-bit Bitstream Preamble (0xFFFFFBDB3).	
	11111111...11111111	32-bit Dummy	
Frame (Reset CRC)	00111011	LSC_RESET_CRC Command.	Exclude
	0	CRC Comparison Flag (0 = no).	Exclude
	000...0000	23-bit Command Information (23 zeros).	Exclude
Frame (Verify ID) See Note <sup>6</sup>	11100010	VERIFY_ID Command.	Start: Include if <sup>6</sup>
	0	CRC Comparison Flag (0 = no) for Verify ID command.	Include if <sup>6</sup>
	000...0000	23-bit Command Information (23 zeros).	Include if <sup>6</sup>
	(DeviceID[31..0])	32-bit Device ID.	Include if <sup>6</sup>
Frame (Control Register 0)	00100010	LSC_PROG_CNTRL0 Command.	Include
	0	CRC Comparison Flag (0 = no).	Include
	000...0000	23-bit Command Information (23 zeros).	Include
	(CtlReg0[31..0])	Control Register 0 Data (See Note <sup>8</sup> ).	Include
Frame (Reset Address)	01000110	LSC_INIT_ADDRESS Command.	Include
	0	CRC Comparison Flag (0 = no).	Include
	000...0000	23-bit Command Information (23 zeros).	Include
Frame (Write Increment)	10000010	LSC_PROG_INCR_RTI Command.	Include
	1	CRC Comparison Flag (1 = yes).	Include
	0	CRC Comparison at End Flag (0 = Execute CRC comparison at the end of each data frame).	Include
	0	Include dummy bits in bitstream.	Include
	1	Dummy definition (1 = use Bit[3:0] as dummy byte count).	Include
	0001	Dummy definition.	Include
	(16-bit Frame Size)	Number of configuration data frames included in operand (see Table 103). For example: 2819 Frames = 0x0B03 or b000101100000011.	Include
Frame (Data 0)	(Frame 0 Data)	x Data bits for Frame 0 (See Table 103 and Note <sup>2</sup> ).	
Include: End			
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
Frame (Data 1)	(Frame 1 Data)	x Data bits for Frame 1 (See Table 103 and Note <sup>2</sup> ).	
Include: End			
	(CRC[15..0])	16-bit CRC (See Note 4).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include
•	•	•	•
Frame (Data n-1)	(Frame n-1 Data)	x Data bits for Frame n-1 (See Table 103 and Note <sup>2</sup> ).	
Include: End			
	(CRC[15..0])	16-bit CRC (See Note <sup>4</sup> ).	CRC
	11111111	Dummy byte (8 bits of ones)	Start: Include

<b>Frame</b>	<b>Contents (D0..D7)</b>	<b>Description</b>	<b>CRC</b>
Frame	10100010	LSC_PROG_SED_CRC Command.	Include if <sup>3</sup>
(SED CRC)	0	CRC Comparison Flag (0 = no).	Include if <sup>3</sup>
(Optional)	000...0000	23-bit Command Information (23 zeros).	Include if <sup>3</sup>
See Note <sup>3</sup>	(CRC[31..0])	32-bit SED CRC (See Note <sup>3</sup> ).	Include if <sup>3</sup>
Frame	11001110	ISC_PROGRAM_SECURITY Command (See Note <sup>7</sup> ).	Include if <sup>7</sup>
(Program Security)	0	CRC Comparison Flag (0 = no).	Include if <sup>7</sup>
(Optional) See Note <sup>7</sup>	000...0000	23-bit Command Information (23 zeros).	Include if <sup>7</sup>
Frame	11000010	ISC_PROGRAM_USERCODE Command.	Include
(Usercode)	1	CRC Comparison Flag (1 = yes).	Include
	000...0000	23-bit Command Information (23 zeros).	Include
	(U[31..0])	32-bit Usercode Data.	Include: End
	(CRC[15..0])	16-bit CRC (See Note <sup>4</sup> ).	CRC
Frame	11110110	EBR Address Command.	Start: Include if 5
(EBR Address)	0	CRC Comparison Flag (0 = no).	Include if <sup>5</sup>
(Optional)	000...0000	23-bit Command Information (23 zeros).	Include if <sup>5</sup>
See Note <sup>5</sup>	000000000000	11-bit dummy (11 zeros).	Include if <sup>5</sup>
	(10-bit EBR ID)	10-bit EBR ID. One = 0000000001.	Include if <sup>5</sup>
	(11-bit Address)	11-bit EBR Beginning address (default address 0).	Include if <sup>5</sup>
Frame	10110010	LSC_EBR_WRITE Command.	Include if <sup>5</sup>
(EBR Write)	1	CRC Comparison Flag (1 = yes).	Include if <sup>5</sup>
(Optional)	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands).	Include if <sup>5</sup>
See Note <sup>6</sup>	0	Exclude dummy bytes from bitstream.	Include if <sup>5</sup>
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings).	Include if <sup>5</sup>
	0000	Dummy definition.	Include if <sup>5</sup>
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001. For 1024*18/72 = 256 EBR, the size = 100000000.	Include if <sup>5</sup>
	(Data)	Data frame for one EBR. For 128 EBR, the data is treated in 2Kx9 mode. The data order is B0[8:0], B1[8:0], B2[8:0], ...	Include: End if <sup>5</sup>
	(CRC[15..0])	16-bit CRC (See Note <sup>4</sup> ).	CRC
•	•	•	•
•	•	•	•
•	•	•	•
Frame (EBR Address) (Optional) See Note <sup>6</sup>	11110110	EBR Address Command.	Start: Include if <sup>5</sup>
	0	CRC Comparison Flag (0 = no).	Include if <sup>5</sup>
	000...0000	23-bit Command Information (23 zeros).	Include if <sup>5</sup>
	000000000000	11-bit dummy (11 zeros).	Include if <sup>5</sup>
	(10-bit EBR ID)	10-bit EBR ID. One = 0000000001.	Include if <sup>5</sup>
	(11-bit Address)	11-bit EBR Beginning address (default address 0).	Include if <sup>5</sup>

Frame	Contents (D0..D7)	Description	CRC
Frame (EBR Write) (Optional) See Note <sup>5</sup>	10110010	LSC_EBR_WRITE Command.	Include if <sup>5</sup>
	1	CRC Comparison Flag (1 = yes).	Include if <sup>5</sup>
	1	CRC Comparison at End Flag (1 = Execute CRC comparison at the end of all operands).	Include if <sup>5</sup>
	0	Exclude dummy bytes from bitstream.	Include if <sup>5</sup>
	1	Dummy definition (1 = use the next 4-bit Dummy Definition settings).	Include if <sup>5</sup>
	0000	Dummy definition.	Include if <sup>5</sup>
	(16-bit Size)	Number of 72-bit frames in the operand. One = 0000000000000001. For 1024*18/72 = 256 EBR, the size = 100000000.	Include if <sup>5</sup>
	(Data)	Data frame for one EBR. For 128 EBR, the data is treated in 2Kx9 mode. The data order is B0[8:0], B1[8:0], B2[8:0], ...	Include: End if <sup>5</sup>
	(CRC[15..0])	16-bit CRC (See Note <sup>4</sup> ).	CRC
Frame (Program Done)	01011110	ISC_PROGRAM_DONE Command.	Exclude
	0	CRC Comparison Flag (0 = no).	Exclude
	000...0000	23-bit Command Information (23 zeros).	Exclude
Frame (DUMMY)	1111...1111	4 bytes DUMMY command (all ones).	Exclude

1. Data is byte bounded and must be padded with ones (1). For example, a 6-bit data frame of b111010 is written in the bitstream as b11111010 to make it byte bounded.
2. Only the data frame bits, highlighted in yellow, can be compressed.
3. If SED option is not chosen, place command DUMMY (NOOP) (32 bits of 1) on the command space. NOOP command is not included in CRC calculation. This is necessary to make the file size consistent with or without security option chosen by users. It must be replaced with the LSC\_PROG\_SED\_CRC frame by Bitgen if the user implements SED. Include in CRC calculation if LSC\_PROG\_SED\_CRC command. Do not include in CRC calculation if NOOP.
4. The CRC must be flipped so that CRC[0] is the LSB and CRC[15] is the MSB. For example, CRC = 0x0823 (CRC[15]...CRC[0]) in the bit file above is b1100010000010000. Refer to Appendix Generic Configuration Specification for CRC calculation details.
5. Insert a Write EBR Frame for each initialized EBR, or for each group of EBR Frames that will be initialized to the same value. Omit if there is no EBR initialization data.
6. By default, this frame must be the Verify ID command. Bitgen shall have a switch to allow the Verify ID frame to be replaced with NOOP (all 1's). Include in CRC calculation if Verify ID command. Do not include in CRC calculation if NOOP.
7. If security option is not chosen, place command NOOP (32 bits of 1) on the command space. NOOP command is not included in CRC calculation. This is necessary to make the file size consistent with or without security option chosen by users. Include in CRC calculation if Program Security command. Do not include in CRC calculation if NOOP.
8. The Control Register 0 must be set to the SW Default value, setting Control Register 0 bits [31..30] = [0..0].

## Read Back Bitstream Format

If required, Diamond will generate a binary read back file. The read back file will contain an ASCII comment string, device ID, control register 0 data, frame data, dummy bits, LUT, EBR, and usercode data. It will not contain a header, commands, or CRC. The data will be ready to be shifted into the devices as required by the programming flow. The data is listed from MSB to LSB, where the MSB is the first bit shifted into TDI and the LSB the last bit. The MSB is also the first bit shifted out of TDO. The data for each frame is byte bounded. The byte must be padded with all ones (1). For example, a 6-bit data frame of b111010 would be written in the readback file as b11111010. The format is shown in Table 3.

**Table 8-5. ECP5 Read Back File Format**

<b>Frame</b>	<b>Contents (D0..D7)</b>	<b>Description</b>
Comments	(Comment String)	ASCII Comment (Argument) String
Comment Terminator	11111110	Read Back File Comment Terminator = 0xFE (binary file (.rbk) only)
Device ID	(ID[0..31])	Expected 32-bit Device ID (See Note <sup>1</sup> )
Control Register 0	(CtlReg0[0..31])	Control Register 0 Data
Device Specific Padding	1111...1111	Terminator bytes (all 1's). See Table 103 for number of bytes.
Frame 0	1111...1111	Dummy (Stop) bytes (all 1's). See Table 103 for number of bytes.
	(Frame 0 Data)	Data for Frame 0 (byte bounded, padded with zeros)
Frame 1	1111...1111	Dummy (Stop) bytes (all 1's). See Table 103 for number of bytes.
	(Frame 1 Data)	Data for Frame 1 (byte bounded, padded with zeros)
•	•	•
•	•	•
•	•	•
Frame n-1	1111...1111	Dummy (Stop) bytes (all 1's). See Table 103 for number of bytes.
	(Frame n-1 Data)	Data for Frame n-1 (byte bounded, padded with zeros)
Usercode	(U[0..31])	32-bit Usercode Data

## ECP5 Device Specific

**Table 8-6. ECP5 Device Specifics**

<b>Device Name</b>	<b>Configuration Frames (n)</b>	<b>Byte Bound Padding Bits</b>	<b>Data Bits per Frame (w)</b>	<b>Total Data Bits per Frame (x)</b>	<b>CRC Bits</b>
LFE5-25	7562	0	592	592	16
LFE5-45	9470	2	846	848	16
LFE5-85	13294	0	1136	1136	16

**Table 8-7. ECP5 Device ID**

<b>Device Name</b>	<b>32 bit IDCODE</b>
LFE5U -25	0x41111043
LFE5U-45	0x41112043
LFE5U-85	0x41113043
LFE5UM -25	0x01111043
LFE5UM-45	0x01112043
LFE5UM-85	0x01113043

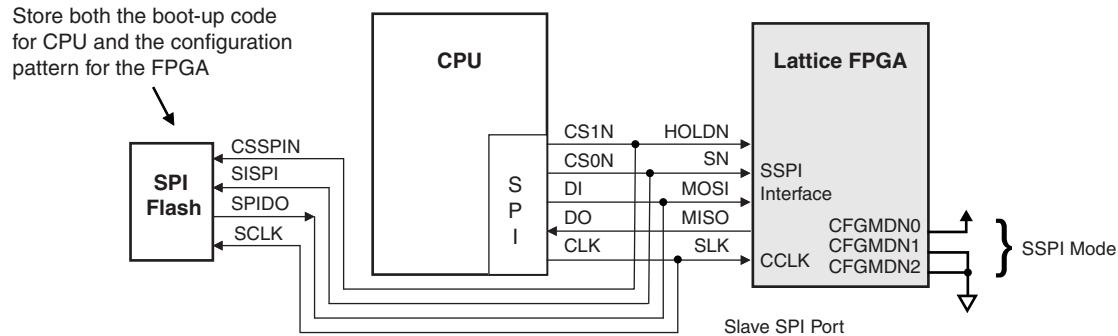
## Appendix C. Advanced Applications – The Slave SPI Port and SPI Flash Interface

The system diagram shown in Figure 9 illustrates an application of the Slave SPI interface. In this example, the CPU can independently access the external SPI flash for CPU boot-up code or for the FPGA bitstream. The FPGA selects Slave SPI as the primary boot source. When the CPU needs to configure the FPGA, it can drive CS0N low then shift the WRITE\_EN into the device command to set the FPGA device into the Slave SPI configuration mode.

In this example, the CPU reads the configuration data from the SPI Flash and shifts it into the FPGA. There are two methods this can be done.

1. The CPU reads the entire bitstream from the SPI Flash before shifting it into the FPGA. This method is much simpler to support, but requires sufficient RAM to store the entire bitstream.
2. The CPU can read a frame of the bitstream from the SPI Flash device, shift the frame into the FPGA, and repeat for each frame. The method requires less RAM, and it does not require the CPU to have knowledge of the bitstream format. However it does require use of the HOLDN pin.

**Figure 9. Example Slave SPI System Diagram (Slave SPI Mode)**

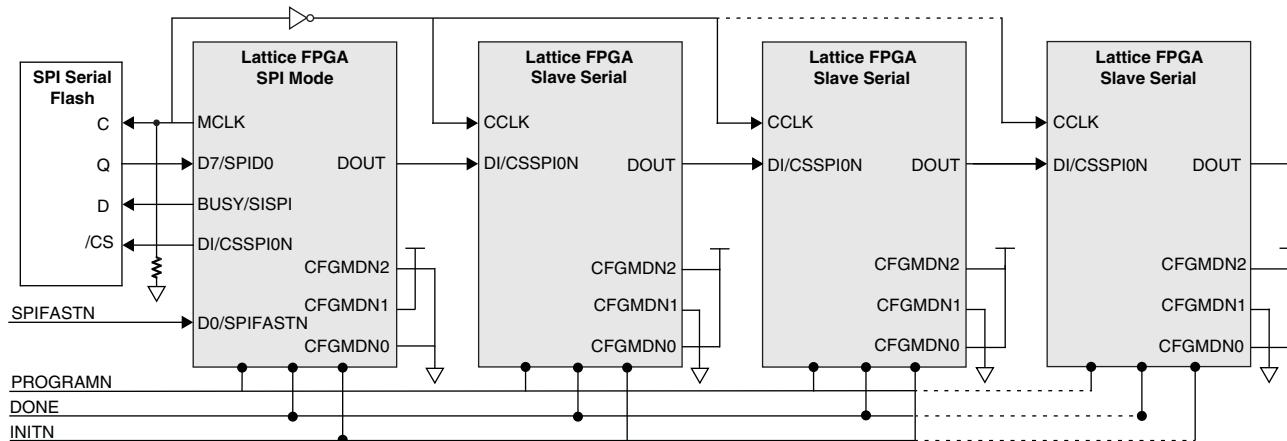


## Appendix D. Advanced Applications – Master SPI sysCONFIG Daisy Chaining

With a sufficiently large SPI Flash, multiple FPGAs can be configured as shown in Figure 8-1. The requirements are listed below.

- The first device must be a ECP5 device in Master SPI mode.
- The rest of the FPGA devices must be in Slave Serial Configuration (SCM) mode. The bitstream files for the daisy chain can be merged using the Deployment Tool.
- The DONE pin of all the FPGA devices must be connected together. This allows the devices to detect when the last device is done configuring.
- The ‘Synchronous to External DONE Pin’ option (DONE\_EX) must be enabled in the Lattice Diamond Spread-sheet view for all Lattice FPGA devices in the sysCONFIG chain.
- The Bypass option must be set by using the Diamond ‘Bitgen’ properties for all Lattice FPGA devices in the sysCONFIG chain.

**Figure 8-1. Master SPI sysCONFIG Daisy Chain**



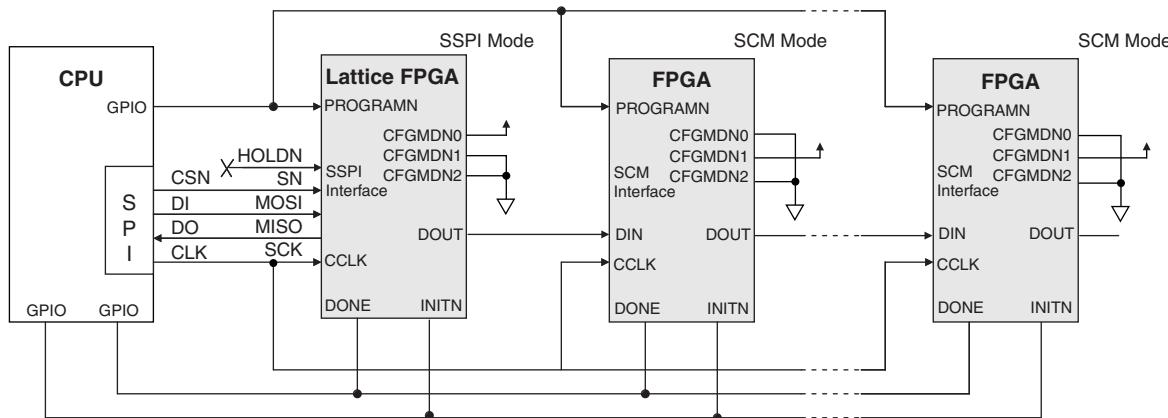
Note: The dashed lines indicate the connection is optional. The inverter for MCLK guarantees the hold time for data input to the daisy chained FPGAs.

## Appendix E. Advanced Applications – Slave SPI sysCONFIG Daisy Chaining

Even though the Slave SPI port does not explicitly support daisy chaining, a sysCONFIG chain of other FPGA devices can be configured using the SPI port of a CPU. A block diagram is shown in Figure 8-2. The requirements are listed below.

- The first device must be a ECP5 device in Slave SPI mode.
- The rest of the FPGA devices must be in Slave Serial Configuration (SCM) mode. The bitstream files for the daisy chain can be merged using the Deployment Tool.
- The DONE pin of all the FPGA devices must be connected together. This allows the devices to detect when the last device is done configuring.
- The ‘Synchronous to External DONE Pin’ option (DONE\_EX) must be enabled in the Lattice Diamond Spread-sheet view for all Lattice FPGA devices in the sysCONFIG chain.
- The Bypass option must be set by using the Diamond ‘Bitgen’ properties for all Lattice FPGA devices in the sysCONFIG chain.

**Figure 8-2. Slave SPI sysCONFIG Daisy Chain**



Note: The dashed lines indicate the connection is optional.

## Introduction

The ECP5™ family architecture is based on LatticeECP3 with increased granularity, low power and low cost. Up to four channels of embedded SERDES with associated Physical Coding Sublayer (PCS) logic are arranged by dual channel base.

Two channels of the LFE5UM are built into one unit called DCU (Dual Channel Unit). Each DCU contains one reference clock input and one TxPLL. This Dual-channel based architecture provides high clock to SERDES channel ratio than its predecessor FPGAs (ECP2M and ECP3).

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

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

## Features

- **Up to Four Channels of High-Speed SERDES with Increased Granularity**
  - 270 Mbps to 3.2 Gbps for Generic 8b10b, 10-bit SERDES and 8-bit SERDES modes. Refer to Table 9-1.
  - 270 Mbps to 3.2 Gbps per channel for all other protocols
  - 3.2 Gbps operation with low 85 mW power per channel
  - Receive equalization and transmit de-emphasis with both pre-cursor and post-cursor, for small form factor backplane operation
  - Supports PCI Express, Gigabit Ethernet (1GbE and SGMII) and XAUI, plus multiple other standards
  - Supports user-specified generic 8b10b mode
- **Multiple Clock Rate Support**
  - Separate reference clocks for each DCU 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 four channels of full-duplex data supported per device.
  - Multiple protocol support on one chip.
  - Supports popular 8b10b-based packet protocols.
  - SERDES Only mode allows direct 8-bit or 10-bit interface to FPGA logic.
- **Multiple Protocol Compliant Clock Tolerance Compensation (CTC) Logic**
  - Compensates for frequency differential between reference clock and received data rate.
  - Allows user-defined skip pattern of two or four bytes in length.
- **Integrated Loopback Modes for System Debugging**
  - Three loopback modes are provided for system debugging
- **Easy to Use Design Interface in Lattice Diamond™ Design Tool**
  - System Planner/Builder provides an easy to use GUI interface to assist users to instantiate the SERDES as a link, instead of individual SERDES channels
  - The tools allow the users to plan all resources to complete the design of a functional link, including SERDES/PCS, Lattice IPs or user design IPs, and all interface logic.

## New Features Over LatticeECP3™ SERDES/PCS

- LFE5UM implements the SERDES block in Dual-Channel architecture. This architecture provides more clock resources per channel base, which allows the users to have more flexibility on utilizing the channels.
- Bit-slip feature simplifies user logic on Word Alignment. This function is important for CPRI and JESD204A/B applications.
- Supports JESD204A/B - ADC and DAC converter I/F: 312.5 Mbps to 3.125 Gbps.
- H-Bridge output driver reduces power consumption.

## Using This Technical Note

The Lattice Diamond design tools support all modes of the PCS. Most modes are dedicated to applications for a specific industry standard data protocol. Other modes are more general purpose in nature and allow designers to define their own custom application settings. Diamond design tools allow the user to define the mode for each dual in their design. This technical note describes operation of the SERDES and PCS for all modes supported by Diamond.

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 DS1044, [ECP5 Family Data Sheet](#). Operation of the PCS logic is provided in the PCS section of this document. A table of all status and control registers associated with the SERDES and PCS logic which can be accessed via the SCI (SerDes Client Interface) Bus is provided in the appendices. Package pinout information is provided in the Pinout Information section of DS1044, [ECP5 Family Data Sheet](#).

## Standards Supported

The supported standards are listed in Table 9-1.

**Table 9-1. Standards Supported by the SERDES/PCS**

Standard	Data Rate (Mbps)	System Reference Clock (MHz)	FPGA Clock (MHz)	Number of Link Width	Encoding Style
PCI Express 1.1	2500	100	250	x1, x2, x4	8b10b
Gigabit Ethernet SGMII	1250	125	125	x1	8b10b
	2500	125	250	x1	8b10b
	3125	156.25	156.25	x1	8b10b
	3125	156.25	156.25	x4	8b10b
XAUI	3125	156.25	156.25	x4	8b10b
Serial RapidIO (SR/LR)	1250	125	125	x1, x4	8b10b
	2500	125	250		
	3125	156.25	156.25		
CPRI-E.6.LV	614.4	61.44	61.44	x1	8b10b
E.12.LV	1228.8	61.44, 122.88	122.88		
E.24.LV	2457.6	122.88	122.88		
E.30.LV	3072	153.6	153.6		
SD-SDI (259M, 344M)	270, 360, 540	27, 36, 54	27	x1	NRZI/Scrambled
HD-SDI (292M)	1483.5	74.175, 148.35	74.175, 148.35,	x1	NRZI/Scrambled
	1485	74.25, 148.50	74.25, 148.5		

Standard	Data Rate (Mbps)	System Reference Clock (MHz)	FPGA Clock (MHz)	Number of Link Width	Encoding Style
3G-SDI (424M)	2967	148.35	148.35	x1	NRZI/Scrambled
	2970	148.5	148.5		
JESD204A/B	312.5 - 3125	31.25 - 156.25	31.25 - 156.25		
10-Bit SERDES	270 - 3125	27 - 320	27 - 160	x1, x2, x3, x4	N/A
8-Bit SERDES	270 - 3125	27 - 320	27 - 160	x1, x2, x3, x4	N/A
Generic 8b10b	270 - 3125	27 - 320	27 - 160	x1, x2, x3, x4	8b10b

## Architecture Overview

The SERDES/PCS block is arranged in duals containing logic for two full-duplex data channels.

Figure 9-1 shows the arrangement of SERDES/PCS duals on the LFE5UM-85.

**Figure 9-1. LFE5UM-85 Block Diagram**

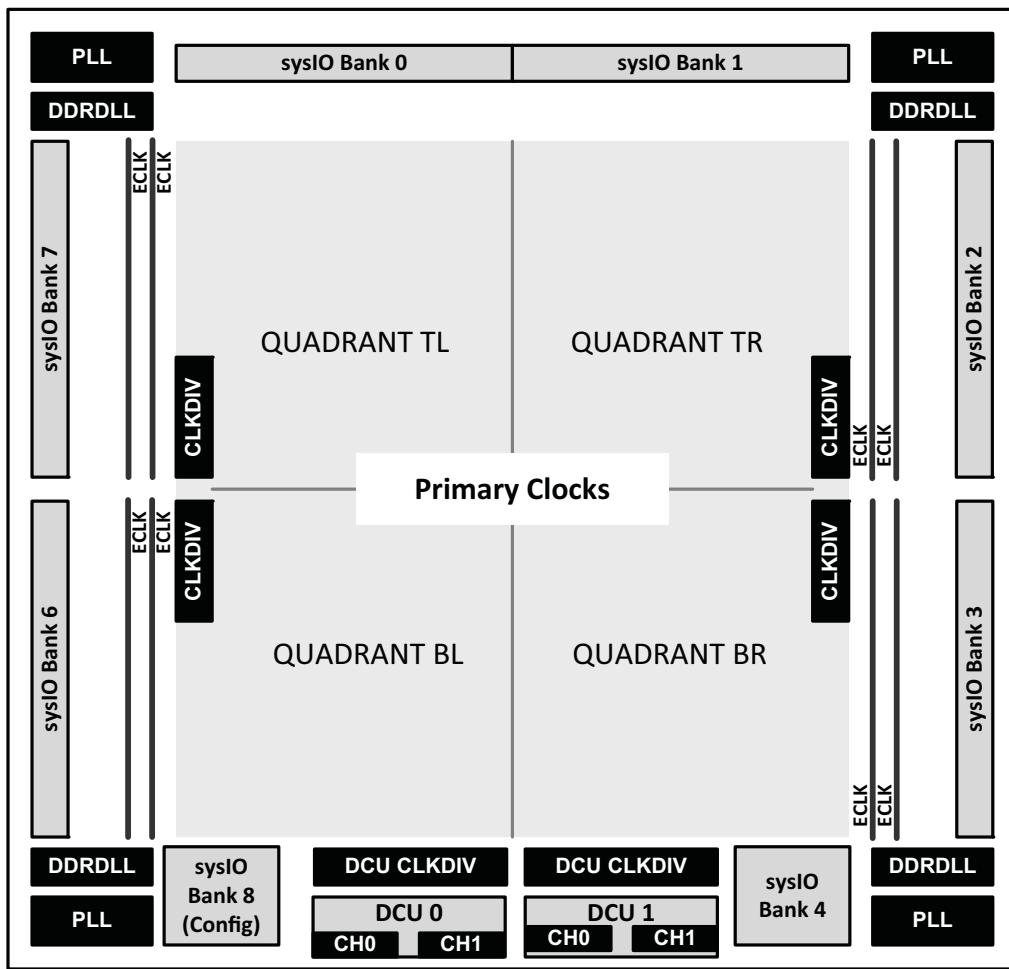


Table 9-2 shows the number of available SERDES/PCS channels for each LFE5UM device.

**Table 9-2. Number of SERDES/PCS Channels Per LFE5UM Device**

Package	LFE5UM-25	LFE5UM-45	LFE5UM-85
SerDes/PCS DCUs	1	2	2
SerDes/PCS Channels	2	4	4

LFE5UM SerDes block is defined by a dual base architecture (DCU). Each DCU includes two SerDes/PCS full-duplex Rx/Tx channels, one common AUX channel between the two SerDes channels that consists of one TX PLL and the associated EXTREF block.

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

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

Since each dual has its own reference clock, different duals can support different standards on the same chip. This feature makes the LFE5UM device family ideal for bridging between different standards

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

Figure 9-2 is a simplified block diagram of two DCUs.

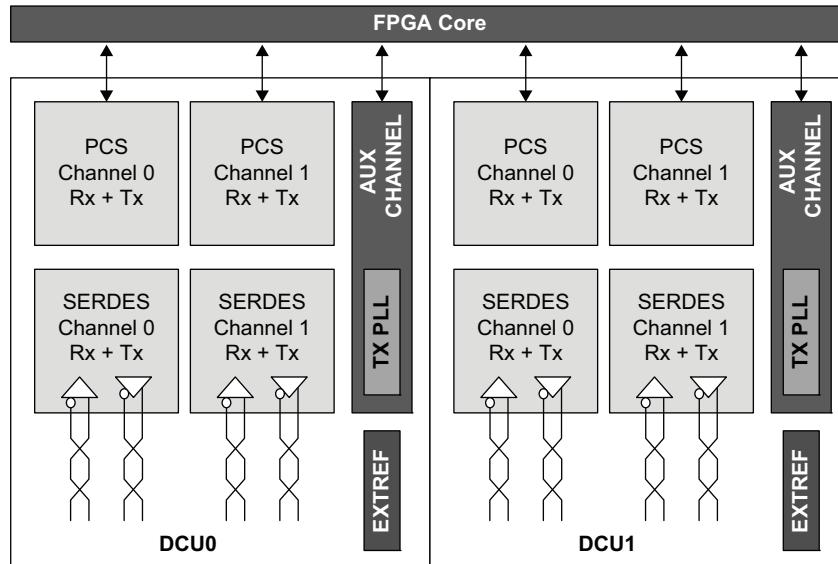
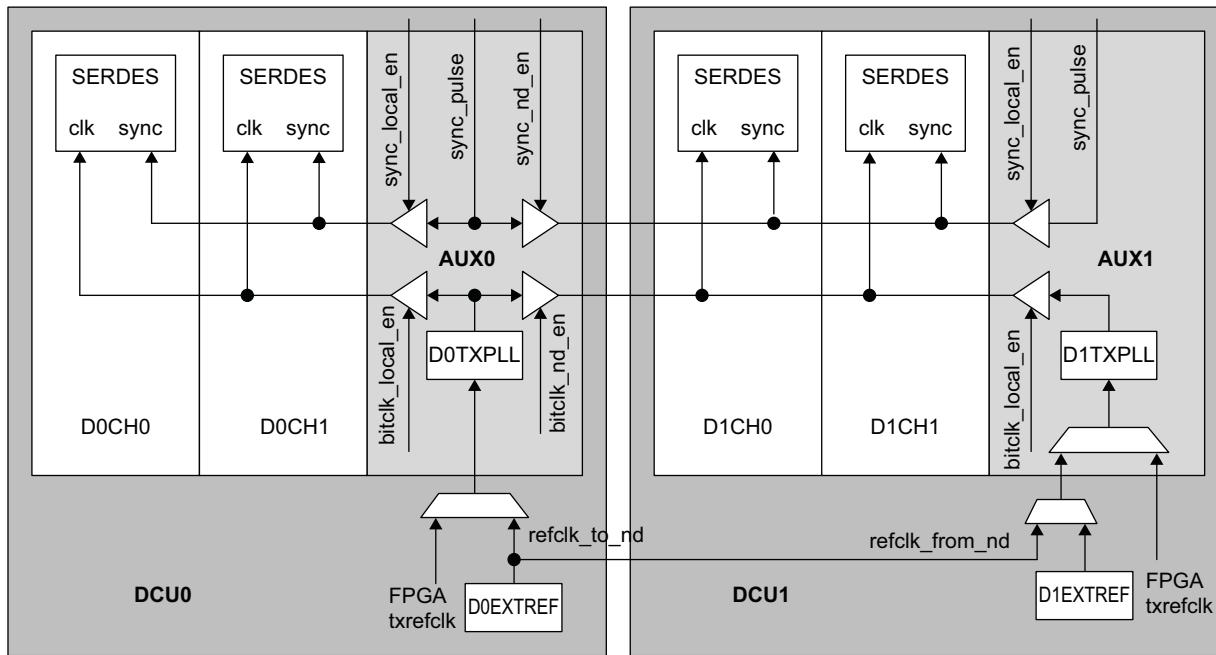
**Figure 9-2. Simplified Block Diagram of Two DCUs in LFE5UM-45 and LFE5UM-85**


Figure 9-3 shows hardware arrangement of two dual and the sharing of clocking resources.

**Figure 9-3. Two DCUs Pair with Clock Sharing**



When implementing an X4 link (such as XAUIC X4, or PCIe X4), one DCU Aux channel resource can be shared between two adjacent DCUs with the clock structure shown in Figure 9-3.

### Multi-Protocol Design Consideration

The dual-channel based DCU architecture in LFE5UM serves the purpose of providing more flexibility to the user in implementing different protocols. This is achieved by sharing resources across different DCUs when a wide interface is needed across two DCUs. Also, the LFE5UM DCUs allow sharing the same DCU with different protocols, provided the protocols have data-rates that are multiplicity of each other.

Different combinations of protocols within a DCU are permitted subject to certain conditions. One of the most basic requirements for two protocols sharing the same DCU is that the two protocols must have data-rate that are either the same, or can be divided by the rate divider (Div1, Div2, Div11). This restriction is due to these two protocols share the same clock from the TxPLL.

A transmit clock derived from the TxPLL in DCU0 can be extended to drive the complementary neighboring DCU1 channels, providing the capability of the TxPLL of DCU0 to be used by four channels. This allows implementation of X4 link to be easily achievable.

Table 9-3 lists the support of mixed protocols within a DCU.

**Table 9-3. LFE5UM Mixed Protocol Pair**

Protocol	&	Protocol
SRIO	&	GbE
PCIe 1.1	&	GbE
PCIe 1.1	&	SRIO
SD/HD/3G SDI	&	SD/HD/3G SDI
PCIe 1.1	&	PCIe 1.1
JESD204A/B	&	JESD204A/B
SRIO 1.25G/2.5G	&	SRIO 1.25G/2.5G

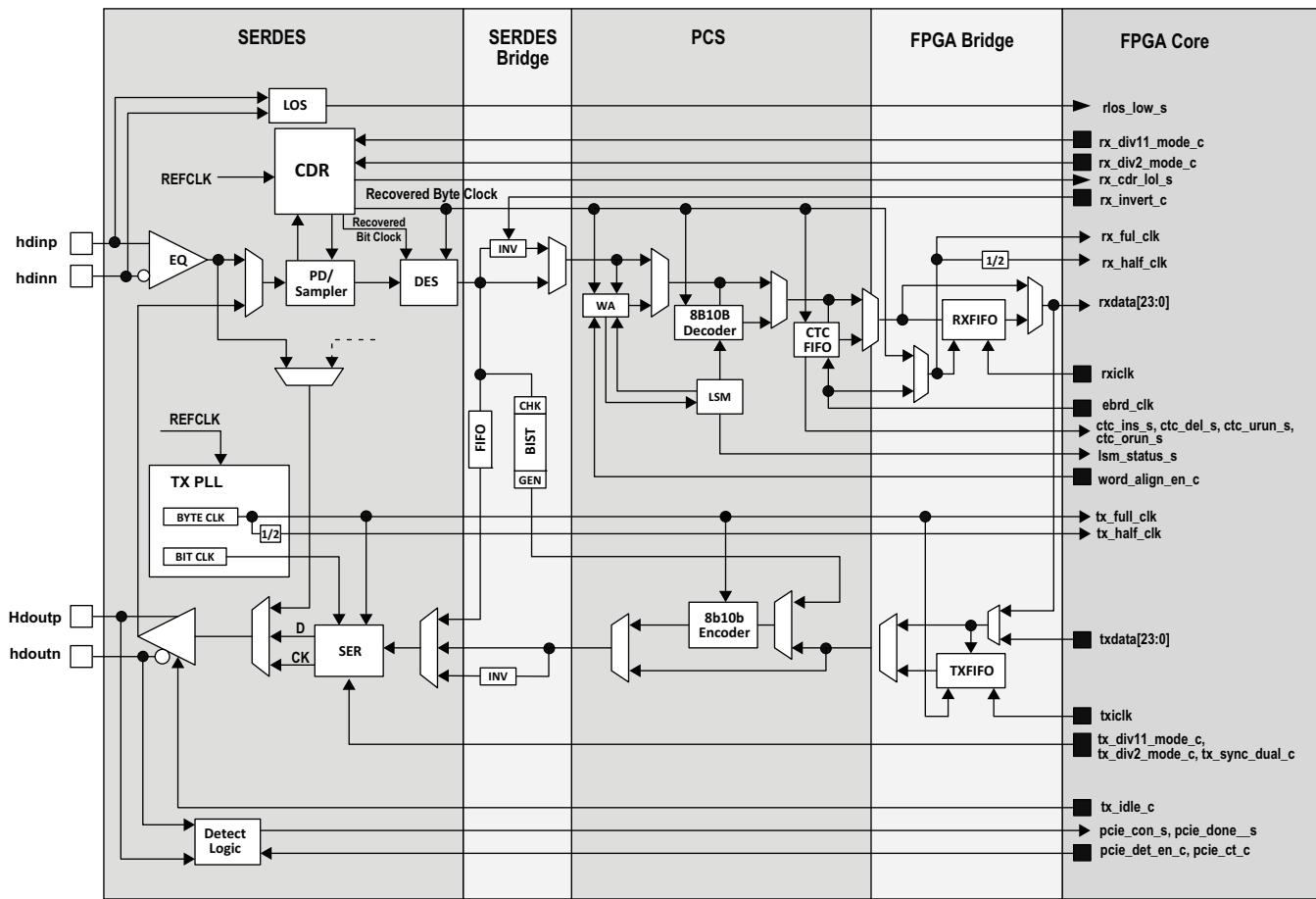
Protocol		Protocol
CPRI E6/E12	&	CPRI E6/E12
CPRI E12/E24	&	CPRI E12/E24

PCIe must be without spread spectrum to share with other protocols in the same DCU.

## Detailed Channel Block Diagram

Figure 9-4 is a detailed block diagram representation of the major functionality in a single channel of the LFE5UM SERDES/PCS. This diagram shows all the major blocks and the majority of the control and status signals that are visible to the user logic in the FPGA. This diagram also shows the major sub-blocks in the channel – SERDES, SERDES Bridge, PCS Core and the FPGA Bridge.

**Figure 9-4. LFE5UM SERDES/PCS Detailed Channel Block Diagram**



## Clocks and Resets

A SERDES/PCS dual supplies per-channel locked reference clocks and per-channel recovered receive clocks to the FPGA logic interface. Each dual provides clocks on both primary and secondary FPGA clock routing. The PCS/FPGA interface also has ports for the transmit clock (one per dual) and receive clocks (one per channel) supplied from the FPGA fabric.

Each dual has reset inputs to force reset of both or individual on the SERDES and PCS logic in a dual. In addition, separate resets are provided to reset the clock generation of the Rx and Tx channels. When the clock generation is reset (either in Tx or Rx, or both), the corresponding SERDES logic needs to be reset in order for the SERDES logic and PCS logic to be synchronized.

## Transmit Data Bus

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

## Receive Data Bus

The signals for the receive path are from the FPGA Bridge in the PCS Block to the FPGA. The data path may be geared 2:1 to the internal PCS data path which is 8 bits wide. The data bus width at the FPGA interface is 16 bits wide. When the data is geared 2:1, the lower bits (rxdata[9:0] or rxdata[7:0]) correspond to the word that has been received first and the higher bits (rxdata[19:10] or rxdata[15:8]) correspond to the word that has been received second. Table 9-4 describes the use of the data bus for each protocol mode.

The 2:1 gearing in LFE5UM can be configured as user-controlled thru the FPGA logic. This allows the user to change the gearing from the core, when different rates in the protocol are selected (for example: HD-SDI and 3G-SDI)

Table 9-4 shows how the signals are assigned in different protocols (channel0 is shown as an example).

**Table 9-4. Data Bus Usage by Mode**

Data Bus PCS Cell Name4	G8B10 B	CPRI	PCI Express	SRIO	Gigabit Ethernet	XAUI	8-Bit SERDES	10-Bit SERDES	SDI
ff_tx_d_0_0					txdata_ch0[0]				
ff_tx_d_0_1					txdata_ch0[1]				
ff_tx_d_0_2					txdata_ch0[2]				
ff_tx_d_0_3					txdata_ch0[3]				
ff_tx_d_0_4					txdata_ch0[4]				
ff_tx_d_0_5					txdata_ch0[5]				
ff_tx_d_0_6					txdata_ch0[6]				
ff_tx_d_0_7					txdata_ch0[7]				
ff_tx_d_0_8					tx_k_ch0[0]	txc_ch0[0]	GND	txdata_ch0[8]	
ff_tx_d_0_9					tx_force_disp_ch0[0]1	GND			txdata_ch0[9]
ff_tx_d_0_10					tx_disp_sel_ch0[0]1	GND	xmit_ch0[0]2	GND	
ff_tx_d_0_11					GND	pci_ei_en_ch0[0]	tx_disp_correct_ch0[0]	GND	
ff_tx_d_0_12					txdata_ch0[8]				txdata_ch0[10]
ff_tx_d_0_13					txdata_ch0[9]				txdata_ch0[11]
ff_tx_d_0_14					txdata_ch0[10]				txdata_ch0[12]
ff_tx_d_0_15					txdata_ch0[11]				txdata_ch0[13]
ff_tx_d_0_16					txdata_ch0[12]				txdata_ch0[14]
ff_tx_d_0_17					txdata_ch0[13]				txdata_ch0[15]
ff_tx_d_0_18					txdata_ch0[14]				txdata_ch0[16]
ff_tx_d_0_19					txdata_ch0[15]				txdata_ch0[17]
ff_tx_d_0_20					tx_k_ch0[1]	txc_ch0[1]	GND	txdata_ch0[18]	
ff_tx_d_0_21					tx_force_disp_ch0[1]1	GND			txdata_ch0[19]
ff_tx_d_0_22					tx_disp_sel_ch0[1]1	GND	xmit_ch0[1]2	GND	
ff_tx_d_0_23	GND	pci_ei_en_c_h0[1]	GND		tx_disp_correct_t_ch0[1]	GND			

Data Bus PCS Cell Name4	G8B10 B	CPRI	PCI Express	SRIO	Gigabit Ethernet	XAUI	8-Bit SERDE S	10-Bit SERDE S	SDI
ff_rx_d_0_0					rxdata_ch0[0]				
ff_rx_d_0_1					rxdata_ch0[1]				
ff_rx_d_0_2					rxdata_ch0[2]				
ff_rx_d_0_3					rxdata_ch0[3]				
ff_rx_d_0_4					rxdata_ch0[4]				
ff_rx_d_0_5					rxdata_ch0[5]				
ff_rx_d_0_6					rxdata_ch0[6]				
ff_rx_d_0_7					rxdata_ch0[7]				
ff_rx_d_0_8					rx_k_ch0[0]		rx_c_ch0[0]	NC	rxdata_ch0[8]
ff_rx_d_0_9	rx_disp_err_ch0[0]	rxstat_us0_ch0[0]	rx_disp_err_ch0[0]	NC	rxdata_ch0[9]				
ff_rx_d_0_10	rx_cv_err_ch0[0]3	rxstat_us0_ch0[1]	rx_cv_err_ch0[0]3	NC					
ff_rx_d_0_11	NC	rxstat_us0_ch0[2]	NC						
ff_rx_d_0_12	rxdata_ch0[8]	rxdata_ch0[10]							
ff_rx_d_0_13	rxdata_ch0[9]	rxdata_ch0[11]							
ff_rx_d_0_14	rxdata_ch0[10]	rxdata_ch0[12]							
ff_rx_d_0_15	rxdata_ch0[11]	rxdata_ch0[13]							
ff_rx_d_0_16	rxdata_ch0[12]	rxdata_ch0[14]							
ff_rx_d_0_17	rxdata_ch0[13]	rxdata_ch0[15]							
ff_rx_d_0_18	rxdata_ch0[14]	rxdata_ch0[16]							
ff_rx_d_0_19	rxdata_ch0[15]	rxdata_ch0[17]							
ff_rx_d_0_20	rx_k_ch0[1]	rx_c_ch0[1]	NC	rxdata_ch0[18]					
ff_rx_d_0_21	rx_disp_err_ch0[1]	rxstat_us1_ch0[0]	rx_disp_err_ch0[1]	NC	rxdata_ch0[19]				

Data Bus PCS Cell Name4	G8B10 B	CPRI	PCI Express	SRIO	Gigabit Ethernet	XAUI	8-Bit SERDE S	10-Bit SERDE S	SDI
ff_rx_d_0_22	rx_cv_err_ch0[1:3]	rxstat us1_c h0[1]	rx_cv_err_ch0[1:3]	NC					
ff_rx_d_0_23	NC	rxstat us1_c h0[2]	NC						

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

## Control and Status Signals

Table 9-5 describes the control and status signals.

**Table 9-5. Control and Status Signals and their Functions**

Signal Name	Description
<b>Transmit Control Signals</b>	
tx_k_ch[1:0]	Per channel, active-high control character indicator.
tx_force_disp_ch[1:0]	Per channel, active-high signal which instructs the PCS to accept disparity value from disp_sel_ch[1:0] input.
tx_disp_sel_ch[1:0]	Per channel, disparity value supplied from FPGA logic. Valid when force_disp_ch[1:0] is HIGH.
tx_correct_disp_ch[1:0]	Corrects disparity identifier when asserted by adjusting the 8B10B encoder to begin in the negative disparity state.
<b>Receive Status Signals</b>	
rx_k_ch[1:0]	Per channel, active-high control character indicator.
rx_disp_err_ch[1:0]	Per channel, active-high signal driven by the PCS to indicate a disparity error was detected with the associated data.
rx_cv_err_ch[1:0]	Per channel, code violation signal to indicate an error was detected with the associated data.

### Control Signals

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

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

### Status Signals

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

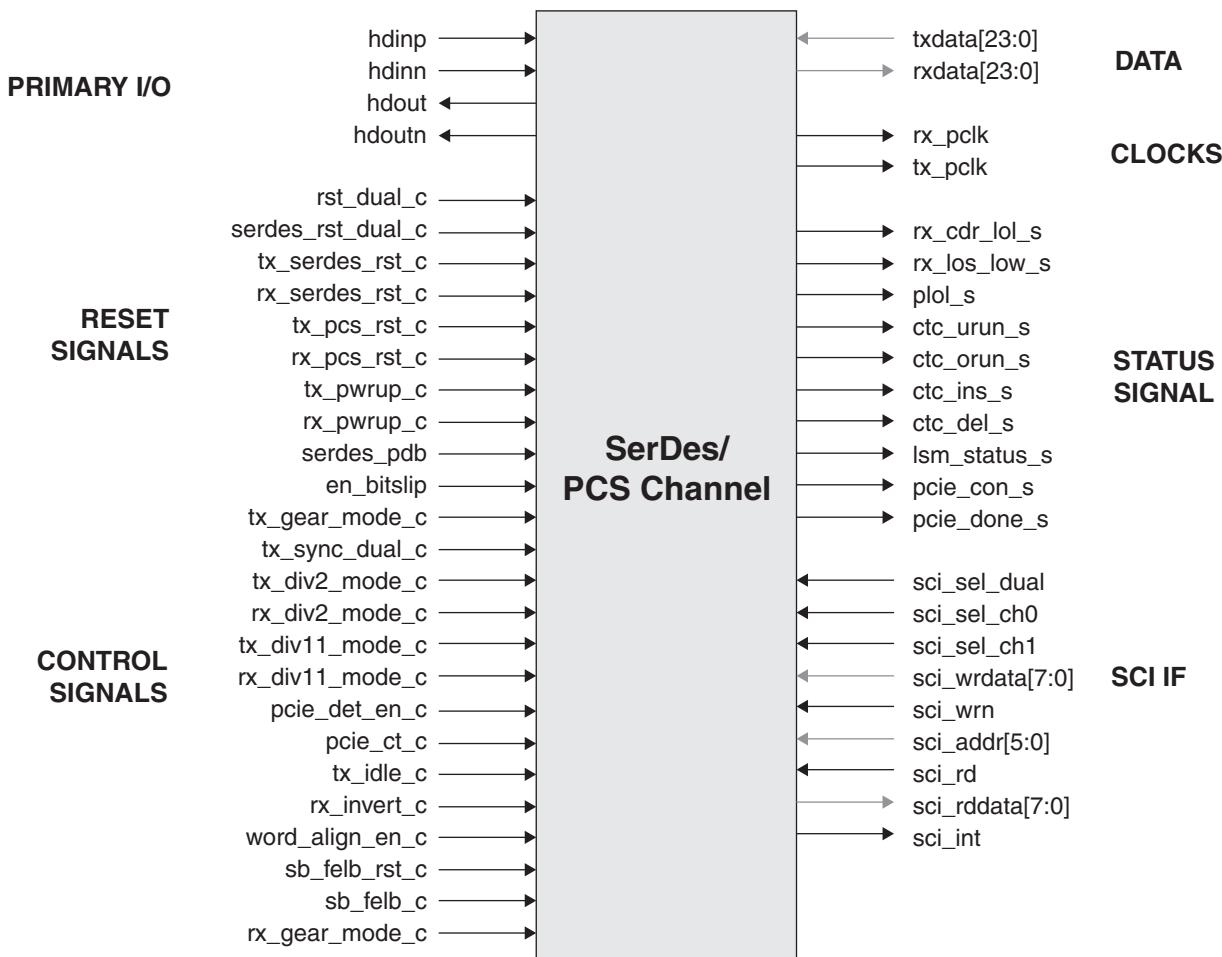
{signal}\_s is the status signal to the FPGA core from the FPGA bridge. All of the status signals are asynchronous from the SERDES/PCS. These should be synchronized to a clock domain before they are used in the FPGA design.

Please refer to the Mode-Specific Control/Status Signals section for detailed information about these signals.

## SERDES/PCS

The DCU contains two 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 dual SERDES/PCS macro performs the serialization and de-serialization function for two lanes of data. In addition, the TxPLL within the DCU provides the system clock for the FPGA logic. The dual 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 9-5.

**Figure 9-5. SERDES/PCS Block Signal Interface**



## I/O Descriptions

Table 9-6 lists all default and optional inputs and outputs to/from a dual.

**Table 9-6. SERDES/PCS I/O Descriptions**

Signal Name	I/O	Type	Description
<b>Primary I/O, SERDES Dual</b>			
hdinp	I	CH	High-speed input, positive
hdinn	I	CH	High-speed input, negative
hdoutp	O	CH	High-speed output, positive
hdoutn	O	CH	High-speed output, negative
<b>Transmit/Receive Data Bus (See Table 9-3 and Table 9-4 for detailed data bus usage)</b>			

Signal Name	I/O	Type	Description
rxdata[23:0]	O	CH	received data
txdata[23:0]	I	CH	transmit data
<b>Control Signals</b>			
tx_sync_dual_c	I	DL	Serializer Reset Rising edge Transition = Reset Level = Normal Operation
tx_div2_mode_c	I	CH	Transmit Rate Mode (Full Rate/Half Rate) Select 1 = Half Rate 0 = Full Rate
rx_div2_mode_c	I	CH	Receive Rate Mode (Full Rate/Half Rate) Select 1 = Half Rate 0 = Full Rate
tx_div11_mode_c	I	CH	Transmitter Rate Mode Select (SDI protocol only) 1: Div11 Rate 0: Full Rate
rx_div11_mode_c	I	CH	Receiver Rate Mode Select (SDI protocol only) 1: Div11 Rate 0: Full Rate
pcie_det_en_c	I	CH	FPGA logic (user logic) informs the SerDes block that it will be requesting for a PCI Express Receiver Detect operation 1 = Enable PCI Express Receiver Detect
pcie_ct_c	I	CH	1 = Request transmitter to do far-end receiver detection 0 = Normal data operation
tx_idle_c	I	CH	0 = Normal operation
rx_invert_c	I	CH	Control the inversion of received data. 1 = Invert the data 0 = Do not invert the data
tx_gear_mode_c	I	CH	Transmit Gearing Mode Select (OR'ed with tx_data_width setting. Use with tx_data_width = 0) 1 = 16/20-Bit Mode 0 = 8/10-Bit Mode
rx_gear_mode_c	I	CH	Receiver Gearing Mode Select (OR'ed with rx_data_width setting. Use with rx_data_width = 0) 1 = 16/20-Bit Mode 0 = 8/10-Bit Mode
word_align_en_c	I	CH	Control comma aligner. 1 = Enable comma aligner 0 = Lock comma aligner at current position
sb_felb_c	I	CH	SerDes Bridge Parallel Loopback 1 = Enable loopback from RX to TX 0 = Normal data operation
sb_felb_RST_c	I	CH	SerDes Bridge Parallel Loopback FIFO Clear 1 = Reset Loopback FIFO 0 = Normal loopback operation
en_bitslip	I	CH	SerDes Word Alignment to shift byte clock by 1 UI 1 = Slip Rx byte clock by 1 UI for Word Alignment 0 = No slip
<b>Status Signals</b>			
rx_cdr_lol_s	O	CH	1 = Receive CDR Loss of Lock 0 = Lock maintained
rx_los_low_s	O	CH	Loss of Signal (LO THRESHOLD RANGE) detection for each channel.

Signal Name	I/O	Type	Description
ctc_urun_s	O	CH	1 = Receive clock compensator FIFO underrun error 0 = No FIFO errors.
ctc_orun_s	O	CH	1 = Receive clock compensator FIFO overrun error 0 = No FIFO errors.
ctc_ins_s	O	CH	1 = SKIP Character Added by CTC
ctc_del_s	O	CH	1 = SKIP Character Deleted by CTC
lsm_status_s	O	CH	1 = Lane is synchronized to commas 0 = Lane has not found comma
pcie_con_s	O	CH	Result of far-end receiver detection 1 = Far end receiver detected 0 = Far end receiver not detected
pcie_done_s	O	CH	1 = Far-end receiver detection complete 0 = Far-end receiver detection incomplete
rst_dual_c	I	DL	Active high, asynchronous input. Resets all SerDes channels including the auxiliary channel and PCS.
serdes_rst_dual_c	I	DL	Active high, asynchronous input to SerDes dual. Gated with software register bit fpga_reset_enable. By default fpga_reset_enable is '1'.
tx_serdes_rst_c	I	DL	Resets LOL signal in PLL
rx_serdes_rst_c	I	CH	Resets selected digital logic in the SerDes Receive channels
tx_pcs_rst_c	I	CH	Active high, asynchronous input. Resets individual Dual TX channel logic only in PCS.
rx_pcs_rst_c	I	CH	Active high, asynchronous input. Resets individual Dual RX channel logic only in PCS.
serdes_pdb	I	DL	Power down control for entire SerDes dual including AUX and channels. 0 = Power down 1 = Power up
tx_pwrup_c	I	CH	Active low Transmit Channel Power Down. 0 = Transmit Channel should be powered down
rx_pwrup_c	I	CH	Active low Receive Channel Power Down. 0 = Receive Channel should be powered down
<b>Clocks</b>			
rx_pcclk	O	CH	Receive Channel Recovered Primary Clock. Direct connection to Primary clock network. When gearing is not enabled, this output equals the receive full rate clock. When 2:1 gearing is enabled, it is a divide-by-2 half-rate clock.
tx_pcclk	O	CH	Transmit Primary Clock. Direct connection to Primary clock network. When gearing is not enabled, this output equals the transmit full rate clock. When 2:1 gearing is enabled, it is a divide-by-2 half-rate clock.
<b>SCI Interface Signals</b>			
sci_sel_dual	I	DL	sci dual select
sci_sel_ch0	I	CH	sci channel0 select
sci_sel_ch1	I	CH	sci channel1 select
sci_wrdata[7:0]	I	CH	sci write data

Signal Name	I/O	Type	Description
sci_rddata[7:0]	O	CH	sci read data bus
sci_wrn	I	CH	sci write strobe
sci_rd	I	CH	sci read data select
sci_addr[5:0]	I	CH	sci address bus
sci_int	O	CH	sci interrupt output

## SERDES/PCS Functional Description

LFE5UM devices have one to two duals of embedded SERDES/PCS logic. Each dual, in turn, supports two independent full-duplex data channels. A single channel can support a data link and each dual can support up to two such channels.

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

Refer to Figure 9-4 for each of the items below.

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

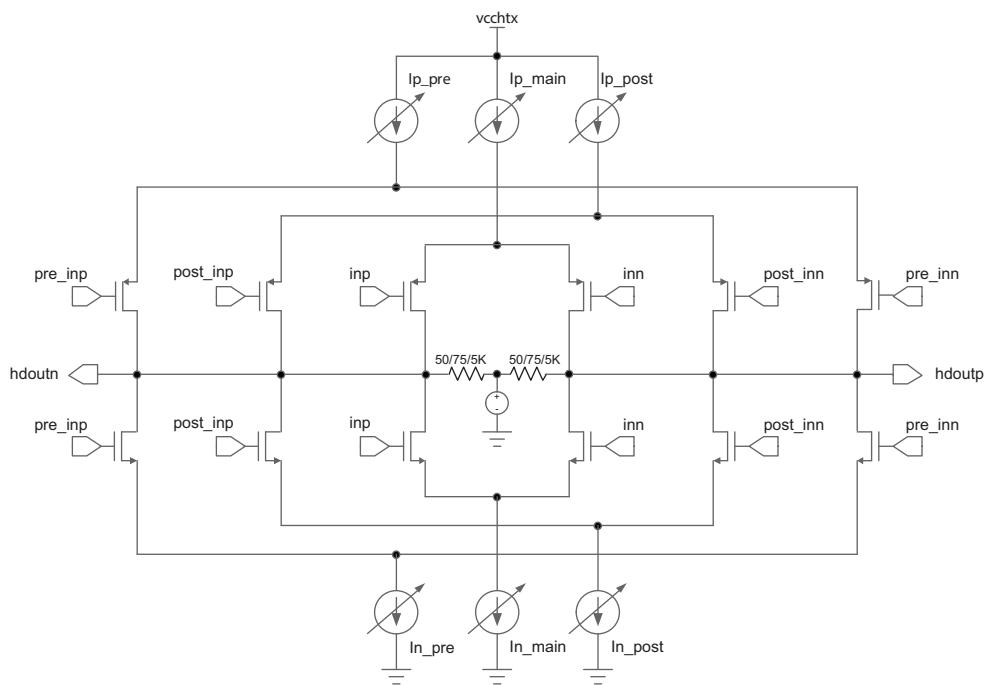
## SERDES Buffers

### Tx Differential Output Driver

The TX output buffer is a high speed line driver, and is used to send serialized data off-chip. It includes functions such as variable amplitude settings, variable termination resistance values, and various pre/post-cursor De-emphasis settings. TX output buffer has a H-Bridge structure as shown in Figure 9-6. This provides lower power consumption to reduce overall TX power. The output voltage swing is maintained by having separate power supply pad for driver only ( $V_{CCHTX} = 1.2$  V). PCI Express requirements for electrical idle and receiver detection is implemented in TX driver and output pads.

Figure 9-6 shows the Transmitter H-Bridge Driver with the differential termination, pre- and post- cursor and associated current sources.

**Figure 9-6. Transmitter H-Bridge Driver with Termination and De-emphasis**



# SERDES

## Linear Equalizer

The Linear Equalizer (LEQ) is used to equalize the channel loss. Generally this is used to selectively amplify the high frequency components more which are attenuated more over a long backplane— similar to the function performed by De-emphasis on the transmitter. As the data rate of the transmission advances over multi-Gpbs, frequency-dependent attenuation can cause severe InterSymbol Interference (ISI) in the receive signal. LEQ performs the recovery of the signal from the interference. Four levels of Equalization can be selected, based on the user transmission channel conditions.

## **De-Emphasis**

De-emphasis refers to a system process designed to increase the magnitude of some frequencies with respect to the magnitude of other frequencies. De-emphasis improves the overall signal-to-noise ratio by minimizing the adverse effects of such phenomena as attenuation differences.

LFE5UM SERDES Transmit channel provides +/- one tap for de-emphasis (pre-cursor and post cursor). Both taps can be programmed individually, to provide 12 settings in each.

## Reference Clocks

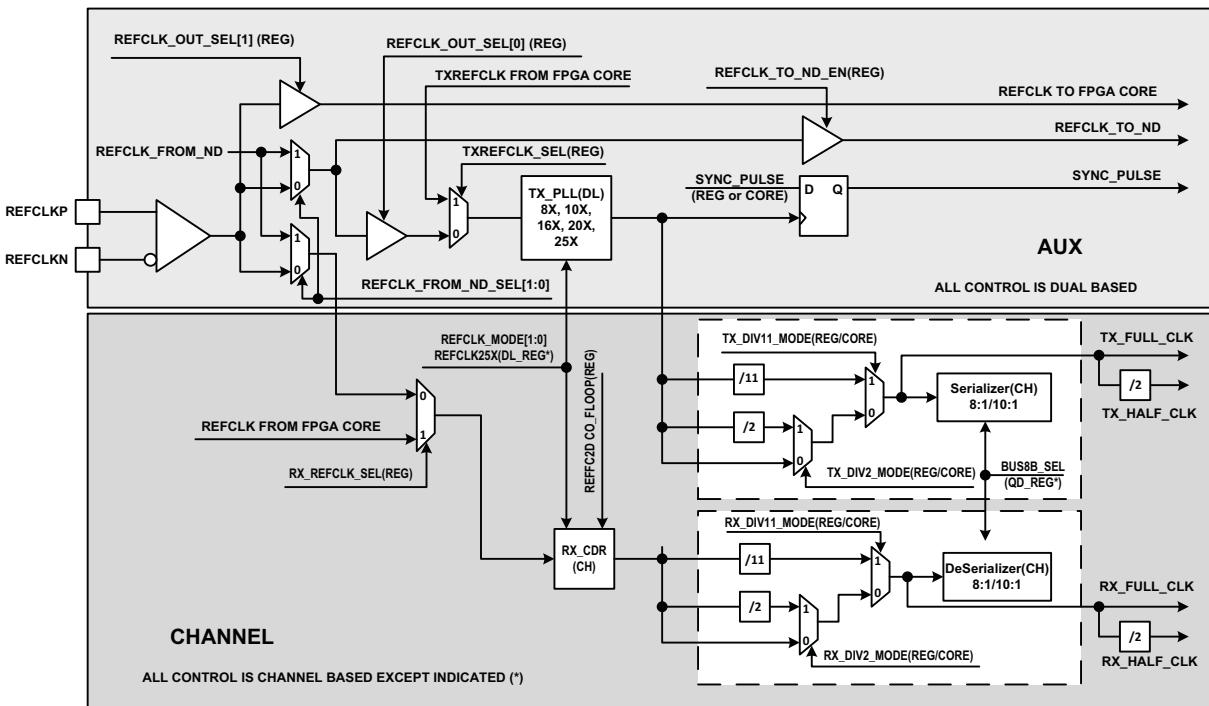
The SERDES dual contains two channels with both RX and TX circuits, and an auxiliary channel that contains the TX PLL. The reference clock to the TX PLL can be provided either by the primary differential reference clock pins, by the neighbor dual's reference clock or by the FPGA core. In addition, the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

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

## SERDES Clock Architecture

Figure 9-7 shows the clocking structure inside a DCU. There is one Aux channel in each DCU. There are two Tx/Rx channels. Various control bits for the different blocks are shown. These could be dual-based control register bits or channel-based control register bits. In some cases, it can be channel control port based. Some are a combination of both register and control ports. Using both modes enables dynamic control of certain functional properties.

**Figure 9-7. SERDES Clock Architecture**



The important components of the LFE5UM SERDES clocking architecture include:

- Per Rx and Tx diver (DIV) modes – DIV1, DIV2, DIV11. Suitable for dynamic rate switching.
- REFCLK sharing between DCUs – REFCLK input from DCU0 can be shared by DCU1. Used in x4 link application.
- Multi-channel transmit synchronization using the tx\_sync\_dual\_c signal from the FPGA

#### Rate Modes

The TX in each channel can be independently programmed to run at either FULL\_RATE, HALF\_RATE (DIV2), or DIV11 mode.

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

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

It should be noted when Rx DIV selection is changed dynamically from the core, the Rx of the channels should be reset. The DIV is inside the CDR lock loop, and change of the value requires the loop to be relocked again.

The TX PLL and the two CDR PLLs generally run at the same frequency, which is a multiple of the reference clock frequency. Table 9-7 illustrates the various clock rate modes possible. The bit clock indicated is a multiple of the reference clock frequency.

**Table 9-7. TXPLL and RX CDRPLL Supported Modes**

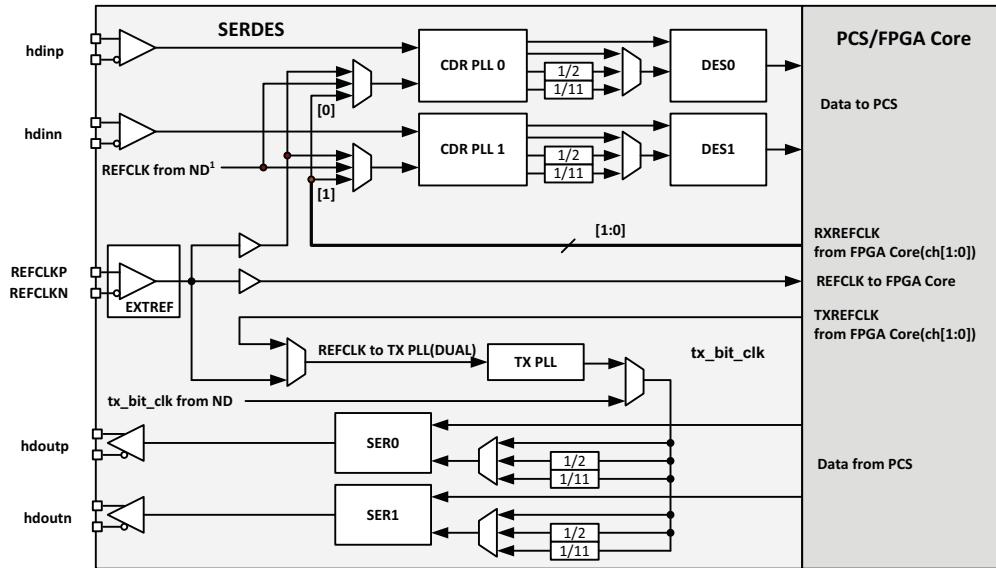
Reference Clock Mode	refclk_mode(Dual)	Bus_Width	Bit Clock(Full)	Bit Clock (div2, div11)
20x	0	10	Refclk x 20	Refclk x 10
16x	0	8	Refclk x 16	Refclk x 8
10x	1	10	Refclk x 10	Refclk x 5
8x	1	8	Refclk x 8	Refclk x 4
25x	-	8	Refclk x 25	Refclk x 12.5
25x	-	10	Refclk x 25	Refclk x 12.5
20x	0	10	Refclk x 20	Refclk x 20/11 <sup>1</sup>

1. DIV11 mode.

### Reference Clock from FPGA Core

As shown in Figure 9-8, the TX reference clock can be brought from the FPGA core. In this case, the extra jitter caused by the FPGA resources to route the clock to the SERDES will be passed onto the transmit data and may violate TX jitter specifications on a case-by-case basis. Caution should be used when using an FPGA-originated SERDES TX reference clock.

The Rx reference clock can also be brought from the FPGA core. Each Rx channel can have independent RxRefClk signal. The Rx reference clock is used in the channel's CDR to acquire initial frequency lock, before switches to lock to the data. High jitter on RxRefClk when is brought in from FPGA can cause difficulty for Rx to obtain initial frequency lock. It can also cause the Rx channel RLOL (Rx Loss Of Lock) signal to go H because the Loss-of-Lock detection circuit compares the recover clock with the RxRefClk.

**Figure 9-8. Reference Clock Block Diagram**


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

Each TX Serializer and RX Deserializer can be split into full data rate and div2 rate or div11 rate depending on the protocol, allowing for different data rates in each direction and in each channel. Refer to Figure 9-8 for more information.

### Reference Clock Sources

refclkp, refclkn

Dedicated LVDS input. This clock source is the preferred choice unless different clock sources for RX and TX are used. The input can interface to different electrical standards, such as CML, LVDS, or LVPECL.

#### FPGA txrefclk, rxrefclk

Reference clock from FPGA logic. This clock signal can be routed from FPGA I/O pins, or from FPGA sysCLOCK PLL. In either case, the PCLK clock tree should be used to route the clock to minimize FPGA core noise coupling. When the clock is routed from the pin, the shared PCLK pins should be used to directly connect the pin to the PCLK clock tree. The clock input on the pin can be LVDS or single-ended.

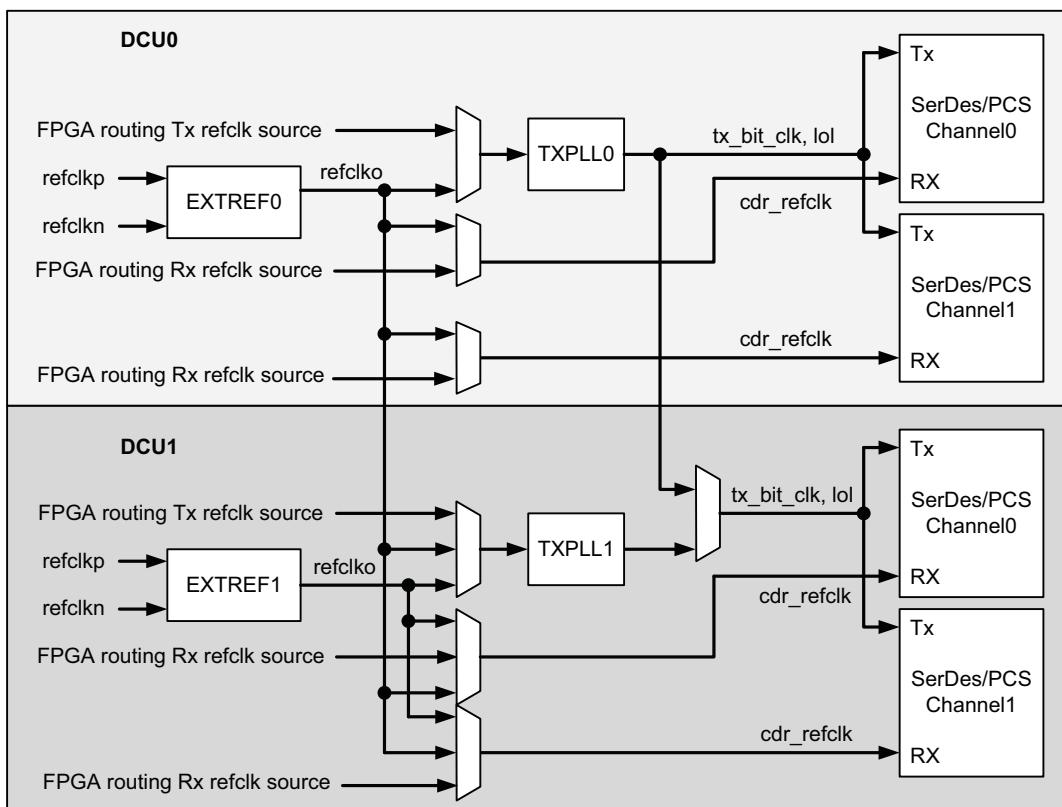
When an FPGA PLL is used as the reference clock, the reference clock to the PLL should be assigned to a dedicated PLL input pad. The FPGA PLL output jitter may not meet system specifications at higher data rates. Use of an FPGA PLL is not recommended in jitter-sensitive applications.

#### Clock Distribution in Complementary DCUs

Figure 9-9 shows the clock distribution for two DCU devices. When x4 link is used, the LFE5UM SERDES provides sharing of the High Speed bit clocks between the two DCUs. This minimizes Transmit skew between the channels residing in different DCUs.

Also shown in the diagram is sharing of RefClk. This sharing of RefClk allow the Rx channels to use the same clock source, without having to route to two different REFCLK pins externally.

**Figure 9-9. Clock Distribution Diagram for a Two Complementary DCU Pair**



#### Spread Spectrum Clocking (SSC) Support

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

rate in the range not exceeding 30KHz to 33KHz. Along with the +/- 300ppm tolerance limit, both ports require the same bit rate clock when the data is modulated with an SSC.

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

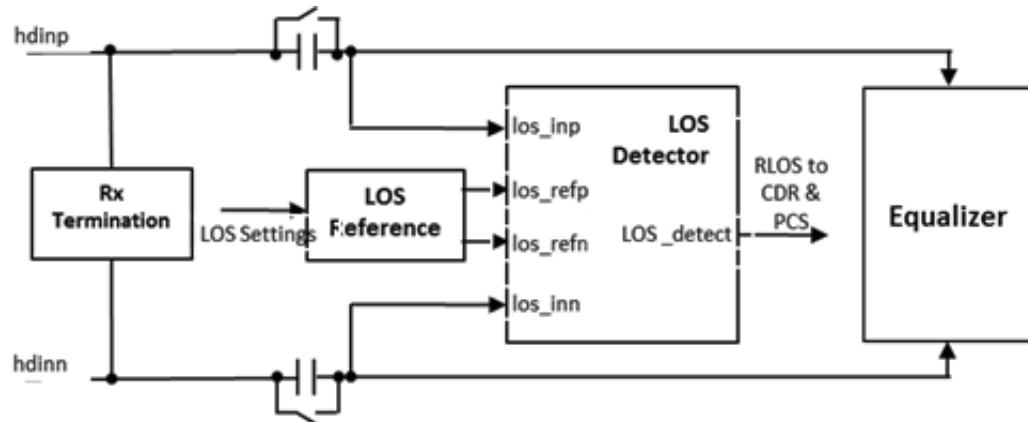
While the LFE5UM architecture will allow the mixing of a PCI Express channel and a Gigabit Ethernet, Serial RapidIO or SGMII channel within the same dual, using a PCI Express SSC as the transmit reference clock will cause a violation of the Gigabit Ethernet, Serial RapidIO and SGMII transmit jitter specifications.

## **Loss of Signal**

Each channel contains a Loss of Signal (LOS) detector circuit at receiver input as shown in Figure 9-10. LOS detector picks up the data signal from channel receiver input, and compares its amplitude with a threshold voltage. If the input signal amplitude is lower than the threshold voltage, LOS detector issues loss of signal output.

Detected LOS indicates that amplitude of input signal is too low. In that case receiver CDR may not be able to recover data from incoming signal. It is a good indication of any error in Channel link. The timing of absent signal determines meaning of handshaking between link endpoints.

**Figure 9-10. Loss of Signal Detector connection to Receiver Inputs**



LOS detection is affected by data rate, data pattern and channel degradation. LOS block shares the signal in the signal path to the Equalizer

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

Description	Min.	Typ.	Max.	Units
Time to detect that signal is lost (rx_los_low 0 to 1)	-	8	10	ns
Time to detect that signal is present (rx_los_low 1 to 0)	-	8	10	ns

## **Loss Of Lock**

Both the transmit PLL and the individual channel CDRs have digital counter-based, loss-of-lock detectors.

If the transmit PLL loses lock, the loss-of-lock for the PLL is asserted and remains asserted until the PLL re-acquires lock. When the PLL loses lock, it is likely to be caused by a reference clock problem.

If a CDR loses lock, the loss-of-lock for that channel is asserted and locking to the reference clock retrains the DCO in the CDR. When the DCO re-training is achieved, loss-of-lock for that channel is de-asserted and the CDR is switched back over to lock to the incoming data. The CDR will either remain locked to the data, or will go back out

of lock again in which case the re-training cycle will repeat. For detailed information on CDR loss-of-lock, please refer to the SERDES/PCS Reset section.

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

Description	Min.	Typ.	Max.	Units
Time to detect that loop is unlocked (tx_pll_lol, rx_cdr_lol goes from 0 to 1)	-	200	500	us
Time to detect that loop is locked (tx_pll_lol, rx_cdr_lol goes from 1 to 0)	-	200	500	us

## TX Lane-to-Lane Skew

Most multi-channel protocol standards require TX lane-to-lane skew is within a certain specification. A control signal in the Transmitter (that can be controlled either by a register bit or by a FPGA signal), tx\_sync\_dual\_c, resets all the active Tx channel to start serialization with bit0.

## Transmit Data

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

### 8b10b Encoder

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

### Serializer

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

## Receive Data

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

### Deserializer

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

### Word Alignment (Byte Boundary Detect)

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

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

- Word alignment control from either embedded Link State Machine (LSM) or from FPGA control. Supported in 8-bit SERDES Only, 10-bit SERDES Only and SDI modes, in addition to 8b10b packet mode.
- Ability to set two programmable word alignment characters (typically one for positive and one for negative disparity) and a programmable per bit mask register for alignment comparison. Alignment characters and the mask register is set on a per quad basis. For many protocols, the word alignment characters can be set to "XX00000011" (jhgfiedcba bits for positive running disparity comma character matching code groups K28.1, K28.5, and K28.7) and "XX01111100" (jhgfiedcba bits for negative running disparity comma character matching code groups K28.1, K28.5, and K28.7). However, the user can define any 10-bit pattern.
- The first alignment character is defined by the 10-bit value assigned to attribute COMMA\_A. This value applies to all channels in a PCS quad.
- The second alignment character is defined by the 10-bit value assigned to attribute COMMA\_B. This value

applies to all channels in a PCS quad. • The mask register defines which word alignment bits to compare (a ‘1’ in a bit of the mask register means check the corresponding bit in the word alignment character register). The mask registers defined by the 10-bit value assigned to attribute COMMA\_M. This value applies to all channels in a PCS quad. When the attribute CHx\_RXWA (word alignment) is set to “ENABLED” and CHx\_ILSM (Internal Link State Machine) is set to “ENABLED”, one of the protocol-based Link State Machines will control word alignment. For more information on the operation of the protocol-based Link State Machines, see the Protocol-Specific Link State Machine section below.

## 8b10b Decoder

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

## External Link State Machine Option

When the attribute CHx\_RXLSM (Internal Link State Machine) is set to DISABLED, and CHx\_RXWA (word alignment) is set to ENABLED, the control signal word\_align\_en\_ch[1:0]\_c is used to enable the word aligner. This signal should be sourced from a FPGA external Link State Machine implemented in the FPGA fabric. When the word\_align\_en\_ch[1:0]\_c is high, the word aligner will lock alignment and stay locked. It will stop comparing incoming data to the user-defined word alignment characters and will maintain current alignment on the first successful compare to either COMMA\_A or COMMA\_B. If re-alignment is required, pulse the word\_align\_en\_ch[1:0]\_c low to high. The word aligner will re-lock on the next match to one of the user-defined word alignment characters. If desired, word\_align\_en\_ch[1:0]\_c can be controlled by a Link State Machine implemented externally to the PCS quad to allow a change in word alignment only under specific conditions.

When a Link State Machine is selected and enabled for a particular channel, that channel’s lsm\_status\_ch[1:0]\_s status signal will go high upon successful link synchronization.

## Idle Insert for Gigabit Ethernet Mode

This is required for Clock Compensation and Auto Negotiation (the latter is done in FPGA logic)

This module automatically inserts /I2/ symbols into the receive data stream during auto-negotiation. Whilst auto-negotiating, the link partner will continuously transmit /C1/ and /C2/ ordered sets. The clock-compensator will not delete these ordered sets as it is configured to only insert/delete /I2/ ordered sets. In order to prevent overruns and underruns in the clock-compensator, /I2/ ordered sets must be periodically inserted to provide insertion/deletion opportunities for the clock compensator.

Whilst performing auto-negotiation, this module will insert a sequence of 8 /I2/ ordered sets (2 bytes each) every 2048 clock cycles. As this module is after the 8b10b decoder, this operation will not introduce any running disparity errors. These /I2/ ordered sets will not be passed on to the FPGA receive interface as the GMII interface is driven to IDLE by the Rx state machine during auto-negotiation. Once auto-negotiation is complete, /I2/ insertion is disabled to prevent any corruption of the received data.

Note that this state machine is active only during auto negotiation. The auto negotiation state machine and the GigE receive state machines are implemented in the soft logic. This state machine depends on the signal “xmit” from the auto negotiation state machine. This signal is provided on the TX data bus. Even though this signal is relatively static (especially after auto negotiation is done) it is currently decided to put this in TX data bus. It provides a signal – “rx\_even\_out”. This will be sent out on the receive data bus to the FPGA logic.

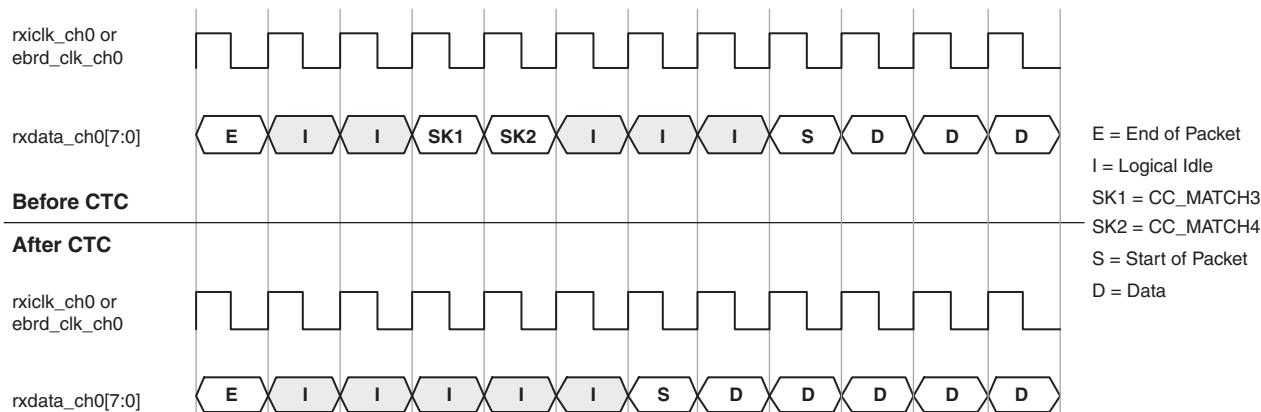
## Clock Tolerance Compensation (CTC)

The Clock Tolerance Compensation (CTC) module performs clock rate adjustment between the recovered receive clocks and the locked reference clock. Clock compensation is performed by inserting or deleting bytes at pre-defined positions, without causing loss of packet data. A 16-byte CTC FIFO is used to transfer data between the two clock domains and will accommodate clock differences of up to the specified ppm tolerance for the LFE5UM SERDES (see DC and Switching Characteristics section of DS1044, [ECP5 Family Data Sheet](#)).

A channel has the Clock Tolerance Compensation block enable when that channel's attribute CHx\_CTC is set to "ENABLED". The CTC is bypassed when that channel's attribute CHx\_CTC is set to "DISABLED". The following diagrams show 2- and 4-byte deletion.

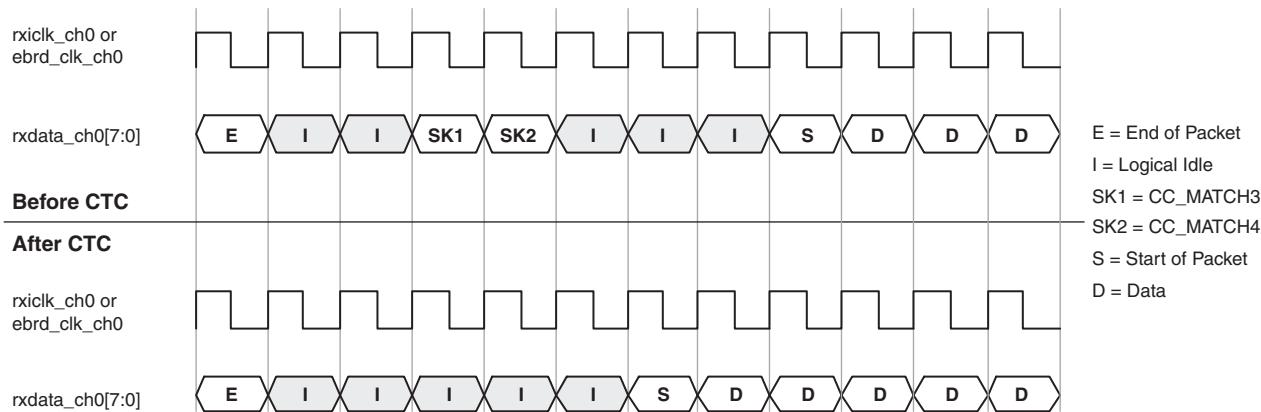
A diagram illustrating 2-byte deletion is shown in Figure 9-11.

**Figure 9-11. Clock Tolerance Compensation 2-Byte Deletion Example**



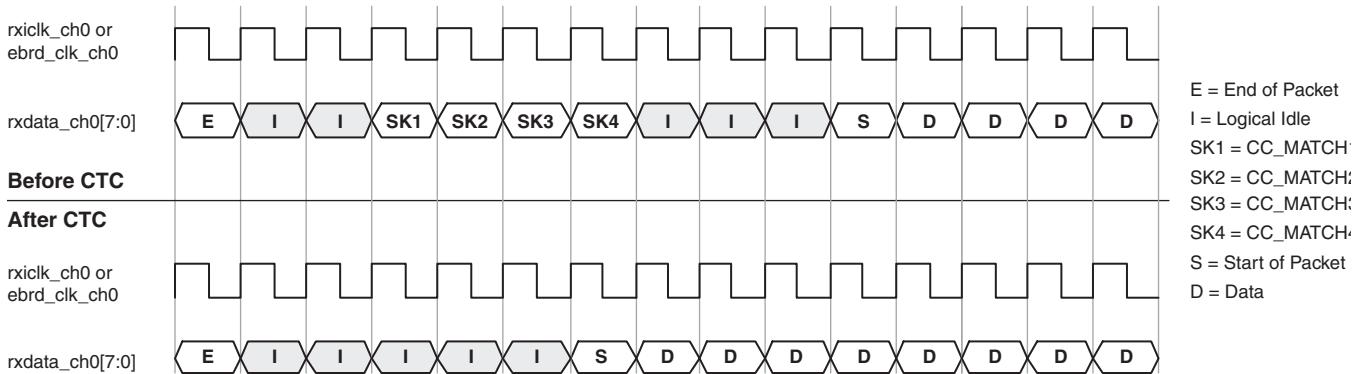
A diagram illustrating 2-byte insertion is shown in Figure 9-12.

**Figure 9-12. Clock Tolerance Compensation 2-Byte Insertion Example**



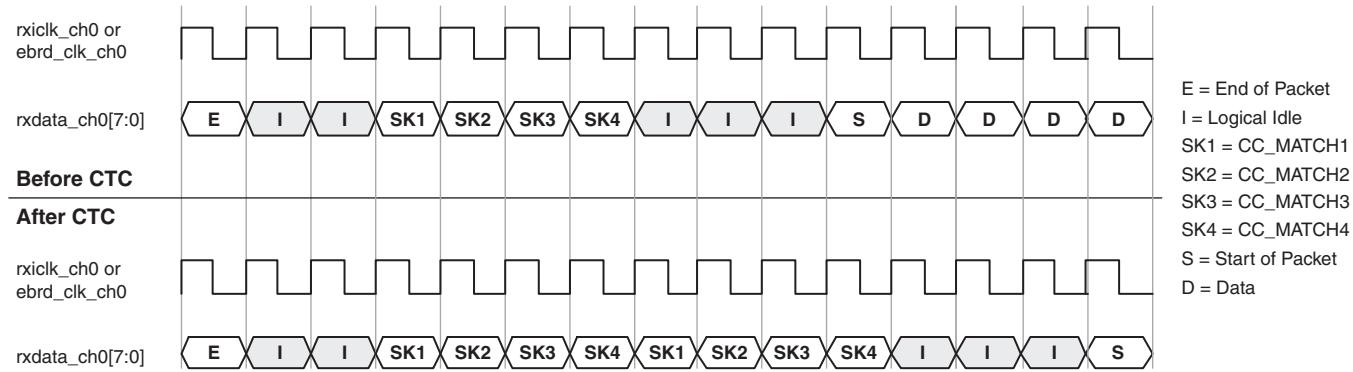
A diagram illustrating 4-byte deletion is shown in Figure 9-13.

**Figure 9-13. Clock Tolerance Compensation 4-Byte Deletion Example**



A diagram illustrating 4-byte insertion is shown in Figure 9-14.

**Figure 9-14. Clock Tolerance Compensation 4-Byte Insertion Example**



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

- Set the insertion/deletion pattern length using the CC\_MATCH\_MODE attribute.** This sets the number of skip bytes the CTC compares to before performing an insertion or deletion. Values for CC\_MATCH\_MODE are “2” (2-byte insertion/deletion) and “4” (4-byte insertion/deletion). The minimum interpacket gap must also be set as appropriate for the targeted application. The interpacket gap is set by assigning values to attribute CC\_MIN\_IPG. Allowed values for CC\_MIN\_IPG are “0”, “1”, “2”, and “3”. The mini-mum allowed interpacket gap after skip character deletion is performed based on these attribute settings is described in Table 9-10.
- The skip byte or ordered set must be set corresponding to the CC\_MATCH\_MODE chosen.** For 4-byte insertion/deletion (CC\_MATCH\_MODE = “4”), the first byte must be assigned to attribute CC\_MATCH1, the second byte must be assigned to attribute CC\_MATCH2, the third byte must be assigned to attribute CC\_MATCH3, and the fourth byte must be assigned to attribute CC\_MATCH4. Values assigned are 10-bit binary values.

For example:

If a 4-byte skip ordered set is /K28.5/D21.4/D21.5/D21.5, then “CC\_MATCH1” should be “0110111100”, “CC\_MATCH2” = “0010010101”, “CC\_MATCH3” = “0010110101” and “CC\_MATCH4” = “0010110101”.

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

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

## Calculating Minimum Interpacket Gap

Table 9-10 shows the relationship between the user-defined values for interpacket gap (defined by the CC\_MIN\_IPG attribute), and the guaranteed minimum number of bytes between packets after a skip character deletion from the PCS. The table shows the interpacket gap as a multiplier number. The minimum number of bytes between packets is equal to the number of bytes per insertion/deletion times the multiplier number shown in the table. For example, if the number of bytes per insertion/deletion is 4 (CC\_MATCH\_MODE is set to “4”), and the minimum interpacket gap attribute CC\_MIN\_IPG is set to “2”, then the minimum interpacket gap is equal to 4 (CC\_MATCH\_MODE = “4”) times 3 (Table 8-10 with CC\_MIN\_IPG = “2”) or 12 bytes. The PCS will not perform a skip character deletion until the minimum number of interpacket bytes have passed through the CTC.

**Table 9-10. Minimum Interpacket Gap Multiplier**

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

## Protocol Modes

### Generic 8b10b Mode

The Generic 8b10b mode of the SERDES/PCS block is intended for applications requiring 8b10b encoding/decoding without the need for additional protocol-specific data manipulation. The LFE5UM SERDES/PCS block can support Generic 8b10b applications up to 3.2 Gbps per channel. In Generic 8b10b mode, the word aligner can be controlled from the embedded PCS Link State Machine (LSM).

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

To achieve link synchronization in this mode, the following conditions need to be satisfied on the 8b10b patterns received at the SERDES channel’s receiver input (hdinp\_ch[0-3]/hdinn\_ch[0-3]):

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

- Two comma characters have to be an even number of word clock cycles apart Additional information:
- It takes roughly four good comma/data pairs for the LSM to reach link synchronization
- It takes four consecutive errors (illegal 8b10b encoded characters, code violations, disparity errors, uneven number of clock cycles between commas) to cause the LSM to unlock
- A CDR loss of lock condition will cause the LSM to unlock by virtue of the many code violation and disparity errors that will result
- When the internal reset sequence state machine is used, a CDR loss of lock (`rx_cdr_lol_ch[3:0]_s`) or a loss of signal (`rx_los_low_ch[3:0]_s`) condition will cause the RX reset sequence state machine to reset the SERDES and cause the LSM to unlock.

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

Valid (even) comma boundary:

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

Valid (odd) comma boundary:

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

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

Alternatively, the LSM can be disabled, and the word aligner is controlled from the fabric `word_align_en_ch[3:0]_c` input pin.

### Transmit Path

- Serializer
- 8b10b encoder

### Receive Path

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

### Gigabit Ethernet and SGMII Modes

The Gigabit Ethernet mode of the LFE5UM SERDES/PCS block supports full compatibility, from the Serial I/O to the GMII/SGMII interface of the IEEE 802.3-2002 1000 BASE-X Gigabit Ethernet standard.

### Transmit Path

- Serializer
- 8b10b encoding

### Receive Path

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

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

This is required for Clock Compensation and Auto-negotiation. Auto-negotiation is done in FPGA logic. The Lattice Gigabit Ethernet PCS IP core provides the auto-negotiation discussed below.

Idle pattern insertion is required for clock compensation and auto-negotiation. Auto-negotiation is done in FPGA logic. This module automatically inserts /I2/ symbols into the receive data stream during auto-negotiation. While auto-negotiating, the link partner will continuously transmit /C1/ and /C2/ ordered sets. The clock-compensator will not delete these ordered sets as it is configured to only insert/delete /I2/ ordered sets. In order to prevent overruns and underruns in the clock-compensator, /I2/ ordered sets must be periodically inserted to provide insertion/deletion opportunities for the clock compensator.

While performing auto-negotiation, this module will insert a sequence of eight /I2/ ordered sets (2 bytes each) every 2048 clock cycles. As this module is after the 8b10b decoder, this operation will not introduce any running disparity errors. These /I2/ ordered sets will not be passed on to the FPGA receive interface as the GMII interface is driven to IDLE by the RX state machine during auto-negotiation. Once auto-negotiation is complete, /I2/ insertion is disabled to prevent any corruption of the received data.

Note that this state machine is active only during auto-negotiation. The auto-negotiation state machine and the GbE receive state machines are implemented in the soft logic. This state machine depends on the signal xmit\_ch[1:0] from the auto-negotiation state machine. This signal is provided on the TX data bus. Though this signal is relatively static (especially after auto-negotiation) it is included in the TX data bus.

**Table 9-11. GbE IDLE State Machine Control and Status Signals**

Module Signal	Direction	Description
xmit_ch[1:0]	In	From FPGA logic Auto-Negotiation State Machine

### Gigabit Ethernet Idle Insert and correct\_disp\_ch[1:0] Signals

The correct\_disp\_ch[1:0] signal is used on the transmit side of the PCS to ensure that an interpacket gap begins in the negative disparity state. Note that at the end of an Ethernet frame, the current disparity state of the transmitter can be either positive or negative, depending on the size and data content of the Ethernet frame.

However, from the FPGA soft-logic side of the PCS, the current disparity state of the PCS transmitter is unknown. This is where the correct\_disp\_ch[1:0] signal comes into play. If the correct\_disp\_ch[1:0] signal is asserted for one clock cycle upon entering an interpacket gap, it will force the PCS transmitter to insert an IDLE1 ordered-set into the transmit data stream if the current disparity is positive. However, if the current disparity is negative, then no change is made to the transmit data stream.

From the FPGA soft-logic side of the PCS, the interpacket gap is typically characterized by the continuous transmission of the IDLE2 ordered set which is as follows: tx\_k\_ch=1, txdata= 0xBC tx\_k\_ch=0, txdata=0x50.

Note that in the PCS channel, IDLE2s mean that current disparity is to be preserved. IDLE1s mean that the current disparity state should be flipped. Therefore, it is possible to ensure that the interpacket gap begins in a negative disparity state. If the disparity state before the interpacket gap is negative, then a continuous stream of IDLE2s are transmitted during the interpacket gap. If the disparity state before the interpacket gap is positive, then a single IDLE1 is transmitted followed by a continuous stream of IDLE2s.

In the FPGA soft-logic side of the PCS, the interpacket gap is always driven with IDLE2s into the PCS. The correct\_disp\_ch[3:0] signal is asserted for one clock cycle, k\_ctrl=0, data=0x50 when the interpacket gap first begins. If necessary, the PCS will convert this IDLE2 into an IDLE1. For the remainder of the interpacket gap, IDLE2s should be driven into the PCS and the correct\_disparity\_chx signal should remain deasserted.

For example, if a continuous stream of 512 bytes of Ethernet frames and 512 bytes of /I/ are sent, it can be observed that:

- During the first interpacket gap, all negative disparity /I2/s are seen (K28.5(-) D16.2(+))
- During the next interpacket gap, the period begins with positive disparity /I1/ (K28.5 (+), D5.6 (+/- are the same)), then all remaining ordered sets are negative disparity /I2/s
- During the next interpacket gap, all negative disparity /I2/s are seen
- During the next interpacket gap, the period begins with positive disparity /I1/ (K28.5 (+), D5.6 (+/- are the same)), then all remaining ordered sets are negative disparity /I2/s

A number of programmable options are supported within the encoder module. These are:

- Ability to force negative or positive disparity on a per-word basis
- Ability to input data directly from the FIFO Bridge - external multiplexer
- Ability to replace code words dependant upon running disparity (100BASE-X and FC)
- Software register controlled bypass mode

## XAUI Mode

With the Lattice XAUI IP Core, the XAUI mode of the SERDES/PCS block supports full compatibility from Serial I/O to the XGMII interface of the IEEE 802.3-2002 XAUI standard. XAUI Mode supports 10 Gigabit Ethernet.

### Transmit Path

- Serializer
- Transmit State Machine which performs translation of XGMII idles to proper ||A||, ||K||, ||R|| characters according to the IEEE 802.3ae-2002 specification
- 8b10b Encoding

### Receive Path

- Deserializer
- Word alignment based on IEEE 802.3-2002 defined alignment characters.
- 8b10b Decoding
- The XAUI Link State Machine is compliant to Figure 48-7 - PCS synchronization state diagram of IEEE 802.3ae-2002 with one exception. Figure 48-7 requires that four consecutive good code groups be received in order for the LSM to transition from one SYNC\_ACQUIRED\_{N} (N=2,3,4) to SYNC\_ACQUIRED\_{N-1}. Instead, the actual LSM implementation requires five consecutive good code groups to make the transition.

- Clock Tolerance Compensation logic in PCS is disabled in XAUI mode. MCA (Multi-Channel Alignment) and CTC are done in the XAUI IP core.
- x4 multi-channel alignment should be done in FPGA core logic.

## **PCI Express Revision 1.1 (2.5 Gbps) Mode**

The PCI Express mode of the SERDES/PCS block supports x1, x2, and x4 PCI Express applications.

### **Transmit Path**

- Serializer
- 8b10b Encoding
- Receiver Detection
- Electrical Idle

### **Receive Path**

- Deserializer
- Word alignment based on the Sync Code
- 8b10b Decoding
- Link Synchronization State Machine functions incorporating operations defined in the PCS Synchronization State Machine (Figure 48-7) of the IEEE 802.3ae-2002 10GBASE-X Specification.
- x2 or x4 PCI Express operation with one PCS quad set to PCI Express mode.
- Clock Tolerance Compensation logic capable of accommodating clock domain differences.
- x2 or x4 multi-channel alignment should be done in FPGA core logic.

Table 9-12 describes the PCI Express mode specific ports.

**Table 9-12. PCI Express Mode Specific Ports**

Signal	Direction	Class	Description
pcie_done_ch[1:0]_s	Out	Channel	1 = Far-end receiver detection complete 0 = Far-end receiver detection incomplete
pcie_con_ch[1:0]_s	Out	Channel	Result of far-end Receiver detection 1 = Far-end receiver detected 0 = Far-end receiver not detected
pcie_det_en_ch[1:0]_c	In	Channel	FPGA logic (user logic) informs the SERDES block that it will request a PCI Express Receiver Detection operation 1 = Enable PCI Express Receiver Detect 0 = Normal Operation
pcie_ct_ch[1:0]_c	In	Channel	1 = Request transmitter to do far-end receiver detection 0 = Normal Operation
rxstatus[2:0]	Out	Channel	Per channel PCI Express receive status port. RxStatus is an encoded status of the receive data path. 2 bits wide if in 16-bit bus mode.

## **PCI Express Termination**

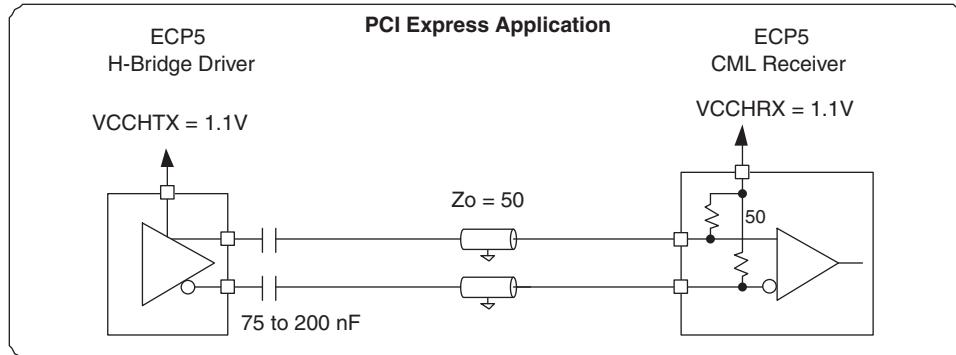
At the electrical level, PCI Express utilizes two uni-directional low voltage differential signaling pairs at 2.5 Gbps for each lane. Transmit and receive are separate differential pairs, for a total of four data wires per lane. An input receiver with programmable equalization and output transmitters with programmable de-emphasis permits optimization of the link. The PCI Express specification requires that the differential line must be common mode terminated at the receiving end. Each link requires a termination resistor at the far (receiver) end. The nominal resistor

values used are 100 ohms. This is accomplished by using the embedded termination features of the CML inputs as shown in Figure 9-15. The specification requires AC coupling capacitors (CTX) on the transmit side of the link. This eliminates potential common-mode bias mismatches between transmit and receive devices. The capacitors must be added external to the Lattice CML outputs.

### PCI Express L2 State

For the PCI Express L2 state, the rx\_pwrup\_c signal should not be de-asserted to power-down the rx channel. This will force the RX termination to high impedance and will not allow the far-end to detect the receiver. Rather, the rx\_pcs\_RST\_c signal should be used to hold the channel in reset to save power.

**Figure 9-15. PCI Express Interface Diagram**



**Table 9-13. Differential PCI Express Specifications**

Symbol	Parameter	Min.	Nom.	Max.	Units	Comments	Location
ZTX-DIFF-DC	DC Differential TX Impedance	80	100	120	Ohm	TX DC Differential mode low impedance. ZTX-DIFF-DC is the small signal resistance of the transmitter measured at a DC operating point that is equivalent to that established by connecting a 100 Ohm resistor from D+ and D- while the TX is driving a static logic one or logic zero.	Internal
ZRX-DIFF-DC	DC Differential Input Impedance	80	100	120	Ohm	RX DC Differential mode impedance during all LTSSM states. When transmitting from a Fundamental Reset to Detect (the initial state of the LTSSM), there is a 5ms transition time before receiver termination values must be met on all un-configured lanes of a port	Internal
CTX	AC Coupling Capacitor	75	-	200	nF	All transmitters shall be AC coupled. The AC coupling is required either within the media or within transmitting component itself	External

### PCI Express Electrical Idle Transmission

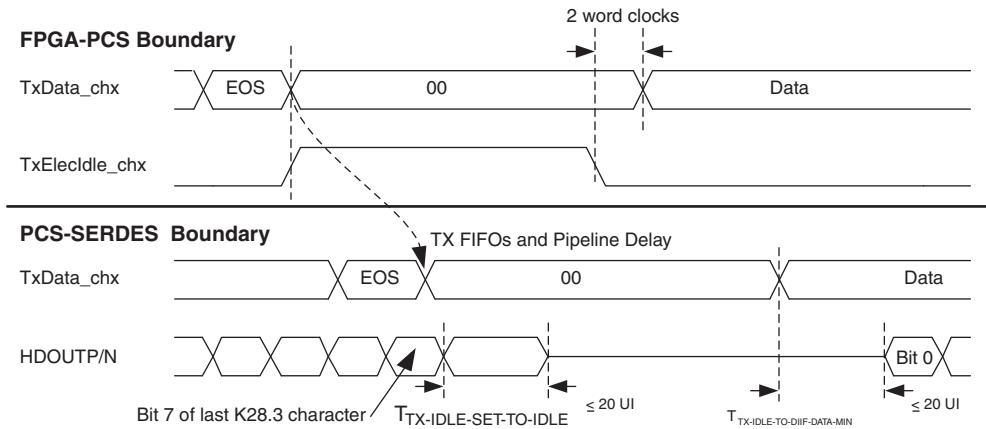
Electrical Idle is a steady state condition where the transmitter P and N voltages are held constant at the same value (i.e., the Electrical Idle Differential Peak Output Voltage, VTX-IDLE-DIFFp, is between 0 and 20mV). Electrical Idle is primarily used in power saving and inactive states.

Per the PCI Express base specification, before a transmitter enters Electrical Idle, it must always send the Electrical Idle Ordered Set (EIOS), a K28.5 (COM) followed by three K28.5 (IDL). After sending the last symbol of the Electrical Idle Ordered Set, the transmitter must be in a valid Electrical Idle state as specified by TTX-IDLE-SET-TO-IDLE is less than 20UI.

To achieve this, the Electrical Idle Enable (tx\_idle\_chx\_c) from the FPGA core logic to the PCS is sent in along with each transmit data. This signal is pipelined similarly all the way to the PCS-SERDES boundary. For all valid data this signal is LOW. To initiate Electrical Idle, the FPGA logic pulls this signal HIGH on the clock after it transmits the last K28.5 (IDL) symbol. As the signal is pipelined to the PCS-SERDES boundary, the relationship between the transmit data and this signal is exactly the same as on the FPGA-PCS boundary.

14UI after the rising edge of the Electrical Idle enable signal at the PCS-SERDES boundary the last bit (bit7) of the last K28.3 (IDL) symbol is transmitted. 16UI (<20UI) later the transmit differential buffer achieves Electrical Idle state.

**Figure 9-16. Transmit Electrical Idle**



As long as the FPGA core logic deems that the transmitter needs to stay in Electrical Idle state it needs to clock in data (preferably all zeros) along with the Electrical Idle Enable (tx\_idle\_chx\_c) signal active (HIGH). The transmitter is required to stay in the Electrical Idle state for a minimum of 50UI (20ns) (TTX-IDLE-MIN).

### PCI Express Electrical Idle Detection

Each channel in the quad has a loss-of-signal detector. The Electrical Idle is detected once two out of the three K28.3 (IDL) symbols in the Electrical Idle Ordered Set (EOS) have been received. After the Electrical Idle Ordered Set is received, the receiver should wait for a minimum of 50ns (TTX-IDLE-MIN) before enabling its Electrical Idle Exit detector.

These signals (one per channel, four per quad) should be routed through the PCS, and should be made available to the FPGA core. The required state machine(s) to support electrical idle can then be constructed in the FPGA core.

### PCI Express Receiver Detection

Figure 9-17 shows a Receiver Detection sequence. A Receiver Detection test can be performed on each channel of a quad independently. Before starting a Receiver Detection test, the transmitter must be put into electrical idle by setting the tx\_idle\_ch#\_c input high. The Receiver Detection test can begin 120 ns after tx\_elec\_idle is set high by driving the appropriate pci\_det\_en\_ch#\_c high. This puts the corresponding SERDES Transmit buffer into receiver detect mode by setting the driver termination to high impedance and pulling both differential outputs to VCCOB through the high impedance driver termination.

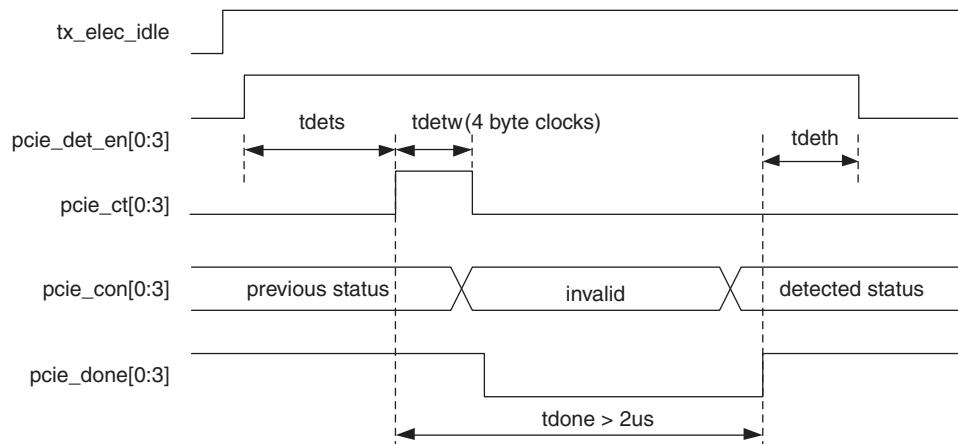
Setting the SERDES Transmit buffer into receiver detect state takes up to 120 ns. After 120 ns, the receiver detect test can be initiated by driving the channel's pcie\_ct\_ch#\_c input high for four byte (word) clock cycles. The corresponding channel's pcie\_done\_ch#\_s is then cleared asynchronously. After enough time for the receiver detect test to finish has elapsed (determined by the time constant on the transmit side), the pcie\_done\_ch#\_s receiver detect status port will go high and the Receiver Detect status can be monitored at the pcie\_con\_ch#\_s port. If at that time

the pcie\_con\_ch#\_s port is high, then a receiver has been detected on that channel. If, however, the pcie\_con\_ch#\_s port is low, then no receiver has been detected for that channel. Once the Receiver Detect test is complete, tx\_idle\_ch#\_c can be deasserted.

Receiver detection proceeds as follows:

1. The user drives pcie\_det\_en high, putting the corresponding TX driver into receiver detect mode. This sets the driver termination to high-impedance (5K ohm) and pulls both outputs of the differential driver toward common mode through the high-impedance driver termination. The TX driver takes some time to enter this state so the pcie\_det\_en must be driven high for at least 120ns before pcie\_ct is asserted.
2. The user drives pcie\_ct high for four byte clocks.
3. SERDES drives the corresponding pcie\_done low.
4. SERDES drives the internal signal (corresponding to pcie\_ct) for as long as it is required (based on the time constant) to detect the receiver.
5. SERDES drives the corresponding pcie\_con connection status.
6. SERDES drives the corresponding pcie\_done high
7. The user can use this asserted state of pcie\_done to sample the pcie\_con status to determine if the receiver detection was successful.

**Figure 9-17. PCI Express Mode Receiver Detection Sequence**



## PCI Express Power Down Mode

rx\_serdes\_rst\_ch[1:0] reset signal should be used instead of rx\_pwrup\_ch[1:0] signal. This allows the RX termination to remain enabled at 50 Ohms so that the far-end transmitter can detect that a receiver is connected.

## PCI Express Beacon Support

This section highlights how the LFE5UM PCS can support Beacon detection and transmission. The PCI Express requirements for Beacon detection are presented with the PCS support for Beacon transmission and detection.

### Beacon Detection Requirements

- Beacon is required for exit from L2 (P2) state.
- Beacon is a DC-balanced signal of periodic arbitrary data, which is required to contain some pulse widths  $\geq 2\text{ns}$  (500 MHz) and  $< 16\text{us}$  (30 KHz).
- Maximum time between pulses should be  $< 16\text{ us}$ .
- DC balance must be restored within  $< 32\text{ us}$ .

- For pulse widths > 500 ns, the output beacon voltage level must be 6 db down from VTX-DIFFp-p (800 mV to 1200 mV).
- For pulse widths < 500 ns, the output beacon voltage level must be  $\leq$  VTX-DIFFp-p and  $\geq$  3.5 db down from VTX-DIFFp-p.

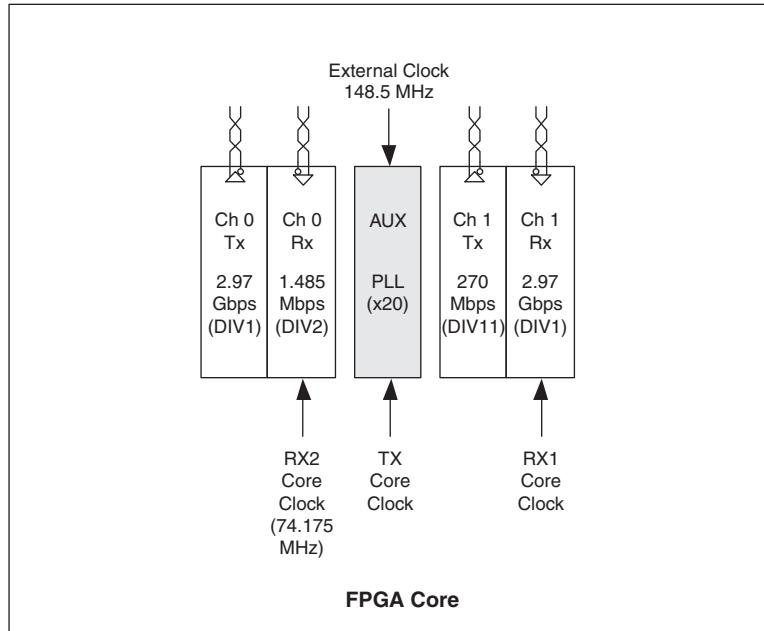
### **PCS Beacon Detection Support**

- The signal loss threshold detection circuit senses if the specified voltage level exists at the receiver buffer.
- This is indicated by the rlos\_lo\_ch[1:0] signal.
- This setting can be used both for PCI Express Electrical Idle Detection and PCI Express beacon detection (when in power state P2).
- The remote transmitting device can have a beacon output voltage of 6 db down from VTX-DIFFpp (i.e., 201 mV). If this signal can be detected, then the beacon is detected.

### **PCS Beacon Transmission Support**

Sending the K28.5 character (IDLE) (5 ones followed by 5 zeroes) provides a periodic pulse with of 2 ns occurring every 2 ns (1.0 UI = 400 ps, multiplied by 5 = 2 ns). This meets the lower requirement. The output beacon voltage level can then be VTX-DIFFp-p. This is a valid beacon transmission.

**Figure 9-18. Example: 3G/HD/SD RX/TX At Different Rate**



To support the major application requirements, a selectable DIV per RX and TX is supported. In LFE5UM, DIV11 has been added. One potential multi-rate configuration is to provide a 148.5MHz REFCLK from the primary pins to the TX PLL. The TX PLL would be in the x20 mode. The resulting output clock would be 2.97 GHz. Then, by using the DIV2 for 1.485Gbps and DIV11 for 270Mbps, a very quick switchover can be achieved without having to re-train and lock the PLL.

### **Serial RapidIO (SRIO) Mode**

This section describes the operation of the Serial RapidIO mode of the SERDES/PCS block. The LFE5UM supports 1x and 4x Serial RapidIO applications with one PCS quad. SRIO1.0 highlights multiple frequency support, 3.125 Gbps, 2.5 Gbps and 1.25 Gbps. The ratio of these rates is 2.5:2:1. Supporting all of these rates with integer dividers in the same quad is not possible, but the ratio of 2.5 Gbps to 1.25 Gbps is 2:1 (full rate : half rate).

**Transmit Path**

- Serializer
- 8b10b encoding

**Receive Path**

- Deserializer
- Word alignment based on the Sync Code Group as defined in the RapidIO Physical Layer 1x/4x LP-Serial specification.
- 8b10b decoding
- Clock Tolerance Compensation logic capable of accommodating clock domain differences

**Serial Digital Video and Out-Of-Band Low Speed SERDES Operation**

The LFE5UM SERDES/PCS supports any data rates that are slower than what the SERDES TX PLL and RX CDR natively support (<250Mbps: Out-Of-Band signal, OOB), by bypassing the receiver CDR and associated SERDES/PCS logic (e.g., 100Mbps Fast Ethernet, SD-SDI at 143Mbps or 177Mbps). Though these out-of-band paths primarily use low data rates, higher rates can be used for other functional reasons. See the Multi-Rate SMPTE Support section of this document for more information.

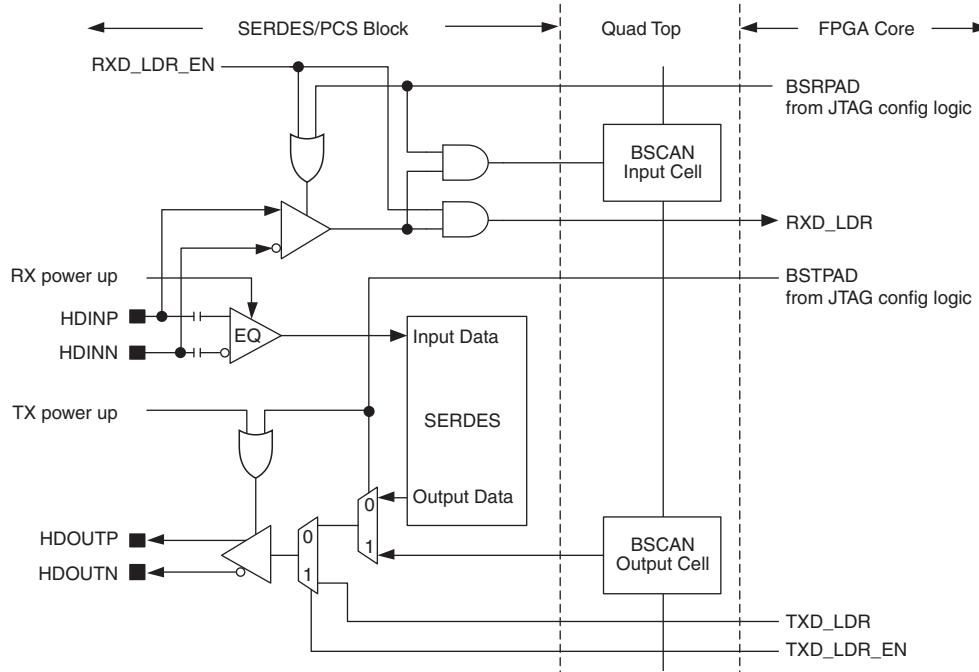
In addition, for SD-SDI, these rates sometimes must co-exist on the same differential RX pair with HD-SDI rates (i.e., SD-SDI rates may be active and then the data rate may switch over to HD-SDI rates). Since there is no way to predict which of these two rates will be in effect, it is possible to send the input data stream to two SERDES in parallel, a high-speed SERDES (already in the quad) and a lower-speed SERDES (implemented outside the quad). One possible implementation is shown in Figure 9-19.

There is an input per channel RXD\_LDR, low data rate single-ended input from the RX buffer to the FPGA core. In the core a low-speed Clock Data Recovery (CDR) block or a Data Recovery Unit (DRU) can be built using soft logic. A channel register bit, RXD\_LDR\_EN, can enable this data path. If enabled by another register bit, a signal from the FPGA can also enable this in LFE5UM.

In the transmit direction, it is also possible to use a serializer built in soft logic in the FPGA core and use the TXD\_LDR pin to send data into the SERDES. It will be muxed in at a point just before the de-emphasis logic near where the regular high-speed SERDES path is muxed with the boundary scan path. This is shown conceptually in Figure 9-19. The low data rate path can be selected by setting a channel register bit, TX\_LDR\_EN.

Alternatively, on the output side, the high-speed SERDES is used to transmit either high-speed data, or lower speed data using decimation (the SERDES continues to run at high-speed, but the output data can only change every nth clock where n is the decimation factor).

**Figure 9-19. Conceptual Low Date Rate Path for RX and TX; Example for Out-of-Band Application**



## Common Public Radio Interface (CPRI)

The goal of CPRI is to allow base station manufacturers and component vendors to share a common protocol and more easily adapt platforms from one user to another.

### Transmit Path

- Serializer
- Transmit State Machine is set to Gigabit Ethernet Mode
- 8b10b Encoding

### Receive Path

- Deserializer
- Word alignment based on IEEE 802.3-2002 1000 BASE-X defined alignment characters
- 8b10b Decoding

CPRI does not specify mechanical or electrical interface requirements. In terms of scope, CPRI has a much narrower focus than OBSAI. CPRI looks solely at the link between the RRH and the baseband module(s). In CPRI nomenclature, those modules are known as Radio Equipment (RE) and Radio Equipment Control (REC), respectively. In other words, CPRI is specifying the same interface as the OBSAI RP3 specification. CPRI primarily covers the physical and data link layer of the interface. It also specifies how to transfer the user plane data, control and management (C&M) plane data and the synchronization plane data.

CPRI has had better “traction” for two reasons - the muscle of the companies backing it and the focus on just one interface link (between the RF modules and the Baseband modules) and even at that focusing primarily on the physical and data link layers.

CPRI allows four line bit rate options; 614.4 Mbps, 1.2288 Gbps, 2.4576 Gbps and 3.072 Gbps; at least one of these rates needs to be supported. The higher line rate is always compared to the one that is immediately lower.

CPRI does not have a mandatory physical layer protocol, but the protocol used must meet the BER requirement of  $1 \times 10^{-12}$ , which is less stringent than OBSAI. It also specifies the clock stability and the phase noise requirements.

CPRI also recommends two electrical variants: high voltage (HV) and low voltage (LV). HV is guided by 1000Base-CX specifications in IEEE 802.3-2002 clause 39 with 100-ohm impedance. LV is guided by XAUI. LV is recommended for all rates and will be the focus for this device.

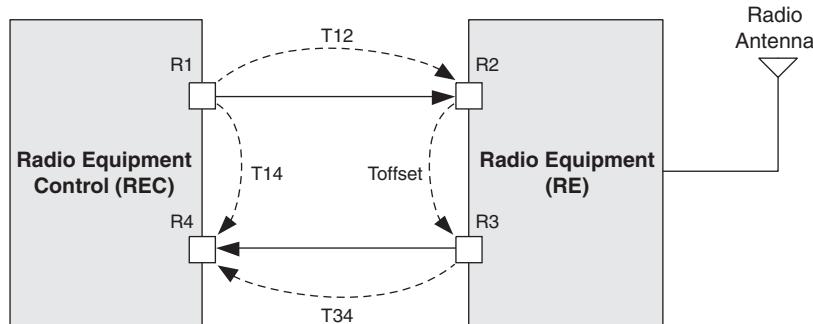
It is important to understand two link layer requirements when dealing with the CPRI and OBSAI specifications:

- Link delay accuracy and cable delay calibration
- Startup synchronization

#### **Link Delay Accuracy and Cable Delay Calibration**

The REs or RRHs are frequency locked to the REC or BTS. Thus, in this synchronous system it is necessary to calibrate all delays between RRHs and the BTS to meet air-interface timing requirements. The interface requires the support of the basic mechanisms to enable calibrating the cable delay on links and the round trip delay on single- and multi-hop connections. Specifically, the reference points for delay calibration and the timing relationship between input and output signals at RE (Radio Equipment) are defined. All definitions and requirements are described for a link between REC Master Port and RE Slave Port in the single-hop scenario as shown in Figure 9-20.

**Figure 9-20. Link Between REC Mater Port and RE Slave Port (Single Hop Scenario)**



Reference points R1-4 correspond to the output point (R1) and the input point (R4) of REC, and the input point (R2) and the output point (R3) of an RE terminating a particular logical connection. The antenna is shown as Ra for reference.

- T12 is the delay of the downlink signal from the output point of REC (R1) to the input point of RE (R2), essentially the downlink cable delay.
- T34 is the delay of the uplink signal from the output point of RE (R3) to the input point of the REC (R4), essentially the uplink cable delay.
- Toffset is the frame offset between the input signal at R2 and the output signal at R3.
- T14 is the frame timing difference between the output signal at R1 and the input signal at R4 (i.e., the round trip delay - RTT).

Delay measurement is accomplished using frame timing. CPRI has a 10ms frame based on the UMTS radio frame number or Node B Frame Number, also known as BFN. Each UMTS Radio Frame has 150 hyperframes (i.e., each HyperFrame is 66.67us) with the corresponding hyperframe number (HFN =  $0 \leq Z \leq 149$ ). Each hyperframe has 256 ( $0 \leq W \leq 255$ ) basic frames (i.e. each basic frame is 260.42ns = Tchip or Tc).

An RE determines the frame timing of its output signal (uplink) to be the fixed offset (Toffset) relative to the frame timing of its input signal (downlink). Toffset is an arbitrary value, which is greater than or equal to 0 and less than

256 Tc (it cannot slip beyond a hyperframe). Different REs may use different values for Toffset. REC knows the value of Toffset of each RE in advance (pre-defined value or RE informs REC by higher layer message).

To determine T14, the downlink BFN and HFN from REC to RE is given back in uplink from the RE to the REC. In the case of an uplink-signaled error condition, the REC treats the uplinks BFN and HFN as invalid. So,  $T_{14} = T_{12} + \text{Toffset} + T_{34}$ .

As stated earlier the system is synchronous. Further, assuming that hyperframes are of fixed length and the RRH-BTS interconnect (cable length) is equal in both directions (i.e.,  $T_{12} = T_{34}$ , both optical fibers are in one bundle), the interconnect delay devolves down to  $(T_{14} - \text{Toffset})/2$ . The method for determining T14 has been discussed earlier. So the major component that affects delay calibration is Toffset. Thus, the interconnect delay is the difference in hyperframe arrival and departure times measured at each side of the link.

Delay calibration requirements are driven by 3GPP and UTRAN requirements specifically requirements R-21 in the CPRI specification (CPRI v3.0 page 20), which states that the accuracy of the round trip delay measurement of the cable delay of one link is  $\pm T_c/16$ . Additionally, requirement R-20 states that the round trip time absolute accuracy of the interface, excluding the round trip group delay on the transmission medium (excluding the cable length), shall meet a similar requirement ( $\pm T_c/16$  for T14). Taking into account the previous discussion, the absolute link delay accuracy in the downlink between REC master port and RE slave port excluding the cable length is half of the above requirement ( $\pm T_c/32$  or approximately 8ns (8.138ns)). Thus, both T14 and Toffset need absolute accuracy less than  $\pm 8$ ns.

Next it is important to determine how many bits of uncertainty can be acceptable for the different rates. Essentially, the various CPRI and OBSAI bit rates can be multiplied by 8.138ns to determine the number of bits worth of indeterminism/variance is acceptable. The impact of this will become clear subsequently when the SERDES serial/parallel data path is discussed.

Most SERDES have a certain level of uncertainty that is introduced in the serializing and de-serializing process. Thus, a SERDES with 16-bit bus architecture may have twice the delay uncertainty as a SERDES with a 8-bit architecture because the number of bits per word is doubled.

TX and RX latency respectively in Table 9-17 is listed. The table also lists the variability between the latency. This variability directly contributes to the absolute delay accuracy required from earlier discussion. The variability comes from three sources: TX FPGA Bridge FIFO, RX FPGA Bridge FIFO and RX Clock Tolerance Compensation FIFO. Since the CPRI system is a synchronous system, the RX CTC FIFO is bypassed and the RX recovered clock is used.

The remaining contributors to the latency variability are the FPGA Bridge FIFO. This FIFO can be bypassed if the interface to the FPGA is 8-bit bus mode. In 16-bit interface mode, the FPGA Bridge FIFO cannot be bypassed because the 2:1 gearing is done via the FIFO.

## SDI (SMPTE) Mode

The SDI mode of the LFE5UM SERDES/PCS block supports all three SDI modes, SD-SDI, HD-SDI and 3G-SDI.

### Transmit Path

- Serializer

### Receive Path

- Deserializer
- Optional word alignment to user-defined alignment pattern.

The following data rates are the most popular in the broadcast video industry.

- SD-SDI (SMPTE259M): 270Mbps

- HD-SDI (SMPTE292M): 1.485 Gbps,  $1.485 \text{ Gbps} / 1.001 = 1.4835 \text{ Gbps}$
- 3G-SDI (SMPTE424M): 2.97 Gbps,  $2.97 \text{ Gbps} / 1.001 = 2.967 \text{ Gbps}$

Most designers have indicated that they would like to see support for all these rates. The reason is that in a broadcast studio, or a satellite head-end or cable head-end, they do not necessarily have prior knowledge of what the RX data rate will be.

The switchover time between different rates should be as low as possible. The time to re-lock the CDR is unavoidable. In the LFE5UM SERDES, the PLL does not have to be re-locked. This is possible because the LFE5UM has per RX and TX dividers. Video links generally have a uni-directional nature (i.e., different channels can run at different rates, and more importantly, the RX and TX in the same channel can run at different rates).

Also, depending on the geography where the equipment is deployed, either the full HD/3G-SDI rates (Europe/Asia) are used while transmitting video or the fractional rates (North America - NTSC). This allows us to develop two potential solution example cases for multi-rate SMPTE support with high quad utilization.

Please note that simultaneous support of 3G/HD Full TX Rate(s) and Fractional TX Rate(s) is not possible in the same SERDES quad. In general, based on the above, geographically partitioned usage is an acceptable limitation.

## Example of Dual-based Channel Arrangements

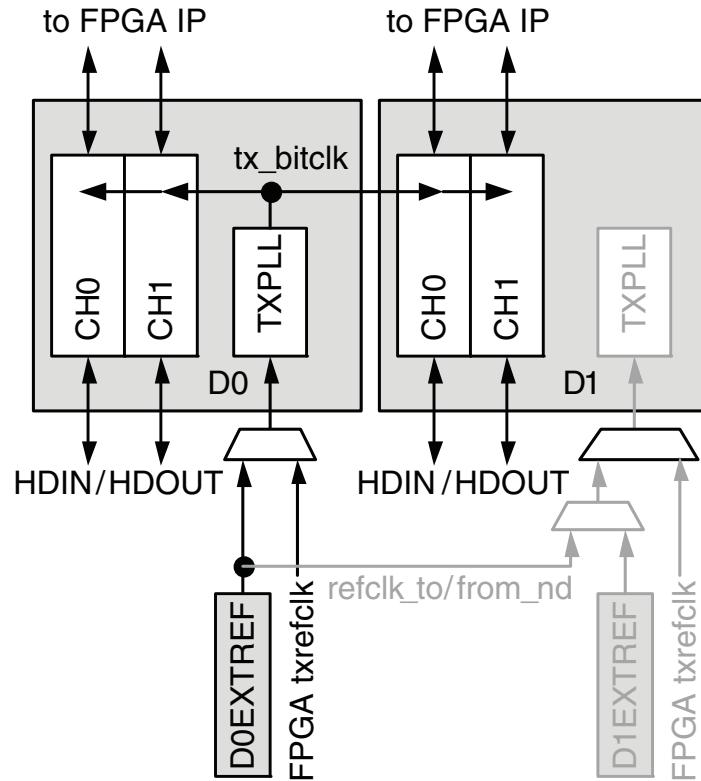
This section shows 5 examples of how instances created in users' view, and how they are placed and connected physically. The user will see how the improved granularity works in dual base architecture.

### Tx Case 1:

All channels in Dual0 and Dual1 use D0 REFCLK(D0EXTREFB or FPGA txrefclk) and D0TXPLL

In this example, all 4channels use same tx\_bitclk from D0TXPLL.

Figure 9-21. Tx Case 1

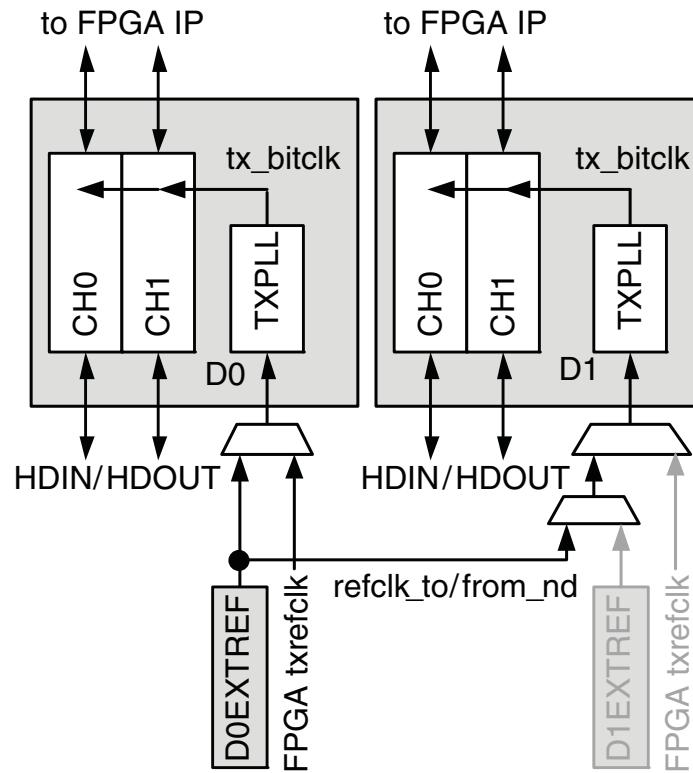


### Tx Case 2:

Dual0 channels use D0 Refclk(D0EXTREFB or FPGA txrefclk) and D0TXPLL.

Dual1 channels use D0 Refclk(D0EXTREFB or FPGA txrefclk) and D1TXPLL

Figure 9-22. Tx Case 2



In this example, both D0TXPLL and D1TXPLL use D0 Refclk. The tx\_bitclks may have different frequency depending on the txpll multiplier. txpll multiplier is not a user attribute but is automatically calculated with data rate and refclk frequency.

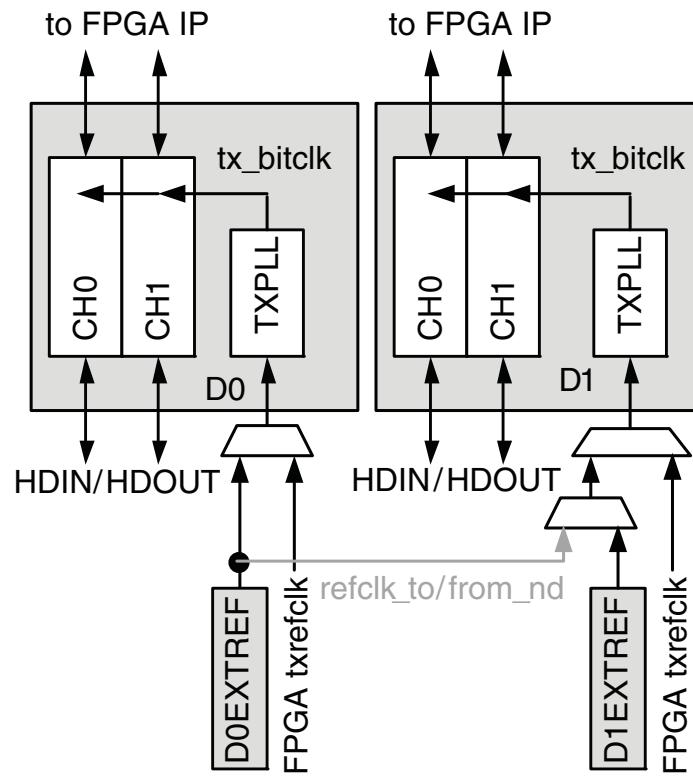
### Tx Case 3:

Dual0 channels use D0 Refclk(D0EXTREFB or FPGA txrefclk) and D0TXPLL

Dual1 channels use D1 Refclk(D1EXTREFB or FPGA txrefclk) and D1TXPLL

In this example, no clocks are crossing Dual boundary.

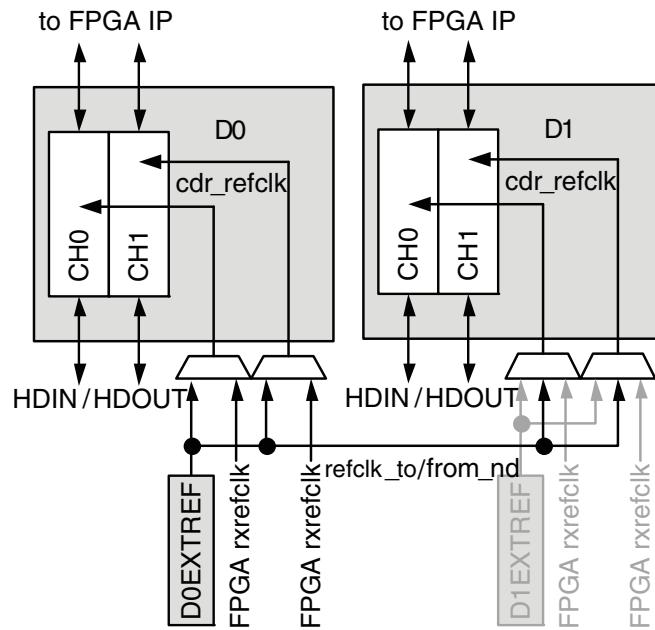
Figure 9-23. Tx Case 3



### Rx Case1:

All channels in Dual0 and Dual1 use D0 REFCLK(D0EXTREFB or FPGA rxrefclk[cdr\_refclk])

Figure 9-24. Rx Case 1

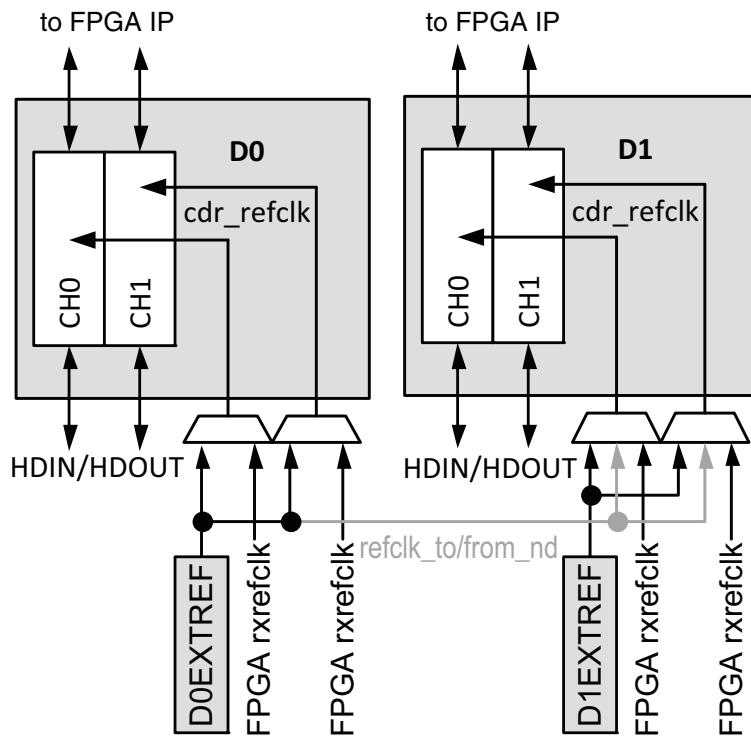


### Rx Case 2:

Dual0 channels use Dual0 Refclk(D0EXTREFB or FPGA rxrefclk).

Dual1 channels use Dual1 Refclk(D1EXTREFB or FPGA rxrefclk).

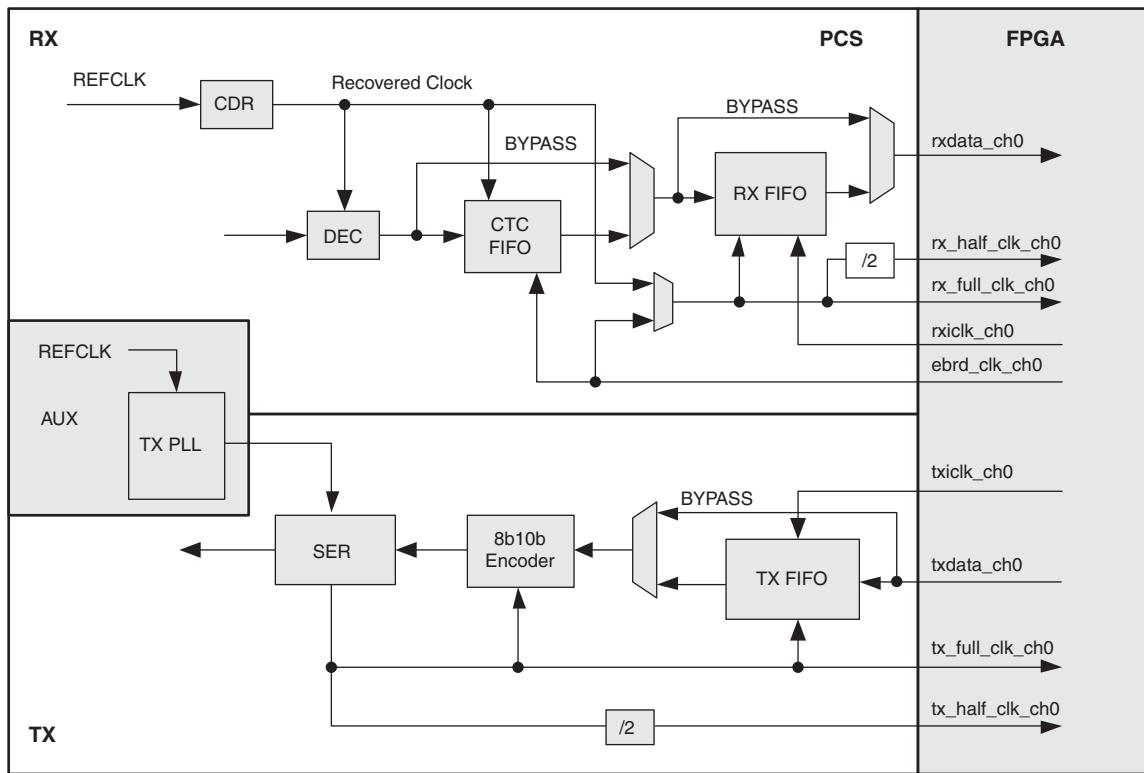
**Figure 9-25. Rx Case 2**



### FPGA Interface Clocks

Figure 9-26 shows a conceptual diagram of the later stage of the PCS core and the FPGA Bridge and the major clocks that cross the boundary between PCS and the FPGA.

Figure 9-26. Conceptual PCS/FPGA Clock Interface Diagram



In the above diagram and in the subsequent clock diagrams in this section, please note that suffix “i” indicates the index [1:0] i.e., one for each channel. It is a requirement that if any of the selectors to the clock muxes are changed by writes to register bits, the software agent will reset the logic clocked by that muxed clock.

The PCS outputs 8 clocks. There are two transmit clocks (per channel) and two receive clocks (per channel). The two transmit clocks provide full rate and half rate clocks and are all derived from the TX PLL. There are also two clocks (full and half) per receive channel. These clocks are muxed into two clocks output per channel (rx\_pclk, tx\_pclk). These clocks are routed to PCLK to minimize skew and jitter.

Illustration on how the clocks are used can be seen in the Interface cases shown in Table 9-13.

Table 9-14. Seven User Interface Cases between SerDes/PCS Dual and FPGA Core

Interface	Datapath Width	RX CTC FIFO	RX Down Sample FIFO	TX Up Sample FIFO	Notes
Case I_a	8/10 bit	Yes	Yes	Yes	(2)
Case I_b	8/10 bit	Bypass	Yes	Yes	(2)
Case I_c	8/10 bit	Yes	Bypass	Bypass	(2)
Case I_d	8/10 bit	Bypass	Bypass	Bypass	(2)
Case II_a	16/20 bit	Yes	Yes	Yes	(1), (2)
Case II_b	16/20 bit	Bypass	Yes	Yes	(1), (2)

1. When using a 16/20-bit datapath width, the TX phase-shift (upsample) FIFO and the RX phase-shift FIFO (downsample) are always used. They cannot be bypassed. It is not required that both RX and TX have the same FPGA interface datapath width simultaneously. There is independent control available. For the sake of brevity, they have been represented together in the same use case.  
2. The TX phase-shift (upsample) FIFO and the RX phase-shift FIFO (downsample) don't need to be bypassed together. They are independently controllable. Again, for the sake of brevity, they have been represented here in the same case.

## 2:1 Gearing

For guaranteed performance of the FPGA global clock tree, it is recommended to use a 16/20-bit wide interface for SERDES line rates greater than 2.5 Gbps. In this interface, the FPGA interface clocks are running at half the byte clock frequency.

Even though the 16/20-bit wide interface running at half the byte clock frequency can be used with all SERDES line rates, the 8/10-bit wide interface is preferred when the SERDES line rate is low enough to allow it (2.5 Gbps and below) because this results in the most efficient implementation of IP in the FPGA core.

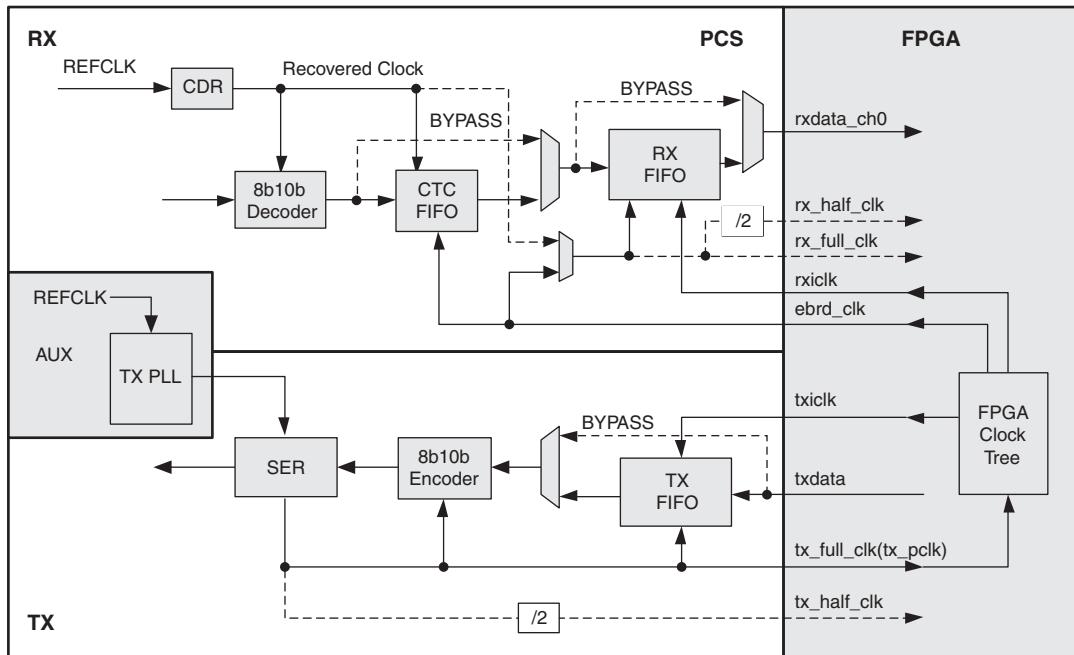
The decision matrix for the six interface cases is explained in Table 9-14.

**Table 9-15. Decision Matrix for Six Interface Cases**

SERDES Line Rate	Dapath Width	Multi-Channel Alignment Required?	CTC Required?	RX FIFO Required?	Interface Case
2.5 Gbps and below	8/10 Bit (1:1 Gearing)	No, Single-Channel Link	Yes	Yes	Case I_a <sup>1</sup>
				No	Case I_c <sup>1</sup>
		Yes, Multi-Channel Link	No	Yes	Case I_b <sup>2</sup>
				No	Case I_d <sup>2</sup>
		Must Bypass, CTC Not Used		Yes	Case I_b <sup>3</sup>
				No	Case I_d <sup>3</sup>
3.2 Gbps and below	16/20 Bit (2:1 Gearing)	No, Single-Channel Link	Yes	Yes	Case II_a <sup>4</sup>
		Yes, Multi-Channel Link	Must Bypass, CTC Not Used	Yes	Case II_b <sup>5</sup>
				Yes	Case II_b <sup>6</sup>
<ol style="list-style-type: none"> <li>1. This case is intended for single-channel links at line rates of 2.5 Gbps and lower (8/10-bit wide interface) that require clock tolerance compensation in the quad. CTC is required when both ends of the link have separate reference clock sources that are within +/- 300ppm of each other. Case I_a is used if the IP in the core requires the RX phase-shift FIFO. Case I_b is used if the IP does not require this FIFO.</li> <li>2. This case is intended for single-channel links at line rates of 2.5 Gbps and lower (8/10-bit wide interface) that do NOT require clock tolerance compensation in the quad. CTC is not required when both ends of the link are connected to the same reference clock source. This is often the case for chip-to-chip links on the same circuit board. There is exactly 0ppm difference between the reference clocks and so CTC is not required and can be bypassed. CTC is also not required in the quad when this function is performed by the IP in the core.</li> <li>3. This case is intended for multi-channel links at line rates of 2.5 Gbps and lower (8/10-bit wide interface). Multi-channel alignment MUST be done in the FPGA design. Since multi-channel alignment must be done prior to CTC, the CTC FIFO in the quad MUST be bypassed when multi-channel alignment is required and so both multi-channel alignment and CTC (if required) are done by the FPGA design.</li> <li>4. This case is intended for single-channel links at line rates of 3.2 Gbps and lower that require a 2:1 gearbox between the quad and the FPGA core (16/20 bit wide interface). Clock tolerance compensation is included in the quad. CTC is required when both ends of the link have separate reference clock sources that are within +/- 300ppm of each other.</li> <li>5. This case is intended for single-channel links at line rates of 3.2 Gbps and lower that require a 2:1 gearbox between the quad and the FPGA core (16/20-bit wide interface). Clock tolerance compensation is NOT included in the quad. CTC is not required when both ends of the link are connected to the same reference clock source. This is often the case for chip-to-chip links on the same circuit board. There is exactly 0ppm difference between the reference clocks and so CTC is not required and can be bypassed. CTC is also not required in the quad when this function is performed by the FPGA design.</li> <li>6. This case is intended for multi-channel links at line rates of 3.2 Gbps and lower that require a 2:1 gearbox between the quad and the FPGA core (16/20-bit wide interface). Multi-channel alignment MUST be done by the FPGA design. Since multi-channel alignment must be done prior to CTC, the CTC FIFO in the quad MUST be bypassed when multi-channel alignment is required and so both multi-channel alignment and CTC (if required) are done by the FPGA design.</li> </ol>					

## Case I\_a: 8/10bit, CTC FIFO NOT Bypassed

Figure 9-27. 8/10-Bit, CTC FIFO and RX/TX FIFOs NOT Bypassed



1. The TX FIFO acts as a Phase Shift FIFO only in this case.
2. The RX FIFO acts as a Phase Shift FIFO only in this case.
3. The quad-level full rate clock from the TX PLL (tx\_full\_clk) has direct access to the FPGA center clock mux. This is a relatively higher performance path. A Global Clock Tree out of the Center Clock Mux is used to clock the user's interface logic in the FPGA. Some leaf nodes of the clock tree are connected to the FPGA Transmit Input clock (txiclk), the CTC FIFO Read Clock per Channel (ebrd\_clk) through the FPGA Receive Input clock (rxiclk). This case is possibly the most common single-channel use case.

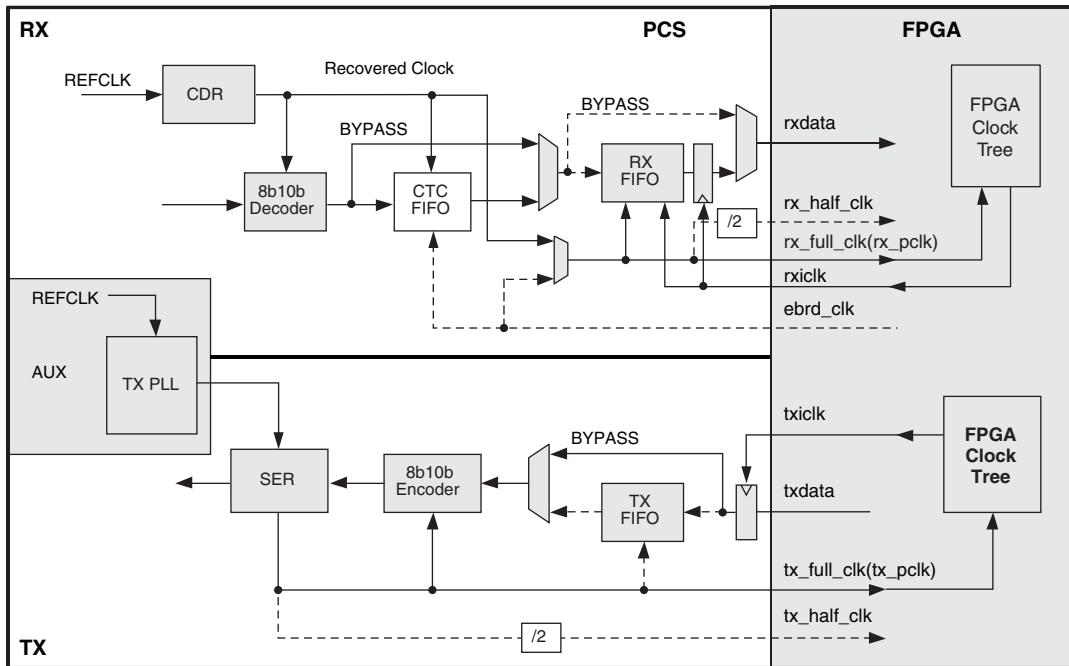
### Example of Clock and Data Signals Interface in FPGA Logic (Case I\_a)

Below is a portion of the SERDES/PCS module instantiation in the top module which describes how clock and data ports are mapped in Verilog.

```
.txiclk_ch0(txclk),
.rxiclk_ch0(txclk),
.rx_full_clk_ch0(),
.tx_full_clk_ch0(txclk),
.tx_half_clk_ch0(),
.txdata_ch0(txdata_2_pcs),
.rxdelay_ch0(rxdata_from_pcs),
.tx_k_ch0(txkcntl_2_pcs),
.rx_k_ch0(rxkcntl_from_pcs),
```

ebrd\_clk\_ch0 is routed automatically by the software, depending on the case.

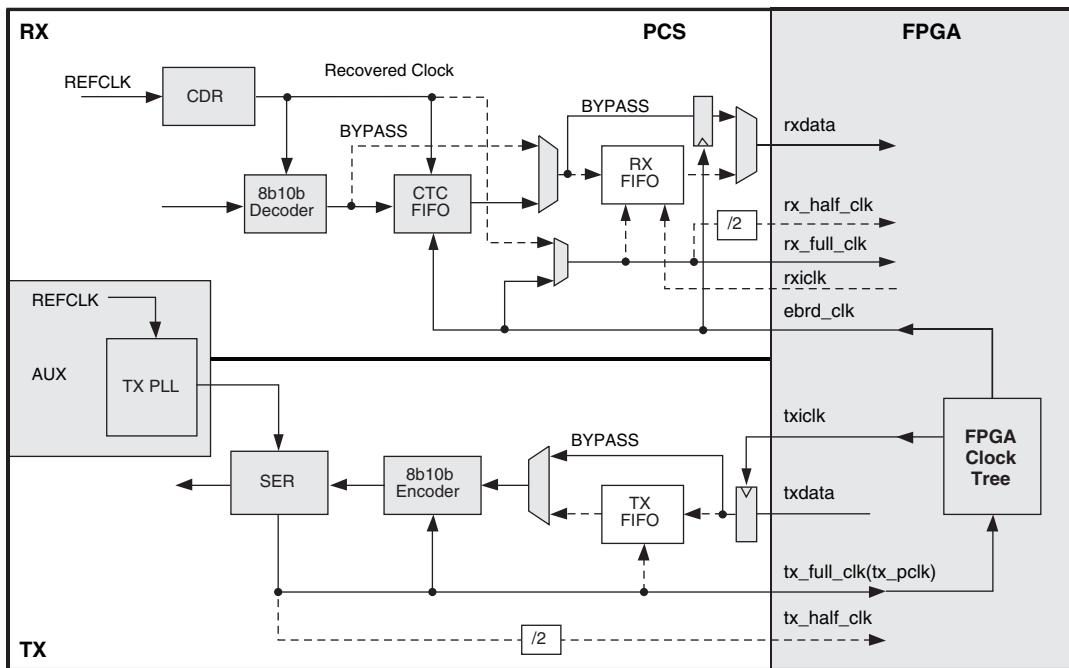
Note that tx\_full\_clk\_ch0 uses wire name 'txclk' and feeds both txi\_clk\_ch0 and rxi\_clk\_ch0, as shown in Figure 9-27.

**Case I\_b: 8 /10bit, CTC FIFO Bypassed**
*Figure 9-28. 8/10-Bit, CTC FIFO Bypassed*


1. The TX FIFO acts as a Phase Shift FIFO only in this case.
2. The RX FIFO acts as a Phase Shift FIFO only in this case.
3. The TX FPGA Channel input clock is clocked similarly as in the previous case using a clock tree driven by a direct connection of the full rate transmit FPGA output clock to the FPGA center clock mux. Once the CTC FIFO is bypassed, the recovered clock needs to control the write port of the RX FIFO. The recovered clock of each channel may need to drive a separate local or global clock tree (i.e., up to four local or global clock trees per quad). The clock tree will then drive the FPGA receive clock input to control the read port of the RX FIFO. The reason for bypassing the CTC FIFO in this case is most likely for doing multi-channel alignment in the FPGA core. It implies that CTC using an elastic buffer will be done in the FPGA core. The CTC FIFOs can be written by either the recovered clocks or by a master recovered clock. The read of the CTC FIFO will be done using the TX clock via the TX clock tree.

### Case I\_c: 8/10bit, FPGA Bridge FIFOs Bypassed

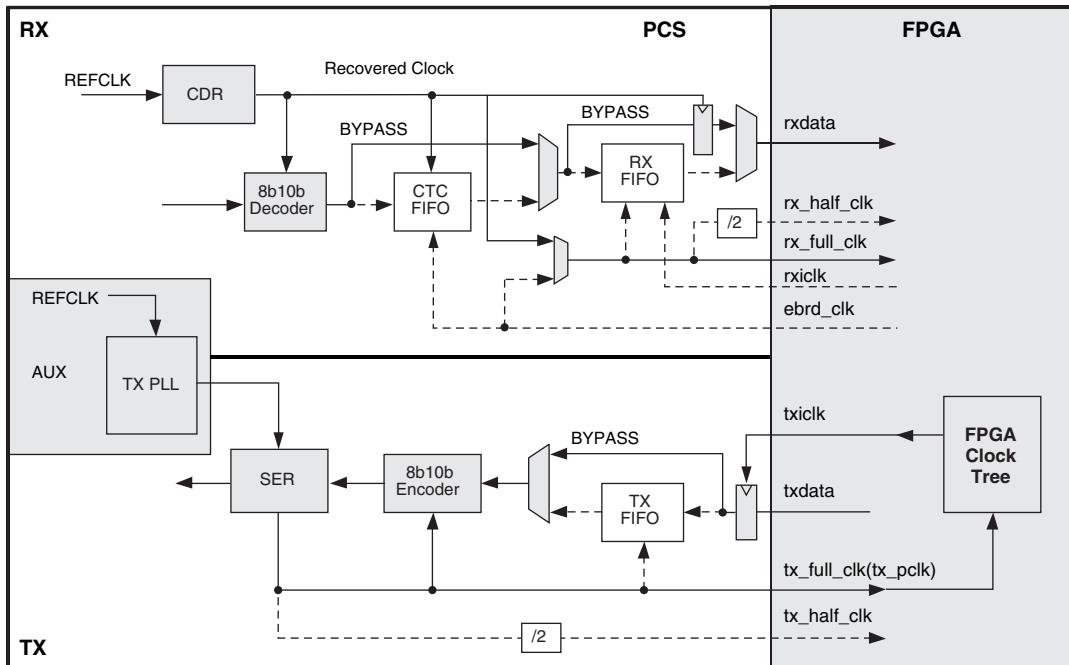
Figure 9-29. 8/10-Bit, RX/TX FIFO Bypassed



1. The TX channel clocking is similar to the previous two cases. On the RX channel, the FPGA input clock is now ebrd\_clk. The FPGA TX clock tree drives this clock. In this case, ebrd\_clk is automatically routed by the software.

### Case I\_d: 8/10bit, CTC FIFO and FPGA Bridge FIFOs Bypassed

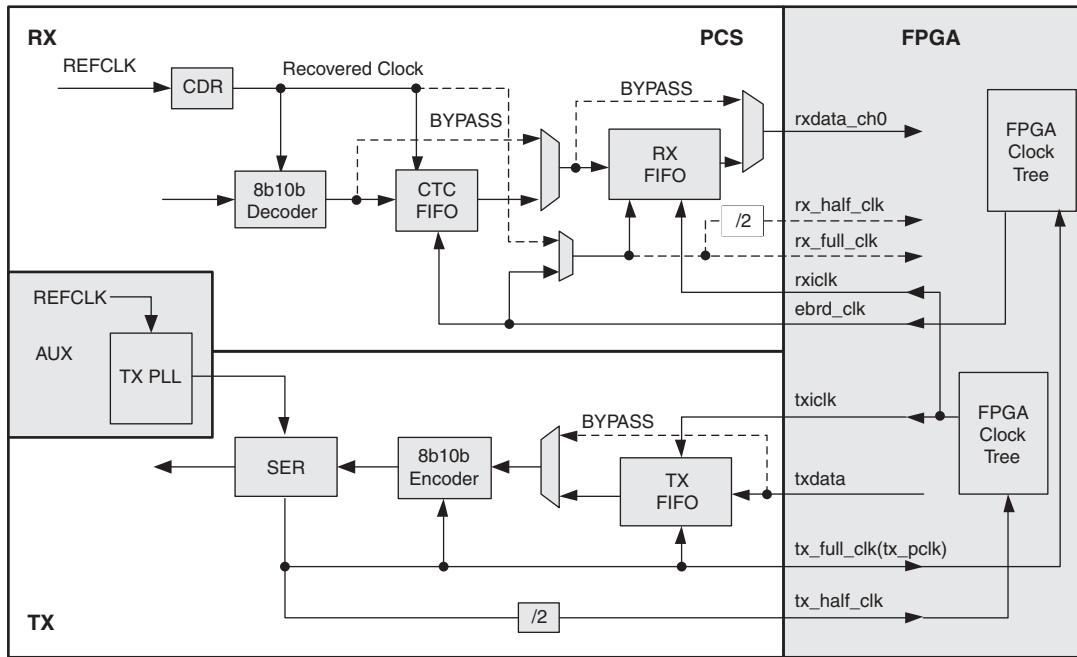
Figure 9-30. 8/10-Bit, CTC FIFO and RX/TX FIFOs Bypassed



1. FPGA clock trees can be interchangeably thought of as clock domains in this case. The TX channel clock-ing is similar to the previous three cases. On the RX channel, the recovered channel RX clock is sent out to the FPGA. This case is useful for supporting video applications.

### Case II\_a: 16/20bit, CTC FIFO NOT Bypassed

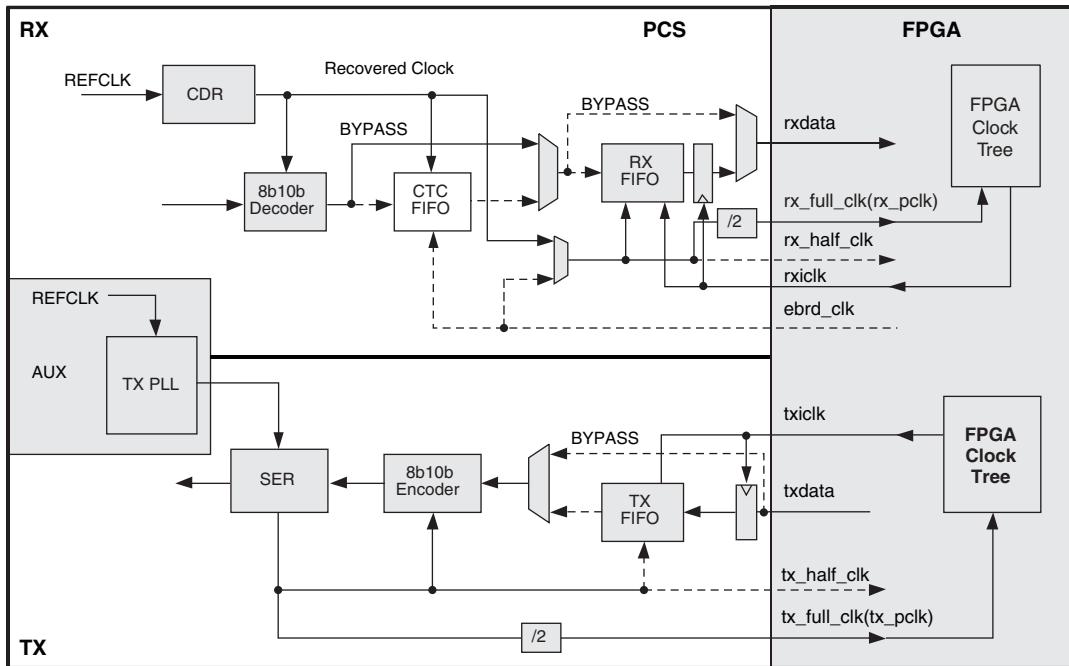
*Figure 9-31. 16/20-Bit, CTC FIFO and RX/TX FIFOs NOT Bypassed*



1. The TX FIFO acts both as a Phase Shift FIFO and Upsample FIFO in this case.
2. The RX FIFO acts both as a Phase Shift FIFO and Downsample FIFO in this case.
3. This is a very common single channel use case when the FPGA is unable to keep up with full byte frequency. Two clock trees are required. These clock trees are driven by direct access of transmit full-rate clock and transmit half-rate clock to the FPGA clock center mux. The full-rate clock tree drives the CTC FIFO read port and the RX FIFO write port. The half-rate clock tree drives the RX FIFO and the FPGA logic.

## Case II\_b: 16/20bit, CTC FIFO Bypassed

Figure 9-32. 16/20-Bit, CTC FIFO Bypassed



1. The TX FIFO acts both as a Phase Shift FIFO and Upsample FIFO in this case.
2. The RX FIFO is acting both as a Phase Shift FIFO and Downsample FIFO in this case.
3. This is a very common multi-channel alignment use case when the FPGA is unable to keep up with full byte frequency. The receive clock trees (up to four) can be local or global. They are running a half-rate clock. The transmit clock tree is driven by direct access of the transmit half-rate clock to the FPGA clock center mux.

## SERDES/PCS Block Latency

Table 9-16 provides transmit and receive latencies, respectively, in the SerDes as well as different stages inside the PCS. The latency is in number of parallel word clocks.

**Table 9-16. Transmit/Receive SerDes/PCS Latency**

Item	Transmit Data Latencies <sup>1</sup>	Description	Min	Typ	Max	Exact	Byp	Unit <sup>4</sup>
T1	FPGA Bridge	Gearing Disabled (with different clocks)	1	3	5	N/A	1	byte_clk
		Gearing Disabled (with same clocks)	N/A	N/A	N/A	3	1	byte_clk
		Gearing Enabled <sup>2</sup>	1	3	5	N/A	N/A	word_clk
T2	8b10b Encoder		N/A	N/A	N/A	2	1	byte_clk
T3	SerDes Bridge		N/A	N/A	N/A	2	1	byte_clk
T4	Serializer	x8 mode (BUS8BIT_SEL = 0)	N/A	N/A	N/A	15 + Δ1	N/A	UI + ps
		x10 mode (BUS8BIT_SEL = 1)	N/A	N/A	N/A	18 + Δ1	N/A	UI + ps
T5	Driver	De-emphasis ON	N/A	N/A	N/A	1 + Δ2	N/A	UI + ps
		Pre-emphasis OFF	N/A	N/A	N/A	0 + Δ3	N/A	UI + ps
	Receive Data Latencies <sup>2</sup>	Description	Min	Typ	Max	Exact	Byp	Units 10
R1	Input Buffer	Equalization ON	N/A	N/A	N/A	Δ1	N/A	UI + ps
		Equalization OFF	N/A	N/A	N/A	Δ2	N/A	UI + ps
R2	De-Serializer	x8 mode (BUS8BIT_SEL = 0)	N/A	N/A	N/A	10 + Δ3	N/A	UI + ps
		x10 mode (BUS8BIT_SEL = 1)	N/A	N/A	N/A	12 + Δ3	N/A	UI + ps
R3	SerDes Bridge		N/A	N/A	N/A	2	1	byte_clk
R4	Word Aligner <sup>3</sup>		N/A	N/A	N/A	4	0	byte_clk
R5	8B10B Decoder		N/A	N/A	N/A	1	1	byte_clk
R6	CTC		7	15	23	N/A	1	byte_clk
R7	FPGA Bridge	Gearing Disabled (with different clocks)	1	3	5	N/A	1	byte_clk
		Gearing Disabled (with same clocks)	N/A	N/A	N/A	3	1	byte_clk
		Gearing Enabled	1	3	5	N/A	N/A	word_clk

1. Δ1 = -100 ps, Δ2 = +88 ps, Δ3 = +112 ps.

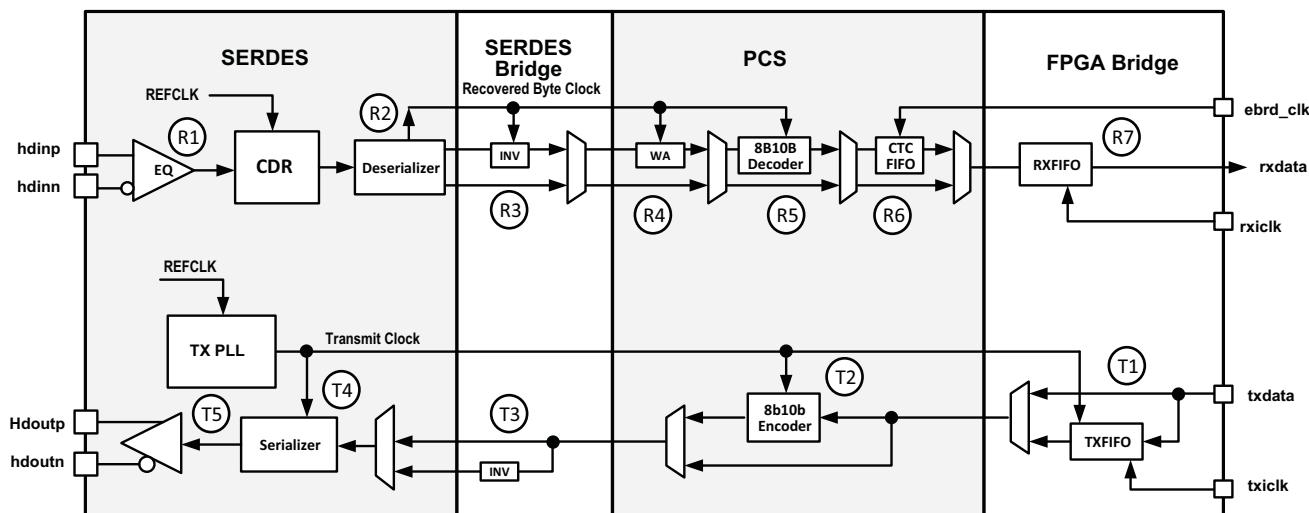
2. Δ1 = +118 ps, Δ2 = +132 ps, Δ3 = +721 ps.

3. Table 9-16 shows word aligner latency depending on word alignment offset. The exact offset can be found in channel status register.

**Table 9-17. Word aligner Latency vs. Offset**

wa_offset	Latency (Word Clocks)
0	4.0
1	3.9
2	3.8
3	3.7
4	3.6
5	3.5
6	3.4
7	3.3
8	3.2
9	3.1

**Figure 9-33. Transmitter and Receiver Latency Block Diagram**



## SERDES Client Interface

### Introduction

LFE5UM provides SCI interface to the core thru general routing.

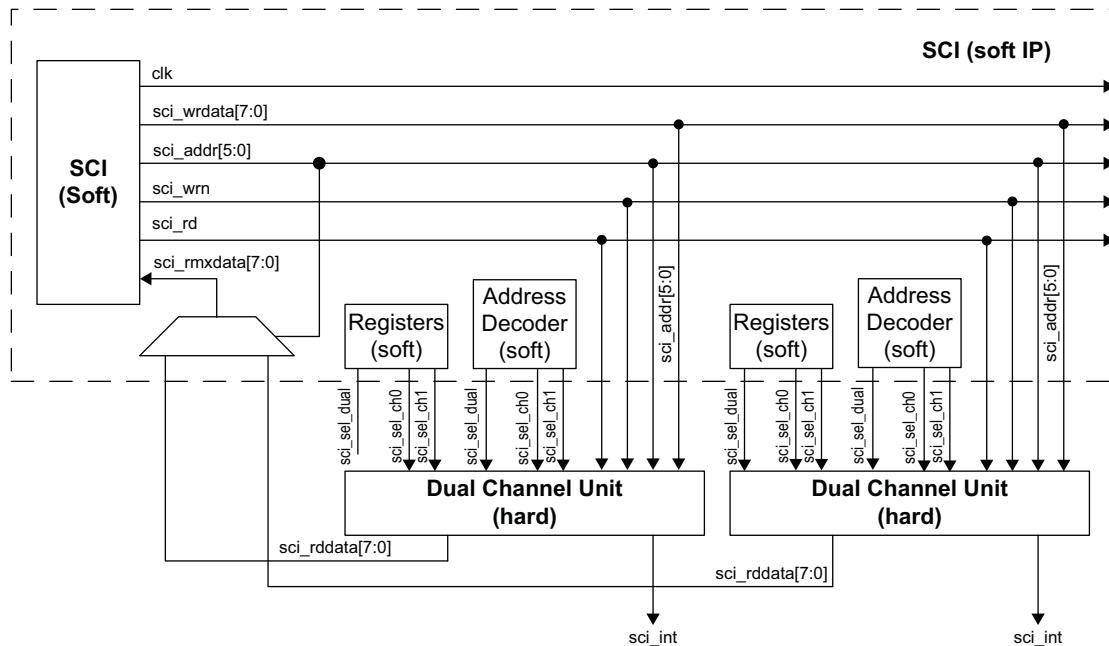
The settings in most of the configuration bits are shadowed into registers that can be read/write by the FPGA core. LFE5UM provides the interface signals that can be controlled by the core. User logic can change these settings via his logic. Care should be taken when the settings are changed, and is recommended and, most of the time, required to perform a reset to the SERDES and PCS, in order to obtain correct operation after the changes.

Table 9-18 describes the addressing map of control registers and Figure 9-34 shows the block diagram of SCI Interface.

Table 9-18. SCI address map for up to four SerDes/PCS duals

Address Bits	Description
SCIADDR[5:0]	<b>Register address bits:</b> 000000 = Select register 0 000001 = Select register 1 ... 111110 = Select register 62 111111 = Select register 63
SCIADDR[8:6]	<b>Channel address bits</b> 000 = Select channel 0 001 = Select channel 1 010 = Reserved 011 = Reserved 100 = Select aux channel 101 = Reserved 110 = Reserved 111 = Reserved
SCIADDR[10:9]	<b>Dual address bits</b> 00 = Select Dual0 01 = Select Dual1 10 = Reserved 11 = Reserved

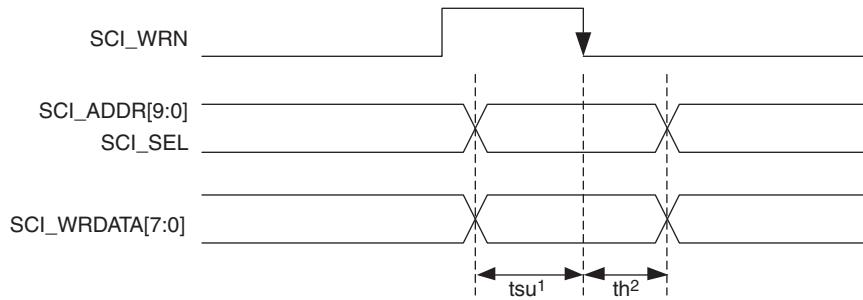
Figure 9-34. LFE5UM SCI(SerDes Client Interface) Block Diagram



The SCI is an 8-bit asynchronous control interface to access the SERDES/PCS channels and TX PLL inside the DCU. The scisel lines are used to select a resource and then the sciwrn or sci\_rd ports are used to latch in the command to write or read 8-bit data from sci\_wrdata or to sci\_rddata respectively. In addition there is a sci\_int port to allow an interrupt to be sent from any of the DCU resources to the FPGA. Refer to Table 9-18 for ports description.

Read and write operations through this interface are asynchronous. In the WRITE cycle the write data and write address must be set up and held in relation to the falling edge of the SCI\_WR. In the READ cycle the timing has to be in relation with the SCI\_RD pulse. Figure 9-35 and Figure 9-36 show the WRITE and READ cycles, respectively.

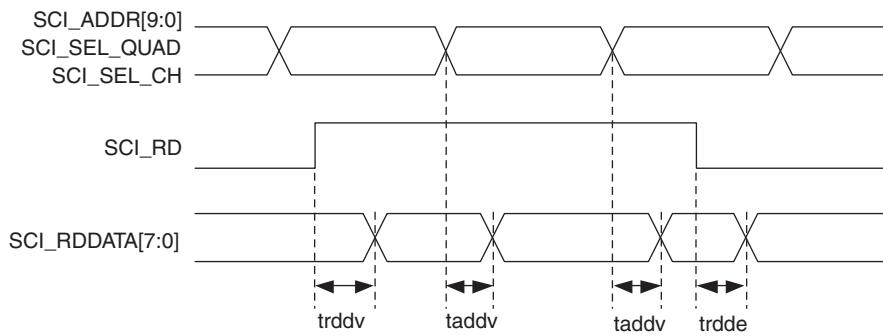
**Figure 9-35. SCI WRITE Cycle, Critical Timing**



1. tsu is the setup time for address and write data prior to the falling edge of the write strobe.
2. th is the hold time for address and write data after the falling edge of the write strobe.

Note: To avoid accidental writing to control registers, registers should be used at the SCI input ports to drive them low at power-up reset.

**Figure 9-36. SCI READ Cycle, Critical Timing**



**Table 9-19. Timing Parameter**

Parameter	Typical Value	Units
$t_{SU}$ , $t_{RDDV}$ , $t_{ADDV}$	1.1	ns
$t_H$ , $t_{RDDE}$	0.9	ns

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

Write:

- Cycle 1: Set sci\_addr[5:0], sciw\_data[7:0], sci\_sel = 1'b1
- Cycle 2: Set sci\_wrn from 0  $\geq$  1
- Cycle 3: Set sci\_wrn from 1  $\geq$  0, sci\_sel = 1'b0

Read:

- Cycle 1: Set sci\_addr[5:0], sci\_sel = 1'b1
- Cycle 2: Set sci\_rd from 0  $\geq$  1
- Cycle 3: Obtain reading data from sci\_rddata[7:0]
- Cycle 4: Set sci\_rd from 1  $\geq$  0

## Interrupts and Status

The status bit may be read via the SCI, which is a byte wide and thus reads the status of eight interrupt status signals at a time. The SCI\_INT signal goes high to indicate that an interrupt event has occurred. The user is then required to read the DIF status register that indicates whether the interrupt came from the quad or one of the channels. This register is not cleared on read. It is cleared when all interrupt sources from the quad or channel are cleared. Once the aggregated source of the interrupt is determined, the user can read the registers in the associated quad or channel to determine the source of the interrupt. Table 9-19 and Table 9-20 list all the sources of interrupt.

**Table 9-20. Dual Interrupt Sources**

Dual SCI_INT Source	Description	Register Name
int_dl_out	Dual Interrupt. If there is an interrupt event anywhere in the dual, this register bit will be active. This register bit is cleared when all interrupt events have been cleared	PCS Dual Status Register DL_20
int_ch_out[1:0]	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 Dual Status Register DL_20
ls_sync_statusn[1:0]_int ls_sync_status[1:0]_int	Link Status Low (out of sync) channel interrupt & Link Status High (in sync) channel interrupt	PCS Dual Status Register DL_22_INT
~PLOL, PLOL	Interrupt generated on ~PLOL and PLOL - PLL Loss of Lock	PCS Dual Status Register DL_25

**Table 9-21. Channel Interrupt Sources**

Channel SCI_INT Source	Description	Register Name
fb_tx_fifo_error_int fb_rx_fifo_error_int cc_overrun_int cc_underrun_int	FPGA Bridge TX FIFO Error Interrupt FPGA Bridge RX FIFO Error Interrupt CTC FIF Overrun and Underrun Interrupts	PCS Channel General Interrupt Status Register CH_INT_23
pci_det_done_int rls_lo_int ~rls_lo_int rl_l_int ~rl_l_int	Interrupt generated for pci_det_done Interrupt generated for rls_lo Interrupt generated for ~rls_lo Interrupt generated for rl_l Interrupt generated for ~rl_l	PCS Dual Status Register DL_INT_20

## Dynamic Configuration of the SERDES/PCS Dual

The SERDES/PCS dual can be controlled by registers that are accessed through the optional SERDES Client Interface.

When controlled by the configuration memory cells, it is a requirement that the SERDES/PCS duals 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 dual must be handled automatically by the hardware. In other words, use of the SCI is optional. The SERDES/PCS dual does NOT assume that the soft IP is present in the FPGA core.

## SERDES Debug Capabilities

### PCS Loopback Modes

The LFE5UM family provides three loopback modes controlled by control signals at the PCS/FPGA interface for convenient testing of the external SERDES/board interface and the internal PCS/FPGA logic interface.

#### RX-to-TX Serial Loopback Mode

This mode loops serial receive data back onto the transmit buffer without passing through the CDR or de-serializer. Select the RX-to-TX Serial Loopback option in the Clarity Designer (System Builder/Planner) GUI and SW will set necessary control bits.

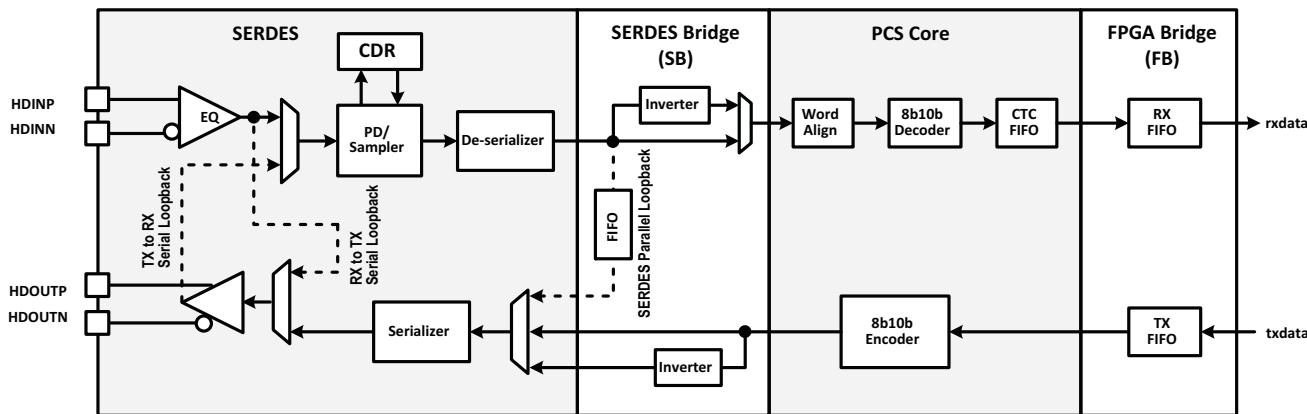
### **TX-to-RX Serial Loopback Mode**

This mode loops back serial transmit data back onto the receiver CDR block. Select the TX-to-RX Serial Loopback option in the Clarity Designer (System Builder/Planner) GUI and SW will set necessary control bits.

### **SERDES Parallel Loopback Mode**

Loops parallel receive data back to the transmit data path without passing through the PCS logic. When disabled in the Clarity Designer (System Builder/Planner) GUI, the parallel loopback mode can be dynamically controlled from the FPGA core control signals sb\_felb\_c and sb\_felb\_rst\_c. If the dynamic feature of this loopback mode is not used, the two control signals should be tied to ground. When the loopback mode is enabled in the Clarity Designer (System Builder/Planner) GUI, the control register bit for the loopback mode is set. The control signals from the FPGA core, sb\_felb\_c and sb\_felb\_rst\_c, are not available in the PCS module.

**Figure 9-37. LFE5UM SERDES Three Loopback Modes**



### **ECO Editor**

The Diamond ECO Editor supports all of the attributes of the primitives. The ECO Editor includes all settings that the user sees in the GUI tab, including pre-defined setting based on protocols.

ECO Editor is a graphical representation of all the user configurable settings, and allows the user to change the settings in that environment. Care must be taken on making these changes, because the changes will not be checked by the design SW tools for correctness.

ECO Editor changes only settings inside the SERDES. Changes in the SERDES that would affect the external connection requires the user to re-run the SW flow. Changes made in ECO Editor that do not change external connection will just require the user to re-generate the bitstream with the same placement and routing done in the core.

Please refer to ECO Editor User Guide in Diamond for more information.

## **Other Design Considerations**

### **Simulation of the SERDES/PCS**

All SERDES/PCS simulation models are located in the installation directory, under ....\cae\_library\simulation\black-box directory

### **16/20-Bit Word Alignment**

The SERDES/PCS recognizes byte boundary by aligning to COMMA characters, but the PCS receiver cannot recognize the 16-bit word boundary. When Word Aligner is enabled, the PCS can only do BYTE alignment. The 16-bit

word alignment should be done in the FPGA fabric and is fairly straight forward. The simulation model works in the same way. It can be enhanced if users implement an alignment scheme as described below.

For example, if transmit data at the FPGA interface are:

```
YZABCDEFGHIJKLM... (each letter is a byte, 8-bit or 10-bit)
```

Then the incoming data in PCS after 8b10b decoder and before rx\_gearbox are:

```
YZABCDEFGHIJKLM...
```

After rx\_gearbox, they can become:

1. ZY}{BA}{DC}{FE}{HG}{JI}{LK}....

or

2. AZ}{CB}{ED}{GF}{IH}{KJ}{ML}...

Clearly, sequence 2 is not aligned. It has one byte offset, but 16/20-bit alignment is needed. Let's say the special character 'A' should be always placed in the lower byte.

character 'A' should be always placed in the lower byte.

Flopping one 20-bit data combines with the current 16/20-bit data to form 32/40-bit data as shown below:

1. {DCBA} {HGFE} {LKJI}...

```
^  
| **Found the A in lower 10-bit, set the offset to '0`, send out aligned  
data 'BA`
```

Next clock cycle:

```
{FEDC} {JIHGF} {NMLK}...  
^  
| **send out aligned data 'DC'
```

etc.

After the 16/20-bit alignment, the output data are:

```
{ZY} {BA} {DC} {FE} {HG} {JI} {LK}...
```

2. {CBAZ} {GFED} {KJIH}....

```
^  
| **Found the A in upper 10-bit, set the offset to '10', send out aligned  
data 'BA'
```

Next clock cycle:

```
{EDCB} {IHGF} {MLKJ}...  
^  
| **send out aligned data 'DC'
```

etc.

After the 20-bit alignment, the output data are:

```
{ZY} {BA} {DC} {FE} {HG} {JI} {LK}...
```

*Note: The LSB of a 8/10-bit byte or a 16/20-bit word is always transmitted first and received first.*

### Unused Dual/Channel and Power Supply

On unused dual and channels, VCCA should be powered up. VCCHRX, VCCHTX, HDINP/N, HDOUTP/N and REF-CLKP/N should be left floating. Unused channel outputs are tristated, with approximately 10 KOhm internal resistor connecting between the differential output pair. During configuration, HDOUTP/N are pulled high to VCCHTX.

Even when a channel is used as Rx Only mode or Tx Only mode, both the VCCHTX and VCCHRX of the channel must be powered up. Unused SERDES is configured in power down mode by default.

## SERDES/PCS Reset

### Reset and Power-Down Control

The SERDES dual has reset and power-down controls for the entire macro and also for each transmitter and receiver as shown in Figure 9-38. The reset signals are active high and the power-down is achieved by driving the pwrup signals low. The operation of the various reset and power-down controls are described in the following sections.

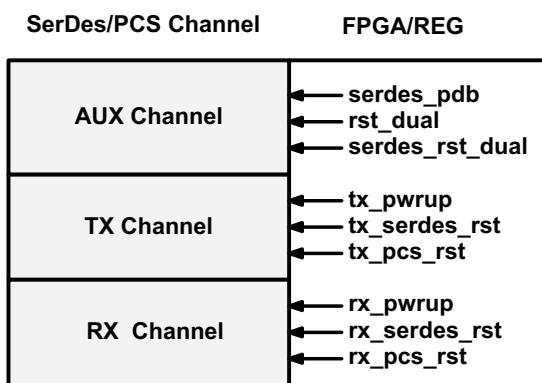
*Note: When the device is powering up and the chip level power-on-reset is active, the SERDES control bits (in the PCS) will be cleared (or will take on their default value). This will put the SERDES quad into the power-down state.*

After configuration is complete, the whole device, including the SerDes/PCS duals, reaches a functional state without further intervention from the user. The reset sequence to initialize the dual is handled by the hardware.

### Reset and Power-Down Control

The SerDes dual has reset and power-down controls for the whole macro as well as individual reset and power-down controls for each transmitter and receiver as shown in Figure 9-38. The reset signals are active high and the power-down signals are active low. The operation of the various reset and power-down controls are described in following sections.

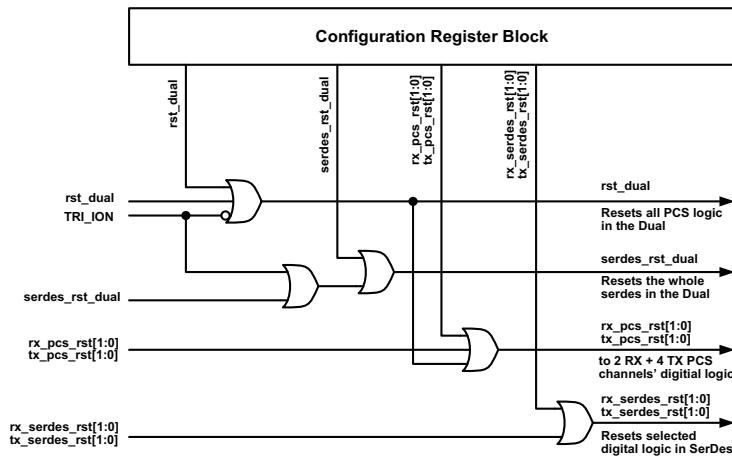
**Figure 9-38. Reset and Power-Down Control**



### Reset Generation

Reset generation and distribution are handled by the clocks and the resets block. The internal reset signal for all blocks within the PCS is active low with an asynchronous assert and a synchronous de-assert. The reset logic is shown in Figure 9-39 and the corresponding table is in Table 9-22.

**Figure 9-39. Global Reset Diagram**



Note in Figure 9-39, each of the reset signal controlled by FPGA logic has a corresponding register bit. These register bits can be accessed thru the SCI interface, and user can write “1” to the register to activate the reset, and write “0” to release it.

Table 9-22 describes the part of SERDES/PCS that are affected by the different resets.

**Table 9-22. Reset Table**

Reset Signal	PCS TX	PCS RX	SERDES TX	SERDES RX	TX PLL	PLOL	RX CDR	CONTROL REGS
tx_pcs_rst[1:0] (fpga)	X							
tx_pcs_rst[1:0] (register)	X							
rx_pcs_rst[1:0] (fpga)		X						
rx_pcs_rst[1:0] (register)		X						
rst_dual (fpga)	X	X						
rst_dual (register)	X	X						
serdes_rst_dual (fpga)			X	X	X	X	X	
serdes_rst_dual (register)			X	X	X	X	X	
rx_serdes_rst[1:0] (fpga)				X			X	
rx_serdes_rst [1:0] (register)				X			X	
tx_serdes_rst (fpga)			X		X	X		
tx_serdes_rst (register)			X		X	X		
(Configuration)	X	X	X	X	X	X	X	X

Table 9-23 describes the power down signals.

**Table 9-23. Power-Down Control Description**

Signal		Description
FPGA	Register	
serdes_pd		Active-low asynchronous input to the SERDES quad, acts on all channels including the auxiliary channel. When driven low, it powers down the whole macro including the transmit PLL. All clocks are stopped and the macro power dissipation is minimized. After release, both the TX and RX reset sequences should be followed.
tx_pwrup_ch[0:3]_c	tpwrup[0:3]	Active-high transmit channel power-up – Powers up the serializer and output driver. After release, the TX reset sequence should be followed.
rx_pwrup_ch[0:3]_c	rpwrup[0:3]	Active-high receive channel power-up – Powers up CDR, input buffer (equalizer and amplifier) and loss-of-signal detector. After release, the RX reset sequence should be followed.

**Table 9-24. Power-Down/Power-Up Timing Specification**

Parameter	Description	Min.	Typ.	Max.	Units
tPWRDN	Power-down time after serdes_pd	20			ns
tPWRUP	Power-up time after serdes_pd	20			ns

## Reset Sequence

There are many different reasons users may want to assert a reset, but there are times when he wants to only reset one channel, or just the Rx or Tx alone. The different resets provide a lot of flexibility for the users. This is especially important considering if the link shows only Rx issue, he does not have to reset the Tx channel, in order to maintain connection to the far end, or visa versa. Also, it is important that when a Dual is connected to two different links, clearing one Tx channel should keep the other channel running. Since two Tx channels in the Dual is sharing the same TxPLL, and if TxPLL is not the source of problem, then the users do not need to reset everything associated with the channel he wants to re-initialize.

However, users have to maintain proper sequencing between PCS and SERDES and sometimes PLL/CDR so that the three blocks are properly synchronized. The proper sequence should involve:

1. PLL needs to be locked first. This goes with TxPLL and CDR PLL. In the case of both are being reset, CDR PLL should wait for TxPLL lock, before it releases the reset on it to start to clock on the CDR.
2. Once the PLLs are locked, the SERDES reset should be released. This properly generates the byte clock that synchronizes to the bit clock inside the SERDES
3. Last to be released is the PCS reset, where all the user logic.

Please contact Lattice Marketing for a sample of reset sequence logic on LFE5UM.

## Clarity Designer (System Builder/Planner) Overview

The Clarity Designer (System Builder/Planner) is an embedded tool in the Diamond Software. With the Clarity Designer tool, the user can specify the full function of the link he wants to implement. This includes SERDES channels, the PCS, the reference clock, and the IP and/or wrapper.

### Diamond Overview

For LFE5UM device family following primitive names are used.

DCU -> DCUA

EXTREF -> EXTREFB

SerDes/PCS HW is composed of three functional blocks, as shown in Figure 9-5. Because the functional block TxPLL has limited sharing, it implemented as part of the SerDes/PCS tab in the GUI.

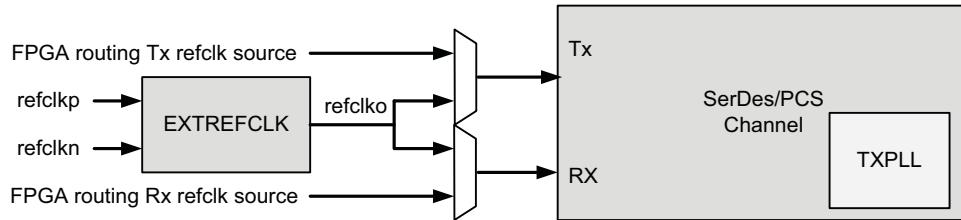
D0CH0: DCU0 Channel0

D0EXTREF: DCU0 EXTREF

## Top Level Block Diagram

This section describes the usage of three blocks, SerDes/PCS channel, TXPLL and EXTREF. A top level diagram in Figure 9-40 shows how the three blocks are integrated.

**Figure 9-40. Three Components Connectivity (for 1-lane link)**



### SerDes/PCS Channel

The SerDes physical channel and its corresponding PCS block are combined into single SerDes/PCS channel block, and multiple channels can be instantiated together as one instance, serving as multi-lane link. When a soft IP is included as part of the instance in the Clarity Designer (System Builder/Planner), Diamond software will route the connection signals from SerDes/PCS to/from the soft IP as defined in the IP.

Refer to Figure 9-5 for the SerDes/PCS channel primitive and Table 9-5 signal description.

### TXPLL

This block is embedded into SerDes/PCS GUI tab but is captured as a separate primitive because the output of the PLL can be shared and used in SerDes/PCS channels outside of its own DCU.

### EXTREF

This is the input buffer block for the reference clock input. PLLs connects to this buffer element when external reference clock source is used.

Single-ended reference clock function is supported, with proper settings of termination resistor and DC bias on the pins. Refer to Lattice technical note, Electrical Recommendations for Lattice SERDES(TN1114) for example circuit.

Connectivity: Captured by SW to provide the implicit connections that user defines. The routing of this is transparent to the users. All users have to do make the connections in the Clarity Designer (System Builder/Planner).

Clock routing: This is a dual based routing block that is captured in the SW. This block connects all the clocking signals in the SerDes/PCS.

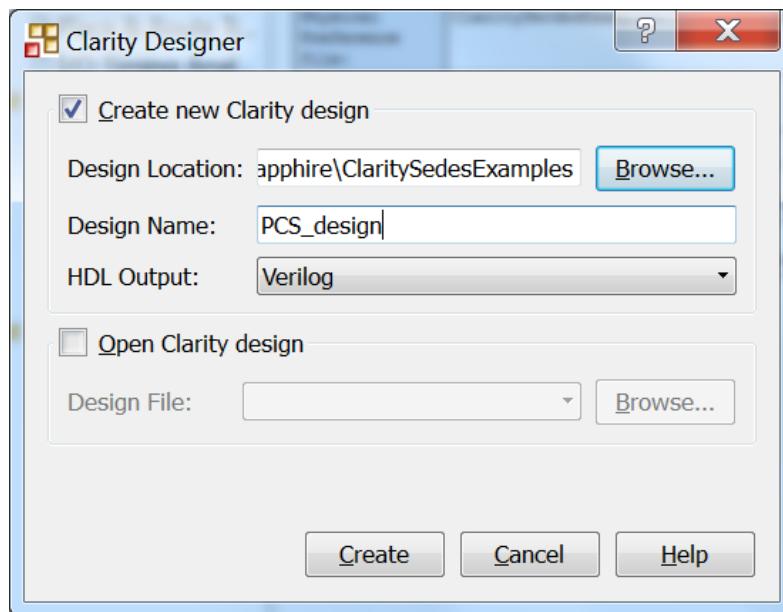
## SERDES/PCS Generation in Clarity Designer (System Builder/Planner)

Clarity Designer (Clarity) is a new tool that targets to provide a more integrated module/function generator environment to the users.

The Clarity allows the user to design the functional block from the top level, pulling in all the necessary modules from the top, then goes into each module to customize what he needs. It is a top-down approach that allows the user looks at his design as a functional block that he can simulate and place in the device. An example of this is an PCIe controller, where the user uses the PCIe IP, the PIPE interface, and the Serdes/PCS blocks.

Once the blocks are identified, the user can push into the GUI of the block, which has similar interface of customizing SERDES on ECP3.

Figure 9-41 shows the view of Clarity when it is opened in Diamond.

**Figure 9-41. Clarity Designer Tool**

The Clarity opens the project in the project directory specifies in the path.

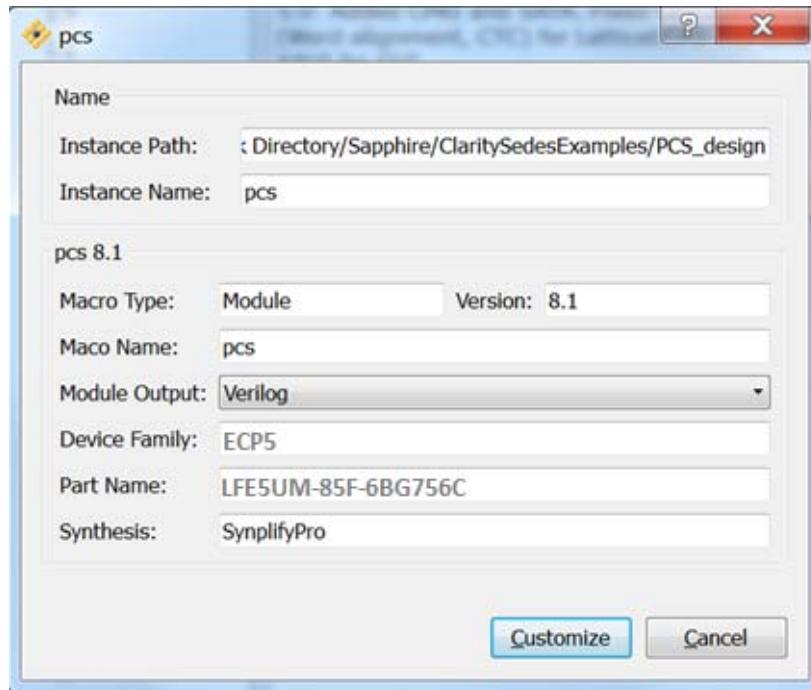
Once the project is created, a list of modules is shown in the catalog. This is shown in Figure 9-42.

Figure 9-42. Clarity Designer Catalog



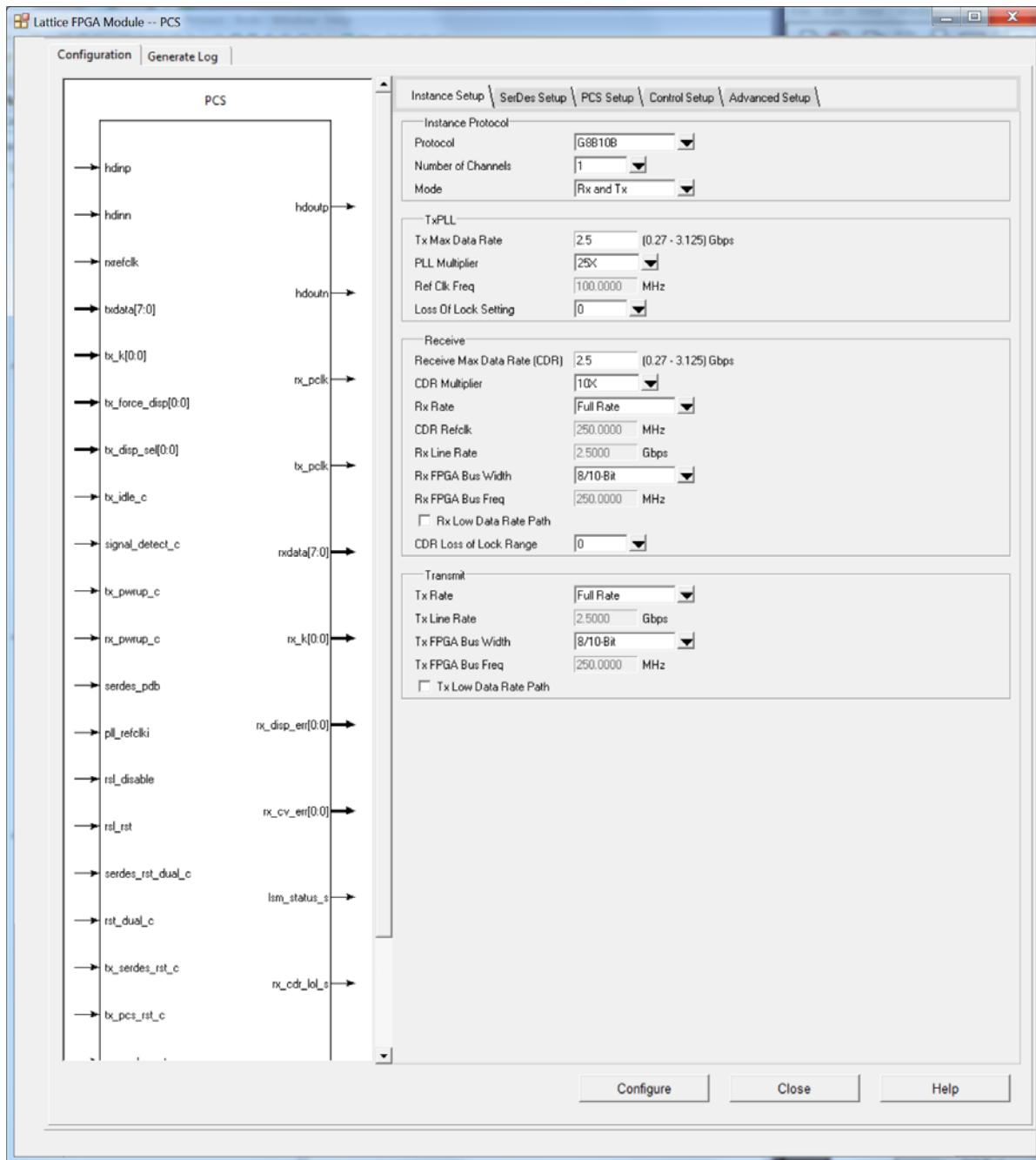
When PCS is selected, a menu opens up to specify the details of the module to be created.

**Figure 9-43. PCS Module in Clarity Designer**



After all the information is filled in, the Custom button brings up the GUI for all the settings of that instance. The first tab is shown in Figure 9-44.

Figure 9-44. Instance Setup Tab



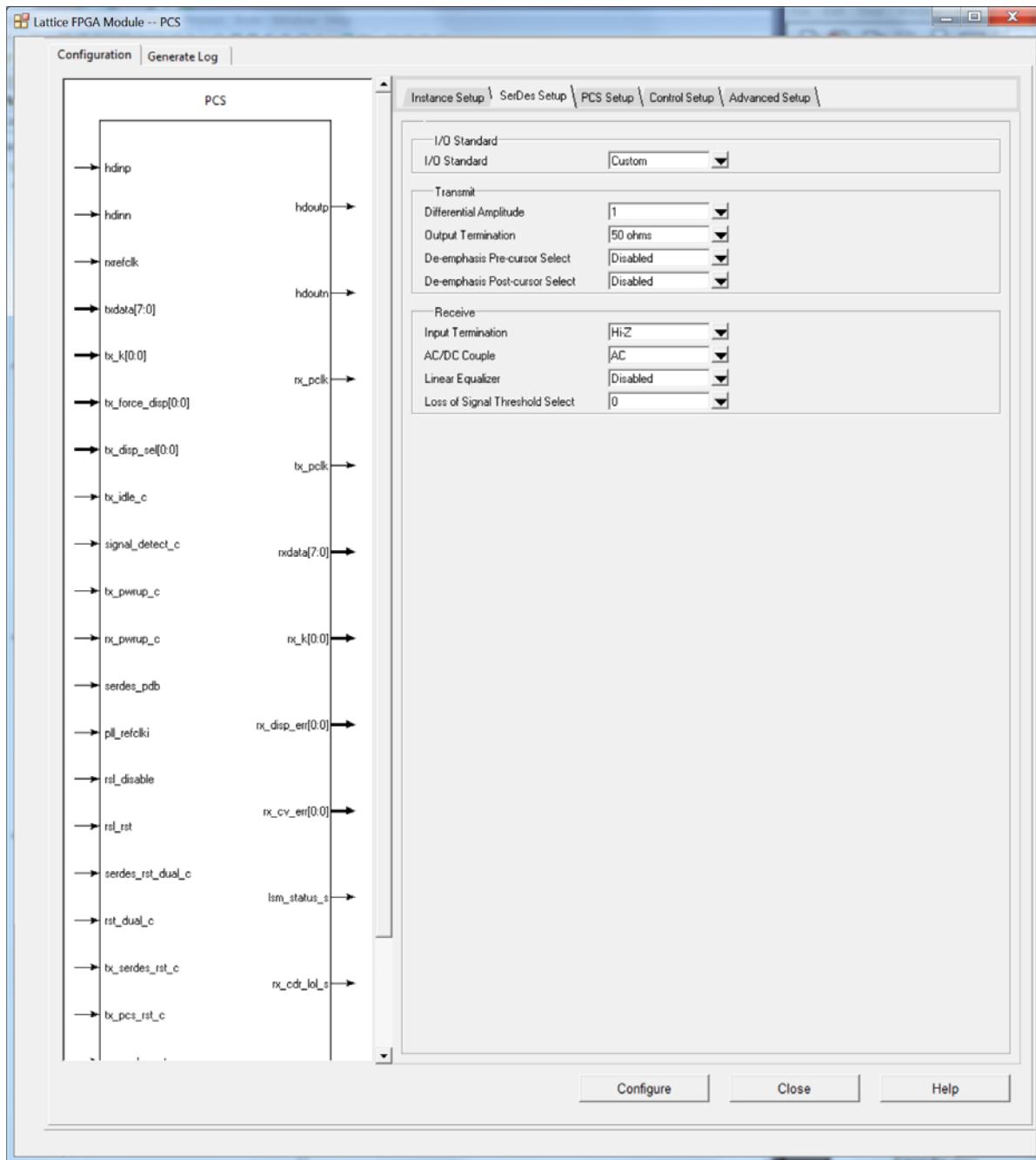
**Table 9-25. Instance Setup Tab Description**

GUI Text	Attribute Names	Values	Default Value
Number of Channels	NUM_CHS	1, 2, 4	1
Protocol	PROTOCOL	See Protocol Table below	G8B10B
Mode	CH_MODE	Rx and Tx, Rx Only, Tx, Only	Rx and Tx
Rx Datapath	RXLDR	On/Off	Off
Tx Datapath	TXLDR	On/Off	Off
Tx Max Data Rate	TX_MAX_RATE	0.27 – 3.125	2.5
PLL Multiplier	TXPLLMULT	8X, 10X, 16X, 20X, 25X	25X
Ref Clk Freq	REFCLK_RATE		100
CDR Loss of Lock Range	CDRLOLRANGE	0-3	0
TxPLL Loss Of Lock Setting	TXPLLLOLTHRESHOLD	0-3	0
Tx Rate Divider	TX_RATE_DIV	Full Rate, Div2 Rate, Div11 Rate	Full Rate
Tx Line Rate	TX_LINE_RATE	135 Mbps – 3.125 Gbps	2.5 Gbps
Receive Max Data Rate (CDR)	CDR_MAX_RATE	0.27 – 3.125	2.5
CDR Refclk	CDR_REF_RATE	27 – 312.5	Calculated
CDR Multiplier	CDR_MULT	8X, 10X, 16X, 20X, 25X	10X
Rx Line Rate	RX_LINE_RATE	135 Mbps – 3.125 Gbps	2.5 Gbps
Tx FPGA Bus Width	TX_DATA_WIDTH	8/10-Bit, 16/20-Bit	8/10- Bit
Tx FPGA Bus Freq	TX_FICLK_RATE	Calculated	Calculated
Rx Rate	RX_RATE_DIV	Full Rate, Div2 Rate, Div11 Rate	Full Rate
Rx FPGA Bus Width	RX_DATA_WIDTH	8/10-Bit, 16/20-Bit	8/10-Bit
Rx FPGA Bus Freq	RX_FICLK_RATE	Calculated	Calculated

Protocols Selectable by the users:

- G8B10B
- PCIe
- GbE
- SGMII
- XAUI
- SDI
- SRIO
- CPRI
- JESD204
- 10BSER
- 8BSER

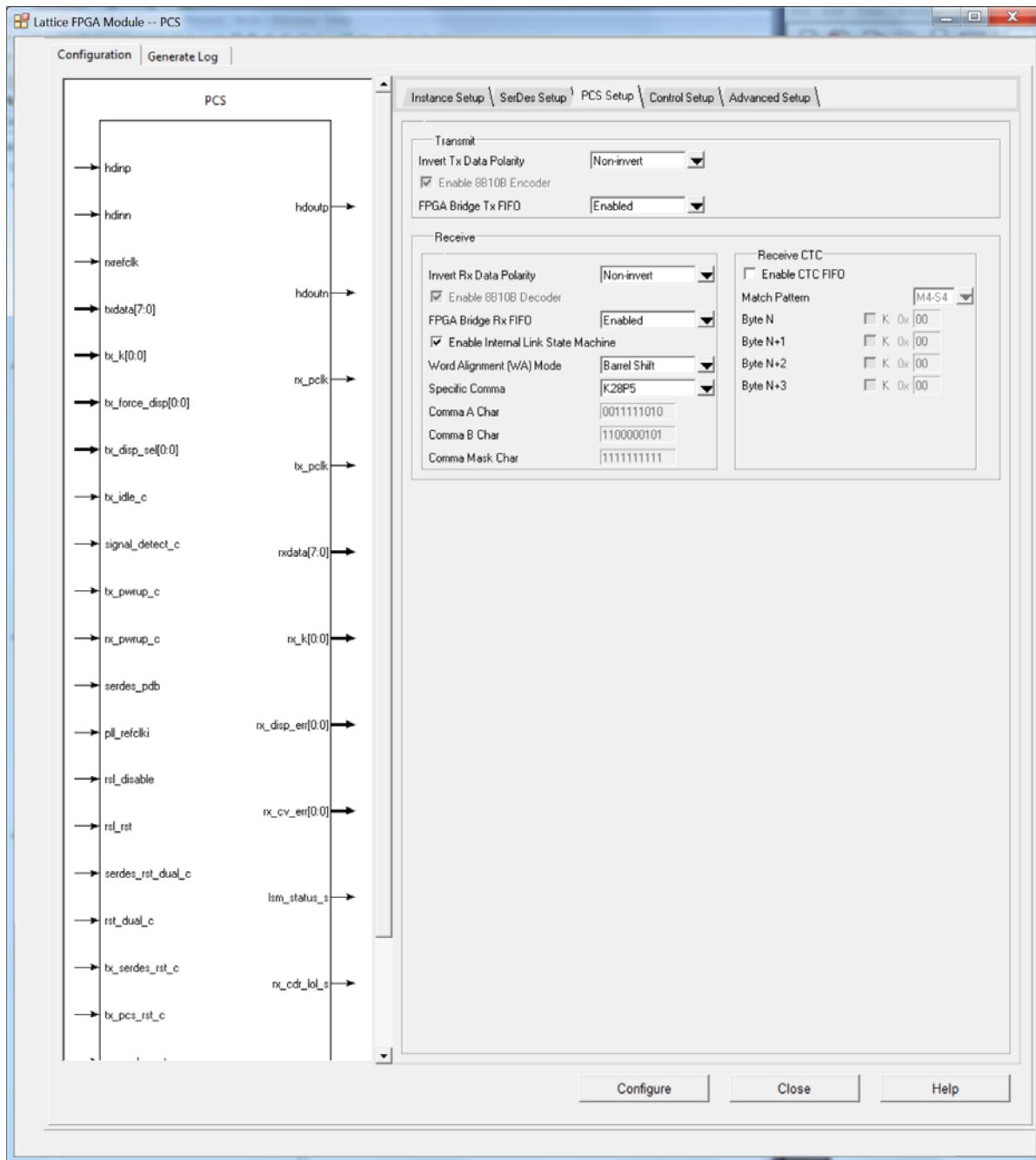
**Figure 9-45. SERDES Setup Tab**



**Table 9-26. SERDES Setup Tab Description**

GUI Text	Attribute Names	Values	Default Value
I/O Standard	IO_TYPE	Protocol dependent, Custom	Based on Protocol
Differential Amplitude	TXAMPLITUDE	Design range 0-9	1
Output Termination	TXDIFFTERM	50, 75, 5K ohms	5K ohms
De-emphasis Pre-cursor Select	TXDEPRE	Disabled, 0-11	Disabled
De-emphasis Post-cursor Select	TXDEPOST	Disabled, 0-11	Disabled
Input Termination	RXDIFFTERM	50, 60, 75 ohms, HiZ	Hi-Z
AC/DC Couple	RXCOUPLING	AC, DC	AC
Linear Equalizer	LEQ	Disabled, 0-3	Disabled
Loss of Signal Threshold Select	RXLOSTTHRESHOLD	0-7	0

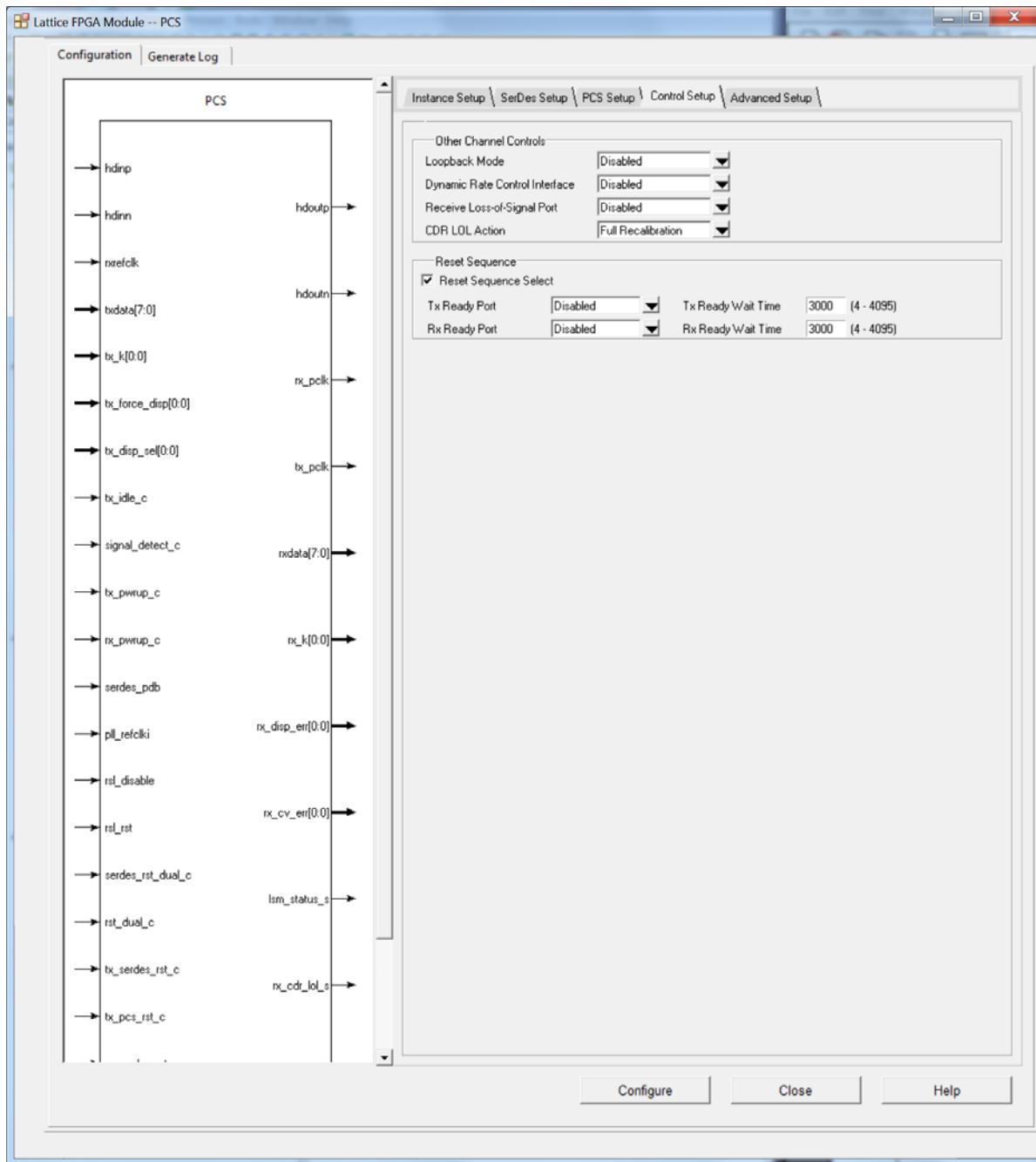
Figure 9-46. PCS Setup Tab



**Table 9-27. PCS Setup Tab Description**

GUI Text	Attribute Names	Values	Default Value
Invert Tx Data Polarity	TXINVPOL	Non-invert, Invert	Non-invert
Enable 8B10B Encoding	TX8B10B	Enabled, Disabled	Enabled
FPGA Bridge Tx FIFO	TXFIFO_ENABLE	Enabled, Disabled	Enabled
Invert Rx Data Polarity	RXINVPOL	Non-invert, Invert, User Port	Non-invert
Enable 8B10B Decoder	RX8B10B	Enabled, Disabled	Enabled
FPGA Bridge Rx FIFO	RXFIFO_ENABLE	Enabled, Disabled	Enabled
Enable Internal Link State Machine	RXLSTM	Enabled, Disabled	Disabled
Word Alignment (WA) Mode	RXWA	Disabled, Barrel Shift, Bit Slip	Disabled
Specific Comma	RXSC	User Defined, K28P5, K28P157	User Defined
Comma A Character	RXCOMMAA	10-bit Binary	1100000101
Comma B Character	RXCOMMAB	10-bit Binary	0011111010
Comma Mask	RXCOMMAM	10-bit Binary	1111111111
Enable CTC FIFO	RXCTC	Enabled, Disabled	Disabled
Match Pattern	RXCTCMATCHPATTERN	M1-S1, M2-S2, M4-S4, M4-S1	M1-S1
Byte N (Kchar, Hex)	RXCTCBYTEN	K, 8-bit Hex	0 00H
Byte N+1 (Kchar, Hex)	RXCTCBYTEN1	K, 8-bit Hex	0 00H
Byte N+2 (Kchar, Hex)	RXCTCBYTEN2	K, 8-bit Hex	0 00H
Byte N+3 (Kchar, Hex)	RXCTCBYTEN3	K, 8-bit Hex	0 00H
Rx Multi-Channel Alignment Enable	RXMCAENABLE	Disabled, Enabled	Disabled
Alignment Character A	ACHARA	K, 8-bit Hex	0 00H
Alignment Character B	CHARB	K, 8-bit Hex	0 00H
Alignment Character Mask	CHARM	K, 8-bit Hex	0 00H

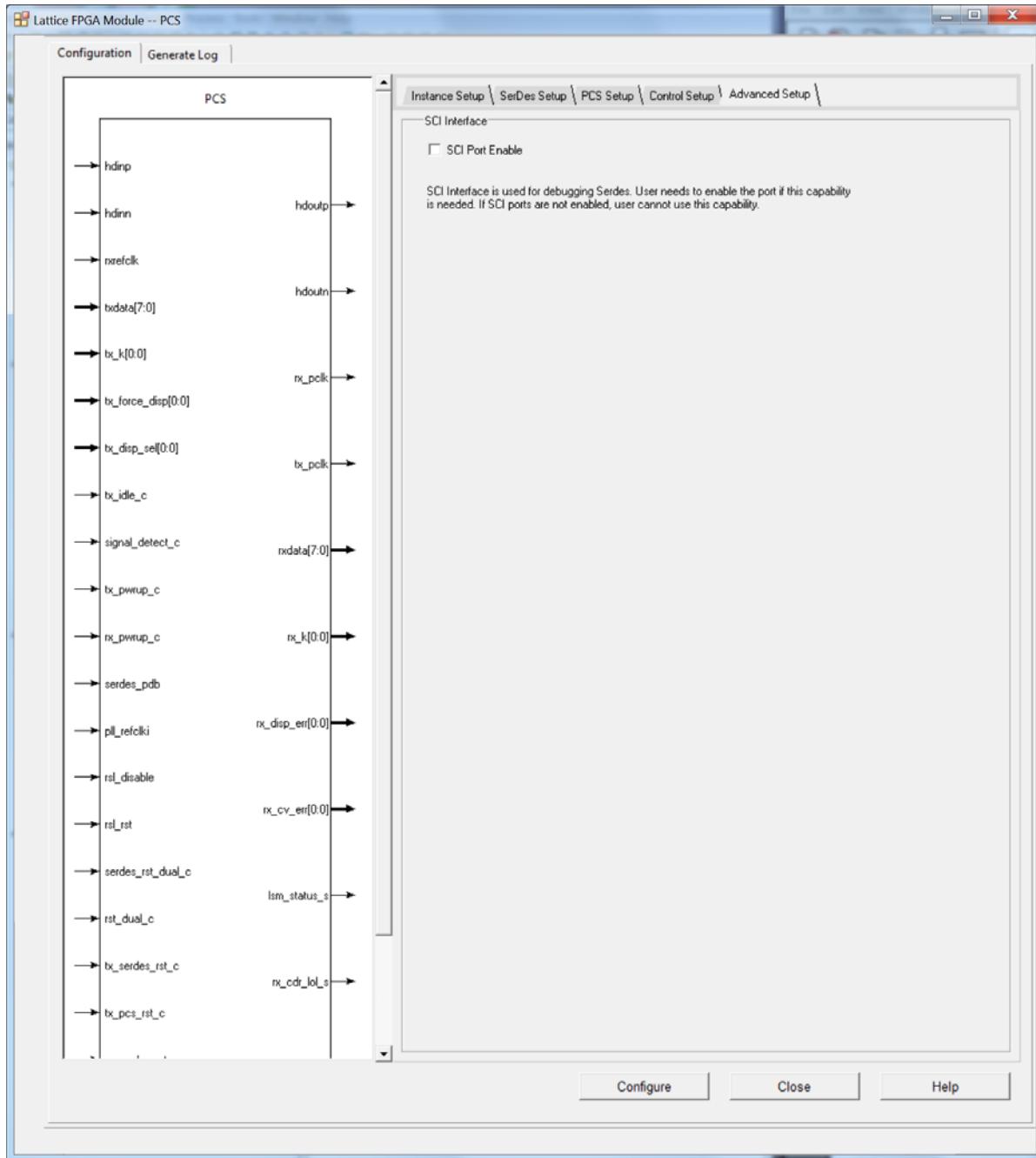
**Figure 9-47. Control Setup Tab**



**Table 9-28. Control Setup Tab Description**

GUI Text	Attribute Names	Values	Default Value
Loopback Mode	LOOPBACK	Disabled, Rx EQ to Tx, Rx Parallel to Tx, Tx Output to Rx	Disabled
Dynamic Rate Control Interface	RCSRC	Disabled, Enabled	Disabled
Receive Loss-of-Signal Port	LOSPORT	Disabled, Enabled	Disabled
CDR LOL Action	CDRLOACTION	"Nothing", "Full Recalibration", "Simple Recalibration"	Full Recalibration
Reset Sequence Select	RSTSEQSEL	None, "Tx Before Rx", "Tx/Rx Independent"	Tx Before Rx

**Figure 9-48. Advanced Setup Tab**



**Table 9-29. Advanced Setup Tab Description**

GUI Text	Attribute Names	Values	Default Value
SCI Port Enable	SCI	Enabled, Disabled	Disabled

## **EXTREF Block**

The EXTREF is the reference clock input buffer primitive for the dedicated external clock inputs to the SerDes TxPLL.

The EXTREF primitive allows the users to select how the reference clock input buffer is configured. There is one EXTREF block in each DCU.

**Figure 9-49. EXTREF Block Diagram**



The EXTREF module takes the differential external reference clock pins (refclkp/n) and sends out a single-ended clock signal to the SerDes TxPLL. The EXTREF can only connect to SerDes TxPLL in the same DCU, or to SerDes TxPLL in the complementary sharing DCU.

### **EXTREF I/O Port Description**

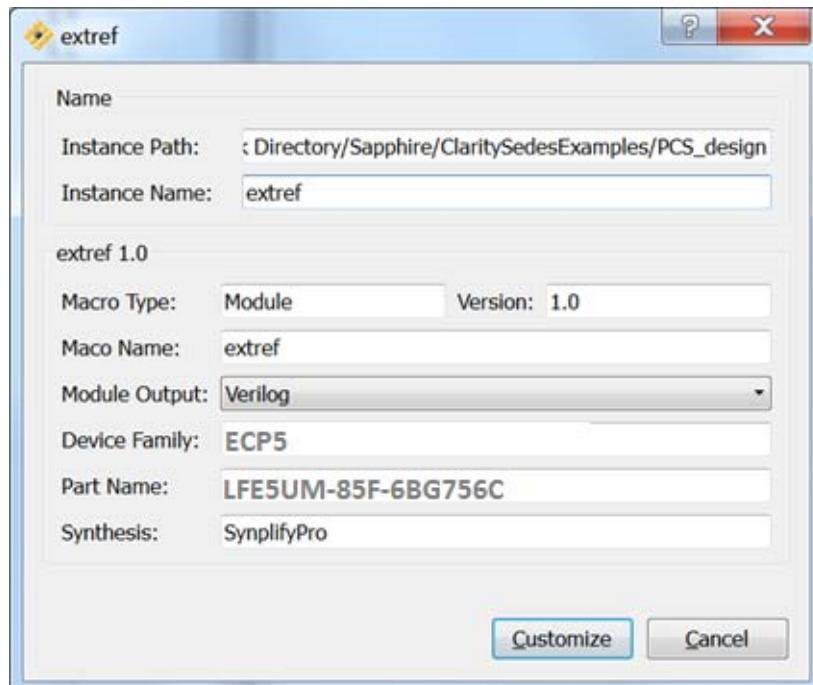
refclkp – External reference clock positive input port

refclkn – External reference clock negative input port

refclko - Single-ended clock signal to be connected to PCS PLL inputs

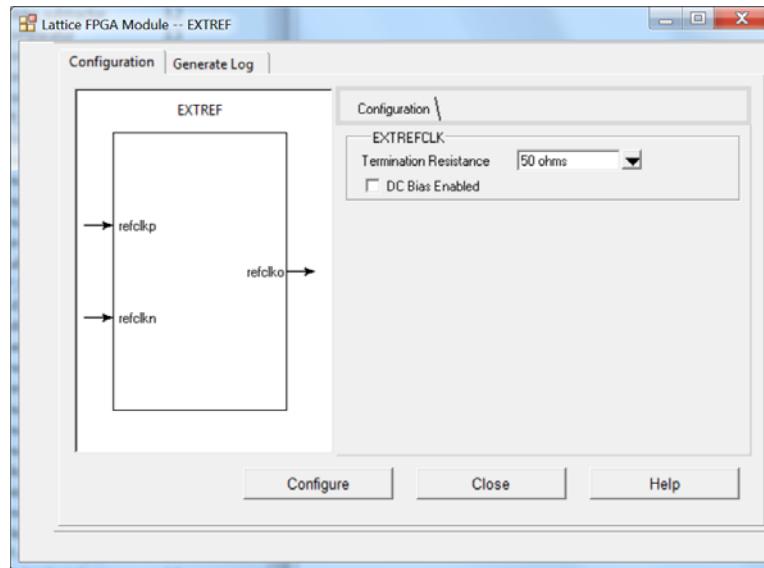
When the REFCLK module is selected, the EXTREF instance table will be brought up

**Figure 9-50. EXTREF Module In Clarity Designer**



The GUI tab and description of the External RERCLK primitive is show below:

**Figure 9-51. EXTREF Tab**



**Table 9-30. EXTREF Tab Description**

GUI Text	Attribute Names	Values	Default Value
Termination Resistance	EXTREFTERMRES	50 ohms, Hi-Z	50 ohms
DC Bias Enabled	EXTREFDCBIAS	Disabled, Enabled	Disabled

## References

- TN1033, [High-Speed PCB Design Considerations](#)
- TN1114, [Electrical Recommendations for Lattice SERDES](#)
- DS1044, [ECP5 Family Data Sheet](#)
- HB1009, [LatticeECP3 Family Handbook](#)
- DS1021, [LatticeECP3 Family Data Sheet](#)
- TN1176, [LatticeECP3 SERDES/PCS Usage Guide](#)
- Clarity Designer Manual
- ECO Editor Manual

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
March 2014	01.0	Initial release.

## Appendix A. Configuration Registers

There are specific dual-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)

The following abbreviations are used to indicate what type of register access for each is supported:

- R/W = Read/Write
- RO = Read Only

### Dual Registers Overview

**Table 9-31. Dual Interface Registers Map**

BA = Base Address (Hex)

BA	Register name	D7	D6	D5	D4	D3	D2	D1	D0
PER DUAL PCS CONTROL REGISTERS (10)									
00	DL_00	reg_sync_toggle	force_int	char_mode	xge_mode	Spare	Spare	Spare	Spare
01	DL_01	Internal use only							
02	DL_02	high_mark[3]	high_mark[2]	high_mark[1]	high_mark[0]	low_mark[3]	low_mark[2]	low_mark[1]	low_mark[0]
03	DL_03	Reserved	Reserved	Reserved	Reserved	Reserved	pfifo_clr_sel	Internal use only	Internal use only
04	DL_04	Internal use only							
05	DL_05	Internal use only							
06	DL_06	Internal use only							
07	DL_07	Internal use only							
08	DL_08	Internal use only							
09	DL_09	Internal use only	Internal use only	ls_sync_status_1_int_c tl	ls_sync_status_0_int_c tl	Reserved	Reserved	ls_sync_statusn_1_int_ ctl	ls_sync_statusn_0_int_ ctl
PER DUAL SERDES CONTROL REGISTERS (6)									
0A	DL_0A	Internal use only	Reserved	tx_refck_sel	refck_dcbias_en	refck_rterm	Reserved	refck_out_sel[1]	Internal use only
0B	DL_0B	refck25x	bus8bit_sel	Reserved	refck_from_nd_sel[1]	refck_from_nd_sel[0]	refck_to_nd_en	refck_mode[1]	refck_mode[0]
0C	DL_0C	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	cdr_lol_set[1]	cdr_lol_set[0]
0D	DL_0D	rg_set[1]	rg_set[0]	rg_en	pll_lol_set[1]	pll_lol_set[0]	Internal use only	Internal use only	Internal use only
0E	DL_0E	Internal use only							
0F	DL_0F	plo_int_ctl	-plo_int_ctl	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
PER DUAL CLOCK RESET REGISTERS (2)									
10	DL_10	Reserved	Reserved	txpll_pwdnb	refck_pwdnb	macropdb	macro_rst	dual_rst	trst
11	DL_11	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
12	DL_12	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
13	DL_13	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
PER DUAL SERDES CONFIGURATION REGISTERS									
15	DL_15	Internal use only							
16	DL_16	Internal use only							
17	DL_17	Internal use only							
18	DL_18	Internal use only							
19	DL_19	Internal use only							
1A	DL_1A	Internal use only							
1B	DL_1B	Internal use only							
1C	DL_1C	Internal use only							
1D	DL_1D	Internal use only							
PER DUAL PCS STATUS REGISTERS (5)									
20	DL_20	Reserved	Reserved		int_dua_out	Reserved	Reserved	int_cha_out[1]	int_cha_out[0]

21	DL_21	Reserved	Reserved	ls_sync_status_1	ls_sync_status_0	Reserved	Reserved	ls_sync_statusn_1	ls_sync_statusn_0
22	DL_22	Reserved	Reserved	ls_sync_status_1_int	ls_sync_status_0_int	Reserved	Reserved	ls_sync_statusn_1_int	ls_sync_statusn_0_int
23	DL_23	Internal use only							
24	DL_24	Internal use only							
PER DUAL SERDES STATUS REGISTERS (4)									
25	DL_25	p1ol	-p1ol	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
26	DL_26	p1ol_int	-p1ol_int	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
27	DL_27	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
28	DL_28	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

## Per Dual PCS Control Register Details

**Table 9-32. PCS Control Register 1 (DL\_00)**

Bit	Name	Description	Type	Default
3:0	Reserved		R/W	0
4	xge_mode	1 = Selects 10Gb Ethernet (selects a different 10G XAUI LSM and some slight differences in 8B10B encoding behavior relative to GE mode). 0 = Depends on Channel Mode Selection	R/W	
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	reg_sync_toggle	Transition = Reset the four Tx Serializers to minimize Tx Lane-to-Lane Skew Level = Normal operation of Tx Serializers	R/W	0

**Table 9-33. PCS Control Register 3 (DL\_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'b1011
7:4	high_mark [3:0]	Clock compensation FIFO high water mark. Mean is 4'b1000	R/W	4'b1011

**Table 9-34. PCS Control Register 4 (DL\_03)**

Bit	Name	Description	Type	Default
0		Internal use only		
1		Internal use only		
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
7:3	Reserved		R/W	0

**Table 9-35. PCS Interrupt Control Register 10 (DL\_09)**

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
3:2	Reserved			
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
7:6	Internal use only			

## Per Dual SerDes Control Register Details

**Table 9-36. SERDES Control Register 1 (DL\_0A)**

Bit	Name	Description	Type	Default
1:0	REFCK_OUT_SEL [1:0]	Refck outputs enable/disable control[0]: 0 = rx_refck_local output disable 1 = rx_refck_local output enable  [1]: 0 = refck2core output disable 1 = refck2core output enable	R/W	2'b00
2	Reserved		R/W	0
3	REFCK_RTERM	Termination at reference clock input buffer 1 = 100 ohm 0 = High Impedance (default)	R/W	
4	REFCK_DCBIAS_EN	1 = Reference clock internal dc bias enabled	R/W	
		0		
5	TX_REFCK_SEL	TxPLL reference clock select		
		1 = ck_core_tx 0 = rx_refck_local	R/W	0
6	Reserved		R/W	0
7	Internal use only			

**Table 9-37. SERDES Control Register 2 (DL\_0B)**

Bit	Name	Description	Type	Default
1:0	REFCK_MODE[1:0]	If REFCK25X=0, then 00 = Internal high speed bit clock is 20x 01 = Internal high speed bit clock is 10x 10 = Internal high speed bit clock is 16x 11 = Internal high speed bit clock is 8x If REFCK25X=1, then xx = Internal high speed bit clock is 25x	R/W	2'b00
2	REFCK_TO_ND_EN	1 = Enable REFCLK to Neighboring Dual	R/W	0
3	REFCK_FROM_ND_SEL[0]	1 = Select TX REFCLK from Neighboring Dual	R/W	0
4	REFCK_FROM_ND_SEL[1]	1 = Select RX REFCLK from Neighboring Dual	R/W	0
5	Reserved			
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 REFCLK = 100 MHz only) 0 = see REFCK_MODE	R/W	0

**Table 9-38. SERDES Control Register 3 - (DL\_0C)**

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
7:2	Reserved			

**Table 9-39. SERDES Control Register 4 (DL\_0D)**

Bit	Name	Description	R/W	Default
2:0	TX_VCO_CK_DIV [2:0]	VCO output frequency select:  00x = Divided by 1      01x = Divided by 2 100 = Divided by 4      101 = Divided by 8 110 = Divided by 16      111 = Divided by 32	R/W	3'b000
4:3	PLL_LOL_SET [1:0]	TxPLL loss of lock setting:  Lock 00 = +/- 300 ppm x2 01 = +/- 300 ppm 10 = +/- 1500 ppm 11 = +/- 4000 ppm  Unlock: = +/- 600 ppm x2 = +/- 2000 ppm = +/- 2200 ppm = +/- 6000 ppm	R/W	2'b00
5	Internal use only			
7:6	Internal use only			

**Table 9-40. SERDES Interrupt Control Register 6 (DL\_0F)**

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.	R/W	0
7	PLOL_INT_CTL	1 = Interrupt enabled for loss of lock on PLOL. 0 = Interrupt not enabled for loss of lock PLOL.	R/W	0

## Per Dual Reset and Clock Control Register Details

**Table 9-41. Reset and Clock Control Register 1 (DL\_10)**

Bit	Name	Description	R/W	Default
0	trst	1 = Reset TxPLL Loss of Lock	R/W	0
1	dual_RST	1 = Assert dual reset	R/W	0
2	macro_RST	1 = Assert macro reset	R/W	0
3	macropdb	0 = Assert power down	R/W	1
4	refck_pwdnb	Reference clock power down control		
		0 = Power down 1 = Power up	R/W	0
5	txpll_pwdnb	TXPLL power down control		
		0 = Power down 1 = Power up	R/W	0
6	Reserved			
7	Reserved			

**Table 9-42. Reset and Clock Control Register 2 (DL\_11)**

Bit	Name	Description	R/W	Default
7:0	Reserved			

**Table 9-43. Serdes Control Register 6 (DL\_12)**

Bit	Name	Description	R/W	Default
7:0	Reserved			

**Table 9-44. Serdes Control Register 7 (DL\_13)**

Bit	Name	Description	R/W	Default
7:0	Reserved			

## Per Dual PCS Status Register Details

**Table 9-45. PCS Status Register 1 (DL\_20)**

Bit	Name	Description	R/W	Int
1:0	int_cha_out [1:0]	Per Channel Interrupt status	RO	N
3:2	Reserved			
4	int_dua_out	Per Dual Interrupt status	RO	N
5	Spare	Delayed global resetn from tri_ion	RO	N
7:6	Reserved			

**Table 9-46. PCS Status Register 2 (DL\_21)**

Bit	Name	Description	R/W	Int
0	ls_sync_statusn_0	1 = Alarm generated on sync_status_0 when it goes low (out of sync) 0 = Alarm not generated on sync_status_0 when it goes low (out of sync)	RO	Y
1	ls_sync_statusn_1	1 = Alarm generated on sync_status_1 when it goes low (out of sync) 0 = Alarm not generated on sync_status_1 when it goes low (out of sync)	RO	Y
3:2	Reserved			
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
7:6	Reserved			

**Table 9-47. PCS Packet Interrupt Status Register 3 (DL\_22)**

Bit	Name	Description	R/W	Int
0	ls_sync_statusn_0_int	1 = Interrupt generated on sync_status_0 when it goes low (out of sync) 0 = Interrupt not generated on sync_status_0 when it goes low (out of sync)	RO CR	Y
1	ls_sync_statusn_1_int	1 = Interrupt generated on sync_status_1 when it goes low (out of sync) 0 = Interrupt not generated on sync_status_1 when it goes low (out of sync)	RO CR	Y
3:2	Reserved			
4	ls_sync_status_0_int	1 = Interrupt generated on sync_status_3 (in sync) 0 = Interrupt not generated on sync_status_3 (in sync)	RO CR	Y
5	ls_sync_status_1_int	1 = Interrupt generated on sync_status_2 (in sync) 0 = Interrupt not generated on sync_status_2 (in sync)	RO CR	Y
7:6	Reserved			

## Per Dual SerDes Status Register Details

**Table 9-48. SERDES Status Register 1 (DL\_25)**

Bit	Name	Description	R/W	Int
5:0	Reserved			
6	PLOL_STSB	1 = PLL lock obtained	RO	Y
7	PLOL_STS	1 = PLL Loss of lock	RO	Y

**Table 9-49. SERDES Interrupt Status Register 2 (DL\_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

## Per Channel Register Overview

**Table 9-50. Channel Interface Registers Map**

BA = Base Address (Hex)

BA	Register name	D7	D6	D5	D4	D3	D2	D1	D0
PER CHANNEL GERNERAL REGISTERS (16)									
00	CH_00	Spare	Spare	Spare	wa_mode	rio_mode	pcie_mode	Spare	uc_mode
01	CH_01	enable_cg_align	prbs_enable	Internal use only	ge_an_enable	Spare	Internal use only	invert_tx	invert_rx
02	CH_02	pfifo_clr	pcie_el_en	pcs_det_time_sel[1]	pcs_det_time_sel[0]	rx_gear_mode	tx_gear_mode	Reserved	Reserved
03	CH_03	Spare	sb_bypass	Internal use only	sb_bist_sel	Internal use only	sel_bist_txd4enc	tx_gear_bypass	fb_loopback
04	CH_04	lsm_disable	signal_detect	rx_gear_bypass	ctc_bypass	dec_bypass	wa_bypass	rx_sb_bypass	sb_loopback
05	CH_05	min_ipg_cnt[1]	min_ipg_cnt[0]	match_4_enable	match_2_enable	Spare	Spare	Spare	Spare
06	CH_06	cc_match_1[7]	cc_match_1[6]	cc_match_1[5]	cc_match_1[4]	cc_match_1[3]	cc_match_1[2]	cc_match_1[1]	cc_match_1[0]
07	CH_07	cc_match_2[7]	cc_match_2[6]	cc_match_2[5]	cc_match_2[4]	cc_match_2[3]	cc_match_2[2]	cc_match_2[1]	cc_match_2[0]
08	CH_08	cc_match_3[7]	cc_match_3[6]	cc_match_3[5]	cc_match_3[4]	cc_match_3[3]	cc_match_3[2]	cc_match_3[1]	cc_match_3[0]
09	CH_09	cc_match_4[7]	cc_match_4[6]	cc_match_4[5]	cc_match_4[4]	cc_match_4[3]	cc_match_4[2]	cc_match_4[1]	cc_match_4[0]
0A	CH_0A	cc_match_4[9]	cc_match_4[8]	cc_match_3[9]	cc_match_3[8]	cc_match_2[9]	cc_match_2[8]	cc_match_1[9]	cc_match_1[8]
0B	CH_0B	udf_comma_mask[7]	udf_comma_mask[6]	udf_comma_mask[5]	udf_comma_mask[4]	udf_comma_mask[3]	udf_comma_mask[2]	udf_comma_mask[1]	udf_comma_mask[0]
0C	CH_0C	udf_comma_a[7]	udf_comma_a[6]	udf_comma_a[5]	udf_comma_a[4]	udf_comma_a[3]	udf_comma_a[2]	udf_comma_a[1]	udf_comma_a[0]
0D	CH_0D	udf_comma_b[7]	udf_comma_b[6]	udf_comma_b[5]	udf_comma_b[4]	udf_comma_b[3]	udf_comma_b[2]	udf_comma_b[1]	udf_comma_b[0]
0E	CH_0E	udf_comma_a[9]	udf_comma_a[8]	udf_comma_b[9]	udf_comma_b[8]	udf_comma_mask[9]	udf_comma_mask[8]		
0F	CH_0F	Reserved	Reserved	Reserved	Reserved	cc_underrun_int_ctl	cc_overrun_int_ctl	fb_rx_fifo_error_int_ctl	fb_tx_fifo_error_int_ctl
PER CHANNEL SERDES CONTROL REGISTERS (15)									
10	CH_10	tx_post_sign	tx_pre_sign	tdrv_post_en	tdrv_pre_en	ldr_core2tx_sel	tx_div11_sel	rate_mode_tx	tpwdnb
11	CH_11	Spare	tx_cm_sel[1]	tx_cm_sel[0]	rterm_tx[4]	rterm_tx[3]	rterm_tx[2]	rterm_tx[1]	rterm_tx[0]
12	CH_12	tdrv_slice3_sel[1]	tdrv_slice3_sel[0]	tdrv_slice2_sel[1]	tdrv_slice2_sel[0]	tdrv_slice1_sel[1]	tdrv_slice1_sel[0]	tdrv_slice0_sel[1]	tdrv_slice0_sel[0]
13	CH_13	tdrv_slice4_cur[1]	tdrv_slice4_cur[0]	tdrv_slice3_cur[1]	tdrv_slice3_cur[0]	tdrv_slice5_sel[1]	tdrv_slice5_sel[0]	tdrv_slice4_sel[1]	tdrv_slice4_sel[0]
14	CH_14	tdrv_slice2_cur[1]	tdrv_slice2_cur[0]	tdrv_slice1_cur[2]	tdrv_slice1_cur[1]	tdrv_slice1_cur[0]	tdrv_slice0_cur[2]	tdrv_slice0_cur[1]	tdrv_slice0_cur[0]
15	CH_15	tdrv_slice5_cur[1]	tdrv_slice5_cur[0]	tdrv_dat_sel[1]	tdrv_dat_sel[0]	lb_ctl[3]	lb_ctl[2]	lb_ctl[1]	lb_ctl[0]
16	CH_16	rxterm_cm[1]	rxterm_cm[0]	rcv_dcc_en	rx_refck_sel	ldr_rx2core_sel	rx_div11_sel	rate_mode_rx	rpwdnb
17	CH_17	rxin_cm[1]	rxin_cm[0]	Reserved	rterm_rx[4]	rterm_rx[3]	rterm_rx[2]	rterm_rx[1]	rterm_rx[0]
18	CH_18	fc2dco_dloop	fc2dco_floop	Reserved	Spare	Reserved	Reserved	Reserved	Reserved
19	CH_19	Spare	req_lv_set[1]	req_lv_set[0]	rx_rate_sel[3]	rx_rate_sel[2]	rx_rate_sel[1]	rx_rate_sel[0]	req_en
1A	CH_1A	band_calib_mode	en_recalib	dco_calib_RST	dco_facq_RST	pden_sel	rx_dco_ck_div[2]	rx_dco_ck_div[1]	rx_dco_ck_div[0]
1B	CH_1B	rlos_sel	rx_los_en	rx_los_hyst_en	rx_los_ceq[1]	rx_los_ceq[0]	rx_los_lv[2]	rx_los_lv[1]	rx_los_lv[0]
1C	CH_1C	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
1D	CH_1D	Spare	Spare	Spare	Spare	Spare	Spare	Spare	Spare
1E	CH_1E		pci_det_done_int_ctl	rlos_int_ctl	~rlos_int_ctl	Reserved	Reserved	riol_int_ctl	~riol_int_ctl
1F	CH_1F	rrst	lane_rx_RST	lane_tx_RST	ff_tx_f_clk_dis	ff_tx_h_clk_en	ff_rx_f_clk_dis	ff_rx_h_clk_en	sel_sd_rx_clk
20	CH_20	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
PER CHANNEL PCS STATUS REGISTERS (6)									
30	CH_30				pfifo_error	cc_underrun	cc_overrun	fb_rx_fifo_error	fb_tx_fifo_error
31	CH_31	prbs_error_cnt[7]	prbs_error_cnt[6]	prbs_error_count[5]	prbs_error_cnt[4]	prbs_error_cnt[3]	prbs_error_cnt[2]	prbs_error_cnt[1]	prbs_error_cnt[0]
32	CH_32				wa_offset[3]	wa_offset[2]	wa_offset[1]	wa_offset[0]	
33	CH_33	Reserved	Reserved	Reserved	Reserved	cc_underrun_int	cc_overrun_int	fb_rx_fifo_error_int	fb_tx_fifo_error_int
34	CH_34	Reserved	ffs_ls_sync_status	fb_nrst_o	fb_txrst_o	Reserved	Reserved	cc_re_o	cc_we_o
35	CH_35	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
PER CHANNEL SERDES STATUS REGISTERS (7)									
36	CH_36		pci_det_done	rlos	~rlos			riol	~riol
37	CH_37	dco_calib_err	dco_calib_done	dco_facq_err	dco_facq_done				pci_connect
38	CH_38	Reserved							
39	CH_39	Reserved							
3a	CH_3A		pci_det_done_int	rlos_int	~rlos_int			riol_int	~riol_int
3b	CH_3B	Reserved							
3c	CH_3C	Reserved							

## Per Channel PCS Register Details

**Table 9-51. PCS Control Register 1 (CH\_00)**

Bit	Name	Description	R/W	Default
0	uc_mode	1 = Selects User Configured mode 0 = Selects other mode (PCIe, RapidIO, 10GbE, 1GbE)	R/W	0
1	Reserved			
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	wa_mode	1 = bitslip word alignment mode 0 = barrel shift word alignment mode	R/W	0
7:5	Reserved		R/W	0

**Table 9-52. PCS Control Register 2 (CH\_01)**

Bit	Name	Description	R/W	Default
0	invert_rx	1 = Invert received data 0 = Do not invert received data.	R/W	0
1	invert_tx	1 = Invert transmitted data 0 = Do not invert transmitted data.	R/W	0
2	Internal use only			
3	Reserved			
4	ge_an_enable	1 = Enable GigE Auto Negotiation 0 = Disable GigE Auto Negotiation	R/W	0
5	Internal use only			
6	Internal use only			
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 9-53. PCS Control Register 3 (CH\_02)**

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
1:0	Reserved			
2	tx_gear_mode	1 = Enable 2:1 gearing for transmit path on selected channels 0 = Disable 2:1 gearing for transmit path on selected channels (no gearing)	R/W	0
3	rx_gear_mode	1 = Enable 2:1 gearing for receive path on selected channels 0 = Disable 2:1 gearing for receive path on selected channels (no gearing)	R/W	0
5:4	pcs_det_time_sel[1:0]	PCS connection detection time 11 = 16 us, 10 = 4 us, 01 = 2 us 00 = 8 us	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 dual 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 9-54. PCS Control Register 4 (CH\_03)**

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	Internal use only			
3	enc_bypass	1 = Bypass 8b10b encoder 0 = Normal operation	R/W	0
4	Internal use only			
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 = Invert TX data after SerDes Bridge 0 = Do not invert TX data after SerDes Bridge (Note: Loopback data is inverted)	R/W	0
7	Reserved			

**Table 9-55. PCS Control Register 5 (CH\_04)**

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 = Invert RX data after SerDesBridge 0 = Normal operation	R/W	0
2	wa_bypass	1 = Bypass word alignment 0 = Do not invert RX data after SerDesBridge(Note: Loopback data is inverted)	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 9-56. PCS Control Register 6 (CH\_05)**

If neither match enable bit is set below, single character skip matching is performed using match 4.

Bit	Name	Description	R/W	Default
3:0	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 9-57. PCS Control Register 7 – CC match 1 LO (CH\_06)**

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

**Table 9-58. PCS Control Register 8 – CC match 2 LO (CH\_07)**

Bit	Name	Description	R/W	Default
7:0	cc_match_2[7:0]	Lower bits of user defined clock compensator skip pattern 2	R/W	8'h00

**Table 9-59. PCS Control Register 9 – CC match 3 LO (CH\_08)**

Bit	Name	Description	R/W	Default
7:0	cc_match_3[7:0]	Lower bits of user defined clock compensator skip pattern 3	R/W	8'hBC

**Table 9-60. PCS Control Register 10 – CC match 4 LO (CH\_09)**

Bit	Name	Description	R/W	Default
7:0	cc_match_4[7:0]	Lower bits of user defined clock compensator skip pattern 3	R/W	8'h50

**Table 9-61. PCS Control Register 11 – CC match HI (CH\_0A)**

Bit	Name	Description	R/W	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 [8] = K control [9] = Disparity error	R/W	2'b01
7:6	cc_match_4 [9:8]	Upper bits of user defined clock compensator skip pattern 4 [8] = K control [9] = Disparity error	R/W	2'b01

**Table 9-62. PCS Control Register 12 – UDF Comma Mask LO (CH\_0B)**

Bit	Name	Description	R/W	Default
7:0	udf_comma_mask [7:0]	Lower bits of user defined comma mask	R/W	8'hFF

**Table 9-63. PCS Control Register 13 – UDF comma a LO (CH\_0C)**

Bit	Name	Description	R/W	Default
7:0	udf_comma_a [7:0]	Lower bits of user defined comma character 'a'	R/W	8'h83

**Table 9-64. PCS Control Register 14 – UDF comma b LO (CH\_0D)**

Bit	Name	Description	R/W	Default
7:0	udf_comma_b [7:0]	Lower bits of user defined comma character 'b'	R/W	8'h7c

**Table 9-65. PCS Control Register 15 – UDF comma HI (CH\_0E)**

Bit	Name	Description	R/W	Default
1:0	Reserved			
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 9-66. PCS Interrupt Control Register 16 (CH\_0F)**

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.	R/W	0
3	cc_underrun_int_ctl	1 = Enable interrupt for cc_underrun 0 = Disable interrupt for cc_underrun.	R/W	0
7:4	Reserved		R/W	0

## Per Channel SerDes Control Register Details

**Table 9-67. SERDES Control Register 1 (CH\_10)**

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
2	tx_div11_sel	0 = Full rate selection for transmit (high definition SMPTE) 1 = Divide by 11 selection for transmit (standard definition SMPTE)	R/W	0
3	ldr_core2tx_sel	1 = Select low speed serial data from FPGA core	R/W	0
4	tdrv_pre_en	1 = TX driver de-emphasis enable 0 = TX driver de-emphasis disable.	R/W	0
5	tdrv_post_en	1 = TX driver post-emphasis enable 0 = TX post-emphasis disable	R/W	0
6	tx_pre_sign	1 = TX preemphasis not inverted 0 = TX preemphasis inverted	R/W	0
7	tx_post_sign	1 = TX post emphasis not inverted 0 = TX post emphasis inverted	R/W	0

**Table 9-68. SERDES Control Register 2 (CH\_11)**

Bit	Name	Description	R/W	Default
4:0	rterm_tx [4:0]	TX resistor termination select. Disabled when PCIe feature is enabled (Ohms) 00000 = 5K 00001 = 80 00100 = 75 00110 = 70 01011 = 60 10011 = 50 11001 = 46 All other values are RESERVED	R/W	4'b0000
6:5	tx_cm_sel[1:0]	Select TX output common mode voltage 00 = power down 01 = 0.6 V 10 = 0.55 V 11 = 0.5 V	R/W	2'b00
7	Internal use only			

**Table 9-69. SERDES Control Register 3 (CH\_12)**

<b>Bit</b>	<b>Name</b>	<b>Description</b>	<b>R/W</b>	<b>Default</b>
1:0	tdrv_slice0_sel[1:0]	TX drive slice enable for slice 0: 00 = Power Down 01 = Select Main Data 10 = Select Pre Data 11 = Select Post Data	R/W	2'b00
3:2	tdrv_slice1_sel[1:0]	TX drive slice enable for slice 1: 00 = Power Down 01 = Select Main Data 10 = Select Pre Data 11 = Select Post Data	R/W	2'b00
5:4	tdrv_slice2_sel[1:0]	TX drive slice enable for slice 2: 00 = Power Down 01 = Select Main Data 10 = Select Pre Data 11 = Select Post Data	R/W	2'b00
7:6	tdrv_slice3_sel[1:0]	TX drive slice enable for slice 3: 00 = Power Down 01 = Select Main Data 10 = Select Pre Data 11 = Select Post Data	R/W	2'b00

**Table 9-70. SERDES Control Register 4 (CH\_13)**

<b>Bit</b>	<b>Name</b>	<b>Description</b>	<b>R/W</b>	<b>Default</b>
3:0	Internal use only			
5:4	tdrv_slice3_cur[1:0]	Controls the output current for slice 3. Every 100 uA increases the output amplitude by 100 mV. 00 = 800 uA 01 = 1600 uA 10 = 2400 uA 11 = 3200 uA	R/W	2'b00
7:6	tdrv_slice4_cur[1:0]	Controls the output current for slice 4. Every 100 uA increases the output amplitude by 100 mV. 00 = 800 uA 01 = 1600 uA 10 = 2400 uA 11 = 3200 uA	R/W	0

**Table 9-71. SERDES Control Register 5 (CH\_14)**

Bit	Name	Description	R/W	Default
2:0	tdrv_slice0_cur[2:0]	TX driver slice 0 current settings. Increases the output current of slice 0 by 100 uA which corresponds to a 100 mV differential increase in output amplitude. 000 = default swing amplitude (100 uA) 001 = 200 uA 010 = 300 uA 011 = 400 uA 100 = 500 uA 101 = 600 uA 110 = 700 uA 111 = 800 uA	R/W	2'b00
5:3	tdrv_slice1_cur[2:0]	TX driver slice 1 current settings. Increases the output current of slice 1 by 100 uA which corresponds to a 100 mV differential increase in output amplitude. 000 = default swing amplitude (100 uA) 001 = 200 uA 010 = 300 uA 011 = 400 uA 100 = 500 uA 101 = 600 uA 110 = 700 uA 111 = 800 uA	R/W	2'b00
7:6	tdrv_slice2_cur[1:0]	Controls the output current for slice 2. Every 100 uA increases the output amplitude by 100 mV. 00 = 800 uA 01 = 1600 uA 10 = 2400 uA 11 = 3200 uA	R/W	2'b00

**Table 9-72. SERDES Control Register 6 (CH\_15)**

Bit	Name	Description	R/W	Default
3:0	lb_ctl[3:0]	Serial loopback control, only one bit should be selected: [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 loopback from equalizer to driver enable [0] = slb_t2r_en serial TX to RX LB enable	R/W	4'b0000
5:4	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	2'b00
7:6	tdrv_slice5_cur[1:0]	Controls the output current for slice 5. Every 100 uA increases the output amplitude by 100mV. 00 = 800 uA 01 = 1600 uA 10 = 2400 uA 11 = 3200 uA	R/W	2'b00

**Table 9-73. SERDES Control Register 7 (CH\_16)**

Bit	Name	Description	R/W	Default
0	rpwdnb	0 = Power down receiver channel 1 = Power up receiver channel	R/W	0
1	rate_mode_rx	0 = Full rate selection for receive 1 = Half rate selection for receive	R/W	0
2	rx_div11_sel	0 = Full rate selection for receive (high definition SMPTE) 1 = Divide by 11 selection for receive (standard definition SMPTE)	R/W	0
3	ldr_rx2core_sel	Enables boundary scan input path for routing the high speed RX inputs to a lower speed Serdes in the FPGA (for out of band application)	R/W	0
4	rx_refck_sel	Rx CDR Reference Clock Select 0 = rx_refck_local 1 = ck_core_rx		
5	rcv_dcc_en	1 = Receiver DC coupling enable. 0 = AC coupling (default)	R/W	0
7:6	rxterm_cm[1:0]	Command mode voltage for RX input termination: 00 = RX Input Supply 01 = Floating (AC Ground) 10 = GND 11 = RX Input Supply	R/W	0

**Table 9-74. SERDES Control Register 8 (CH\_17)**

Bit	Name	Description	R/W	Default
4:0	rterm_rx [4:0]	RX input termination setting (Ohms): 00000 = 5K 00001 = 80 00100 = 75 00110 = 70 01011 = 60 10011 = 50 11001 = 46 All other values are RESERVED	R/W	4'b0000
5	Reserved			
7:6	rxin_cm[1:0]	Common mode voltage for equalizer input in ac coupling: 00 = 0.7 V 01 = 0.65 V 10 = 0.75 V 11 = CMFB	R/W	2'b00

**Table 9-75. Serdes Control Register 9 (CH\_18)**

Bit	Name	Description	R/W	Default
3:0	Reserved			
4	Reserved		R/W	0
5	Reserved		R/W	0
6	fc2dco_floop	1 = Force DCO lock to the frequency loop	R/W	0
7	fc2dco_dloop	1 = Force DCO lock to the data loop	R/W	0

**Table 9-76. Serdes Control Register 10 (CH\_19)**

Bit	Name	Description	R/W	Default
0	req_en	1 = Receiver equalization enable 0 = Receiver equalization disable	R/W	0
4:1	rx_rate_sel [3:0]	Equalizer pole position select: 0000 = TBD 0001 = TBD 0010 = TBD 0011 = TBD 0100 = TBD 0101 = TBD 0110 = TBD 0111 = TBD 1000 = TBD 1001 = TBD 1010 = TBD 1011 = TBD 1100 = TBD 1101 = TBD 1110 = TBD 1111 = TBD,	R/W	4'h0
6:5	req_lvl_set[1:0]	Level setting for equalization: 00 = 6 dB 01 = 9 dB 10 = 12 dB 11 = not used	R/W	2'b00
7	Reserved			

**Table 9-77. Serdes Control Register 11 (CH\_1A)**

Bit	Name	Description	R/W	Default
2:0	rx_dco_ck_div[2:0]	VCO output frequency select: 00x = Divided by 1 01x = Divided by 2 100 = Divided by 4 101 = Divided by 8 110 = Divided by 16 111 = Divided by 32	R/W	3'b000
3	pden_sel	This signal is used to disable the phase detector in the CDR during electrical idle. When set to 1, checks if los is set. If los is 0 then disables the phase detector (or data loop).	R/W	0
4	dco_facq_RST	Reset signal to trigger the DCO frequency acquisition process when it's necessary	RW	0
5	dco_calib_RST	Reset signal to trigger the DCO calibration process when it's necessary	RW	0
6	en_recalib	Enable recalibration: 0 = Disable 1 = Enable re-calibration	R/W	0
7	band_calib_mode	Calibration mode: 0 = Disable 1 = Enable	R/W	0

**Table 9-78. Serdes Control Register 12 (CH\_1B)**

Bit	Name	Description	R/W	Default
2:0	rx_los_lvl[2:0]	Sets differential p-p threshold voltage for loss of signal detection: 000 = TBD 001 = TBD 010 = TBD 011 = TBD 100 = TBD 101 = TBD 110 = TBD 111 = TBD	R/W	3'b000
4:3	rx_los_ceq[1:0]	Sets the equalization value at input stage of LOS detector: 00 = TBD 01 = TBD 10 = TBD 11 = TBD	R/W	2'b00
5	rx_los_hyst_en	Enables hysteresis in detection threshold level	R/W	0
6	rx_los_en	Enables loss of signal detector: 0 = Disabled 1 = Enabled	R/W	0
7	rlos_sel	Enables LOS detector output before enabling CDR phase detector after calibration: 0 = Disabled 1 = Enabled (If the channel is being used, this bit should be set to 1)	R/W	0

**Table 9-79. Serdes Interrupt Control Register 1 (CH\_1E)**

Bit	Name	Description	R/W	Default
0	~rlol_int_ctl	1= Enable interrupt when receiver is locked	R/W	0
1	rlol_int_ctl	1= Enable interrupt for receiver loss of lock	R/W	0
2	Reserved			
3	Reserved			
4	~rlos_int_ctl	1 = Enable interrupt for receiver Rx Loss of Signal when input level meets or is greater than programmed LOW threshold	R/W	0
5	rlos_int_ctl	1 = Enable interrupt for Rx Loss of Signal when input levels fall below the programmed LOW threshold (using rlos_set)	R/W	0
6	pcie_det_done_int_ctl	1 = Enable interrupt for detection of a far-end receiver for PCI Express	R/W	0
7	Reserved			

## Per Channel Reset and Clock Control Register Details

**Table 9-80. Reset and Clock Control Register 1 (CH\_1F)**

Bit	Name	Description	R/W	Default
0	sel_sd_rx_clk	1 = FPGA Bridge write clock and Elastic Buffer read clock driven by serdes recovered clock 0 = FPGA Bridge write clock and Elastic Buffer read clock driven by ff_ebrd_clk	R/W	0
1	ff_rx_h_clk_en	1 = Enable ff_rx_h_clk	R/W	0
2	ff_rx_f_clk_dis	1 = Disable ff_rx_f_clk	R/W	0
3	ff_tx_h_clk_en	1 = Enable ff_tx_h_clk	R/W	0
4	ff_tx_f_clk_dis	1 = Disable ff_tx_f_clk	R/W	0
5	lane_tx_rst	1 = Assert reset signal to transmit logic	R/W	0
6	lane_rx_rst	1 = Assert reset signal to receive logic	R/W	0
7	rrst	1 = Rx reset	RW	0

## Per Channel PCS Status Register Details

**Table 9-81. PCS Status Register 1 (CH\_30)**

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 9-82. PCS Status Register 2 (CH\_31)**

Bit	Name	Description	R/W	Int
7:0	prbs_error_cnt[7:0]	Count of the number of PRBS errors. Clears to zero on read. Sticks at FF.	RO CR	N

**Table 9-83. PCS Status Register 3 (CH\_32)**

Bit	Name	Description	R/W	Int
3:0	wa_offset[3:0]	Word Aligner Offset	RO	N
7:4	Reserved			

**Table 9-84. PCS General Interrupt Status Register 4 (CH\_33)**

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 9-85. PCS Status Register 5 (CH\_34)**

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
3:2	Reserved			
4	fb_txrst_o	1 = FPGA bridge Tx Normal Operation 0 = FPGA bridge Tx reset	RO	N
5	fb_rxrst_o	1 = FPGA bridge Rx Normal Operation 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 9-86. PCS Status Register 6 (CH\_35)**

Bit	Name	Description	R/W	Default
7:0	Reserved			

## Per Channel SerDes Status Register Details

**Table 9-87. SERDES Status Register 1 (CH\_36)**

Bit	Name	Description	R/W	Int
0	~rlol	1 = Indicates that CDR has locked to data.	RO	Y
1	rlol	1 = Indicates CDR loss of lock to data. CDR is locked to reference clock.	RO	Y
3:2	Reserved		RO CR	Y
4	~rlos	1 = Indicates that the input signal detected by receiver is greater than or equal to the programmed LOW threshold	RO CR	Y
5	rlos	1 = Indicates that the input signal detected by receiver is below the programmed LOW threshold	RO CR	Y
6	pci_det_done	1 = Receiver detection process completed by SerDes transmitter. 0 = Receiver detection process not completed by SerDes transmitter.	RO CR	Y
7	Reserved			

**Table 9-88. SERDES Status Register 2 (CH\_37)**

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 9-89. SERDES Interrupt Status Register 5 (CH\_3A)**

Bit	Name	Description	R/W	Int
0	~rlol_int	1 = Interrupt generated for ~rlol	CO CR	Y
1	rlol_int	1 = Interrupt generated for rlol	CO CR	Y
3:2	Reserved			
4	~rlos_int	1 = Interrupt generated for ~rlos	CR RO	Y
5	rlos_int	1 = Interrupt generated for rlos	CR RO	Y
6	pci_det_done_int	1 = Interrupt generated for pci_det_done	CR RO	Y
7	Reserved			

## Appendix B. Register Settings for Various Standards

**Table 9-90. Per Dual Register Settings for Different Standards**

Register	1GbE	10GbE	RapidIO 1x	RapidIO 4x	PCI-Ex 1x	PCI-Ex 4x
comma_a_lo	hex 03	hex 03	hex 03	hex 03	hex 03	hex 03
comma_b_lo	hex fc	hex fc	hex fc	hex fc	hex fc	hex fc
comma_mask_lo	hex 7f	hex 7f	hex 7f	hex 7f	hex 7f	hex 7f

**Table 9-91. Per Channel Register Settings for Different Standards**

Character	1GbE	10GbE	PCI-Ex	RapidIO
K23.7 (F7)	Carrier extend		PAD	
K27.7 (FB)	SOP	ST	Start TLP	A (align)
K28.0 (1C)		SKIP R	SKIP	SC
K28.1 (3C)			FTS	
K28.2 (5C)		SoS	Start DLLP	
K28.3 (7C)		ALIGN A	IDLE	PD
K28.4 (9C)		SEQ		
K28.5 (BC)	+ D5.6 or D16.2			
= IDLE	SYNC K	COMMA	K	
K28.6 (DC)				
K28.7 (FC)				
K29.7 (FD)	EOP	T	END	R (skip)
K30.7 (FE)	ERR	ERR	END BAD	

---

March 2014

Technical Note TN1262

## Introduction

The sysIO buffers in the ECP5™ device give the designer the ability to easily interface with other devices using advanced system I/O standards. This technical note describes the sysIO standards available and how to implement them using Lattice Diamond® design software.

## sysIO Buffer Overview

The ECP5 sysIO interface contains multiple Programmable I/O Cell (PIC) blocks. The primary building block is a quad or pair of GPIO depending on the side of the I/O. The GPIO functions are available on every PIO of all devices. The quad is built of four GPIOs (PIOA, PIOB, PIOC and PIOD) or two GPIOs (PIOA, PIOB). Two adjacent PIOs can be joined to provide a differential I/O pair (labeled as 'T' and 'C'). PIOA and PIOB comprise a differential pair and PIOC and PIOD comprise another pair. One true LVDS driver is connected only to the A/B pair. Each PIO includes a sysIO buffer and I/O logic (IOLeGIC). The ECP5 sysIO buffers support a variety of single-ended and differential signaling standards. The sysIO buffer also supports the DQS strobe signal that is required for interfacing with the DDR memory. The DQS signal from the bus is used to strobe the DDR data from the memory into input register blocks.

The top and bottom sides are grouped into eight IOs with the pitch matches to nine PLC from the core. These IOs will support hot socket with IO standards from 3.3 V to 1.2 V and mainly used for 3.3 V domain IOs. The left and right sides are grouped into 16 IOs that support one DQS group and pitch matches to 12PLC + EBR/DSP from the core. The left/right side IOs will support IO standard from 3.3 V to 1.2 V with no hot socket capability. The left/right side also have one LVDS output driver per four IOs and one differential termination resistor per two IOs. For more information on the architecture of the sysIO buffer, refer to DS1044, [ECP5 Family Data Sheet](#).

The IOLeGIC includes input, output and tri-state 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 IOLeGIC 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 IOLeGIC and the DDR memory interface support are discussed in more detail in TN1265, [ECP5 High-Speed I/O Interface](#).

## Supported sysIO Standards

The ECP5 sysIO buffer supports both single-ended and differential standards. Single-ended standards can be further subdivided into internally ratioed standards such as LVCMOS, LVTTL; and externally referenced standards such as HSUL and SSTL. The buffers support the LVTTL, LVCMOS 1.2 V, 1.5 V, 1.8 V, 2.5 V and 3.3 V standards. In LVCMOS and LVTTL modes, the buffer has individually-configurable options for drive strength, bus maintenance (weak pull-up, weak pull-down). Differential standards supported include LVDS, BLVDS, LVPECL, MLVDS, SLVS (Rx only), differential LVCMOS, differential SSTL and differential HSUL. For better support of video standards, sub-LVDS and MIPI (Rx only) are also supported. Table 10-1 and Table 10-2 list the sysIO standards supported in ECP5 devices.

**Table 10-1. Single-Ended I/O Standards**

Standard	$V_{REF}$	$V_{CCIO}$	Input	Output	Bi-Directional
LVTTL33	—	3.3 <sup>2</sup>	Yes	Yes	Yes
LVCMOS33	—	3.3 <sup>2</sup>	Yes	Yes	Yes
LVCMOS25	—	2.5 <sup>2</sup>	Yes	Yes	Yes
LVCMOS18	—	1.8	Yes	Yes	Yes
LVCMOS15	—	1.5	Yes	Yes	Yes
LVCMOS12	—	1.2 <sup>2</sup>	Yes	Yes	Yes
SSTL18 Class I, II	0.9	—	Yes <sup>1</sup>	Yes	Yes <sup>1</sup>
SSTL15 Class I, II	0.75	—	Yes <sup>1</sup>	Yes	Yes <sup>1</sup>
SSTL135 Class I, II	0.675	—	Yes <sup>1</sup>	Yes	Yes <sup>1</sup>
HSUL12	0.6	—	Yes <sup>1</sup>	Yes	Yes <sup>1</sup>

1. Left and right side I/O only.

2. Required for output only.

**Table 10-2. Differential I/O Standards**

Standard	$V_{REF}$	Input	Output	Bi-Directional
SSTL18D I, II	—			
SSTL135D I, II	—			
SSTL15D I, II	—			
HSUL12D	—			
LVTTL33D	—			
LVCMOS33D	—			
LVCMOS25D	—			
LVCMOS18D	—			
LVDS	—	Yes	A/B pair	Yes
LVDS25E	—	No	Yes	No
BLVDS25	—	Yes	No	No
BLVDS25E	—	No	Yes	Yes
MLVDS25	—	Yes	No	No
MLVDS25E	—	No	Yes	Yes
LVPECL33	—	Yes	No	No
LVPECL33E	—	No	Yes	No
SLVS	—	Yes	No	No
SUBLVDS	—	Yes	No	No
MII D-PHY HS Mode	—	C/D Pair	No	No

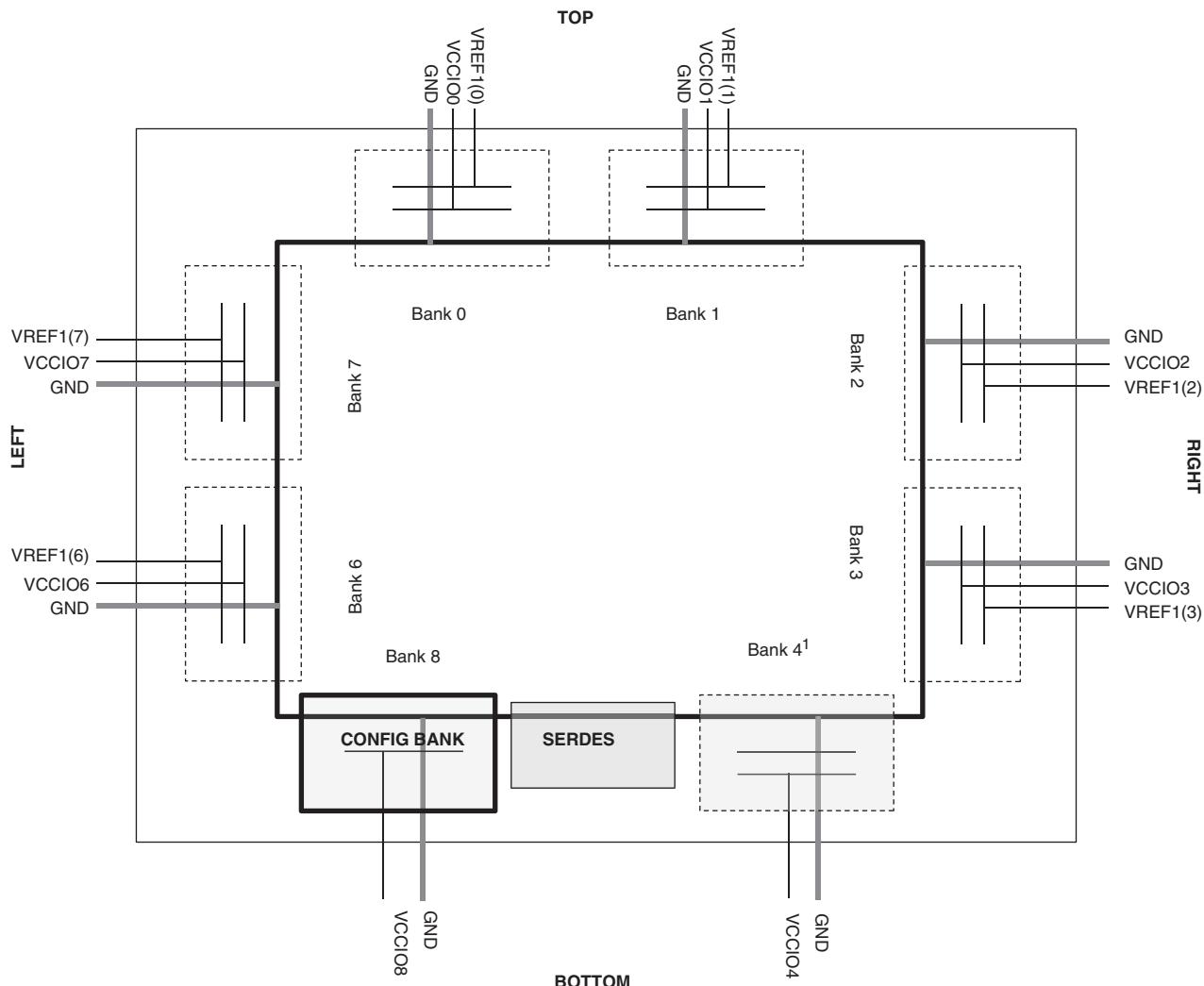
## sysIO Banking Scheme

ECP5 devices have general-purpose programmable sysIO banks and a configuration bank. Each of the general-purpose sysIO banks has a  $V_{CCIO}$  supply voltage and one reference voltage,  $V_{REF1}$ . Every device has two banks on the left, right and top side.

The bottom side implements SERDES channels and only the biggest device 85K has one sysIO bank.

Every ECP5 device has a TAP controller interface bank in the lower left corner of the device. This Bank 8 has four signal pins (TCK, TMS, TDI and TDO) and is powered by  $V_{CCIO8}$ , located on the lower left side of the device, has shared I/O for configuration.

**Figure 10-1. ECP5 sysIO Banking**



1. Only 85K device has this bank.

## V<sub>CC</sub> (1.1 V)

The core power supply, V<sub>CC</sub>, is used to power the device internally before data is captured by the I/O buffers. V<sub>CC</sub> is also used to power the 1.2V (LVCMS12) ratioed buffers so these can be captured independently of V<sub>CCIO</sub>.

## V<sub>CCIO</sub> (1.2 V/1.35 V/1.5 V/1.8 V/2.5 V/3.3 V)

Each bank has a separate V<sub>CCIO</sub> supply that powers the single-ended output drivers in a bank. The bank V<sub>CCIO</sub> is also used to power ratioed input buffers such as LVCMS15 and LVCMS18, as well as extended threshold ratioed buffers.

For unused banks, it is recommended to set V<sub>CCIO</sub> to 0V to minimize power and hold the bank in hot socket.

## V<sub>CCAUX</sub> (2.5V)

In addition the V<sub>CCIO</sub> supply, every bank also has an auxiliary global supply called V<sub>CCAUX</sub>. The bank V<sub>CCAUX</sub> supply is used to power the differential and referenced (SSTL) input buffer. Bank V<sub>CCAUX</sub> is also used to power the push-pull output pre-driver sections.

## **V<sub>CCIO8</sub> (1.2 V/1.5 V/1.8 V/2.5 V/3.3 V)**

The JTAG pins share power supply of Bank 8 V<sub>CCIO</sub> supplies. V<sub>CCIO8</sub> determines the electrical characteristics of the LVCMS JTAG pins, both the output high level and the input threshold. Table 10-3 contains a summary of the required power supplies.

**Table 10-3. Power Supplies**

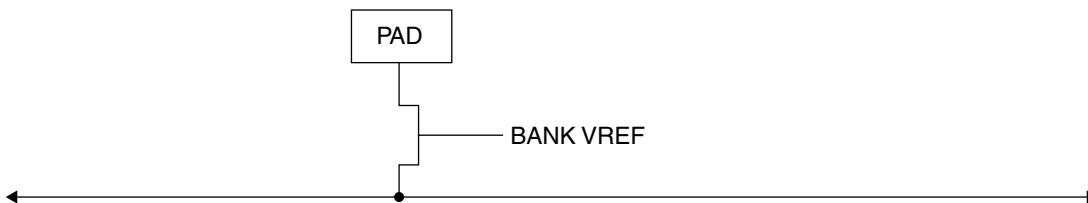
Power Supply	Description	Value <sup>1</sup>
VCC	Core power supply	1.1 V
VCCIO	Power supply for the I/O banks	1.2 V/1.35 V/1.8 V/2.5 V/3.3 V
VCCAUX	Auxiliary power supply	2.5 V
VCCIO8	Power supply for JTAG pins and configuration bank I/O	1.2 V/1.5 V/1.8 V/2.5 V/3.3 V

1. Refer to DS1044, [ECP5 Family Data Sheet](#) for recommended minimum and maximum values.

## **V<sub>REF1</sub>**

Each bank can support one separate V<sub>REF</sub> input voltage, V<sub>REF1</sub>, which is used to set the threshold for the referenced input buffers. A dedicated I/O in each bank can be used to drive the V<sub>REF1</sub> bank reference voltage. An I/O used as a V<sub>REF1</sub> input is also called a VREF1\_DRIVER. A conceptual block diagram is shown in Figure 10-2.

**Figure 10-2. Bank V<sub>REF</sub> from One Specific Pad**



To assign a V<sub>REF</sub> driver, use IO\_TYPE=VREF1\_DRIVER. To assign a V<sub>REF</sub> to a buffer, use VREF=VREF1\_LOAD.

## **Hot Socketing Support**

The I/Os located on the top and bottom sides are fully hot socketable

See DS1044, [ECP5 Family Data Sheet](#) for hot socketing (IDK) requirements.

## **Standby**

Using the Standby modes is a way to dynamically power-down the bank. It disables the differential/reference receiver, true differential driver, current mirrors and bias circuits.

In Standby mode, differential drivers and differential input buffers can be powered down to save power.

The Standby modes are enabled on a bank-by-bank basis. Each bank has user-routed input signals to enable the Standby (dynamic power-down) modes.

Refer to TN1266, [Power Consumption and Management for ECP5 Devices](#) for detailed information.

## **LVDS sysIO Buffer Pairs (A/B and C/D on Left and Right Sides)**

The GPIO are grouped as a quad building block, GPIOA, GPIOB, GPIOC and GPIOD. Each pair consists of two single-ended output drivers and two sets of single-ended input buffers (both ratioed and referenced). One referenced input buffer, per pair, can also be configured as a differential input. In addition to these buffers and drivers, each I/O has a weak pull-up and weak pull-down resistor. The programmability for these 'weak' features is limited to ON and OFF programmability on each I/O independently. The pull modes are always disabled in output mode. Left and right side GPIO has clamp always on. 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 asso-

ciated with the negative side of the differential I/O. The sysIO buffers pairs are grouped as A/B pad pairs or C/D pad pairs. Each sysIO pad pair will support programmable on/off differential input termination of 100 ohms. There is an added LVDS output driver in the A/B pad pairs of all arrays. The C/D pad pairs do not have the true LVDS differential output driver. The LVDS output driver does support tri-state. LVDS can be BIDI.

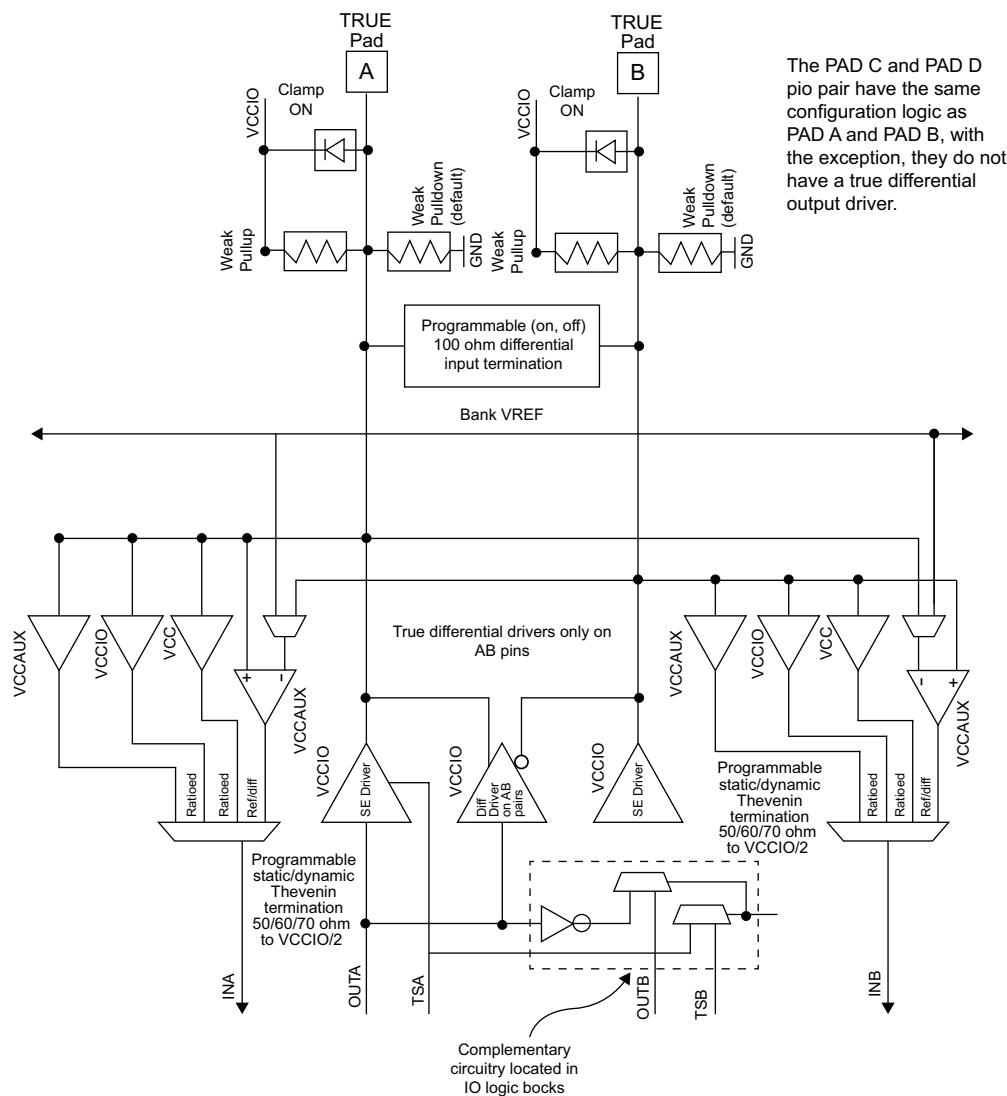
Figure 10-5 describes the detail of the sysIO Buffer Pairs on left and right sides.

### **sysIO Buffer Pair (A/B Pair on Top and Bottom Sides)**

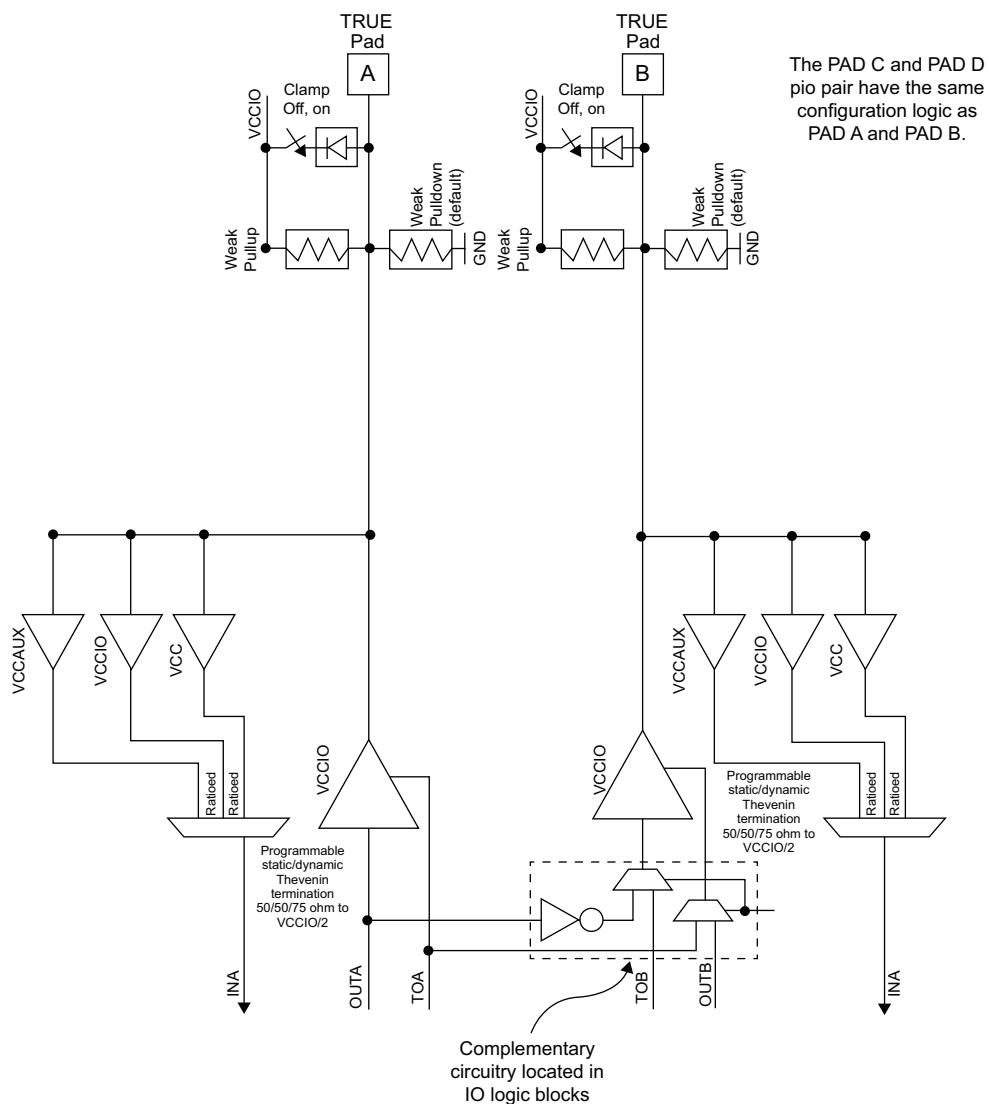
The GPIO are grouped as a pair building block, GPIOA and GPIOB. Each pair consists of two single-ended output drivers and two sets of single-ended input buffers (ratioed only). In addition to these buffers and drivers, each I/O has a weak pull-up and weak pull-down resistor. The programmability for these ‘weak’ features is limited to ON and OFF programmability on each I/O independently. The pull modes are always disabled in output mode. All GPIO on top and bottom side support a clamp that is programmable on or off.

Figure 10-5 describes the detail of the sysIO Buffer Pair on the top and bottom sides.

**Figure 10-3. sysIO Buffer Pair for Left and Right Sides**



**Figure 10-4. SysIO Buffer Pair for Top and Bottom Sides**



## Mixed Voltage Support in a Bank

ECP5 devices support mixed mode inputs in a given bank on all sides of the device. All differential and referenced inputs are supported independent of  $V_{CCIO}$ . When output is configured as an open drain it can be placed independently of  $V_{CCIO}$ . Some of the ratioed buffers are powered by  $V_{CCAUX}$  and  $V_{CC}$  and can therefore be placed independently of  $V_{CCIO}$ .

ECP5 devices support numerous mixed input voltage combinations by using a combination of three ratio receivers. The first receiver is powered by  $V_{CCIO}$  and uses overdrive/underdrive threshold adjustments to support 1.8 V and 1.5 V signaling. The second is a fixed threshold 1.2 V ratio receiver powered by  $V_{CC}$  that supports 1.2 V signaling. The third is powered with  $V_{CCAUX}$  (2.5 V), supports hysteresis, and is used for 3.3 V and 2.5 V signaling.

Table 10-4 lists the ratioed sysIO standards that can be mixed in the same bank.

**Table 10-4. Mixed Voltage I/O Support**

For TOP/BOTTOM BANKS										
V <sub>CCIO</sub> (V)	Input SysIO Standards					Output SysIO Standards				
	1.2 V	1.5 V	1.8 V	2.5 V	3.3 V	1.2 V	1.5 V	1.8 V	2.5 V	3.3 V
1.2 V	✓			✓	✓	✓				
1.35 V	✓			✓	✓					
1.5 V	✓	✓		✓	✓		✓			
1.8 V	✓		✓	✓	✓			✓		
2.5 V	✓			✓	✓				✓	
3.3 V	✓			✓	✓					✓
For LEFT/RIGHT BANKS										
V <sub>CCIO</sub> (V)	Input Signal					Output SysIO Standards				
	1.2 V	1.5 V	1.8 V	2.5 V	3.3 V	1.2 V	1.5 V	1.8 V	2.5 V	3.3 V
1.2 V	✓					✓				
1.35 V	✓									
1.5 V	✓	✓					✓			
1.8 V	✓		✓					✓		
2.5 V	✓			✓					✓	
3.3 V	✓			✓	✓					✓

## sysIO Buffer Configurations

This section describes the various sysIO features available on the ECP5 FPGA.

### Programmable Drive Strength

The single-ended driver has programmable drive strength. The LVCMS/LVTTL drive strength available at each value of V<sub>CCIO</sub> is shown in Table 10-5. The ECP5 single-ended driver is a process, voltage and temperature compensating driver. Therefore, there will be a good tolerance of drive strength. For LVCMS and LVTTL I/O standards, guaranteed minimum drive strength is listed.

The user must consider the maximum allowable current per bank and the package thermal limit current when selecting the drive strength. Table 10-5 shows the available drive settings for each of the output standards.

**Table 10-5. Programmable Drive Values for LVCMS/LVTTL**

1.2 V		1.5 V		1.8 V		2.5 V		3.3 V		Units
I <sub>OLmin</sub>	I <sub>OHighmin</sub>									
4	-4	4	-4	4	-4	4	-4	4	-4	mA
8	8	8	-8	8	-8	8	-8	8	-8	mA
				12 <sup>1</sup>	-12 <sup>1</sup>	12 <sup>1</sup>	-12 <sup>1</sup>	12 <sup>1</sup>	-12 <sup>1</sup>	mA
								16 <sup>1</sup>	-16 <sup>1</sup>	mA

1. Automotive device may not support drive setting.

The SSTL and HSUL nominal drive strengths are optimized for the performance and signal integrity of the I/O interface.

### Programmable Slew Rate

The single-ended output buffer for each device I/O pin has programmable output slew rate control that can be configured for either low-noise (SLEWRATE=SLOW) or high-speed (SLEWRATE=FAST) performance. Each I/O pin has an individual slew rate control that allows designers to specify slew rate control on a pin-by-pin basis. Slew rate control affects both the rising and falling edges. Slew rates vary as a function of drive and PVT conditions. Slow slew rate reduces SSO noise. The software default for slew rate is SLEWRATE=SLOW.

Differential standards are not impacted by slew rate settings. However, slew rate settings have some impact on emulated differential standards, as they use single-ended output buffers and complementary outputs.

### Tri-state Control

On the output side, each single-ended driver has a separate tri-state control. The differential driver has tri-state control as well.

### Open Drain Control

In addition to tri-state control, single-ended drivers also support open drain operation on each I/O independently. Unlike non-open drain outputs that consist of both source and sink components, an open drain output is composed of only the sink section of the output driver.

All LVCMOS and LVTTL 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.

### Complementary Outputs

The single-ended driver associated with the complementary pad can optionally be driven by the complement of the data that drives the single-ended driver associated with the true pad. ECP5 devices use pads A and C as true pads and pads B and D as complement pads. This allows a pair of single-ended drivers to be used to drive complementary outputs. Pads A and B from a PIO pair and pads C and D from another PIO pair. This is used for driving complementary SSTL signals (as required by the differential SSTL clock inputs on synchronous DRAM and synchronous SRAM devices, respectively). It can also be used in conjunction with off-chip resistors to emulate LVPECL33, MLVDS, LVDS25E and BLVDS output drivers. When this option is selected, the tri-state control for the driver associated with the complement pad is driven by the same signal as the tri-state control for the driver associated with the true pad.

### Differential I/O Supported

Differential inputs LVDS, SUBLVDS, MLVDS25, BLVDS, SLVS, MIPI are supported with differential receivers on both A/B pair and C/D pair PIOs and on left and right sides only. 50% of the sysIO buffer pairs on the left and right sides only are true differential outputs. LVDS is supported with a dedicated differential output driver on the A/B PIO pair. The C/D pair pins do not support true differential outputs.

LVDS25E, LVPECL33E, MLVDS25E, and BLVDS25E outputs can be implemented via emulation on all A/B and C/D pin output pairs. These emulated differential outputs require external resistors. Refer to DS1044, [ECP5 Family Data Sheet](#) for detailed information.

### Complementary SSTL Output Support

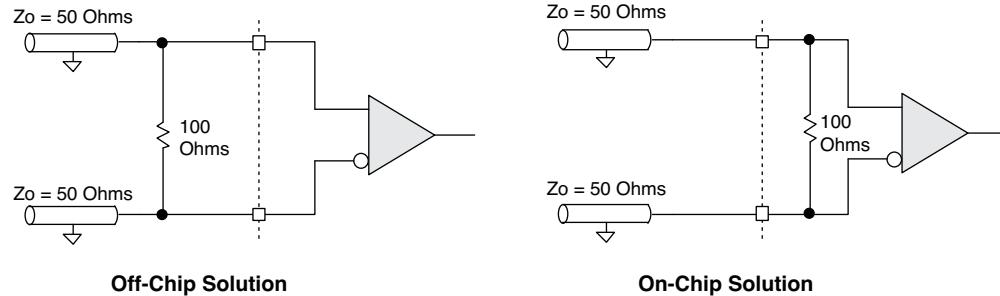
Differential SSTL outputs do not use external resistors, they use the complementary mux contained within each pair of single-ended output drivers.

### Differential Input Termination

The ECP5 device supports on-chip 100 ohm input differential termination between all pairs of all banks on left and right sides. The only value supported is 100 ohm. It is programmable as on and off. When it is on and the I/O type is MIPI or a BIDI, it is dynamic.

Figure 10-5 shows the discrete off-chip and on-chip solutions for dedicated, differential input termination. The differential termination is implemented using parallel legs that turn on and off to compensate for PVT variation. The termination also applies to input termination and is dynamic (enabled when output buffer is put in tri-state) or static (always on) to support MIPI and BIDI applications.

**Figure 10-5. Differential Input Termination**



### Single-Ended Input Termination

ECP5 devices support single-ended input parallel termination to  $V_{CCIO}/2$ . This is done by using output driver legs to emulate termination between the pad and  $V_{CCIO}$  as well as between the pad and VSS. Both static and dynamic termination are supported. Dynamic termination is used to support the DDR2 and DDR3 interface standards. Values of termination are 50 ohms, 75 ohms and 150 ohms. All input parallel terminations use a Thevenin termination scheme. As an example, 50ohms to  $V_{CCIO}/2$  is created by the Thevenin combination of 100 ohms between the pad and  $V_{CCIO}$  and 100 ohms between the pad and VSS.

Figure 10-6 shows the various off-chip, single-ended input termination schemes.

**Figure 10-6. Single-Ended Input Termination**

Termination Type	Off-Chip Solution	On-Chip Solution
Parallel to $V_{CCIO}/2$ at Receiving end. Thevenin		
Parallel to $V_{CCIO}/2$ at Receiving end with Bidirectional Enable/Disable for DDR2 and DDR3	N/A	

### **Programmable CLAMP**

The buffers on top and bottom sysIO have optional clamp diodes that may optionally be specified in the Lattice Diamond design software. The programmable CLAMP can be turned ON or OFF.

### **Differential SSTL and HSUL**

The single-ended driver associated with the complementary 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 HSUL signals (as required by the differential SSTL and HSUL clock inputs on synchronous DRAM and synchronous SRAM devices, respectively).

Refer to the DS1044, [ECP5 Family Data Sheet](#) for a detailed description of the differential HSUL and SSTL implementations.

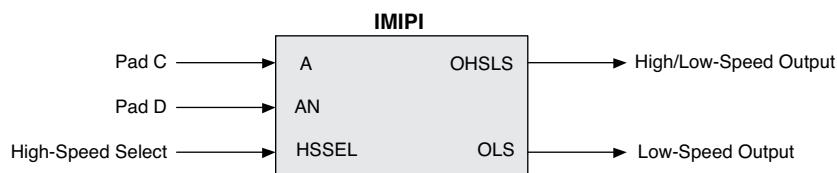
### **MIPI D-PHY Interface**

The MIPI D-PHY Interface is used only in input mode and only on the C/D pad pair. The differential C/D pad pair high-speed and pad C single-ended low-power inputs are handled through the pad C I/O logic. The pad D single-ended low-power inputs are handled through the pad D I/O logic.

- **HS mode:** 100 ohm differential termination is enabled with a differential receiver
- **LP mode:** HS mode is disabled and ratio receiver is enabled on pad C

The primitive shown in Figure 10-7 should be used when implementing the MIPI interface.

**Figure 10-7. MIPI Primitive**



**Table 10-6. IMIPI Port List**

Port	I/O	Description
A	I	Pad C input
AN	I	Pad D input
HSSEL	I	High-speed select signal. This is shared with the tri-state input of the buffer. HSSEL=1: High-speed mode, 100 ohm differential termination is on. Pad C logic select differential signal to IOL for gearing. HSSEL=0: Low-speed mode, 100 ohm termination is turned off. OHSL selected as ratioed LVCMOS input buffer from the I input (pad C), OLS selected as LVCMOS input from the IN input (pad D).
OHSL	O	High-speed or low-speed output, depending on HSSEL
OLS	O	Low-speed output

MIPI is supported via the IMIPI primitive instead of IO\_TYPE in the front-end RTL and simulation.

## Software sysIO Attributes

sysIO attributes can be specified in the HDL, using the Preference Editor GUI or in the ASCII Preference file (.prf) directly. The appendices of this document provide examples of how these can be assigned using each of the methods described above. This section describes each of the attributes in detail.

### **IO\_TYPE**

The IO\_TYPE 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. The BANK  $V_{CCIO}$  attribute is used to specify allowed  $V_{CCIO}$  combinations for each IO\_TYPE.

**Default:** LVCMOS25

**Table 10-7. IO\_TYPE Attribute Values**

sysIO Signaling Standard	IO_TYPE
DEFAULT	LVCMOS25
LVDS 2.5 V	LVDS
Emulated LVDS 2.5 V	LVDS25E
Bus LVDS 2.5 V	BLVDS25
Emulated Bus LVDS 2.5 V	BLVDS25E
LVPECL 3.3 V	LVPECL33
Emulated LVPECL 3.3 V	LVPECL33E
MLVDS	MLVDS
Emulated MLVDS	MLVDS25E
SLVS	SLVS
Sub_LVDS	SUBLVDS
HSUL 1.2 V	HSUL12
Differential HSUL	HSUL12D
SSTL15 Class I and II	SSTL15_I, SSTL15_II
Differential SSTL15 Class I and II	SSTL15D_I, SSTL15D_II
SSTL135 Class I and II	SSTL135_I, SSTL135_II
Differential SSTL135 Class I and II	SSTL135D_I, SSTL135D_II
SSTL18 Class I and II	SSTL18_I, SSTL18_II
Differential SSTL18 Class I and II	SSTL18D_I, SSTL18D_II
Differential LVTTL	LVTTL33D
LVTTL	LVTTL33
LVCMOS 3.3 V	LVCMOS33
LVCMOS 2.5 V	LVCMOS25
LVCMOS 1.8 V	LVCMOS18
LVCMOS 1.5 V	LVCMOS15
LVCMOS 1.2 V	LVCMOS12
Differential LVCMOS 3.3 V	LVCMOS33D
Differential LVCMOS 2.5 V	LVCMOS25D
Differential LVCMOS 1.8 V	LVCMOS18D

## OPENDRAIN

An I/O can be assigned independently to be an open drain when this attribute is turned on.

**Values:** ON, OFF

**Default:** OFF

## DRIVE

The drive strength attribute is available for output standards that support programmable drive strength. The default depends on the I/O standard used.

**Table 10-8. Programmable Output Drive**

Output Standard	DRIVE	DIFFDRIVE	VCCIO
<b>Single-Ended Interfaces</b>			
LVTTL33	4 mA, 8 mA, 12 mA, 16 mA	—	3.3
LVCMOS33	4 mA, 8 mA, 12 mA, 16 mA	—	3.3
LVCMOS25	4 mA, 8 mA, 12 mA	—	2.5
LVCMOS18	4 mA, 8 mA, 12 mA	—	1.8
LVCMOS15	4 mA, 8 mA	—	1.5
LVCMOS12	4 mA, 8 mA	—	1.2
LVTTL33 (open drain)	4 mA, 8 mA, 12 mA, 16 mA	—	Note <sup>2</sup>
LVCMOS33 (open drain)	4 mA, 8 mA, 12 mA, 16 mA	—	Note <sup>2</sup>
LVCMOS25 (open drain)	4 mA, 8 mA, 12 mA, 16 mA	—	Note <sup>2</sup>
LVCMOS18 (open drain)	4 mA, 8 mA, 12 mA, 16 mA	—	Note <sup>2</sup>
LVCMOS15 (open drain)	4 mA, 8 mA, 12 mA <sup>1</sup> , 16 mA	—	Note <sup>2</sup>
LVCMOS12 (open drain)	4 mA, 8 mA, 12 mA <sup>1</sup> , 16 mA	—	Note <sup>2</sup>
HSUL12	4 mA, 8 mA	—	1.2
SSTL135 I	8 mA	—	1.35
SSTL135 II	10 mA	—	1.35
SSTL18 I	8 mA	—	1.8
SSTL18 II	16 mA	—	1.8
SSTL15 I	8 mA	—	1.5
SSTL15 II	10 mA	—	1.5
<b>Differential Interfaces</b>			
LVTTL33D	4 mA, 8 mA, 12 mA, 16 mA	—	3.3
LVCMOS33D	4 mA, 8 mA, 12 mA, 16 mA	—	3.3
LVCMOS25D	4 mA, 8 mA, 12 mA	—	2.5
SSTL1.35D I	8 mA	—	1.35
SSTL1.35D II	10 mA	—	1.35
SSTL18D I	8 mA	—	1.8
SSTL18D II	16 mA	—	1.8
SSTL15D I	8 mA	—	1.5
SSTL15D II	10 mA	—	1.5
HSUL12D	4 mA, 8 mA	—	1.2
LVDS	—	3.5	2.5
LVDS25E <sup>1</sup>	8 mA	—	2.5
BLVDS25E <sup>1</sup>	16 mA	—	2.5
MLVDS25E <sup>1</sup>	16 mA	—	2.5
LVPECL33E <sup>1</sup>	16 mA	—	3.3

1. Emulated with LVCMOS drivers and external resistors.

2. Independent of V<sub>CCIO</sub>.

## DIFFDRIVE

DIFFDRIVE attribute is available for the LVDS output standard. The default value is set to 3.5 mA.

**Values:** 3.5 mA

**Default:** 3.5 mA

## TERMINATION

This attribute sets the on-chip input parallel termination to  $V_{CCIO}/2$ . Parallel termination is achieved using a Thevenin termination scheme. This programmable option can be set for each I/O individually. Both static and dynamic terminations are available.

**Values:** OFF, 50, 75, 150

**Default:** OFF

## DIFFRESISTOR

This attribute is used to provide differential termination (dynamic differential). It is available only for differential IO\_TYPES.

**Values:** OFF, 100

**Default:** OFF

## CLAMP

The CLAMP options can be enabled for each I/O independently. CLAMP is available on only top and bottom sysIO banks. CLAMP is not available when an output is set to open drain.

**Values:** ON, OFF

**Default:** OFF

## PULLMODE

The PULLMODE options can be enabled for each I/O pin independently. The PULLMODE settings are not available when I/O pins are programmed output-only. It is available for I/O pins in Input mode and Bidi mode.

**Values:** UP, DOWN, NONE

**Default:** DOWN

## SLEWRATE

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 is not a valid attribute for inputs.

**Values:** FAST, SLOW, NA

**Default:** SLOW

*Note: LVTTL and LVCMS support fast and slow slew rates.*

## Hysteresis

The ratioed input buffers have an input hysteresis option. The HYSTERESIS option can be used to change the amount of hysteresis for the LVTTL33, LVCMOS33 and LVCMOS25 input and bi-directional I/O standards.

The HYSTERESIS option for each of the input pins can be set independently.

**Values:** ON, OFF

**Default:** Default for LVCMOS33, LVCMOS25 and LVTTL33 is ON. Default for all other IO\_TYPES is OFF.

## V<sub>REF</sub>

The V<sub>REF</sub> option **must** be enabled for referenced input buffers (HSUL and SSTL). The V<sub>REF</sub> can be specified in the HDL or in the Design Planner GUI.

**Values:** OFF, VREF1\_LOAD

**Default:** VREF1\_LOAD (software assigns the dedicated pin to be V<sub>REF</sub>).

## DIN/DOUT

This attribute can be used when an I/O register needs to be assigned. Using DIN asserts an input register and using DOUT asserts an output register in the design. By default, the software will attempt to assign the I/O registers if applicable. Users can turn this OFF by using a synthesis attribute or the Preference Editor. These attributes can only be applied on registers.

## LOC

This attribute can be used to make pin assignments to the I/O ports in the design. This attribute is used only when the pin assignments are made in HDL source code. Pins can also be assigned directly using the GUI in the Preference Editor. See the appendices of this document for further information.

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

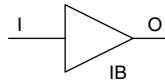
## Revision History

Date	Version	Change Summary
August 2013	01.0	Initial release

## Appendix A. sysIO Primitive Symbols and Instance Examples

### Primitive Symbols

**IB:** Input Buffer



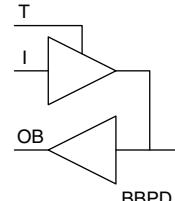
INPUT: I  
OUTPUT: O

**OBCO:** Output Complementary Buffer



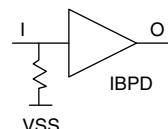
INPUT: I  
OUTPUTS: OT, OC

**BBPD:** Bi-directional Buffer with Pull-down



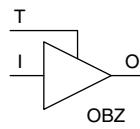
INPUTS: I, T  
OUTPUT: O  
INOUT: B

**IBPD:** Input Buffer with Pull-down



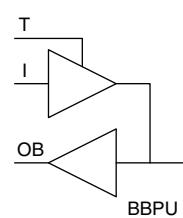
INPUT: I  
OUTPUT: O

**OBZ:** Output Buffer with Tristate



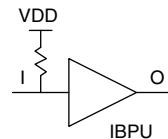
INPUTS: I, T  
OUTPUT: O

**BBPU:** Bi-directional Buffer with Pull-up



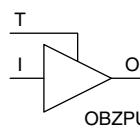
INPUTS: I, T  
OUTPUT: O  
INOUT: B

**IBPU:** Input Buffer with Pull-up



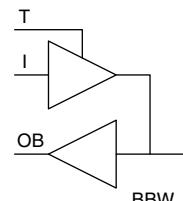
INPUT: I  
OUTPUT: O

**OBZPU:** Output Buffer with Tristate and Pull-up



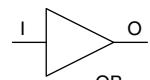
INPUTS: I, T  
OUTPUT: O

**BBW:** Bi-directional Buffer with Keeper Mode



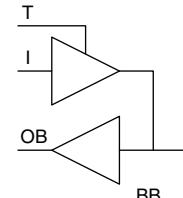
INPUTS: I, T  
OUTPUT: O  
INOUT: B

**OB:** Output Buffer



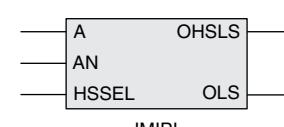
INPUT: I  
OUTPUT: O

**BB:** Bi-directional Buffer



INPUTS: I, T  
OUTPUT: O  
INOUT: B

**MIPi**



INPUTS: A, AN, HSSEL  
OUTPUTS: OHSLS, OLS

## Instance Examples

### Input Buffer (IB)

VHDL:

```
component IB
    port (I: in std_logic; O: out std_logic);
end component;

Inst_IB: IB
    port map (I=>clk, O=>buf_clk);
```

Verilog:

```
IB IB_inst (.I(Data[7]), .O(buf_Data7));
```

### Output Buffer (OB)

VHDL:

```
component OB
    port (I: in std_logic; O: out std_logic);

Inst_OB0: OB
    port map (I=>buf_qo0, O=>q(0));
```

Verilog:

```
IB IB_inst (.I(Data[7]), .O(buf_Data7));
```

### Bi-directional Buffer (BB)

VHDL:

```
component BB
    port (I: in std_logic; T: in std_logic; O: out std_logic;
          B: inout std_logic);
end component;

buf7: BB
    port map (I=>Q_out7, T=>Q_tri7, O=>buf_Data7, B=>Data(7));
```

Verilog:

```
BB buf7 (.I(Q_out7), .T(Q_tri7), .O(buf_Data7), .B(Data[7]));
```

## Appendix B. sysIO Attribute Examples

### IO\_TYPE

VHDL:

```
ATTRIBUTE IO_TYPE: string;
ATTRIBUTE IO_TYPE OF portA: SIGNAL IS "LVCMOS18";
ATTRIBUTE IO_TYPE OF portB: SIGNAL IS "LVCMOS33";
ATTRIBUTE IO_TYPE OF portC: SIGNAL IS "SSTL33_II";
ATTRIBUTE IO_TYPE OF portD: SIGNAL IS "LVCMOS25";
```

Verilog

```
output [4:0] portA /* synthesis IO_TYPE="LVTTL33" DRIVE="16" PULLMODE="UP" SLE-WRATE="FAST" */;
```

### OPENDRAIN

VHDL:

```
ATTRIBUTE OPENDRAIN: string;
ATTRIBUTE OPENDRAIN OF q_lvttl33_17: SIGNAL IS "ON";
```

Verilog:

```
output [4:0] portA /* synthesis attribute OPENDRAIN of q_lvttl33_17 is ON */;
```

### DRIVE

VHDL:

```
ATTRIBUTE DRIVE: string;
ATTRIBUTE DRIVE OF portD: SIGNAL IS "8";
```

Verilog:

```
output [4:0] portA /* synthesis DRIVE = "8" */;
```

### DIFFDRIVE

VHDL:

```
ATTRIBUTE DIFFDRIVE: string;
ATTRIBUTE DIFFDRIVE OF portF: SIGNAL IS "3.5";
```

Verilog:

```
output [4:0] portF/* synthesis IO_TYPE="LVDS" DIFFDRIVE="3.5" */;
```

### TERMINATION

VHDL:

```
ATTRIBUTE TERMINATION: string;
ATTRIBUTE TERMINATION OF portF: SIGNAL IS "50";
```

Verilog:

```
output [4:0] portA /* synthesis IO_TYPE="SSTL18_I" TERMINATION = "50" */;
```

## DIFFRESISTOR

VHDL:

```
ATTRIBUTE DIFFRESISTOR: string;
ATTRIBUTE DIFFRESISTOR OF portF: SIGNAL IS "100";
```

Verilog:

```
output [4:0] portA /* synthesis IO_TYPE="LVDS" DIFFRESISTOR = "100" */;
```

## PULLMODE

VHDL:

```
ATTRIBUTE PULLMODE: string;
ATTRIBUTE PULLMODE OF portF: SIGNAL IS "PULLUP";
```

Verilog:

```
output [4:0] portA /* synthesis IO_TYPE="LVCMOS33" PULLMODE = "PULLUP" */;
```

## SLEWRATE

VHDL:

```
ATTRIBUTE SLEWRATE: string;
ATTRIBUTE SLEWRATE OF portF: SIGNAL IS "FAST";
```

Verilog:

```
output [4:0] portA /* synthesis IO_TYPE="LVCMOS33" SLEWRATE = "FAST" */;
```

## CLAMP

VHDL:

```
ATTRIBUTE CLAMP: string;
ATTRIBUTE CLAMP OF portF: SIGNAL IS "ON";
```

Verilog:

```
output [4:0] portA /* synthesis IO_TYPE="LVCMOS33" CLAMP = "ON" */;
```

## HYSTERESIS

VHDL:

```
ATTRIBUTE HYSTERESIS: string;
ATTRIBUTE HYSTERESIS OF portF: SIGNAL IS "ON";
```

Verilog:

```
output [4:0] portA /* synthesis IO_TYPE="LVCMOS25" HYSTERESIS = "ON" */;
```

## Appendix C. sysIO Buffer Design Rules

1. Only one  $V_{CCIO}$  level is allowed in a given bank.
  - a. If  $V_{CCIO}$  for any bank is set to 2.5 V, it is recommended that it be connected to the same power supply as  $V_{CCAUX}$ , thus minimizing leakage. The software will issue a message in the .pad file to the user about this if the  $V_{CCIO}$  of a bank is set to 2.5 V.
2. When an output is configured as an OPENDRAIN, the PULLMODE is set to NONE and the CLAMP setting is set to OFF.
  - a. When an output is configured as an OPENDRAIN, it can be placed independent of  $V_{CCIO}$ .
3. When a ratioed input buffer is placed in a bank with a different  $V_{CCIO}$  (mixed mode), the Pull mode options of Up are no longer available
4. Left and right banks can support LVDS input buffers. True LVDS outputs are supported on 50% of the sysIO pins of left and right banks. True LVDS outputs are available only on the A and B pairs of the I/O pairs of left and right banks. Emulated differential outputs are available on every output pair. Pad information can be found in the data sheet of the pad file.
  - a. The IO\_TYPE attribute for a differential buffer can only be assigned to the TRUE pad. The Lattice Diamond design tool will automatically assign the other I/O of the differential pair to the complementary pad.
5. DIFFRESISTOR termination is available on all sysIO pairs of left and right banks.
6. If none of the pins is used for a given bank, the  $V_{CCIO}$  of the bank should be grounded except the JTAG bank.

## Appendix D. sysIO Attributes Using the Diamond Spreadsheet View User Interface

sysIO buffer attributes can be assigned using the Spreadsheet View in Lattice Diamond design software. The Port Assignments Sheet lists all the ports in a design and all the available sysIO attributes in multiple columns. Click on each of these cells for a list of all the valid I/O preferences for that port. Each column takes precedence over the next. Therefore, when you choose a particular IO\_TYPE, the columns for the PULLMODE, DRIVE, SLEWRATE and other attributes will only list the valid entries for that IO\_TYPE.

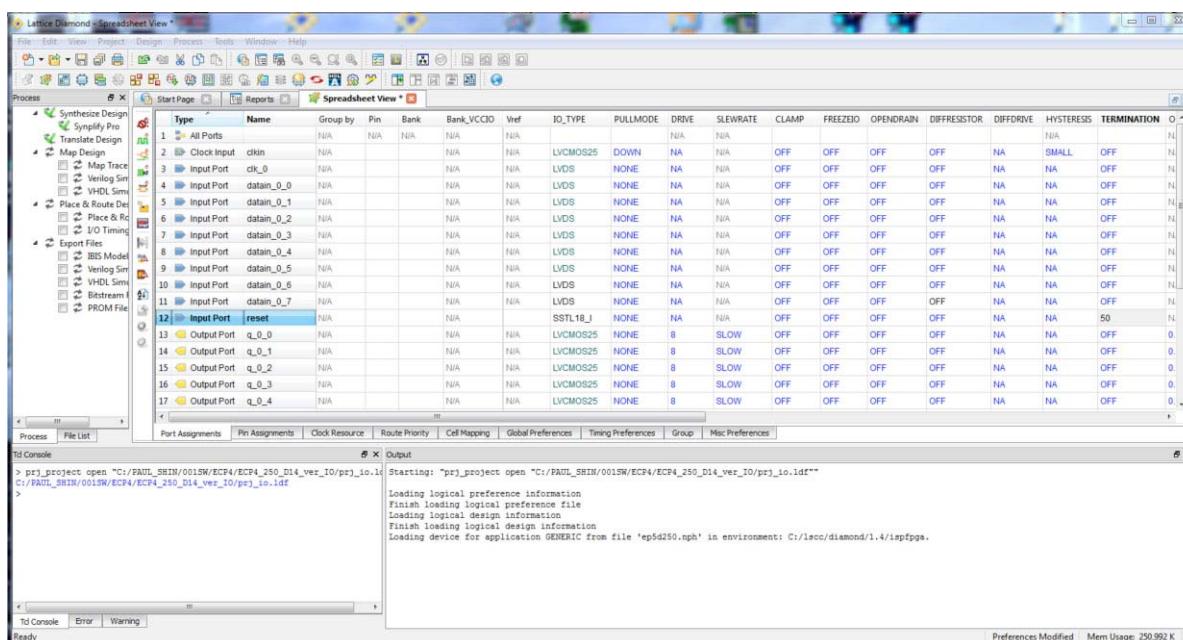
Pin locations can be locked using the Pin column of the Port Assignments Sheet or using the Pin Assignments Sheet. You can right-click on a cell and go to Assign Pins to see a list of available pins.

In Spreadsheet View, go to **Design > Preference PIO DRC** to look for incorrect pin assignments.

You can enter the DIN/DOUT preferences using the Cell Mapping tab. All the preferences assigned using the Spreadsheet view are written into the logical preference file (.lpf).

Figure 10-8 shows the Port Assignments Sheet of the Spreadsheet View. For further information on how to use the Spreadsheet View, refer to the Diamond Help documentation, available in the Help menu option of the software.

**Figure 10-8. Port Attributes Tab of Spreadsheet View**



## Introduction

This usage guide describes the clock resources available in the ECP5™ device architecture. Details are provided for primary clocks, edge clocks, PLLs, the internal oscillator, and clocking elements such as clock dividers, clock multiplexers, and clock stop blocks available in the ECP5 device.

The number of PLLs, Edge clocks, and Clock dividers for each device is listed in Table 11-1.

**Table 11-1. Number of PLLs, Edge Clocks, and Clock Dividers<sup>1</sup>**

Parameter	Description	LFE5-85	LFE5-45	LFE5-25
Number of PLLs	General purpose PLLs.	4	4	2
Number of Edge Clocks	Edge Clocks for high speed applications.	8	8	8
Number of Clock Dividers	Edge Clock Dividers for DDR applications.	4	4	4
Number of PCS Clock Dividers <sup>1</sup>	Clock dividers for domain crossing applications.	2	2	1
Number of DDRDLLs	DDRDLL used to DDR memory and High Speed IO interfaces	4	4	2

1. LFE5U devices do not have PCS Clock Dividers.

It is very important to note that the user needs to validate their pinout so that correct pin placement is used. The Lattice Diamond® tools should be used to validate the pinout while designing the printed circuit board.

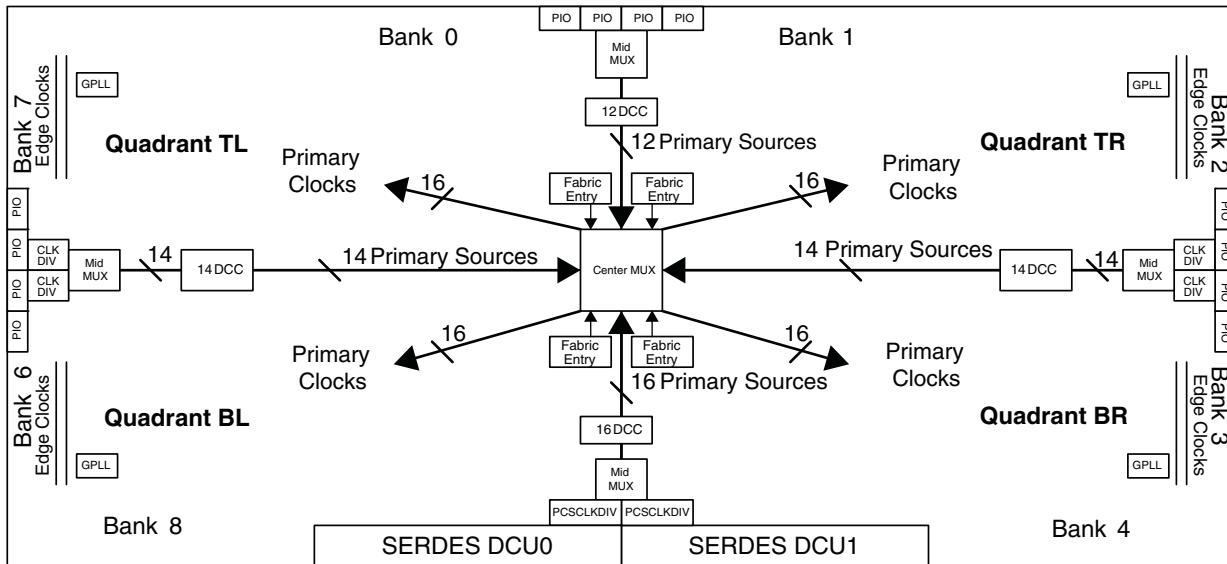
## Clock/Control Distribution Network

ECP5 devices provide global clock distribution in the form of 16 global primary clocks. These Primary clocks can be divided into 16 clocks per each of the four quadrants; however there is a maximum of 60 unique clock input sources. The ECP5 primary clocking structure is enhanced as it features more Edge clock resources, more low-skew Primary clock resources and removes the Secondary clock resources.

## ECP5 Top-Level View

A top level view of the major clocking resources for the ECP5 devices is shown in Figure 11-1.

**Figure 11-1. ECP5 Clocking Structure (LFE5-85)**



Note: LFE4U devices do not have PCS Clock Dividers

# Clocking Architecture Overview

Below is a brief overview of the clocking structure, elements, and PLL. Greater detail is provided starting with the Appendix A. Primary Clock Sources and Distribution section.

## Primary Clock Network

Up to 60 Primary Clock Sources (PLLs, External Inputs, SERDES, etc.) can be selected and routed to the Primary Clock Network. This gives the user an available 60 unique clock domains in the ECP5.

The Primary Clock Network provides low-skew, high fanout clock distribution to all synchronous elements in the FPGA fabric. The Primary Clock Network is divided into four clocking quadrants: Top Left (TL), Bottom Left (BL), Top Right (TR), and Bottom Right (BR). Each of these quadrants has 16 clocks that can be distributed to the fabric in the quadrant. Initially, the Lattice Diamond software automatically routes each clock to all four quadrants; up to a maximum of 16 clocks since each clock is routed to all four quadrants. The user can change how the clocks are routed by specifying a preference in the Lattice Diamond software to locate the clock to specific quadrants.

# Edge Clock Network

Edge clocks are low skew, high speed clock resources used to clock data into/out of the I/O logic of the ECP5. There are two edge clocks per bank located on the left and right sides.

## Overview of Other Clocking Elements

### Edge Clock Dividers (CLKDIVF)

Clock dividers are provided to create the divided down clocks used with the I/O Mux/DeMux gearing logic (SCLK inputs to the DDR) and drives to the Primary Clock routing to the fabric. There are four clock dividers on the ECP5 device.

### PCS Clock Dividers (PCSCLKDIV)

A new clock divider is provided to create phase-matched divided-down clocks for bus-widening/narrowing circuits. A port is provided to dynamically change the divide value of the input clock. The PCSCLKDIV will mainly be used to slow the clock rate from high speed SERDES applications by providing the clocks for domain crossing circuits. There is one PCSCLKDIV per SERDES on an LFE5UM device.

### Dynamic Clock Select (DCSC)

The ECP5 dynamic clock select provides run-time selectable glitchless or non-glitchless operation between two independent clock sources to the primary clock network. This clock select allows the selection of clock sources without leaving the dedicated clock resources in the device. There are two dynamic clock select blocks on the ECP5 device.

### Edge Clock Bridge with Clock Select (ECLKBRIDGECS)

The ECLKBRIDGECS allows non-glitchless ECLK selection between two ECLKs. The ECLKBRIDGECS will allow user bridge ECLK from one side to the other. There are two of these elements and they are used for ECLK clock bridging and ECLK selection.

### Edge Clock Stop (ECLKSYNCFB)

Each ECLK has a block to allow dynamic stopping of the edge clock. This allows the user to start and stop the clock synchronous to an event or external signal. These are important for applications requiring exact clock timing relationships on the inputs, such as DDR memories and video applications.

### Oscillator (OSCG)

An internal programmable rate oscillator is provided. The oscillator can be used for master configuration modes when the FPGA sources the configuration clock, Soft Error Detect (SED), and as a user logic clock source that is available after FPGA configuration. There is one OSCG on the ECP5 device. The oscillator clock output is routed directly to primary clocking.

The oscillator output is not a high-accuracy clock, having a +/- 15% variation in its output frequency. It is mainly used for circuits that do not require a high degree of clock accuracy. Examples of usage would be asynchronous logic blocks such as a timer or reset generator, or other logic that require a constantly running clock.

The STDBY port can be used to turn off the Oscillator output to save power. This port is enabled by a logic '1' and can be connected to a user signal or an I/O pin. The Oscillator must not be turned off when it is needed for operations such as background flash updates or SED.

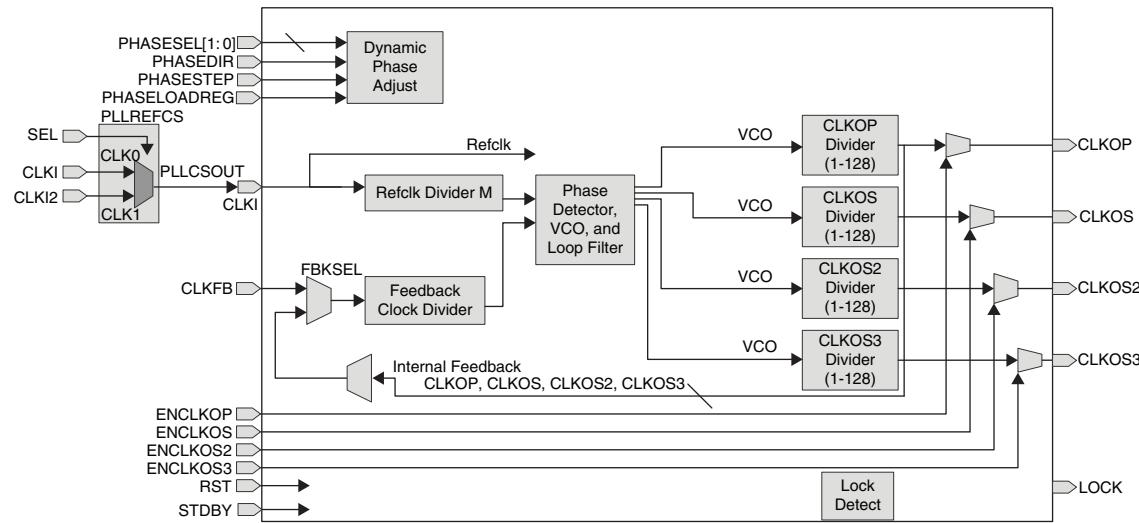
## sysCLOCK PLL Overview

The sysCLOCK PLLs can be used in a variety of clock management applications such as clock injection removal, clock phase adjustment, clock timing adjustment, and frequency synthesis (multiplication and division of a clock). The ECP5™ Clarity Designer PLL GUI shows important timing parameters such as the VCO rate and the PLL loop bandwidth.

PLL Input sources are:

- Dedicated PLL Input Pins
- Primary Clock Routing
- Edge Clock Routing
- FPGA Fabric

**Figure 11-2. ECP5 PLL Block Diagram**



There are four PLLs on the bigger density devices LFE5-85 and LFE5-45 and two PLLs on the smaller density LFE5-25 device. There is one PLL on each corner of the device on the bigger density devices and the smaller density devices have one PLL only on the Lower Left and Lower Right corner. Each PLL has four outputs. All four PLL outputs can go to the Primary Clock network. Only the CLKOP and CLKOS outputs can go to the ECLK network.

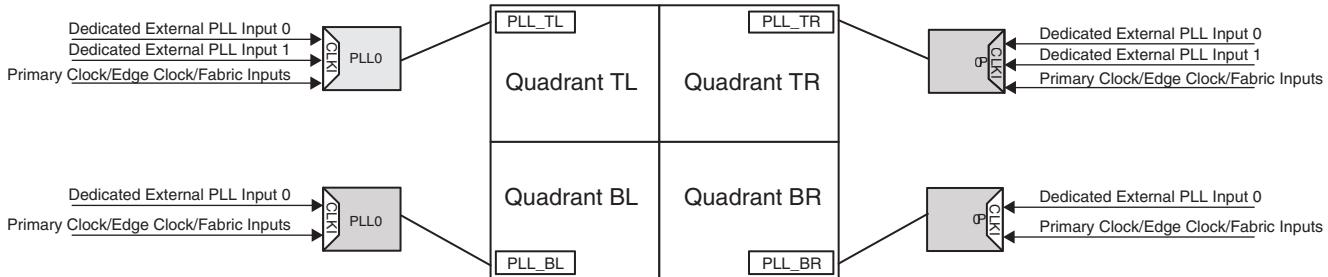
## PLL Features

### Dedicated PLL Inputs

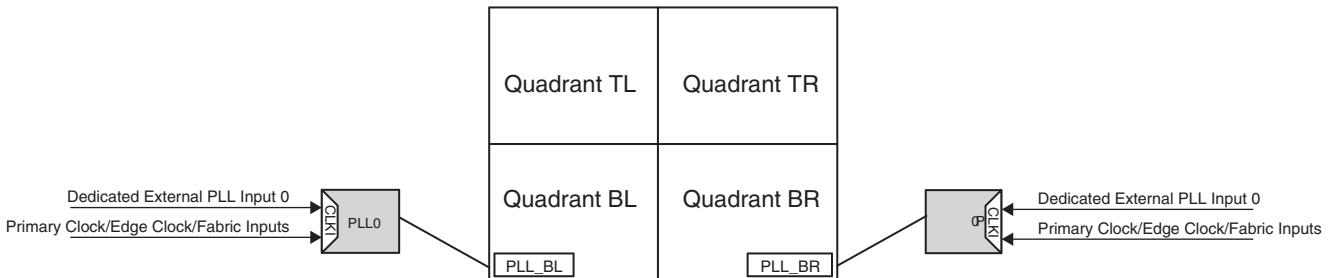
The PLLs have dedicated PLL input pins that are not Primary Clock input pins. Each of the Top Left and Right corner PLLs have two pairs of dedicated PLL input pins, one from the Left/Right side bank and the other from the Top bank. Either one of these can be used as input to the PLL.

The bottom two PLLs have one pair of dedicated input pin on the Left/Right side banks.

**Figure 11-3. PLL Input Pins for LFE5-85 & LFE5-45**



**Figure 11-4. PLL Input Structure for LFE5-25**



### Input PLL Clock Selection (PLLREFCS)

The PLLREFCS component is a non-glitchless multiplexor that allows the user to dynamically select between two PLL input reference clocks. The PLLREFCS has the same input clock sources as the PLL. Since the dedicated PLL inputs are routed to the input of the PLLREFCS components a user can dynamically select between two external reference clock inputs for the top corners of the FPGA. This mux can also be used stand-alone with the PLL in bypass mode for more clock muxing capabilities.

### Clock Injection Delay Removal

The clock injection delay removal feature of the PLL removes the delay associated with the PLL and clock tree. This feature is typically used to reduce clock to out timing and remove the delay differences between the PLL output clock and the data input. This feature is performed by aligning the input clock with a feedback clock from the clock tree. Optional delay may also be added to the feedback path to further reduce the clock injection time.

### Clock Phase Adjustment

The clock phase adjustment feature of the PLL provides the ability to set a specific phase offset between the outputs of the PLL. New to the ECP5 device, phase adjustments can be calculated in much finer increments since the frequency is used to calculate the available phase increments. This feature is detailed further in the Dynamic Phase Adjustment section.

## Frequency Synthesis

The PLL can be used to multiply up or divide down an input clock.

## Additional Features

In addition to the major features, the PLL has several other options that can be used in conjunction with the major modes.

- A Standby mode to reduce power.
- Smaller phase shift increments are supported, based upon frequency.

## Primary Clocks

### Primary Clock Sources

The primary clock network has multiple inputs, called primary clock sources, which can be routed directly to the primary clock routing to clock the FPGA fabric.

The primary clock sources that can get to the primary clock routing are:

- Dedicated Clock Input Pins
- PLL Outputs
- CLKDIV Outputs
- Internal FPGA Fabric Entries (with minimum general routing)
- SERDES/PCS/PCSDIV clocks
- OSC Clock

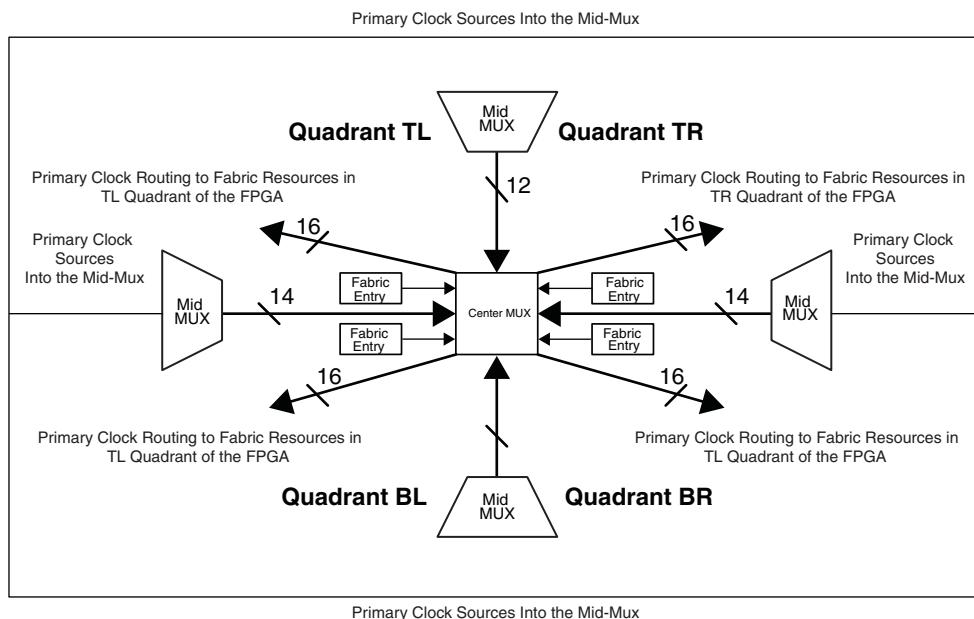
All potential primary clock sources are multiplexed prior to going to the primary clock routing by a mid-mux. There are 56 mid-mux connections and four FPGA fabric connections, 60 total, routed to a multiplexor in the center of the chip called the centermux. From the centermux primary clocks are selected and distributed to the FPGA fabric. The maximum number of unique clock sources is 16 bottom mid-mux sources + 12 top mid-mux sources + 14 left mid-mux sources + 14 right mid-mux sources + 4 direct FPGA fabric entry points (from general routing) = 60. The basic clocking structure is shown in Figure 11-1 and elaborated in [Appendix A. Primary Clock Sources and Distribution](#).

### Primary Clock Routing

The primary clock routing network is made up of low skew clock routing resources with connectivity to every synchronous element of the device. Primary clock sources are selected at the mid-mux, then selected in the centermux and distributed on the primary clock routing to clock the synchronous elements in the FPGA fabric.

The primary clock routing network is divided into four sections, called quadrants. Figure 11-5 is a simplified view of Figure 11-1.

**Figure 11-5. Primary Clock Routing Architecture**



The centermux can source up to 16 independent primary clocks per quadrant which can clock the logic located in that quadrant. The centermux can also route each clock source to all quadrants. The Diamond software will automatically route a primary clock to all four quadrants in the FPGA.

## Dedicated Clock Inputs

The ECP5 device has dedicated pins, called PCLK pins, to bring an external clock source into the FPGA and allow them to be used as FPGA primary clocks. These inputs route directly to the Primary clock network, or to Edge clock routing resources. A dedicated PCLK clock pin must always be used to route an external clock source to FPGA logic and I/O.

If an external input clock is being sourced to a PLL, then in most cases the input clock should use a dedicated PLL input pin. SERDES reference clocks also have dedicated SERDES reference clock pins. The ECP5 device allows a PLL reference clock or a SERDES reference clock to come from an external Primary Clock (PCLK) pin and route through the Primary clock network to drive the reference clock to the SERDES or the input of a PLL.

## PCS Clock Dividers (PCSCLKDIV)

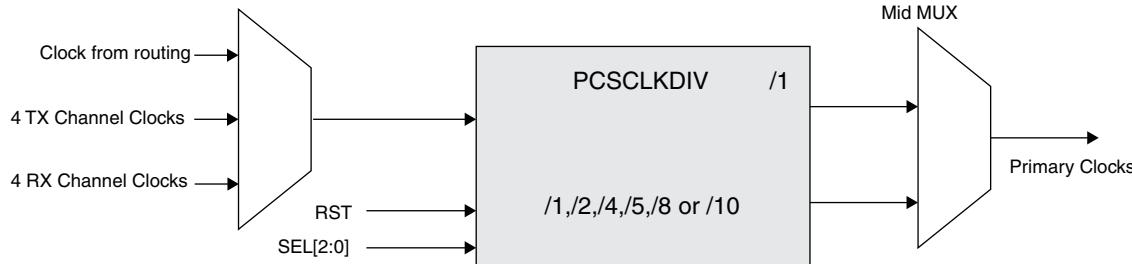
The ECP5 device has a new clock divider called the PCSCLKDIV. The PCSCLKDIV is built with our PCS / IP Core interaction in mind. Its main purpose is to allow our IP cores to do multi-rate clocking for cores that can change data rates. The specific features are:

- Capability to generate different divider clocks
- Input mux to change the divider value on-the-fly.
- Non-glitchless when the divider's select mux input is changed.
- Data width changes in the fabric for widening / narrowing circuits. This element must allow clock domain crossing in the fabric registers.

There is one PCSCLKDIV per SERDES Channel on the bottom of the device where the SERDES blocks are located. The clock outputs of the PCSCLKDIV (CDIV1, CDIVX) are delay and phase matched to each other and has a run-time selectable divider value. The PCSCLKDIV clock input sources are:

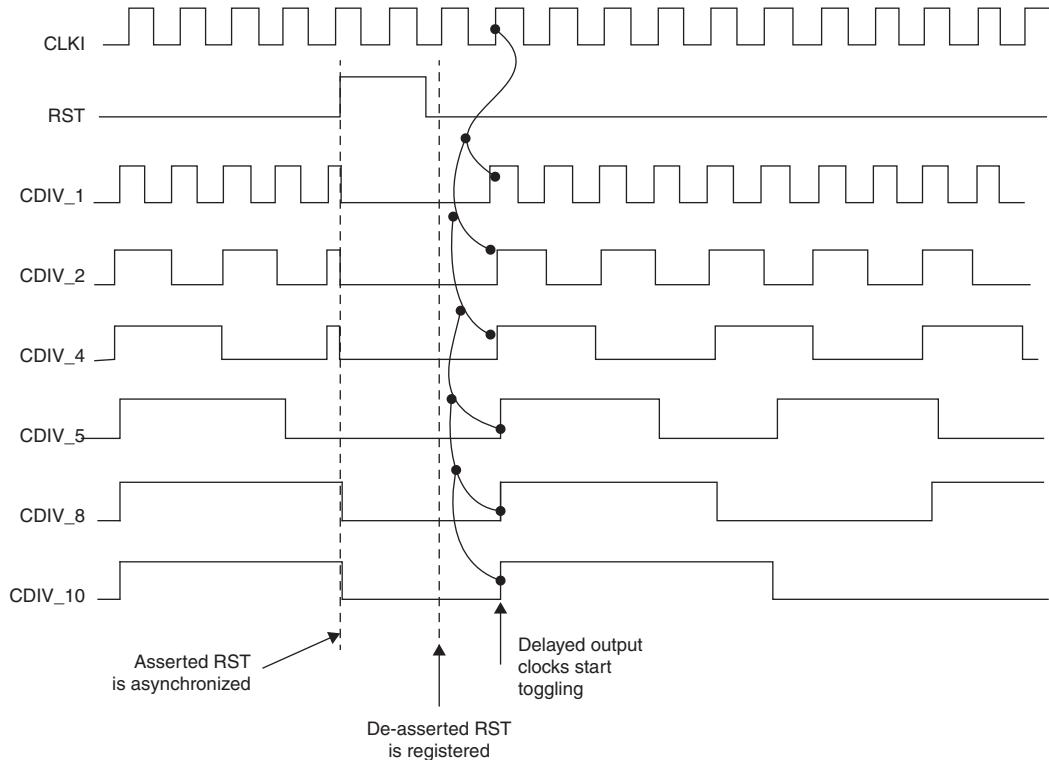
- Up to four TX Channel Clocks
- Up to four RX Channel Clocks
- Clock from Routing

**Figure 11-6. PCSCLKDIV Connection Diagram**



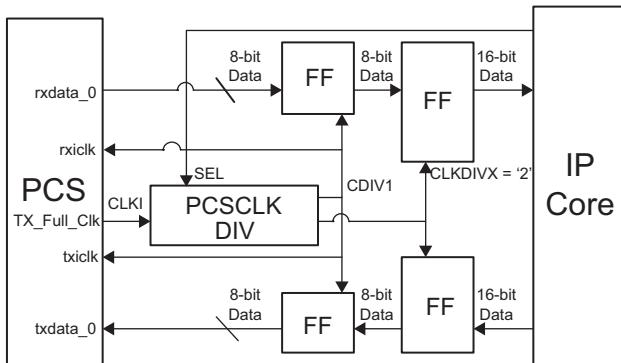
The output PCS channel clocks route directly to the input of the PCSCLKDIV without requiring the use of a primary clock. The CDIV1 and CDIVX outputs route directly to the primary clock routing. Its function is to do bus widening / narrowing circuits in the FPGA fabric to reduce the fabric frequency. The PCSCLKDIV is not suitable for DDR I/O interfaces (ECLK to SCLK domain crossing).

**Figure 11-7. Logical Functional Timing of PCSCLKDIV Block**



An example design is shown in Figure 11-8. It is a flip-flop based clock domain crossing circuit between the CDIV1 and the CDIVX = 2 clock outputs of the PCSCLKDIV. In this example an IP core is using the PCSCLKDIV to change the operating frequency and bus width for a PCS (SERDES) application.

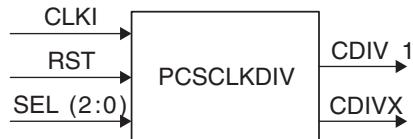
**Figure 11-8. PCSCLKDIV Example Usage**



### PCSCLKDIV Component Definition

The PCSCLKDIV component can be instantiated in the source code of a design as defined in this section. Figure 11-9, Table 11-2 and Table 11-3 define the PCSCLKDIV component. Verilog and VHDL instantiations are included.

**Figure 11-9. PCSCLKDIV Component Symbol**



**Table 11-2. PCSCLKDIV Component Port Definition**

Port Name	I/O	Description
CLKI	I	Clock Input.
RST	I	Reset input – Active High, Asynchronously forces all outputs low. - RST = 0 Clock output outputs are active - RST = 1 Clock output outputs are OFF
SEL(2:0)	I	Signals are used for dynamically changing divide value. - SEL = 000, Output = '0' - SEL = 001, $\div 1$ - SEL = 010, $\div 2$ - SEL = 011, $\div 4$ - SEL = 100, $\div 5$ - SEL = 101, $\div 8$ - SEL = 110, $\div 10$ - SEL = 111, Output = '0'
CDIV1	O	Output is the same frequency as the input clock. Delay matched to CDIVX.
CDIVX	O	Divide by 1, 2, 4, 5, 8, 10 Output Port

**Table 11-3. PCSCLKDIV Component Attribute Definition**

Name	Description	Value	Default
GSR	Enable or Disable Global Reset Signal to Component	ENABLED, DISABLED	DISABLED

## PCSCLKDIV Usage in VHDL

### Component Instantiation

```
Library lattice;
use lattice.components.all;
```

### Component and Attribute Declaration

```
component PCSCLKDIV
Generic (GSR      : string);
Port    (RST      : in STD_LOGIC;
         CLKI     : in STD_LOGIC;
         SEL      : in STD_LOGIC_VECTOR(2 downto 0);
         CDIV1   : in STD_LOGIC;
         CDIVX   : out STD_LOGIC);
end component;
```

### PCSCLKDIV Instantiation

```
attribute GSR : string;
attribute GSR of I1 : label is "DISABLED";
I1: PCSCLKDIV
generic map (
  GSR      => "DISABLED")
port map (
  RST      => RST
 ,CLKI     => CLKI
 ,SEL      => SEL
 ,CDIV1   => CDIV1
 ,CDIVX   => CDIVX);
```

## PCSCLKDIV Usage in Verilog

### Component and Attribute Declaration

```
module PCSCLKDIV (RST, CLKI, SEL, CDIV1, CDIVX);
parameter GSR = "DISABLED"; // "ENABLED", "DISABLED"
input      RST, CLKI;
input [2:0] SEL;
output     CDIVX;
endmodule
```

### PCSCLKDIV Instantiation

```
defparam I1.GSR = "DISABLED";
PCSCLKDIV I1 (
  .RST      (RST)
 ,.CLKI     (CLKI)
 ,.SEL      (SEL)
 ,.CDIV1   (CDIV1)
 ,.CDIVX   (CDIVX));
```

## Dynamic Clock Select (DCSC)

The ECP5 device has two dynamic clock select (DCS) blocks that can drive to any or all the quadrants. The inputs to the DCS block come from all the output of MIDMUXs and local routing that is located at the center of the PLC array core. The output of the DCS is connected to one of the inputs of Primary Clock Center MUX.

Each DCS can select between two independent input clock sources. There are two modes of clock switching, a glitchless mode of operation, or a non glitchless mode where the DCS operates as a regular mux. Figure 11-10 and Figure 11-11 show the glitchless / non-glitchless modes of operation. It should be noted that the clock source used as feedback into the GPLL should not be switched when using the DCS as it will lead to loss of lock for the GPLL.

As indicated there are two modes of clock switching. In non-glitchless mode (input MODESEL='1') the DCS acts like a regular mux, allowing glitches and runt pulses on the output, depending on when the clock is switched. The SEL and MODESEL inputs are driven by signals from the general routing fabric and are used to support the clock switching feature in the DCS.

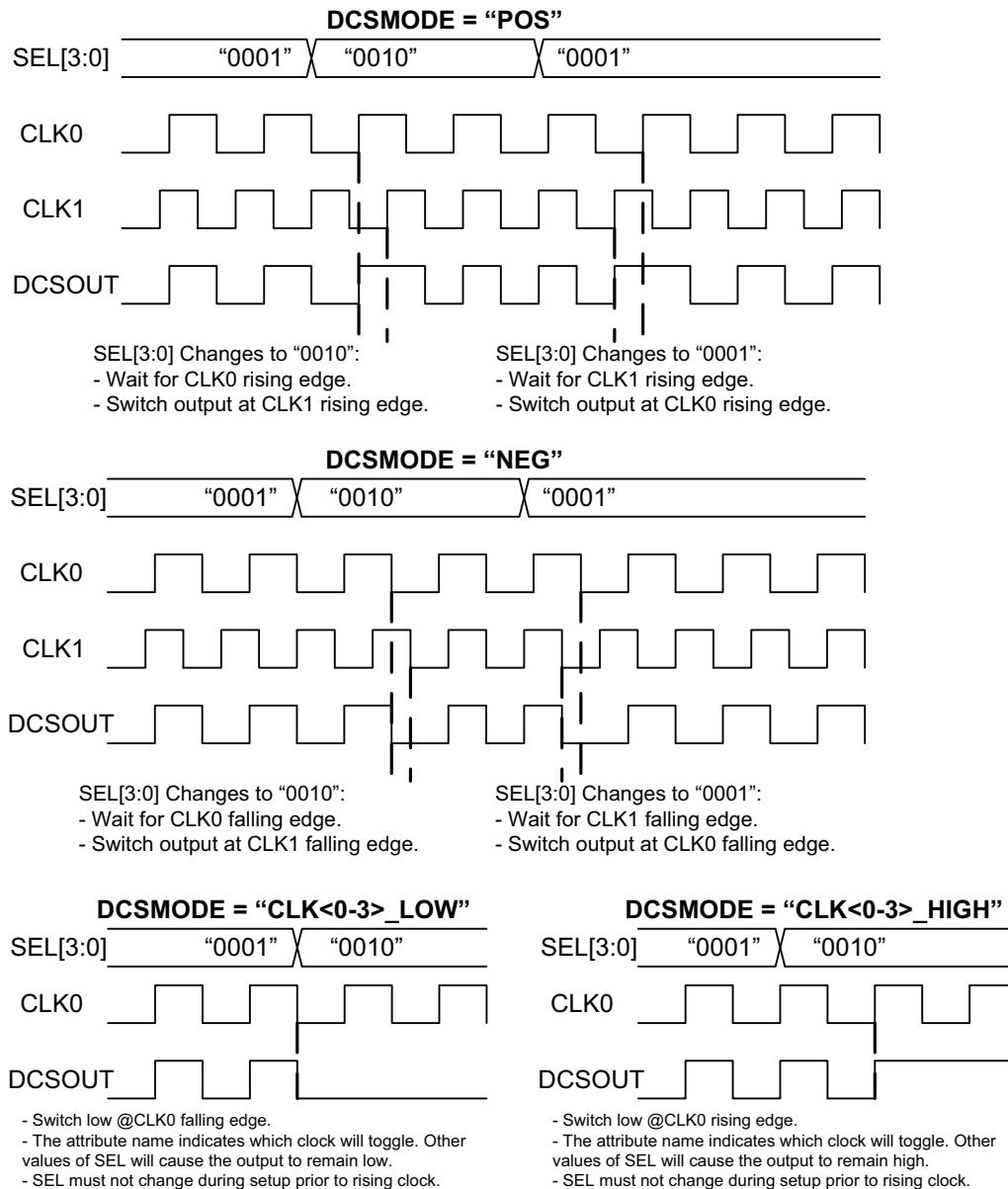
In glitchless mode (input MODESEL='0') the DCS avoids glitches or runt pulses on the output clock, regardless of when the enable signal is toggled. In order to switch between clocks glitchlessly, both input clocks must be oscillating; otherwise the output clock will be '0'.

For glitchless operation, the “DCSMODE” attribute sets the behavior of the DCS output. The additional attribute values and their functions are shown in Table 11-5.

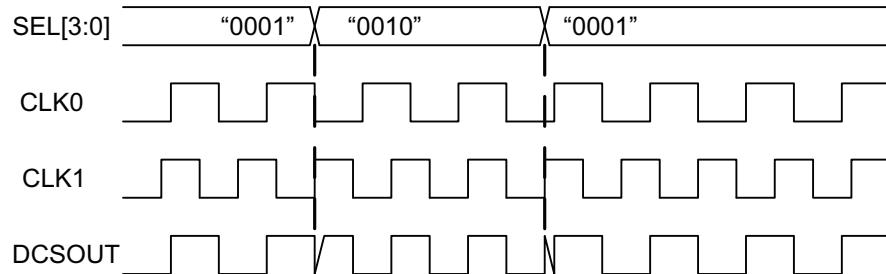
## DCS Timing Diagrams

Figure 11-10 shows timing diagrams to show the operation of the DCS in Glitchless mode in conjunction with the DCSMODE attribute.

**Figure 11-10. Timing Diagrams by “DCSMODE” Attribute Setting, Glitchless Operation (MODESEL=’0’)**



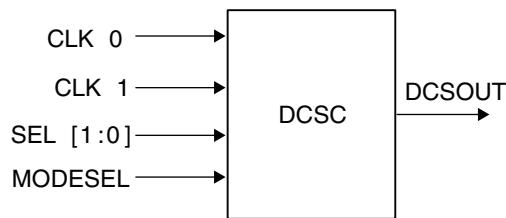
**Figure 11-11. DCS Operation, Non-Glitchless Operation (MODESEL='1')**



### DCSC Component Definition

The DCSC component can be instantiated in the source code of a design as defined in this section.

**Figure 11-12. DCSC Component Symbol**



**Table 11-4. DCSC Component Port Definition**

Port Name	I/O	Description
CLK0	I	Clock Input port 0 – Default.
CLK1	I	Clock Input port 1
SEL[1:0]	I	Input Clock Select bit (See Table 6 below)
MODESEL	I	Selects Glitchless ('0') or Non-Glitchless ('1') behavior.
DCSOUT	O	Clock Output Port

## DCSMODE Attribute

Table 11-5 provides the behavior of the DCS output based on the setting of the DCSMODE attribute when the pin MODESEL = '0'. The MODESEL pin is dynamic and can toggle during operation. Table 11-5 is only valid when MODESEL = '0'.

**Table 11-5. DCSC – DCSMODE Attribute Table when MODESEL='0'**

Attribute Name	Description	SEL[1:0]			Attribute Value
		2'b01	2'b10	2'b00/2'b11	
DCSMODE Attributes	Falling Edge Triggered	Clk0	Clk1	0	NEG
	Rising Edge Triggered	Clk0	Clk1	1	POS
	Disabled Output is Low, Clk0	Clk0	0	0	CLK0_LOW
	Disabled Output is High, Clk0	Clk0	1	1	CLK0_HIGH
	Disabled Output is Low, Clk0	0	Clk1	0	CLK1_LOW
	Disabled Output is High, Clk0	1	Clk1	1	CLK1_HIGH
	Clk0 Buffered	Clk0	Clk0	Clk0	CLK0
	Clk1 Buffered	Clk1	Clk1	Clk1	CLK1
	Tie Low	0	0	0	LOW
	Tie High	1	1	1	HIGH
MODESEL= "1"	Non-Glitchless	Clk0	Clk1	0	-

## DCSC Usage in VHDL

### Component Instantiation

```
Library lattice;
use lattice.components.all;
```

### Component and Attribute Declaration

```
COMPONENT DCSC
  GENERIC(DCSMODE : string := "POS");
  PORT  (CLK0      : IN STD_LOGIC;
         CLK1      : IN STD_LOGIC;
         SEL       : IN STD_LOGIC_VECTOR(1 downto 0);
         MODESEL   : IN STD_LOGIC;
         DCSOUT    : OUT STD_LOGIC);
END COMPONENT;
```

### DCSC Instantiation

```
attribute DCSMODE : string;
attribute DCSMODE of DCSinst0 : label is "POS";
I1: DCSC
generic map(
  DCSMODE  => "POS")
port map  (
  CLK0      => CLK0
 ,CLK1      => CLK1
 ,SEL       => SEL
 ,MODESEL   => MODESEL
 ,DCSOUT    => DCSOUT);
```

## DCSC Usage in Verilog

### Component and Attribute Declaration

```
module DCSC(CLK0,CLK1,CLK2,CLK3,SEL,MODESEL,DCSOUT);
input CLK0;
input CLK1;
input [1:0] SEL;
input MODESEL;
output DCSOUT;
endmodule
```

### DCSC Instantiation

```
defparam DCSInst0.DCSMODE = "POS";
DCSC DCSInst0 (
    .CLK0(CLK0)
, .CLK1(CLK1)
, .SEL(SEL)
, .MODESEL(MODESEL)
, .DCSOUT(DCSOUT) );
```

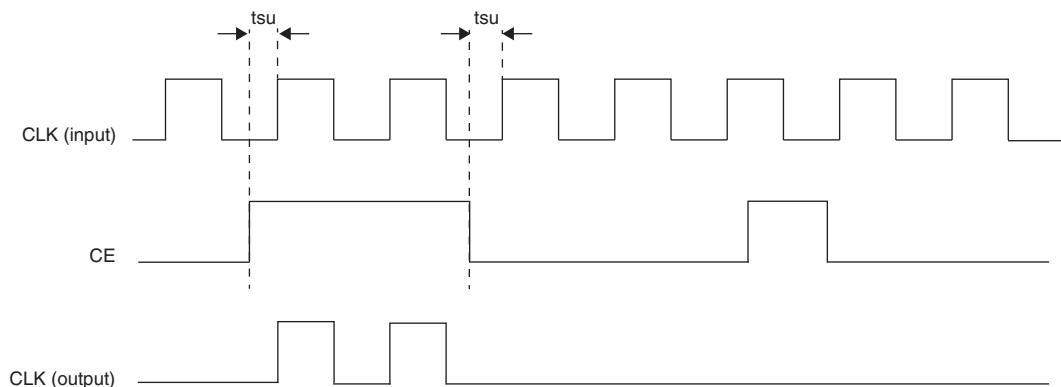
## Dynamic Clock Control (DCCA)

The ECP5 device has a Dynamic Clock Control feature which allows internal logic dynamically to enable or disable quadrant primary clock network. The disable function will not create glitch and increase the clock latency to the primary clock network. Also, this dynamic clock control function can be disabled by a configuration memory fuse to always enable the primary clock network.

This DCC controls the clock sources from the Primary CLOCK MIDMUX before they are fed to the Primary Center MUXs that drive the quadrant clock network. When a clock network is disabled, all the logic fed by that clock does not toggle, hence reducing the overall power consumption of the device.

The ECP5 device clock architecture allows both DCC and DCS to function at the same. It should be noted that the clock source used as feedback into the GPLL should always keep enable signal “high” in the DCC, otherwise it will lead to loss of lock for the GPLL when toggling the enable signal.

**Figure 11-13. Glitchless DCC Functional Waveform**



## DCCA Component Definition

The DCCA component can be instantiated in the source code of a design as defined in this section. Figure 11-14 and Table 11-6 show the DCCA definitions.

**Figure 11-14. DCCA Component Symbol**



**Table 11-6. DCCA Component Port Definition**

Port Name	I/O	Description
CLKI	I	Clock Input port.
CE	I	Clock Enable port – CE = 0 CLKO is disabled (CLKO = '0') – CE = 1 CLKO is enabled (CLKO = CLKI)
CLKO	O	Clock Output Port

## DCCA Usage in VHDL

### Component Instantiation

```

library lattice;
use lattice.components.all;
Component and Attribute Declaration
COMPONENT DCCA
PORT
  (CLKI :IN STD_LOGIC;
   CE   :IN STD_LOGIC;
   CLKO :OUT STD_LOGIC);
END COMPONENT;

```

### DCCA Instantiation

```

I1: DCCA
port map (
  CLKI  => CLKI
 ,CE    => CE
 ,CLKO  => CLKO);
DCCA Usage in Verilog
Component and Attribute Declaration
module DCCA(CLKI,CE,CLKO);
input CLKI;
input CE;
output CLKO;
endmodule

```

### DCCA Instantiation

```

DCCA DCSInst0 (
  .CLKI  (CLKI)
 ,.CE    (CE)
 ,.CLKO  (CLKO));

```

## Internal Oscillator (OSCG)

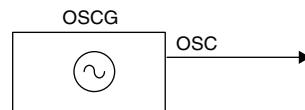
The OSCG element performs multiple functions on the ECP5 device. It is used for configuration, SED, as well as optionally in user mode. In user mode, the OSCG element has the following features:

- It permits a design to be fully self-clocked, as long as the quality of the OSCG element's silicon-based oscillator is adequate.
- If it's unused it can be turned off for power savings.
- It has an input to dynamically control standby/normal operation.
- It has a direct connection to primary clock routing through the left mid-mux.
- It can be configured for operation at a wide range of frequencies via configuration bits.

### OSCG Component Definition

The OSCG component can be instantiated in the source code of a design as defined in this section. Figure 11-15 and Table 11-7 below show the OSCG definitions.

**Figure 11-15. OSCG Component Symbol**



**Table 11-7. OSCG Component Port Definition**

Port Name	I/O	Description
OSC	O	Clock Output Port

**Table 11-8. OSCG Component Attribute Definition**

Name	Description	Value	Default
DIV	Selects Divider Value	2 (~155.0 MHz) – 128 (~2.4 MHz)	2.4 MHz

### OSCG Usage in VHDL

#### Component Instantiation

```
Library lattice;
use lattice.components.all;
```

#### Component and Attribute Declaration

```
component OSCG
Port      (OSC      : out STD_LOGIC);
end component;
```

#### OSCG Instantiation

```
I1: OSCG
port map (
    OSC      => OSC);
```

## OSCG Usage in Verilog

### Component and Attribute Declaration

```
module OSCG (OSC);
output OSC;
endmodule
```

### OSCG Instantiation

```
OSCG I1 (.OSC(OSC))
```

## Edge Clocks

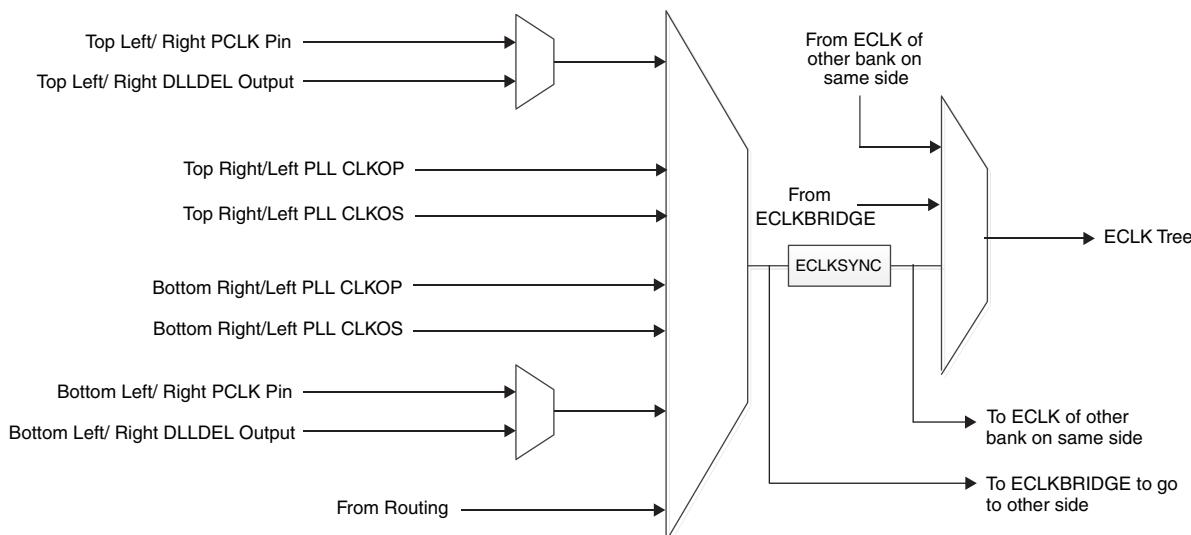
Each ECP5 device I/O bank has four ECLK resources. These clocks, which have low injection time and skew, are used to clock I/O registers. Edge clock resources are designed for high speed I/O interfaces with high fan-out capability. Refer to Appendix B. EDGE CLOCK Sources and Connectivity for detailed information on the ECLK locations and connectivity.

The sources of edge clocks are:

- Dedicated Clock (PCLK) pins
- DLLDEL output
- PLL outputs (CLKOP & CLKOS)
- ECLK Bridge
- Internal nodes

The ECP5 device has Edge Clock (ECLK) at the Left side and right side of the device. There are two ECLK network per bank IO. ECLK Input MUX collects all clock sources available shown in figure below. There are two ECLK Input MUXs, one on the left side and one on the right side. Each of these MUX will generate total of four ECLK Clock sources. Two of them drive the upper IO bank and two of them drive the lower IO bank. Two out of four also drive the ECLK Bridge Switch Block to form an ECLK Bridge high speed clock before drive the ECLK Tree Network.

**Figure 11-16. Edge Clock Sources Per Bank**



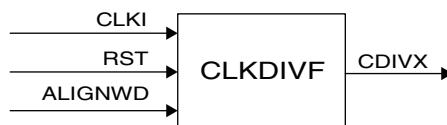
## Edge Clock Dividers (CLKDIVF)

There are four edge clock dividers available in the ECP5 device, two per side of the device. The clock divider provides a single divided output with available divide values of 2 or 3.5. The inputs to the clock dividers are the edge clocks, PLL outputs and Primary Clock Input pins. The outputs of the clock divider drive the primary clock network and are mainly used for DDR I/O domain crossing.

### CLKDIVF Component Definition

The CLKDIVF component can be instantiated in the source code of a design as defined in this section. Figure 11-17, Table 11-9, and Table 11-10 define the CLKDIVF component. Verilog and VHDL instantiations are included.

**Figure 11-17. CLKDIVF Component Symbol**



**Table 11-9. CLKDIVF Component Port Definition**

Port Name	I/O	Description
CLKI	I	Clock Input.
RST	I	Reset input – Active High, Asynchronously forces all outputs low.
- RST = 0 Clock outputs are active		
- RST = 1 Clock outputs are OFF		
ALIGNWD	I	Signal is used for word alignment.
When enabled it slips the output one cycle relative to the input clock.		
CDIVX	O	Divide by 2 or 3.5 Output Port

**Table 11-10. CLKDIVF Component Attribute Definition**

Name	Description	Value	Default
GSR	Enable or Disable Global Reset Signal to Component	ENABLED, DISABLED	DISABLED
DIV	CLK Divider Value	2.0 or 3.5	2.0

The ALIGNWD input is intended for use with high-speed data interfaces such as DDR or 7:1 LVDS Video.

### CLKDIVF Usage in VHDL

#### Component Instantiation

```
library lattice;
use lattice.components.all;
```

#### Component and Attribute Declaration

```
component CLKDIVF
Generic (DIV      : string;
          GSR      : string);
Port    (RST      : in  STD_LOGIC;
          CLKI     : in  STD_LOGIC;
          ALIGNWD  : in  STD_LOGIC;
          CDIVX    : out STD_LOGIC);
end component;
```

### CLKDIVF Instantiation

```

attribute DIV : string;
attribute DIV of I1 : label is "2.0";
attribute GSR : string;
attribute GSR of I1 : label is "DISABLED";

I1: CLKDIVF
generic map (DIV      => "2.0"
             ,GSR     => "DISABLED")
port map    (RST      => RST
             ,CLKI     => CLKI
             ,ALIGNWD => ALIGNWD
             ,CDIVX   => CDIVX);

```

### CLKDIVF Usage in Verilog

#### Component and Attribute Declaration

```

module CLKDIVF (RST, CLKI, ALIGNWD, CDIVX);

parameter DIV = "2.0";           // "2.0", "3.5"
parameter GSR = "DISABLED";     // "ENABLED", "DISABLED"

input  RST, CLKI, ALIGNWD;
output CDIVX;
endmodule

```

### CLKDIVF Instantiation

```

defparam I1.DIV = "2.0";
defparam I1.GSR = "DISABLED";
CLKDIVF I1 (
    .RST      (RST)
    ,.CLKI    (CLKI)
    ,.ALIGNWD (ALIGNWD)
    ,.CDIVX   (CDIVX));

```

## Edge Clock Bridge (ECLKBRIDGECS)

ECLK Clock Bridge provides to bridge the ECLK for banks on the same side or the ECLK of the left side and the right side. The ECLK Bridge enhances the communication of high speed clocks of the two edges with minimum skew to ECLK tree.

There are two ECLK bridge components in the ECP5 device. There are two ECLK muxes on the left and two ECLK muxes on the right and they allow a user to bridge edge clocks to the left and right sides of the chip with minimal skew.

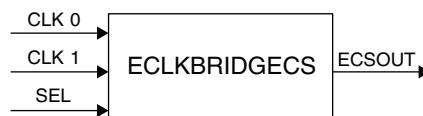
In the edge clock bridge there is a non-glitchless clock select mux that allows a design to switch between two different clock sources for each edge clock. This clock select mux is instantiated using the ECLKBRIDGECS primitive. Not all edge clocks can drive the ECLKBRIDGECS, one of the two ECLKs on the left and right side can be bridged. When connected to the ECLKBRIDGECS it would use up the ECLK1 of both banks on the left side and ECLK0 of both banks on the right side of the device.

## ECLKBRIDGECS Component Definition

The ECLKBRIDGECS component must be instantiated in the source code of a design in order to bridge Edge clocks. Figure 11-18 and Table 11-11 define the ECLKBRIDGECS.

Note that an ECLKSYNCB module has to be always used after the ECLKBRIDGECS instance to be able to connect to the ECLK tree.

**Figure 11-18. ECLKBRIDGECS Component Symbol**



**Table 11-11. ECLKBRIDGECS Component Port Definition**

Port Name	I/O	Description
CLK0	I	Clock Input Port 0 – Default
CLK1	I	Clock Input Port 1
SEL	I	Select Port – SEL = 0 for CLK0 – SEL = 1 for CLK1
ECSOUT	O	Clock Output Port

## ECLKBRIDGECS Usage in VHDL

### Component Instantiation

```
Library lattice;
use lattice.components.all;
```

### Component and Attribute Declaration

```
COMPONENT ECLKBRIDGECS
PORT  (CLK0      :IN STD_LOGIC;
       CLK1      :IN STD_LOGIC;
       SEL       :IN STD_LOGIC;
       ECSOUT   :OUT STD_LOGIC);
END COMPONENT;
```

### ECLKBRIDGECS Instantiation

```
I1: ECLKBRIDGECS
port map  (
    CLK0      => CLK0
  ,CLK1      => CLK1
  ,SEL       => SEL
  ,ECSOUT   => ECSOUT);
```

## ECLKBRIDGECS Usage in Verilog

### Component and Attribute Declaration

```
module ECLKBRIDGECS (CLK0,CLK1,SEL,ECSOUT);
input CLK0;
input CLK1;
input SEL;
output ECSOUT;
endmodule
```

### DCSC Instantiation

```
ECLKBRIDGECS ECSInst0 (
    .CLK0      (CLK0)
    , .CLK1      (CLK1)
    , .SEL       (SEL)
    , .ECSOUT   (ECSOUT) );
```

## Edge Clock Synchronization (ECLKSYNCB)

ECP5 devices have dynamic edge clock synchronization control (ECLKSYNCB) which allows each edge clock to be disabled or enabled glitchlessly from core logic if desired. This allows the designer to synchronize the edge clock to an event or external signal if desired. It also allows the design to dynamically disable a clock and its associated logic in the design when it is not needed and thus save power. Applications such as DDR2, DDR3 and 7:1 LVDS for displays will use this component for clock synchronization.

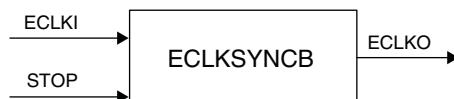
### ECLKSYNCB Component Definition

The ECLKSYNCB component can be instantiated in the source code of a design as defined in this section. Asserting the STOP control signal has the ability to stop the edge clock in order to synchronize the signals derived from ECLK and used in high speed DDR mode applications as DDR memory, generic DDR and 7:1 LVDS.

Control signal STOP is synchronized with ECLK when asserted. When control signal STOP is asserted, the clock output will be forced to low after the fourth falling edge of the input ECLKI. When the STOP signal is released, the clock output starts to toggle at the fourth (4th) rising edge of the input ECLKI clock.

Figure 11-19 and Figure 11-12 show the ECLKSYNCB component definition.

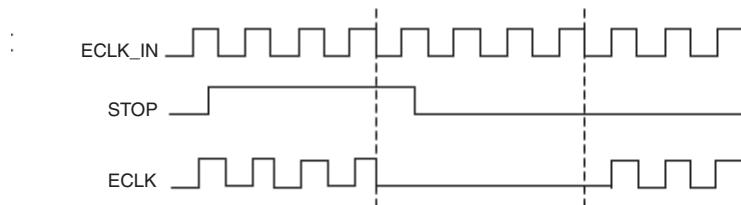
**Figure 11-19. ECLKSYNCB Component Symbol**



**Table 11-12. ECLKSYNCB Component Port Definition**

Port Name	I/O	Description
ECLKI	I	Clock Input port.
STOP	I	Control signal to stop Edge Clock – STOP = 0 Clock is Active – STOP =1 Clock is Off
ECLKO	O	Clock Output Port

**Figure 11-20. ECLKSYNC Functional Waveform**



## ECLKSYNCB Usage in VHDL

### Component Instantiation

```
Library lattice;
use lattice.components.all;
```

### Component and Attribute Declaration

```
COMPONENT ECLKSYNCB
PORT  (ECLKI  :IN STD_LOGIC;
       STOP    :IN STD_LOGIC;
       ECLKO   :OUT STD_LOGIC);
END COMPONENT;
```

### ECLKSYNCB Instantiation

```
I1: ECLKSYNCB
port map (
    ECLKI  => ECLKI
  ,STOP    => STOP
  ,ECLKO   => ECLKO);
```

### ECLKSYNCB Usage in Verilog

```
Component and Attribute Declaration
module ECLKSYNCB (ECLKI,STOP,ECLKO);
input ECLKI;
input STOP;
output ECLKO;
endmodule
```

### ECLKSYNCB Instantiation

```
ECLKSYNCB ECLKSYNCInst0 (
  .ECLKI  (ECLKI)
, .STOP    (STOP)
, .ECLKO   (ECLKO));
```

## General Routing for Clocks

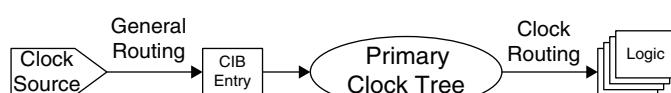
The ECP5 device architecture supports the ability to use data routing, or general routing, for a clock. This capability is intended to be used for small areas of the design to allow additional flexibility in linking dedicated clocking resources and building very small clock trees. General routing cannot be used for edge clocks for applications that use the DDR registers in the I/O components of the FPGA.

Software will limit the distance of a general routing based (gated) clock to one PLC in distance to a primary clock entry point. If the software cannot place the clock gating logic close enough to a primary clock entry point then an error will occur:

ERROR – par: Unable to reach a primary clock entry point for general route clock <net> in the minimum required distance of one PLC.

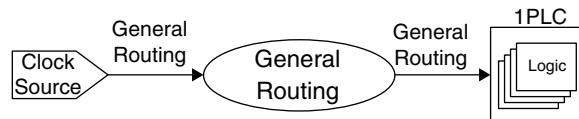
There are multiple entry points to the Primary clock routing throughout the ECP5 device fabric. In this case it is recommended to add a preference for this gated clock to use primary routing.

**Figure 11-21. Gated Clock to the Primary Clock Routing**



For a very small clock domain, the user can limit the distance of a general routing based (gated) clock to one PLC in distance to the logic it clocks. The user must group this logic (UGROUP) with a BBOX = “1, 1” (see Diamond Help > Constraints Reference Guide > Preferences > UGROUP) as well as specify a “PROHIBIT PRIMARY” on the generated clock. If the software cannot place the logic tree within the BBOX, then an error message will occur.

**Figure 11-22. Gated Clock to Small Logic Domain**



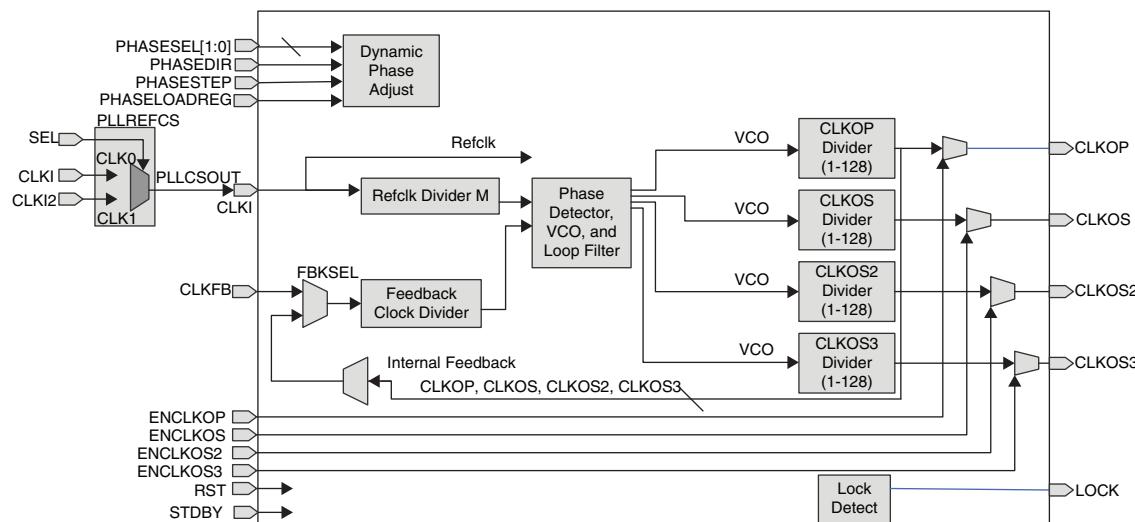
## General routing PCLK pins

Some Dedicated pins (GR\_PCLK) can access some CIBs along the edge of the device and can drive the primary clock routing from this CIB. These will go through some general routing but have direct access to the primary clock from this dedicated CIB. There are four at the left side and right side, four at the top side of the device. These pins can be used when user runs out of PCLK pins. Note that for any DDR interface, it is still required to use dedicated clock pins and clock trees.

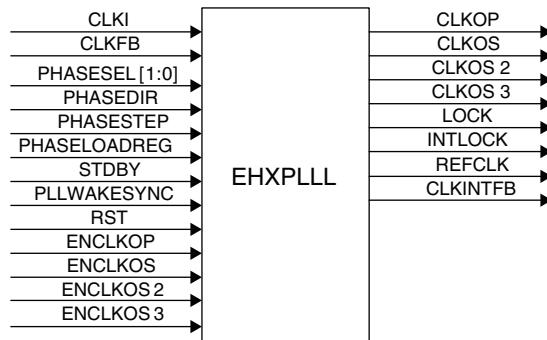
## sysCLOCK PLL

The ECP5 PLL provides features such as clock injection delay removal, frequency synthesis, and phase adjustment. Figure 11-23 shows a block diagram of the ECP5 PLL.

**Figure 11-23. ECP5 PLL Block Diagram**



**Figure 11-24. PLL Component Instance**



**Table 11-13. PLL Component Port Definition**

Signal	I/O	Description
CLKI	I	Input Clock to PLL.
CLKFB	I	Feedback Clock
CLKINTFB	O	Internal Feedback Clock
PHASESEL[1:0]	I	Select the output affected by Dynamic Phase adjustment.
PHASEDIR	I	Dynamic Phase adjustment direction.
PHASESTEP	I	Dynamic Phase adjustment step.
PHASELOADREG	I	Load dynamic phase adjustment values into PLL..
STDBY	I	Standby signal to power down the PLL.
RST	I	Resets the whole PLL.
ENCLKOP	I	Enable PLL output CLKOP
ENCLKOS	I	Enable PLL output CLKOS
ENCLKOS2	I	Enable PLL output CLKOS2
ENCLKOS3	I	Enable PLL output CLKOS3
PLLWAKESYNC	I	Enable PLL switching from internal feedback to user feedback path when PLL wake up
CLKOP	O	PLL main output clock.
CLKOS	O	PLL output clock.
CLKOS2	O	PLL output clock.
CLKOS3	O	PLL output clock.
LOCK	O	PLL LOCK to CLKI, Asynchronous signal. Active high indicates PLL lock.
INTLOCK	O	Internal Lock Signal
REFCLK	O	Output of Reference clock mux

## Functional Description

### Refclk (CLKI) Divider

The CLKI divider is used to control the input clock frequency into the PLL block. The valid input frequency range is specified in the device data sheet.

### Feedback Loop (CLKFB) Divider

The CLKFB divider is used to divide the feedback signal, effectively multiplying the output clock. The VCO block increases the output frequency until the divided feedback frequency equals the input frequency. The output of the feedback divider must be within the phase detector frequency range specified in the device data sheet. This port is only available to user interface when “user clock” option is selected for feedback clock, otherwise this port will be connected by the tool to appropriate signal as selected by the user in the software.

### Output Clock Dividers (CLKOP, CLKOS, CLKOS2, CLKOS3)

The output clock dividers allow the VCO frequency to be scaled up to the maximum range to minimize jitter. Each of the output dividers is independent of the other dividers and each uses the VCO as the source by default. Each of the output dividers can be set to a value of 1 to 128.

### Phase Adjustment (Static Mode)

The CLKOP, CLKOS, CLKOS2, and CLKOS3 outputs can be phase adjusted relative to the enabled unshifted output clock. New to the ECP5 device, phase adjustments are now calculated values in the software tools based on VCO clock frequency. This provides a finer phase shift depending on the required frequency. The clock output selected as the feedback cannot use the static phase adjustment feature since it will cause the PLL to unlock. For example if the FB\_MODE is INT\_OP or CLKOP, then there should be no phase shift on CLKOP. Similar restriction would apply on other clocks.

### Phase Adjustment (Dynamic Mode)

The phase adjustments can also be controlled in a dynamic mode using the PHASESEL, PHASEDIR, PHASESTEP, and PHASELOADREG ports. The clock output selected as the feedback should not use the dynamic phase adjustment feature. Please see the Dynamic Phase Adjustment section in this document for usage details. The clock output selected as the feedback cannot use the dynamic phase adjustment feature since it will cause the PLL to unlock. For example if the FB\_MODE is INT\_OP or CLKOP, then there should be no phase shift on CLKOP. Similar restriction would apply on other clocks.

## PLL Features

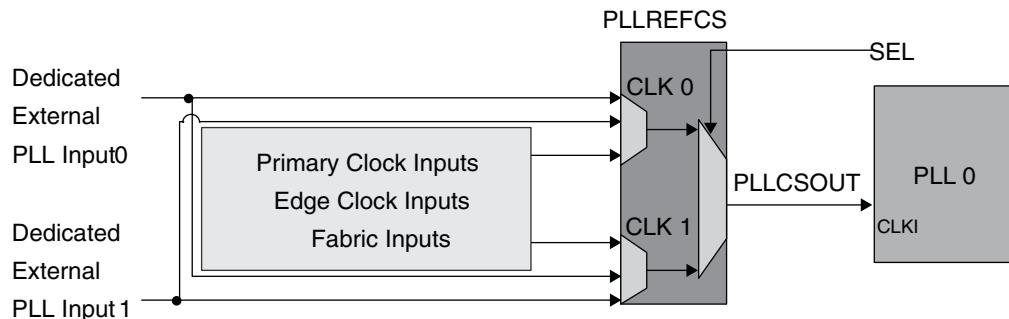
### Dedicated PLL Inputs

Every PLL has a dedicated low skew input that will route directly to its reference clock input. These are the recommended inputs for a PLL. It is possible to route a PLL input from the Primary clock routing, but it incurs more clock input injection delay, which is not natively compensated for using feedback, than a dedicated PLL input. There is one PLLs in each corner of the FPGA on bigger densities. Each of the PLL on the top left and right corners have two pairs dedicated PLL inputs that user can choose from. Both of these inputs can route to the PLL in the top side corner. The PLLs on the bottom corners each have one pair of dedicated PLL input pin.

### PLL Input Clock Mux (PLLREFCS)

Each ECP5 PLL contains an input mux to dynamically switch between two input reference clocks. The output of the PLLREFCS is routed directly into the PLL. There is one PLLREFCS component in each top left and right corner of the FPGA. In order to enhance the clock muxing capability of the ECP5 device, the dedicated clock inputs for the PLLs in each corner are routed to the PLLREFCS component in a corner along with the other potential PLL sources such as edge clocks and primary clocks. This structure is seen in Figure 11-25.

**Figure 11-25. PLL Dedicated Inputs to the PLLREFCS Component for Top Left and Right PLL**



This adds a lot of flexibility for designs that need to switch between two external clocks.

### Standby Mode

The ECP5 device PLL contains a Standby Mode that allows the PLL to be placed into a standby state to save power when not needed in the design. Standby mode is very similar to holding the PLL in reset since the VCO will be turned off and will need to regain lock when exiting standby. In both cases, reset and standby mode, the PLL will retain its programming.

Users MUST hold the PLL in standby for a minimum of 1 ms in order to be sure the PLL analog circuits are fully reset and to have a stable analog startup.

## PLL Inputs and Outputs

### CLKI Input

The CLKI signal is the reference clock for the PLL. It must conform to the specifications in the data sheet in order for the PLL to operate correctly. The CLKI signal can come from a dedicated PLL input pin or from internal routing. The dedicated dual-purpose I/O pin provides a low skew input path and is the recommended source for the PLL. The reference clock can be divided by the input (M) divider to create one input to the phase detector of the PLL.

### CLKFB Input

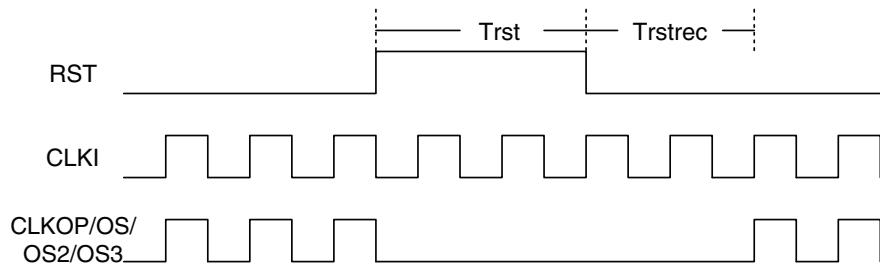
The CLKFB signal is the feedback signal to the PLL. The feedback signal is used by the Phase Frequency Detector inside the PLL to determine if the output clock needs adjustment to maintain the correct frequency and phase. The CLKFB signal can come from a primary clock net (feedback mode = CLKO[P/S/S2/S3]) to remove the primary clock routing injection delay, from a dedicated external dual-purpose I/O pin (feedback mode = UserClock) to account for board level clock alignment, or an internal PLL connection (feedback mode = INT\_O[P/S/S2/S3]) for simple feedback. The feedback clock signal will be divided by the feedback (N) divider to create an input to the VCO of the PLL. A bypassed PLL output cannot be used as the feedback signal.

### RST Input

At power-up an internal power-up reset signal from the configuration block resets the PLL. At runtime an active high, asynchronous, user-controlled PLL reset signal can be provided as a part of the PLL module. The RST signal can be driven by an internally generated reset function or by an I/O pin. This RST signal resets the PLL core (VCO, phase detector, and charge pump) and the output dividers which will cause the outputs to be logic '0'. In bypass mode the output will not be reset.

After the RST signal is de-asserted the PLL will start the lock-in process and will take tLOCK time, about 16 ms, to complete PLL lock. Figure below shows the timing diagram of the RST input. The RST signal is active high. The RST signal is optional. Trst = 1 ms reset pulse width, Trstrec = 1 ns time after a reset before the divider output starts counting again.

**Figure 11-26. RST Input Timing Diagram**



### Dynamic Clock Enables

Each PLL output has a user input signal to dynamically enable / disable its output clock glitchlessly. When the clock enable signal is set to logic ‘0’, the corresponding output clock is held to logic ‘0’.

**Table 11-14. PLL Clock Output Enable Signal List**

Clock Enable Signal Name	Corresponding PLL Output	Clarity Designer Option Name
ENCLKOP	CLKOP	“Clock Enable OP”
ENCLKOS	CLKOS	“Clock Enable OS”
ENCLKOS2	CLKOS2	“Clock Enable OS2”
ENCLKOS3	CLKOS3	“Clock Enable OS3”

This allows the user to save power by stopping the corresponding output clock when not in use. The clock enable signals are optional and will only be available if the user has selected the corresponding option in Clarity Designer. If a clock enable signal is not requested, its corresponding output will be active at all times when the PLL is instantiated unless the PLL is placed into standby mode. The user cannot access a clock enable signal in Clarity Designer when using it for external feedback in order to avoid shutting off the feedback clock input.

### STDBY Input

The STDBY signal is used to put the PLL into a low power standby mode when it is not required. The STDBY signal is optional and will only be available if the user has selected the Standby port option in Clarity Designer. The STDBY signal is active high. When asserted the PLL outputs are pulled to “0” and the PLL will be reset. Users need to stay in the STDBY mode for at least 1 ms to make sure the PLL analog circuits are fully reset and to have a stable analog startup.

### Dynamic Phase Shift Inputs

The ECP5 PLL has five ports to allow for dynamic phase adjustment from FPGA logic. The Dynamic Phase Adjustment section will elaborate on how the user should drive these ports.

### PHASESEL Input

The PHASESEL[1:0] inputs are used to specify which PLL output port will be affected by the dynamic phase adjustment ports. The settings available are shown in the Dynamic Phase Adjustment section. The PHASESEL signal must be stable for 5 ns before the PHASESTEP or PHASELOADREG signals are pulsed. The PHASESEL signal is optional and will be available if the user has selected the “Dynamic Phase Ports” option in Clarity Designer.

**Table 11-15. PHASESEL signal settings definition**

PHASESEL[1:0]	PLL Output Shifted
00	CLKOS
01	CLKOS2
10	CLKOS3
11	CLKOP

### **PHASEDIR Input**

The PHASEDIR input is used to specify which direction the dynamic phase shift will occur, advanced (leading) or delayed (lagging). When PHASEDIR = 0 then the phase shift will be delayed. When PHASEDIR = 1 then the phase shift will be advanced. The PHASEDIR signal must be stable for 5 ns before the PHASESTEP or PHASELOADREG signals are pulsed. The PHASEDIR signal is optional and will be available if the user has selected the Dynamic Phase ports option in Clarity Designer.

**Table 11-16. PHASEDIR signal settings definition**

PHASEDIR	Direction
0	Delayed (lagging)
1	Advanced (leading)

### **PHASESTEP Input**

The PHASESTEP signal is used to initiate a VCO dynamic phase shift for the clock output port and in the direction specified by the PHASESEL and PHASEDIR inputs. This phase adjustment is done by changing the phase of the VCO in 45° increments. The VCO phase changes on the negative edge of the PHASESTEP input after four VCO cycles. This is an active low signal and the minimum pulse width (both high and low) of PHASESTEP pulse is four cycles of VCO running period. The PHASESTEP signal is optional and will be available if the user has selected the Dynamic Phase ports option in Clarity Designer. The PHASESEL and PHASEDIR are required to have a setup time of 5 ns prior to PHASESTEP falling edge.

### **PHASELOADREG Input**

The PHASELOADREG signal is used to initiate a post-divider dynamic phase shift, relative to the unshifted output, for the clock output port and in the direction specified by the PHASESEL and PHASEDIR inputs. A phase shift is started on the falling edge of the PHASELOADREG signal and there is a minimum pulse width of 10 ns from assertion to deassertion. The PHASESEL and PHASEDIR are required to have a setup time of 5 ns prior to PHASELOADREG falling edge. The PHASELOADREG signal is optional and will be available if the user has selected the Dynamic Phase ports option in Clarity Designer.

### **PLL Clock Outputs**

The PLL has four outputs, listed in Table 11-17. All four outputs can be routed to the Primary clock routing of the FPGA. All four outputs can be phase shifted statically or dynamically if external feedback on the clock is not used. They can also statically or dynamically adjust their output duty cycle. The outputs can come from their output divider or the reference clock input (PLL bypass). In bypass mode the output divider can be bypassed or used to divide the reference clock.

**Table 11-17. PLL Clock Outputs and ECLK Connectivity**

Clock Output Name	Edge Clock Connectivity	Selectable Output
CLKOP	Right, Left ECLKs	Always Enabled
CLKOS	Right, Left ECLKs	Selectable via Clarity Designer
CLKOS2	No ECLK Connection	Selectable via Clarity Designer
CLKOS3	No ECLK Connection	Selectable via Clarity Designer

## **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 16 ms. Once lock is achieved, the PLL LOCK signal will be asserted. The LOCK signal can be set in Clarity Designer in either the default “unsticky” frequency lock mode by checking the “Provide PLL Lock Signal” or sticky lock mode by selecting “PLL Lock is Sticky”. In sticky lock mode, once the LOCK signal is asserted (logic ‘1’) it will stay asserted until a PLL reset is asserted. In the default lock mode of “unsticky” frequency lock, if during operation the input clock or feedback signals to the PLL become invalid the PLL will lose lock and the LOCK output will de-assert (logic ‘0’). It is recommended to assert PLL RST to re-synchronize the PLL to the reference clock when the PLL loses lock. The LOCK signal is available to the FPGA routing to implement the generation of the RST signal if requested by the designer. The LOCK signal is optional and will be available if the user has selected the Provide PLL Lock signal option in Clarity Designer.

## **Dynamic Phase Adjustment**

Dynamic phase adjustment of the PLL output clocks can be affected without reconfiguring the FPGA by using the dedicated dynamic phase-shift ports of the PLL.

All four output clocks, CLKOP, CLKOS, CLKOS2 & CLKOS3 have the dynamic phase adjustment feature but only one output clock can be adjusted at a time. Table above shows the output clock selection settings available for the PHASESEL[1:0] signal. The PHASESEL signal must be stable for 5 ns before the PHASESTEP or PHASELOADREG signals are pulsed.

The selected output clock phase will either be advanced or delayed depending upon the value of the PHASEDIR port or signal. Table 11-16 shows the PHASEDIR settings available. The PHASEDIR signal must be stable for 5 ns before the PHASESTEP or PHASELOADREG signals are pulsed.

### **VCO Phase Shift**

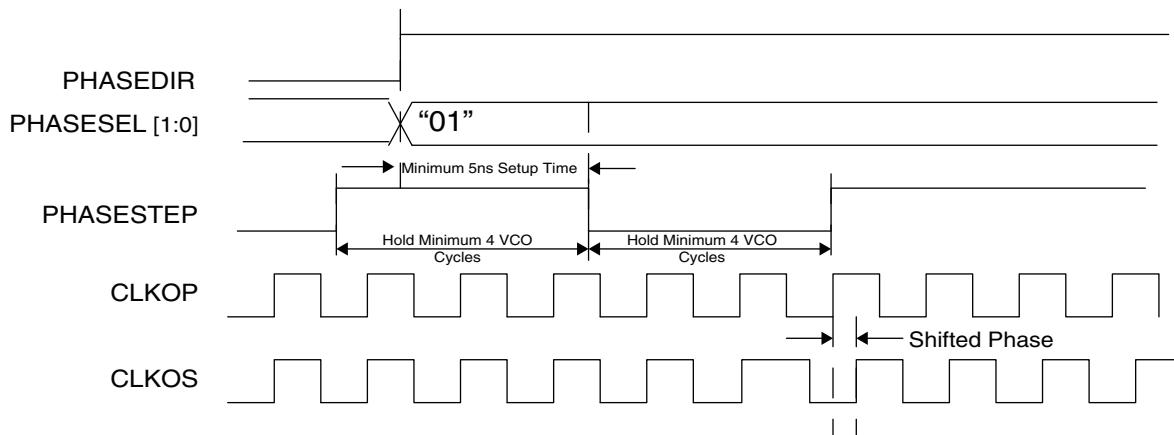
Once the PHASESEL and PHASEDIR have been set, a VCO phase adjustment is made by toggling the PHASESTEP signal from the current setting. Each pulse of the PHASESTEP signal will generate a phase step based on this equation:

$$(CLKO<n>_FPHASE/(8*CLKO<n>_DIV)]*360$$

Where  $<n>$  is the clock output specified by PHASESEL (CLKOP/OS/OS2/OS3). Values for CLKO $<n>$ \_FPHASE and CLKO $<n>$ \_DIV are located in the HDL source file.

The PHASESTEP signal is latched in on the falling edge and is subject to a minimum wait of four VCO cycles prior to pulsing the signal again. One step size is the smallest phase shift that can be generated by the PLL in one pulse. The dynamic phase adjustment results in a glitch free adjustment when delaying the output clock, but glitches may result when advancing the output clock.

**Figure 11-27. PLL Phase Shifting Using the PHASESTEP Signal**



For Example:

PHASESEL[1:0]=2'b00 to select CLKOS for phase shift

PHASEDIR =1'b0 for selecting delayed (lagging) phase

Assume the output is divided by 2, CLKOS\_DIV = 2

The CLKOS\_FPHASE is set to 1

The above signals need to be stable for 5 ns before the falling edge of PHASESTEP and the minimum pulse width of PHASESTEP should be four VCO clock cycles. It should also stay low for four VCO Clock Cycles.

For each toggling of PHASESTEP, you will get  $[1/(8*2)]*360 = 22.5$  degree phase shift (delayed).

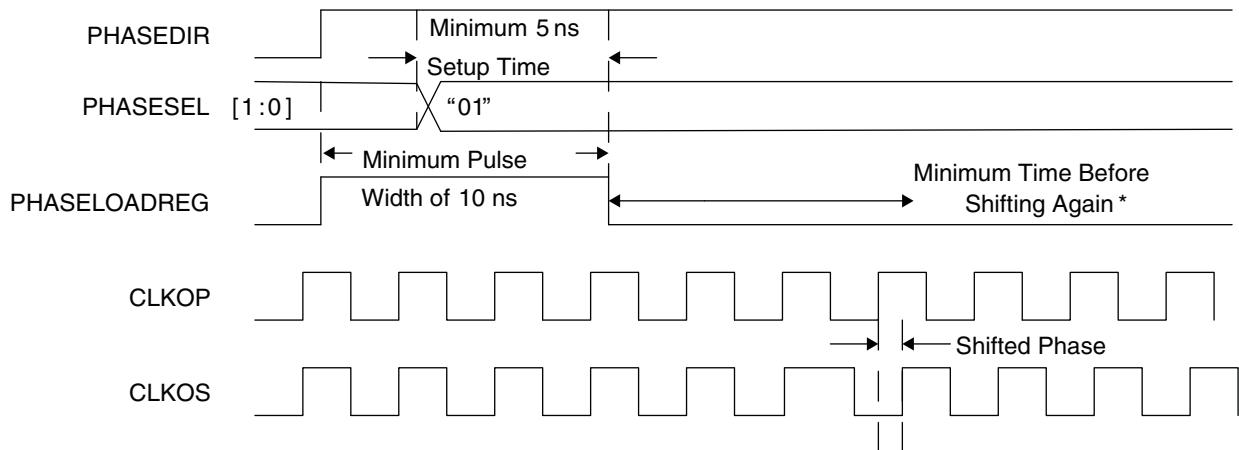
### Divider Phase Shift

Once the PHASESEL and PHASEDIR have been set a post-divider phase adjustment is made by toggling the PHASELOADREG signal. Each pulse of the PHASELOADREG signal will generate a phase shift. The step size relative to the unshifted output is specified by this equation:

$$[(\text{CLKO}_{<n>} \text{ CPHASE} - \text{CLKO}_{<n>} \text{ DIV}) / (\text{CLKO}_{<n>} \text{ DIV} + 1)] * 360^\circ$$

Where  $<n>$  is the clock output specified by PHASESEL (CLKOP/OS/OS2/OS3). Values for  $\text{CLKO}_{<n>} \text{ CPHASE}$  and  $\text{CLKO}_{<n>} \text{ DIV}$  are located in the HDL source file. Please note that if these values are both "1", no shift will be made.

**Figure 11-28. Divider Phase Shift Timing Diagram**



\*Note – Minimum Time Before Shifting Again Equation =  
 $2.5 * (\text{CLKO}_{<n>} \text{ DIV} + 1) + (\text{CLKO}_{<n>} \text{ CPHASE} + 1) * (\text{Period of Divider Clock})$ .

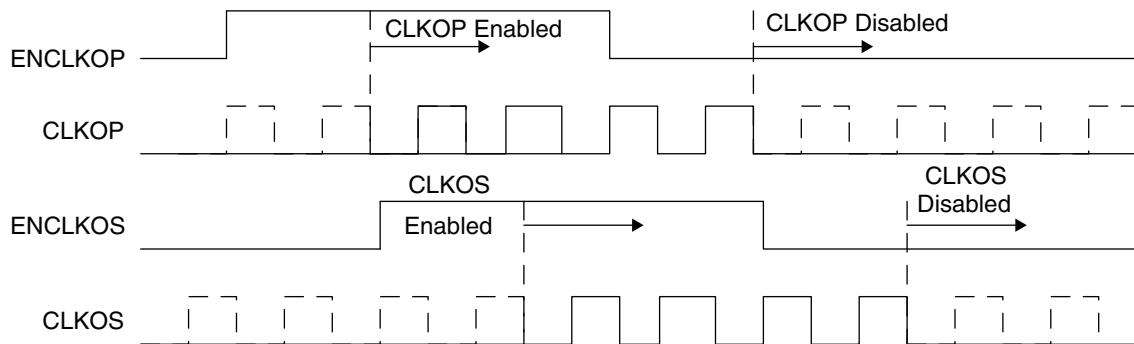
## Low Power Features

The ECP5 PLL contains several features that allow the user to reduce the power usage of a design including Standby mode support and Dynamic clock enable.

### Dynamic Clock Enable

The Dynamic Clock Enable feature allows the user to glitchlessly enable and disable selected output clocks during periods when not used in the design. A disabled output clock will be logic '0'. Re-enabled clocks start on the falling edge of CLKOP. To support this feature each output clock has an independent Output Enable signal that can be selected. The Output Enable signals are ENCLKOP, ENCLKOS, ENCLKOS2, and ENCLKOS3. Each clock enable port has an option in the Clarity Designer GUI to bring the signal to the top level ports of the PLL. If external feedback is used on a port or if the clock's output is not enabled its dynamic clock enable port is unavailable.

**Figure 11-29. Dynamic Clock Enable for PLL Outputs**



### Standby Mode

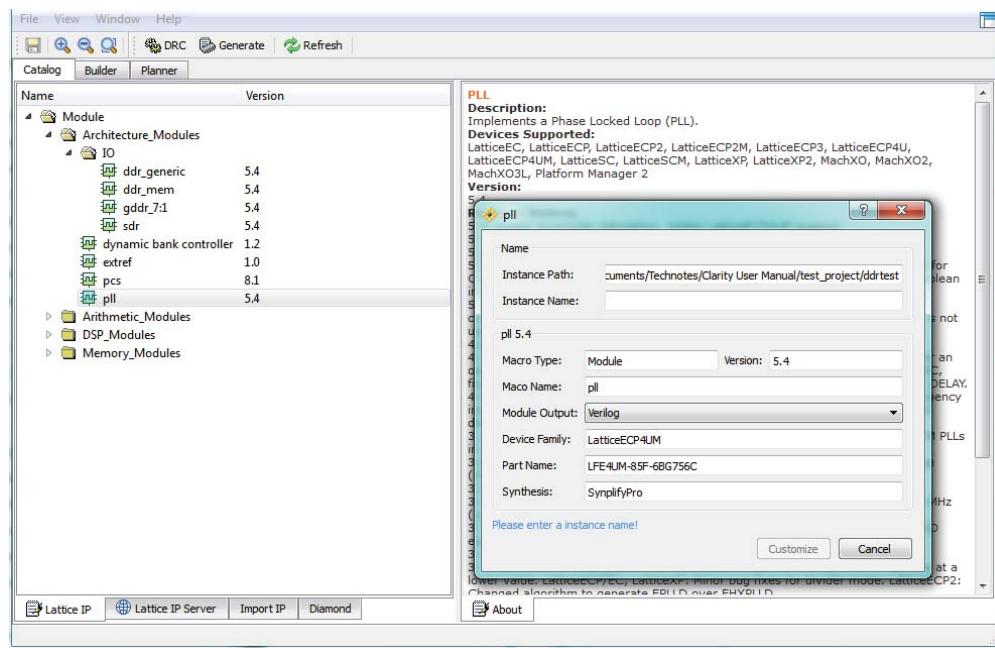
The PLL can also be put into standby mode. This is similar to reset in that the PLL is still powered, however the VCO is not running and the clock outputs driven low. The PLL will enter Standby mode when the STDBY signal is driven high and the outputs will be driven low. Users need to stay in the STDBY mode for at least 1 ms to make sure the PLL analog circuits are fully reset and to have a stable analog startup. The PLL can be restarted when it is needed again and the output clocks will be reactivated. It will take  $T_{lock\_time} = 16$  us to achieve PLL lock again. To support this mode the "Standby Port" option is in the Clarity Designer GUI and will cause the STDBY port to be brought out to the top level of the PLL module.

### PLL Usage in Clarity Designer

It is expected that Clarity Designer will be used to create and configure a PLL. PLL can be found in the **Catalog** tab of Clarity Designer under Module - Architecture Modules. The graphical user interface is used to select parameters for the PLL. The result is an HDL block to be used in the simulation and synthesis flow.

The main window when the PLL is selected is shown in Figure 11-30. When opening Clarity Designer inside a Diamond project, the only entry required is the file name as the other entries are set to the project settings. If Clarity Designer is opened as a stand-alone tool then it is necessary to supply the additional parameters shown on this screen. After entering the module name of choice, clicking on Customize will open the PLL configuration window as shown in Figure 11-30.

**Figure 11-30. Clarity Designer Main Window for PLL module**



### Configuration Tab

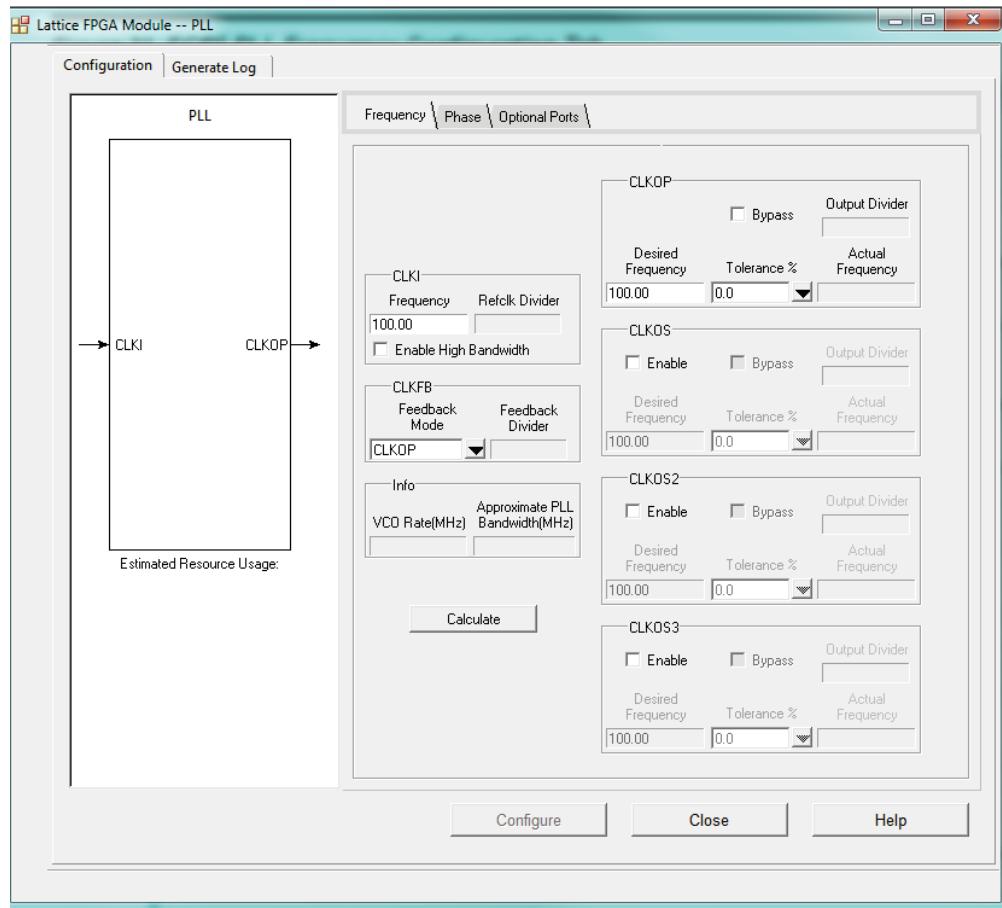
The configuration window lists all user accessible attributes with default values set. Upon completion, clicking Generate generates the source.

### PLL Frequency and Phase Configuration

Enter the input and output clock frequencies and the software will calculate the divider settings. After the input and output frequencies are entered clicking the Calculate button will display the divider values and the closest achievable frequency will be displayed in the “Actual Frequency” text box. If an entered value is out of range it will be displayed in red and an error message will be displayed. The user can also select a tolerance value from the “Tolerance %” drop-down box. When the Calculate button is pressed the calculation will be considered accurate if the result is within the entered tolerance percentage range.

New to the ECP5 PLL GUI, the user enters the desired phase shift and the software will calculate the closest achievable shift. After the desired phase is entered, clicking the Calculate button will display the closest achievable phase shift in the “Actual Phase” text box. If an entered value is out of range it will be displayed in red and an error message will be displayed.

**Figure 11-31. ECP5 PLL Frequency Configuration Tab**



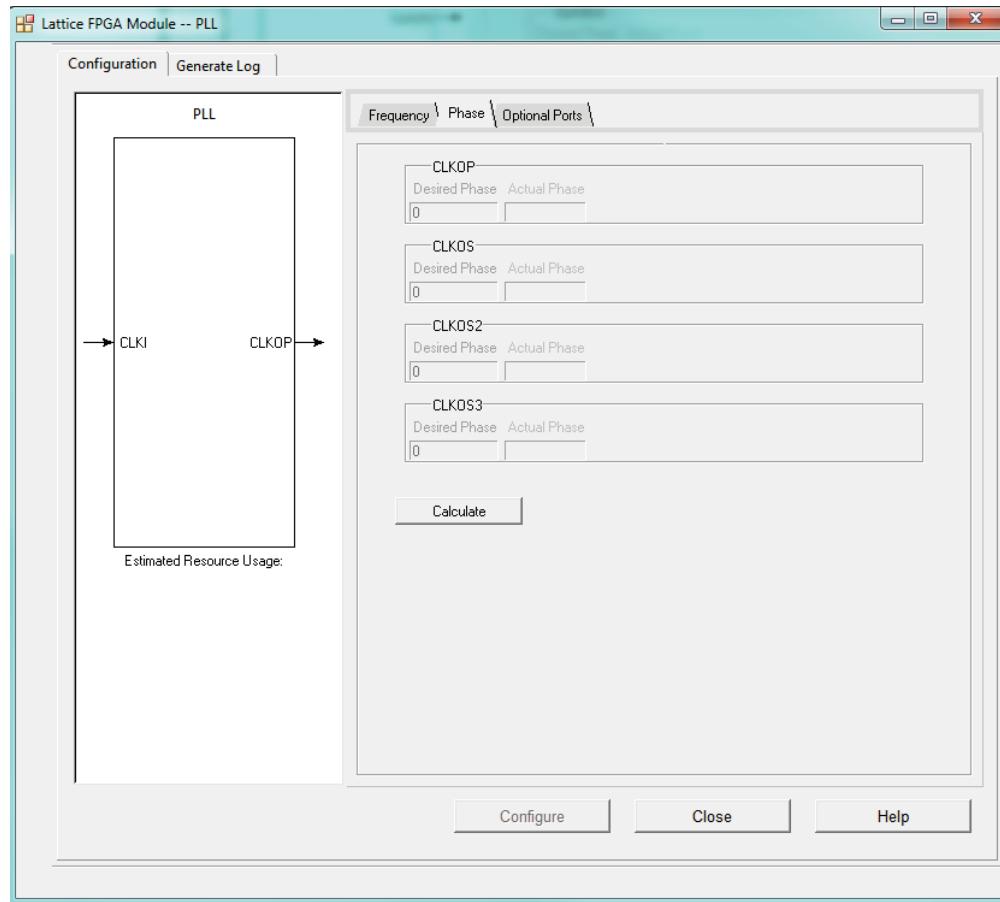
**Table 11-18. Page 1, PLL Frequency Settings, Clarity Designer GUI**

User Parameters	Description	Range	Default	Corresponding HDL Attribute
CLKI	Frequency Input	10 – 400 MHz	100 MHz	FREQUENCY_PIN_CLKI
Refclk Divider – Read Only	Shows the reference clock divider value	–	–	CLKI_DIV
Enable High Bandwidth	Sets the PLL to high bandwidth mode	ON / OFF	OFF	
CLKFB	Feedback mode	CLKOP, CLKOS, CLKOS2, CLKOS3, INT_OP, INT_OS, INT_OS2, INT_OS3, User-Clock	CLKOP	FEEDBK_PATH
	Feedback Divider (read only)	1 – 128	1	CLKFB_DIV

User Parameters	Description	Range	Default	Corresponding HDL Attribute
CLKOP	Enable	ON / OFF	OFF	CLKOP_ENABLE
	Bypass	ON / OFF	OFF	OUTDIVIDER_MU_XA
	Output Divider (read only)	-	-	CLKOP_DIV
	Desired Frequency *1	3.125 – 400 MHz	100 MHz	FREQUENCY_PIN_CLKOP
	Tolerance (%)	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	
	Actual Frequency (read only)	-	-	
CLKOS	Enable	ON / OFF	OFF	CLKOS_Enable
	Bypass	ON / OFF	OFF	OUTDIVIDER_MU_XB
	Clock Divider (read only)	-	-	CLKOS_DIV
	Desired Frequency *1	3.125 – 400 MHz	100 MHz	FREQUENCY_PIN_CLKOS
	Tolerance (%)	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	
	Actual Frequency (read only)	-	-	
CLKOS2	Enable	ON / OFF	OFF	CLKOS2_Enable
	Bypass	ON / OFF	OFF	OUTDIVIDER_MU_XC
	Clock Divider (read only)	-	-	CLKOS2_DIV
	Desired Frequency *1	3.125 – 400 MHz	100 MHz	FREQUENCY_PIN_CLKOS2
	Tolerance (%)	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	
	Actual Frequency (read only)	-	-	
CLKOS3	Enable	ON / OFF	OFF	CLKOS3_Enable
	Bypass	ON / OFF	OFF	OUTDIVIDER_MU_XD
	Clock Divider (read only)	-	-	CLKOS3_DIV
	Desired Frequency *1	3.125 – 400 MHz	100 MHz	FREQUENCY_PIN_CLKOS3
	Tolerance (%)	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	
	Actual Frequency (read only)	-	-	

1. If this clock is selected as feedback, the minimum output frequency that is achievable is 10 MHz.

**Figure 11-32. ECP5 PLL Phase Configuration Tab**

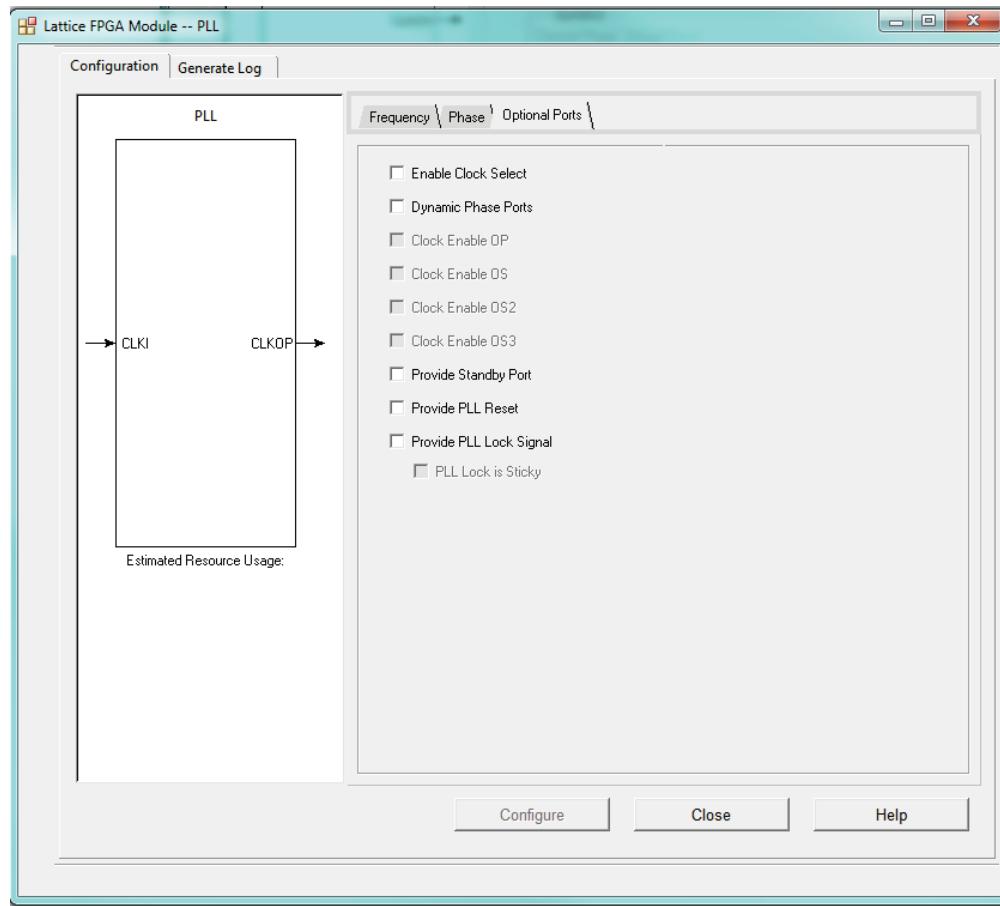


**Table 11-19. Tab 2, PLL Phase Settings, Clarity Designer GUI**

User Parameters	Description	Range	Default	Corresponding HDL Attribute
CLKOP	Desired Phase*1	(Based on Frequency)	100 MHz	CLKOP_CPHASE, CLKOP_FPHASE
	Actual Phase (read only)	—	—	
CLKOS	Desired Phase*1	(Based on Frequency)	100 MHz	CLKOS_CPHASE, CLKOS_FPHASE
	Actual Phase (read only)	—	—	
CLKOS2	Desired Phase*1	(Based on Frequency)	100 MHz	CLKOS2_CPHASE, CLKOS2_FPHASE
	Actual Phase (read only)	—	—	
CLKOS3	Desired Phase*1	(Based on Frequency)	100 MHz	CLKOS3_CPHASE, CLKOS3_FPHASE
	Actual Phase (read only)	—	—	

1. Phase is now a calculated value based on frequency parameters, which gives finer phase resolution.

**Figure 11-33. ECP5 PLL Optional Ports Configuration Tab**



**Table 11-20. Tab 3, PLL Optional Ports, Clarity Designer GUI**

User Parameters	Description	Range	Default	Corresponding HDL Attribute
Enable Clock Select	Enables the input clock mux (PLLREFCS component).	ON / OFF	OFF	
Dynamic Phase ports	Provides Dynamic Phase Shift ports.	ON / OFF	OFF	DPHASE_SOURCE
Clock Enable OP	Provides ENCLKOP; clock enable port for dynamic clock output shutoff.	ON / OFF	OFF	
Clock Enable OS	Provides ENCLKOS; clock enable port for dynamic clock output shutoff.	ON / OFF	OFF	
Clock Enable OS2	Provides ENCLKOS2; clock enable port for dynamic clock output shutoff.	ON / OFF	OFF	
Clock Enable OS3	Provides ENCLKOS3; clock enable port for dynamic clock output shutoff.	ON / OFF	OFF	
Provide Standby Port	Provides STDBY port to put the PLL into standby mode.	ON / OFF	OFF	STDBY_ENABLE
Provide PLL Reset	Provides RST signal.	ON / OFF	OFF	PLLRST_ENA
Provide PLL Lock Signal	Provides the LOCK signal.	ON / OFF	OFF	
PLL Lock is Sticky	Once LOCK goes high it won't de-assert unless the PLL is reset.	ON / OFF	OFF	PLL_LOCK_MODE

For the PLL, Clarity Designer sets attributes in the HDL module that are specific to the data rate selected. Although these attributes can be easily changed, they should only be modified by re-running the GUI so that the performance of the PLL is maintained. After the MAP stage in the design flow, FREQUENCY preferences will be included in the preference file to automatically constrain the clocks produced by the PLL. For a step by step guide to using Clarity Designer, refer to the Clarity Designer User Manual.

## **PLL Reference Clock Switch Primitive (PLLREFCS)**

The ECP5 PLL contains an input mux to dynamically switch between two input reference clocks. This mux is modeled by the PLLREFCS component. This mux may allow glitches and runt pulses through depending on when the clock is switched. It is expected that the input clocks have the same frequency. Table 11-22 defines the I/O ports of the PLLREFCS block.

This component is instantiated in the PLL wrapper when the “Enable Clock Select” option is checked in the Clarity Designer GUI. It can also be directly instantiated and software will automatically assign it to an unused PLL in bypass mode and route the output to the CLKOP port.

**Figure 11-34. PLLREFCS Component Symbol**



**Table 11-21. PLLREFCS Component Port Definition**

Port Name	Description
CLK0	CLK0
CLK1	CLK1
SEL	SEL = '0', CLK0 is selected
SEL = '1', CLK1 is selected	
PLLCSOUT	PLLCSOUT

## **PLLREFCS Usage in VHDL**

### **Component Declaration**

```

COMPONENT PLLREFCS
PORT (
    CLK0      : IN STD_LOGIC;
    CLK1      : IN STD_LOGIC;
    SEL       : IN STD_LOGIC;
    PLLCSOUT : OUT STD_LOGIC);
END COMPONENT;

```

### **PLLREFCS Instantiation**

```

PLLREFCSInst0 : PLLREFCS
PORT MAP (
    CLK0      => CLK_0
    ,CLK1      => CLK_1
    ,SEL       => SELECT
    ,PLLCSOUT => CLK_OUT);

```

### **PLLREFCS Usage in Verilog**

### **Component and Attribute Declaration**

```
module PLLREFCS(CLK0, CLK1, SEL, PLLCSOUT);
input CLK0, CLK1, SEL;
output PLLCSOUT;
endmodule;
```

**PLLREFCS Instantiation**

```
PLLREFCS PLLREFCSInst0 (
    .CLK0      (CLK_0)
    ,.CLK1      (CLK_1)
    ,.SEL       (SELECT)
    ,.PLLCSOUT (CLK_OUT) );
```

**Technical Support Assistance**

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

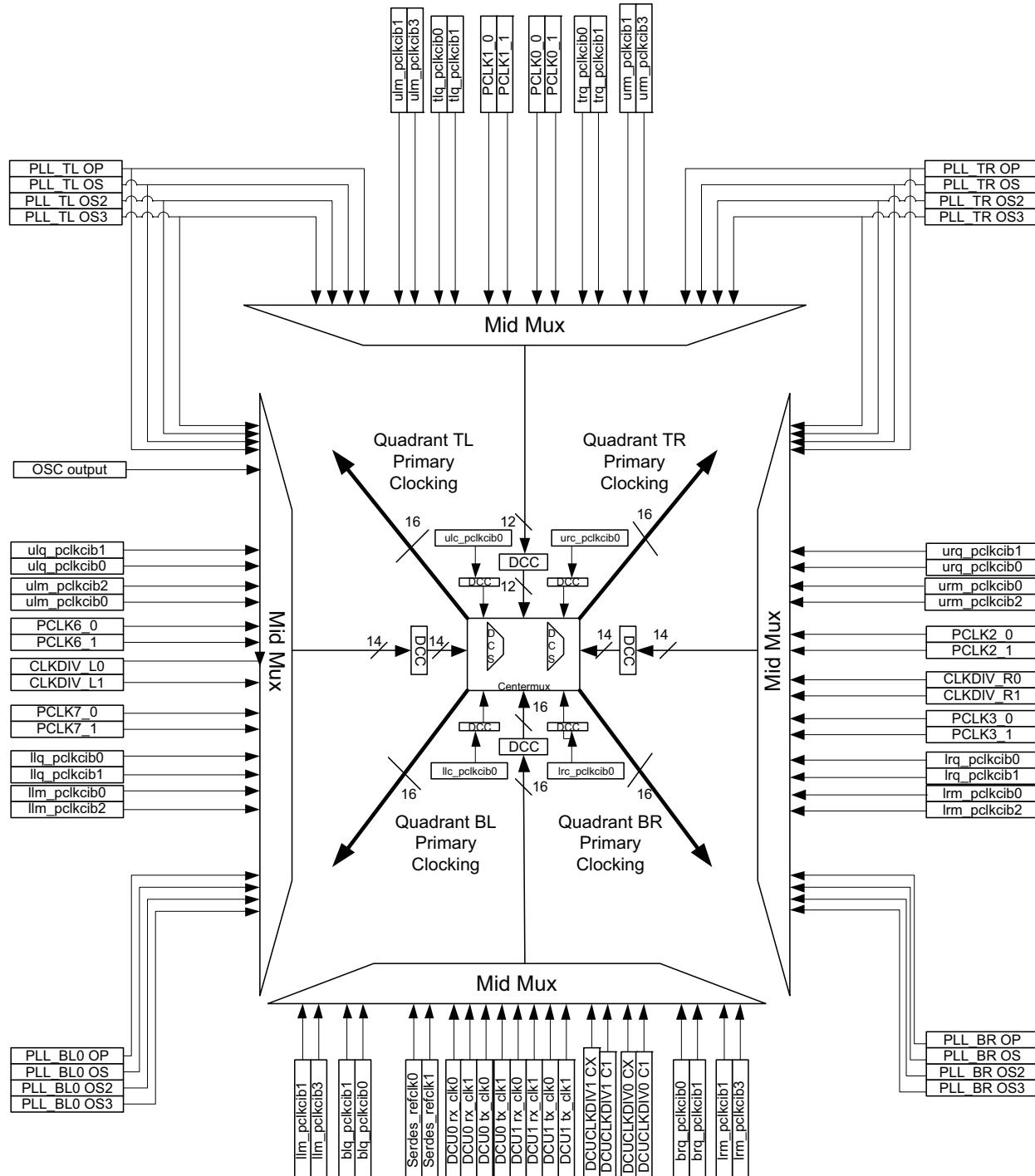
**Revision History**

Date	Version	Change Summary
March 2014	01.0	Initial Release

## Appendix A. Primary Clock Sources and Distribution

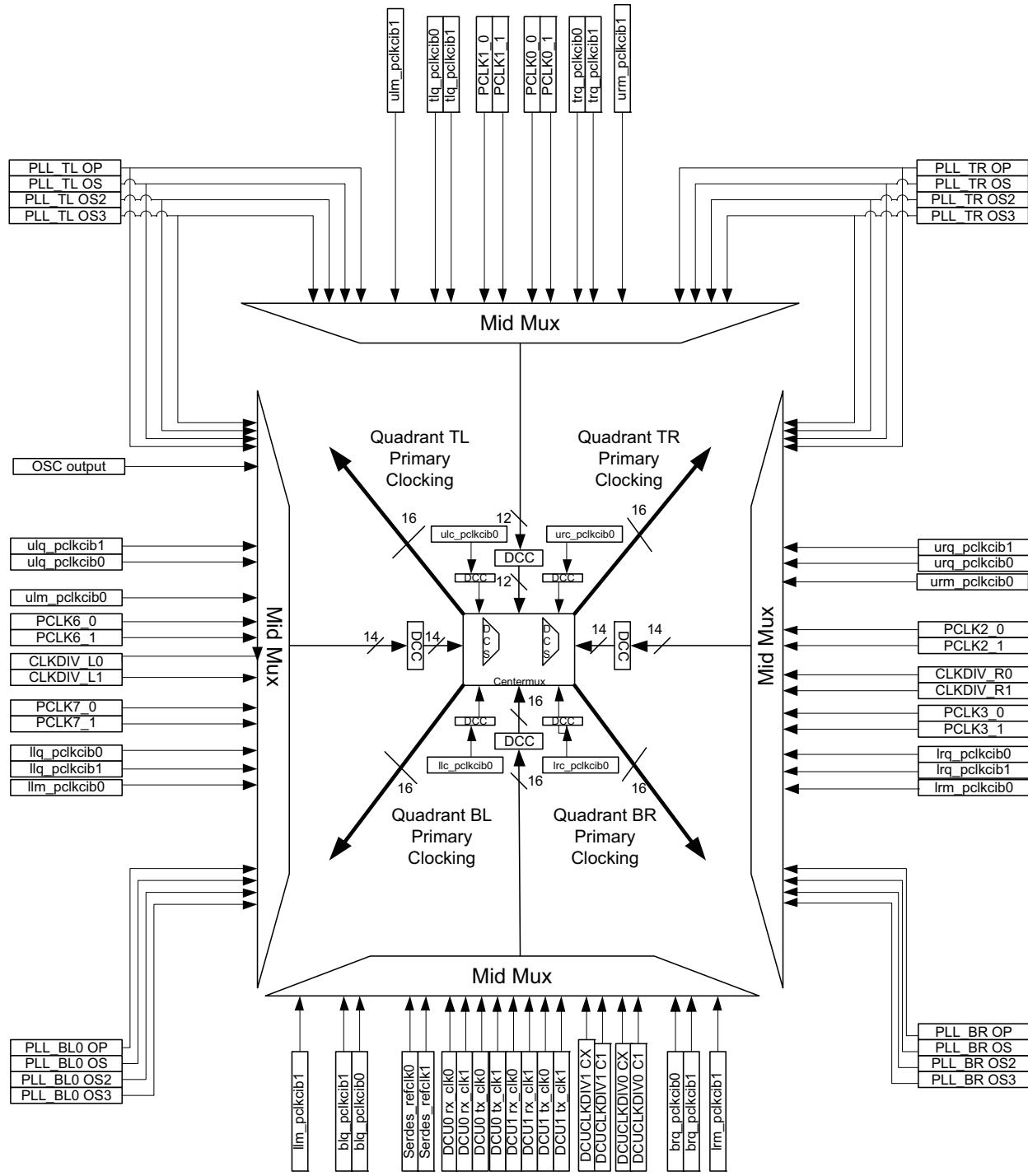
The following figures show the inputs into the Primary Clock Network through the mid-mux into the centermux for each device. There are DCC components at the input of the centermux to allow the user to stop the clock in order to save power. All mid-mux inputs with the nomenclature “<quadrant>\_pclkcb<#>” are fabric entry points to the PCLK network. All mid-mux inputs with the nomenclature “<location>\_DQS\_CLK” are DQS clocks from the I/O.

**Figure 11-35. ECP5 Primary Clock Sources and Distribution, LFE5-85 Devices**

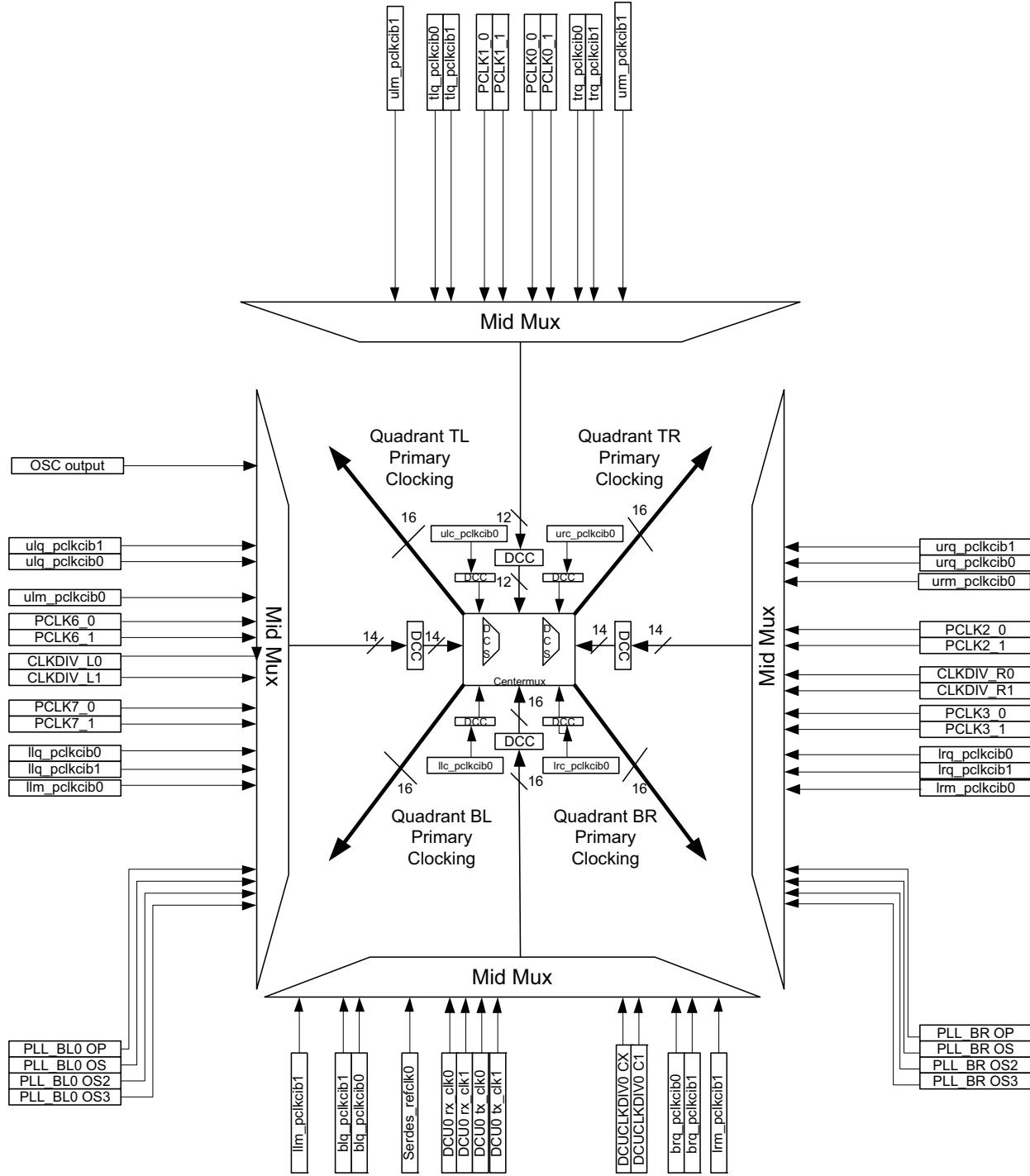


Note: LFE4U devices do not have the DCU & Serdes clock inputs

**Figure 11-36. ECP5 Primary Clock Sources and Distribution, LFE5-45 device**



**Figure 11-37. ECP5 Primary Clock Sources and Distribution, LFE5-25 Device**



Note: LFE4U devices do not have the DCU & Serdes clock inputs

## Appendix B. Pinout Rules for Clocking in ECP5 Devices

In the ECP5 device, as with all other architectures, there are general rules and guidelines for board designers which are required to be followed. These rules will give the best possible timing and allow for a successful design.

In the .csv file where pins are listed, under the “Dual Function” section, you will see the PCLK and PLL input pins listed as below:

Primary Clock Input Pin – PCLKT<Bank>\_<0/1>

Dedicated PLL Input Pin – <LOC>\_GPLL0T\_IN

**Table 11-22. Clock Input Selection Table**

Clock Input	Pin to Use	Clock Routing Resource
Clock Input to Logic Directly	PCLK Input Pin	Uses Primary Clock Routing for the Clock.
Clock Input to PLL Only	PLL Input Pin	Uses a Dedicated PLL Input. No Primary Clock Routing is Used.
Clock Input to Logic and PLL	PCLK Input Pin	Uses Primary Clock Routing for the Clock.
Clock input to more than 2 PLLs	PCLK Input Pin	Uses Primary Clock Routing for the Clock.

## Introduction

This technical note discusses memory usage for the ECP5™ family of FPGA devices. It is intended to be used by design engineers as a guide to integrating the EBR- and PFU-based memories for this device family in Lattice Diamond®.

The architecture of these devices provides many resources for memory-intensive applications. The sysMEM™ Embedded Block RAM (EBR) complements its distributed PFU-based memory. Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM and ROM memories can be constructed using the EBR. The LUTs and PFUs can implement Distributed Single-Port RAM, Dual-Port RAM and ROM. Look-up Tables (LUTs) within PFUs can implement Distributed Single-Port RAM, Dual-Port RAM and ROM.

The capabilities of the EBR Block RAM and PFU RAM are referred in this document. Designers can utilize the memory primitives in three separate ways:

- **Via Clarity Designer** – The Clarity Designer GUI allows users to specify the memory type and size required. The Clarity Designer takes this specification and constructs a netlist to implement the desired memory by using one or more of the memory primitives.
- **Via PMI (Parameterizable Module Inferencing)** – PMI allows experienced users to skip the graphical interface and utilize the Configurable memory primitives on-the-fly from the Lattice Diamond® project navigator. The parameters and the control signals needed either in Verilog or VHDL can be set. The top-level design will have the parameters defined and signals declared so the interface can automatically generate the black box during synthesis.
- **Via the Instantiation of Memory Primitives** – Memory primitives are called directly by the top-level module and instantiated in the user's design. This is an advanced method and requires a thorough understanding of memory hook-ups and design interfaces.

The remainder of this document discusses these approaches.

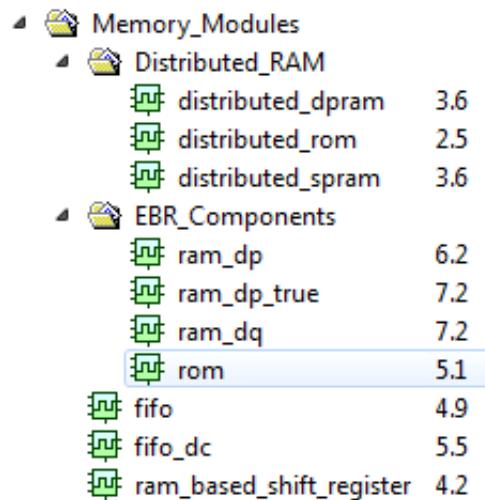
## Memory Generation

Designers can utilize the Clarity Designer to easily specify a variety of memories in their designs. These modules will be constructed using one or more memory primitives along with general purpose routing and LUTs as required. The available modules in the Clarity Designer are:

- Distributed Memory Modules
  - Distributed Dual Port RAM (Distributed\_DPRAM)
  - Distributed ROM (Distributed\_ROM)
  - Distributed Single Port RAM (Distributed\_SPRAM)
- EBR Components (or EBR based Modules)
  - Dual PORT RAM (RAM\_DP\_TRUE)
  - Pseudo Dual Port RAM (RAM\_DP)
  - Single Port RAM (RAM\_DQ)
  - Read Only Memory (ROM)
- First In First Out Memory (FIFO and FIFO\_DC)
- RAM Based Shift Register

Figure 12-1 shows the memory modules under Clarity Designer in Lattice Diamond software.

**Figure 12-1. Memory Modules Available in Clarity Designer**

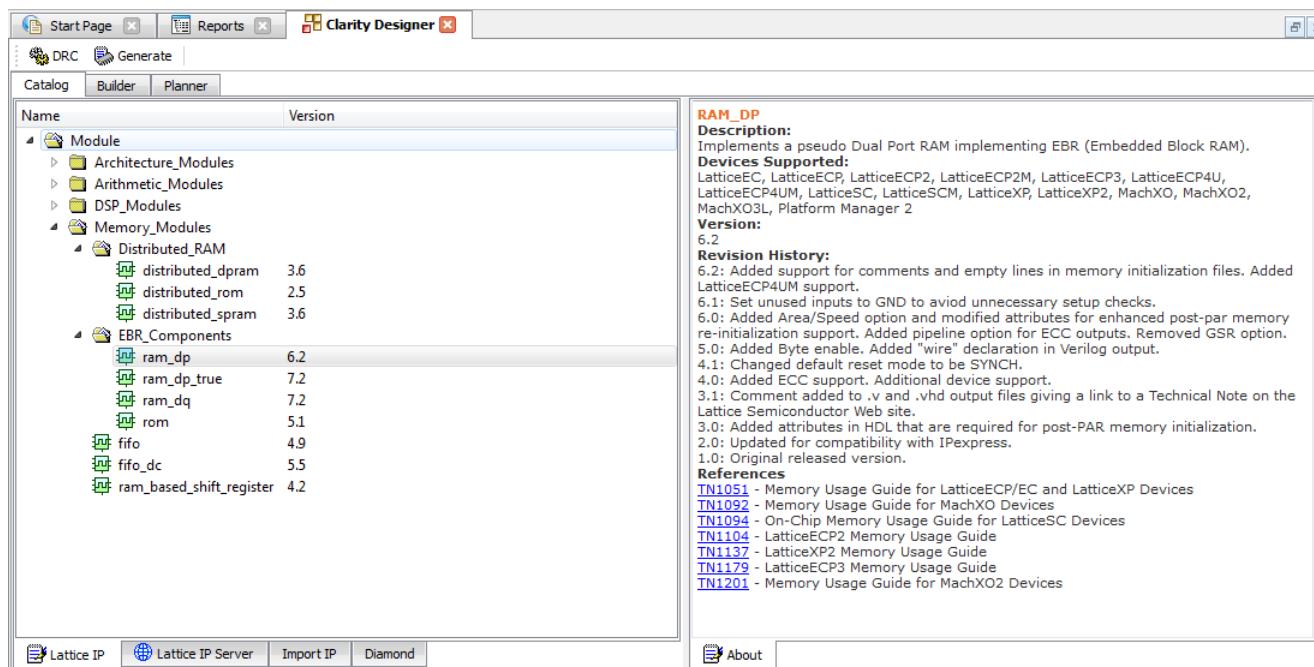


## Clarity Designer Flow

Clarity Designer allows users to generate, create (or open) any of the above modules for ECP5 devices. From the Lattice Diamond software, select **Tools > Clarity Designer**.

Alternatively, you can click on the  button in the toolbar. This opens the Clarity Designer window as shown in Figure 12-2.

**Figure 12-2. Clarity Designer in Lattice Diamond Software**

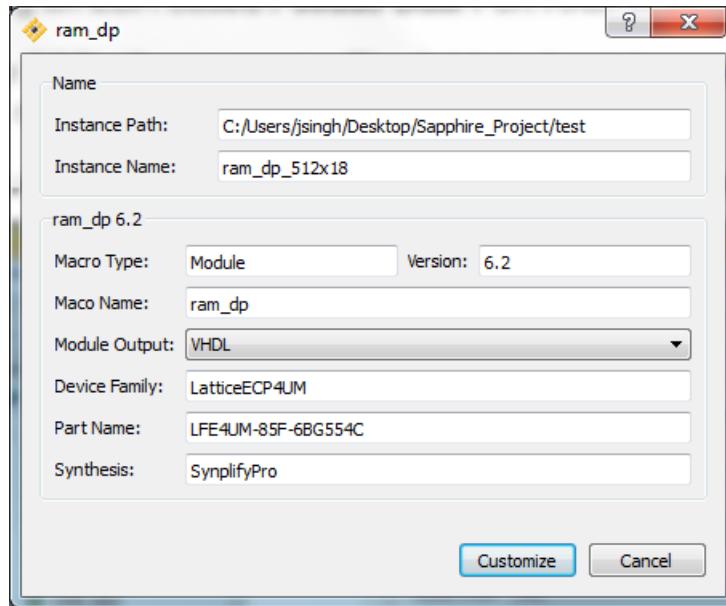


The left section of the Clarity Designer window includes the Module Tree. The Memory Modules are categorized as **Distributed RAM**, **EBR Components** and **FIFOs**. The right section of the window shows the description of the module selected and links to the documentation to find more details about it.

Let us look at an example of generating an EBR based Pseudo Dual Port RAM of size 512 x 18.

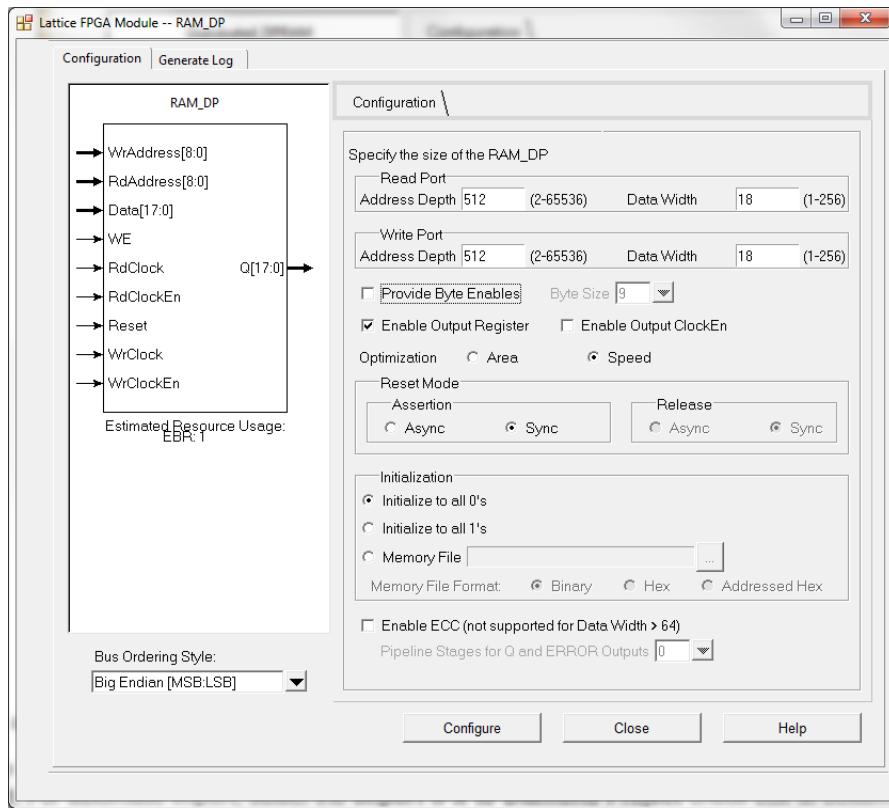
Double-click **ram\_dp** under the **EBR\_ Components**. Fill out the information of the module to generate and click the **Customize** button. This is shown in Figure 12-3

**Figure 12-3. Example: Generating Pseudo Dual Port RAM (RAM\_DP) Using Clarity Designer**



Clicking **Customize** opens another window, as shown in Figure 12-4, where you can customize the Distributed DPRAM.

**Figure 12-4. Example: Generating Pseudo Dual Port RAM (RAM\_DP) Module Customization - General Options**



When all the options of the module being generated are filled in, click **Generate**.

This module, once in the Diamond project, can be instantiated within other modules.

## Utilizing PMI

The parameters and control signals needed can be set in either Verilog or VHDL. The top-level design includes the defined memory parameters and declared signals. The interface can then automatically generate the black box during synthesis and Diamond can generate the netlist on-the-fly. Lattice memories are the same as industry standard memories, so you can get the parameters for each module from any memory-related guide, which is available through the on-line help system.

To do this, the user creates a Verilog or VHDL behavior code for the memory and the synthesis tool automatically identifies it as memory and synthesizes it as a distributed or EBR memory. Memory sizes smaller than 2K bits are automatically mapped to Distributed mode and those larger than 2K bits will be implemented using EBRs. This default option can be over-ridden using the RAM\_STYLE attribute in Synopsys Synplify Pro®.

## Utilizing Direct instantiation of Memory Primitives

Another way to use the memories in the designs is by directly instantiating the memory primitives for the ECP5 devices. When instantiating the primitives, users have to work at the EBR block level. In case there is a need to have a memory that spans multiple modules, users are required to create the cascading memory on their own.

Lattice provides a detailed list of all the primitives in a VHDL/ Verilog file under the cae\_library/synthesis folder in Lattice Diamond software installation folder.

## Memory Features

The RAMs can be generated with Error Correction and Byte Enables that mask selective bits. These features are available in the EBR based RAM modules.

### ECC in Memory Modules

An error-correcting code (ECC) code is a system of adding redundant data, or parity data, to a message, such that it can be recovered by a receiver even when a number of errors were introduced, either during the process of transmission, or on storage.

EBR based Memory modules Clarity Designer allows users to implement ECC. There is a check box to enable ECC in the Configuration tab for the module.

Enable ECC check box allows error correction of single errors and detection of 2-bit errors. This is not supported for data widths higher than 64 bits.

The two bits indicate the error if any, and the following shows you what each of these bits mean:

- Error[1:0] = 00 Indicates there is no error
- Error[0] = 1 Indicates there was a 1-bit error which was fixed
- Error[1] = 1 Indicates there was a 2-bit error which cannot be corrected.

ECC is adding 0~2 cycle of delay to both the data and ERROR outputs base on the Clarity Designer GUI setting. The data and ERROR output are always in sync.

One of the things to note is that the ECC is added in the PFU logic, and hence there logic implemented outside the EBR block and can cause some speed impact. The effective speed at which the memory runs should be determined by running the Trace report.

### Byte Enable

Byte Enable is a feature available in the selected RAM modules where users can mask the bytes written in the RAM. The Byte Enable control can be per 8-bits or 9-bits; the selection can be made in Clarity Designer while generating the module.

Note that the Byte Enable option is available for the memory widths higher than 8-bits (1 byte).

Each bit of the BE signal corresponds to the corresponding 8-bit or 9-bit selection, starting from LSB side. For example, if we add Byte Enable to an 18-bit wide RAM, then Table 12-1 explains how the written data (Data In) is masked for an 8-bit or 9-bit Byte Size.

**Table 12-1. Masked Data In Bits for an 8-Bit or 9-Bit Byte Size**

Byte Enable Bit	Data In Bits that Get Masked (with 8-bit Byte Size)	Data In bits that Get Masked (with 9-Bit Byte Size)
ByteEn(0)	Data(7:0)	Data(8:0)
ByteEn(1)	Data(13:8)	Data(15:9)
ByteEn(2)	Data(19:14)	Data(19:16)

It is to be noted that the ByteEn and ECC are mutually exclusive and they cannot be used together.

## Memory Modules

The following sections discuss the different modules, the size of memory that each EBR block or the Distributive primitive can support and any special options for the module.

When a user specifies the width and depth of the memory in the Clarity Designer, the tool generates the memory by depth cascading and/ or width cascading or EBR blocks or Distributed RAM primitives. Clarity Designer automatically allows users to create memories larger than the width and depth supported for each primitive.

## Memory Cascading

For any memory sizes smaller than that can fit in a single EBR block or the Distributed primitive, the module will utilize the complete block or primitive.

For memory sizes larger than that of a single module, the multiple modules will be cascaded (either in depth or width) to create a larger module.

## Input & Output Register

The architecture of the EBR blocks in ECP5 devices are designed such that the inputs that go into the memory are always registered. This means that the input data and address are always registered at the input of the memory array. The output data of the memory is optionally registered at the output. The user can choose this option by selecting the **Enable Output Register** check box in Clarity Designer while customizing the module.

Control signals like WE and Byte Enable are also registered going in to the EBR block.

## Reset

The EBRs also support the Reset signal. The Reset (or RST) signal only resets input and output registers of the RAM. It does not reset the contents of the memory.

## Timing

In order to correctly write into a memory cell in the EBR block, the correct address has to be registered by the logic.

Hence it is important to note that while running the trace on the EBR blocks, there should be no setup and hold time violations on the address registers (address). Failing to meet these requirements can result in incorrect addressing and hence corruption of memory contents.

During a read cycle, a similar issue can occur that the correct contents are not read if the address is not correctly registered in the memory.

A Post Place and Route timing report in Lattice Diamond design software can be run to verify that no such timing errors occur. Refer to the timing preferences in the Online Help documents.

## Single Port RAM (RAM\_DQ) – EBR Based

ECP5 FPGAs supports all the features of Single Port Memory Module or RAM\_DQ. Clarity Designer allows users to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirement.

Clarity Designer generates the memory module, as shown in Figure 12-5.

**Figure 12-5. Single-Port Memory Module Generated by Clarity Designer**

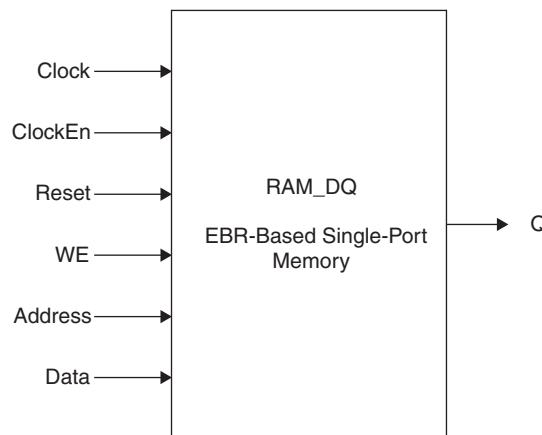
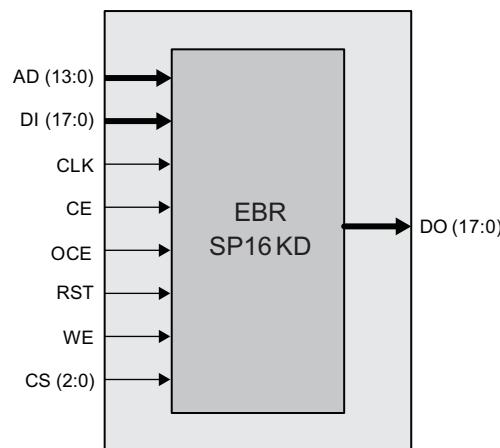


Figure 12-6 provides the primitive that can be instantiated for the Single Port RAM. Primitive name is SP16KD and it can be directly instantiated in the code. Check the details on the port and port names under the primitives available under cae\_library/synthesis folder in Lattice Diamond software installation folder.

It is to be noted that each EBR can accommodate 18K bits of memory; if the memory required is larger than 18K, then cascading can be used, using the CS port (CSA and CSB in this case). See Appendix A. Attribute Definitions CSDECODE section on the cascading of multiple EBR primitives.

**Figure 12-6. Single Port RAM Primitive for ECP5 Devices**



The various ports and their definitions for Single-Port Memory are listed in Table 12-2. The table lists the corresponding ports for the module generated by Clarity Designer and for the EBR RAM\_DQ primitive.

**Table 12-2. EBR-Based Single-Port Memory Port Definitions**

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
Clock	CLK	Clock
ClockEn	CE	Clock Enable
Address	AD[x:0]	Address Bus
Data	DI[y:0]	Data In
Q	DO[y:0]	Data Out
WE	WE	Write Enable
Reset	RST	Reset
—	CS[2:0]	Chip Select

Each EBR block consists of 18,432 bits of RAM. The values for x (address) and y (data) of each EBR block are listed in Table 12-3.

**Table 12-3. Single-Port Memory Sizes for 18K Memories in ECP5 Devices**

Single Port Memory Size	Input Data	Output Data	Address [MSB:LSB]
16,384 x 1	DI	DO	AD[13:0]
8,192 x 2	DI[1:0]	DO[1:0]	AD[12:0]
4,096 x 4	DI[3:0]	DO[3:0]	AD[11:0]
2,048 x 9	DI[8:0]	DO[8:0]	AD[10:0]
1,024 x 18	DI[17:0]	DO[17:0]	AD[9:0]
512 x 36	DI[35:0]	DO[35:0]	AD[8:0]

Table 12-4 shows the various attributes available for the Single-Port Memory (RAM\_DQ). Some of these attributes are user-selectable through the Clarity Designer GUI.

The ones that do not have selectable option in Clarity Designer are handled by the engine. However, users working with the direct primitive instantiation can access these options.

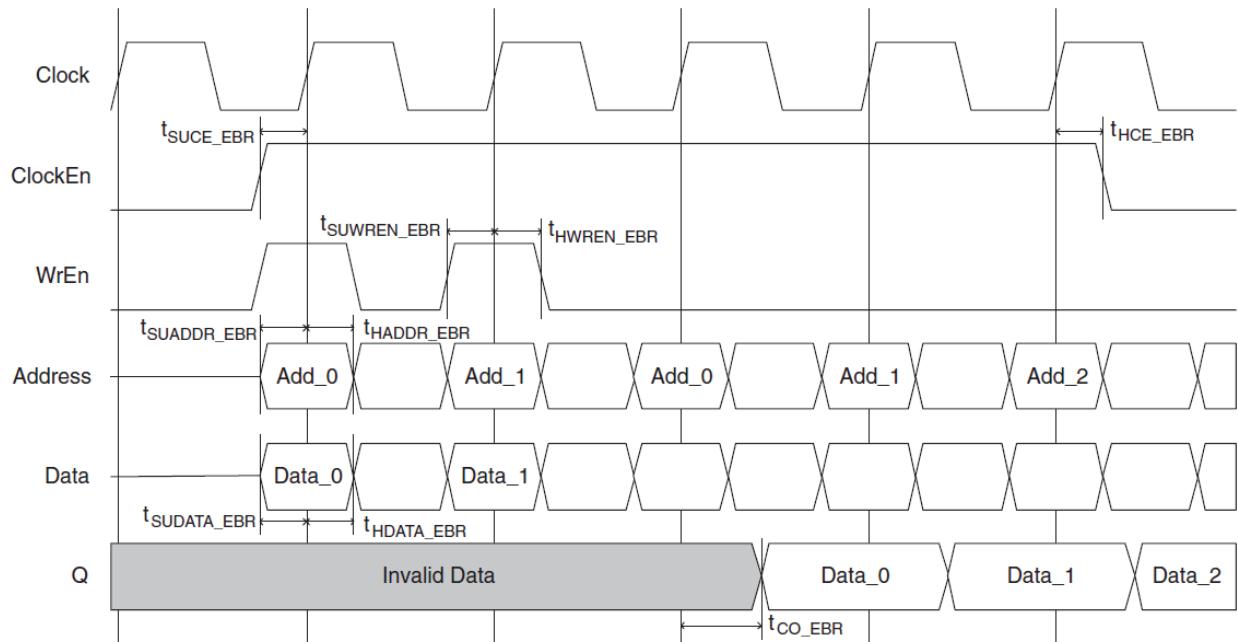
**Table 12-4. Single-Port Memory Attribute Definitions for ECP5 Devices**

Configuration Tab Attributes	Description	Values	Default Value
Address Depth	Address depth of the read and write port	2 – <Max that can fit in the device>	512
Data Width	Data word width of the Read and write port	1 – 256	36
Enable Output Register	Data Out port (Q) can be registered or not using this selection.	TRUE, FALSE	TRUE
Enable Output ClockEn	Clock Enable for the output clock (this option requires Enabling Output Register)	TRUE, FALSE	FALSE
Byte Enables	Allows users to select Byte Enable options	TRUE, FALSE	FALSE
Byte Size	Byte Size selection when Byte Enable option is selected	9, 8	9
Reset Mode	Selection for the Reset to be Synchronous or Asynchronous to the Clock	Async, Sync	Sync
Reset Release	Selection for the release of Asynchronous Reset to be Synchronous or Asynchronous to the Clock	ASYNC, SYNC	SYNC
Optimization	Design optimizations to configure EBR for Speed or Area	Area, Speed	Speed
Initialization	Allows users to initialize their memories to all 1's, 0's or providing a custom initialization by providing a memory file	0's, 1's, File	0's
Memory File	When Memory file is selected, used can browse to the mem file for custom initialization of RAM.		
Memory File Format	This option allows users to select if the memory file is formatted as Binary, Hex or address Hex. (Check Appendix for details on different formats)	Binary, Hex, Addressed Hex	Binary
Enable ECC	Option allows users to enable Error Correction Codes. This option is not available for memory that are wider than 64 bits.	TRUE, FALSE	FALSE
Advanced Tab Attributes	Description	Values	Default Value
Write Mode	Option to select different write modes. Check Timing waveforms for the behavior of RAMs in these modes	NORMAL, WRITE THROUGH, READ BEFORE WRITE	NORMAL

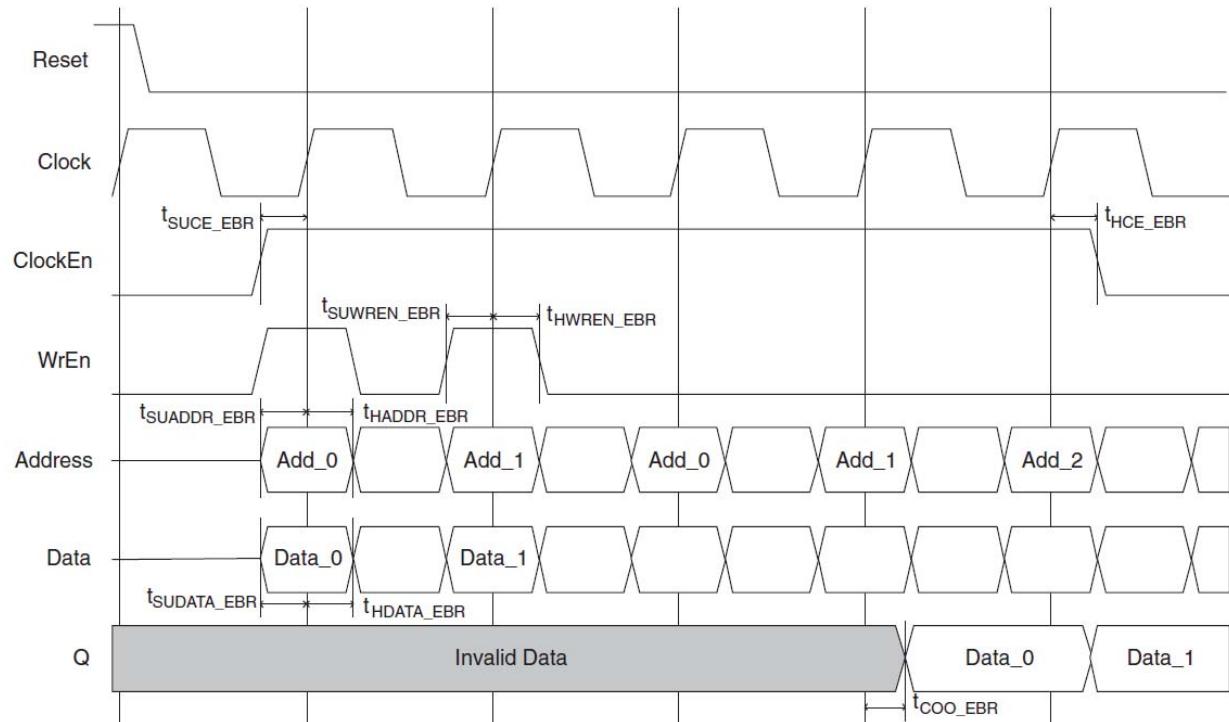
The Single-Port RAM (RAM\_DQ) can be configured as NORMAL, WRITE THROUGH or READBEFOREWRITE modes. Each of these modes affects the data coming out of port Q of the memory during the write operation followed by the read operation at the same memory location.

Additionally, users can select to enable the output registers for RAM\_DQ. The waveforms in the figures in the following pages show the internal timing waveforms for the Single Port RAM (RAM\_DQ) with these options.

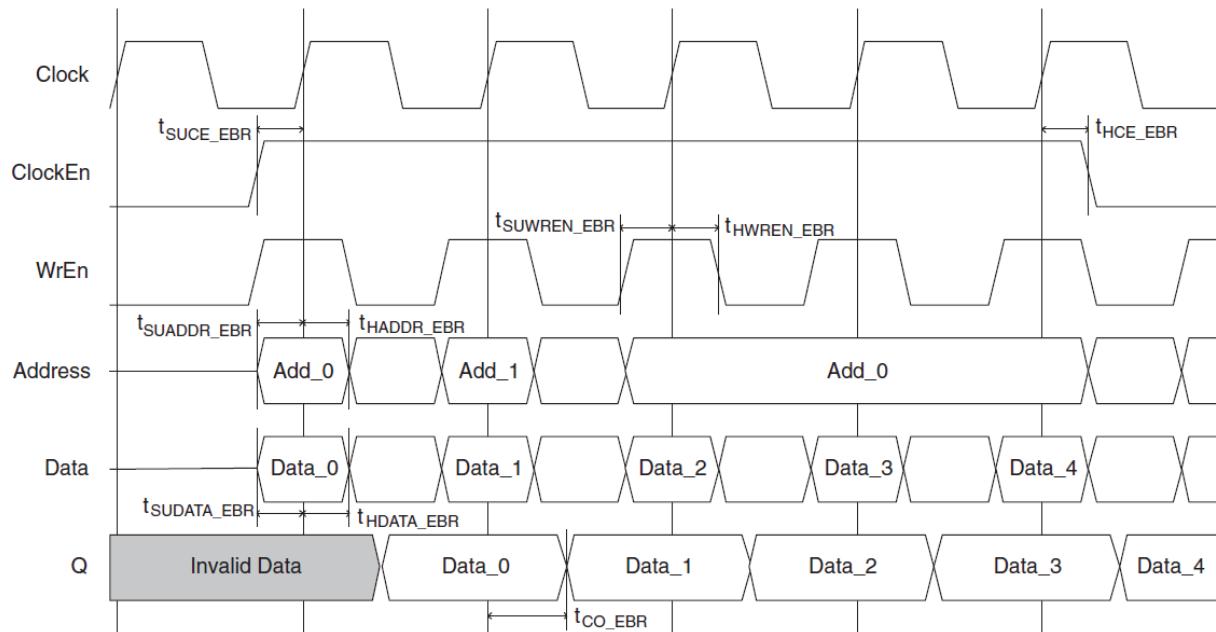
**Figure 12-7. Single Port RAM Timing Waveform in Normal (NORMAL) Mode, without Output Registers**



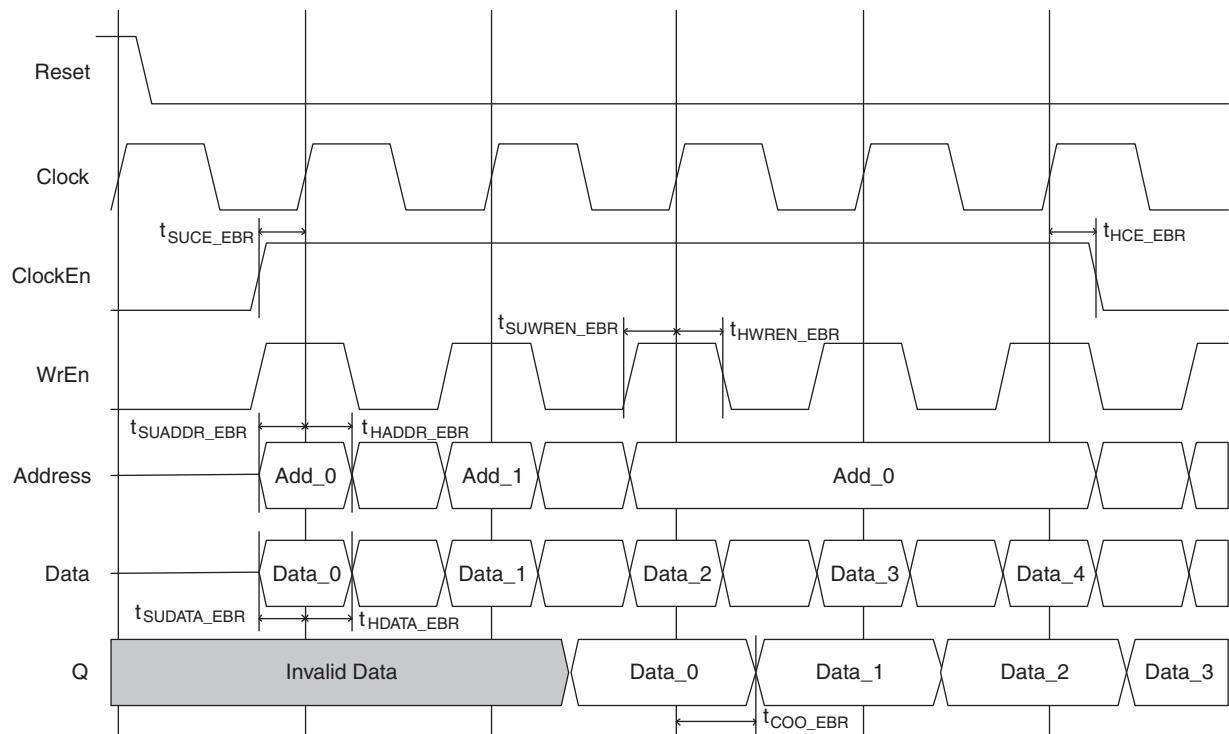
**Figure 12-8. Single Port RAM Timing Waveform in Normal (NORMAL) Mode, with Output Registers**



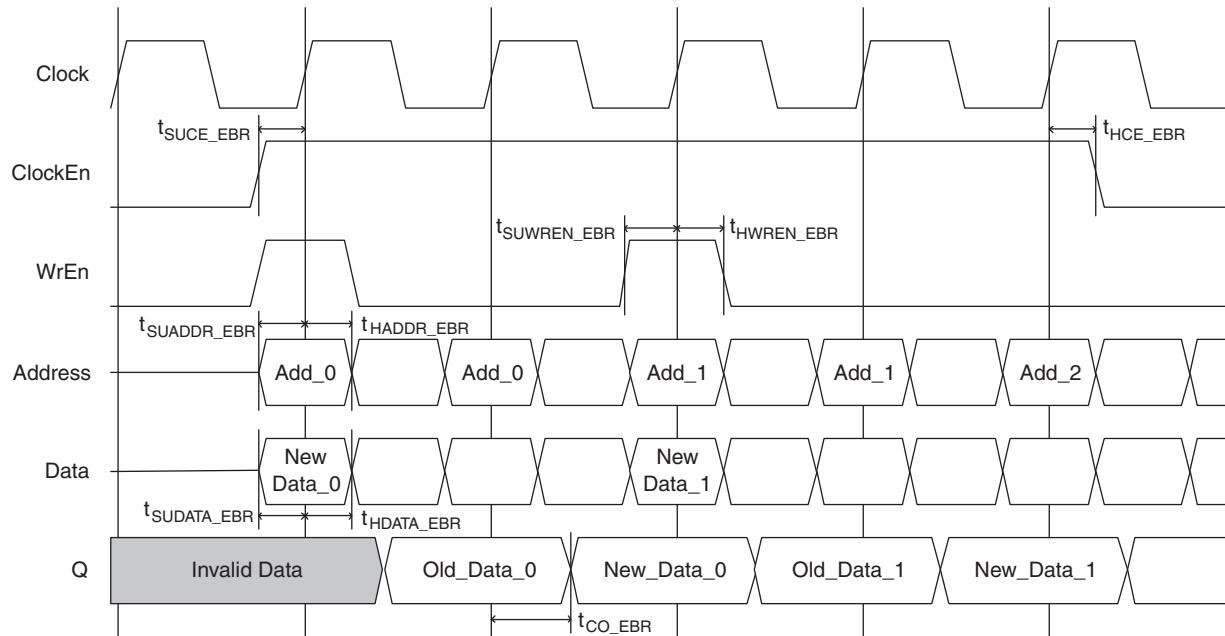
**Figure 12-9. Single Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode, without Output Registers**



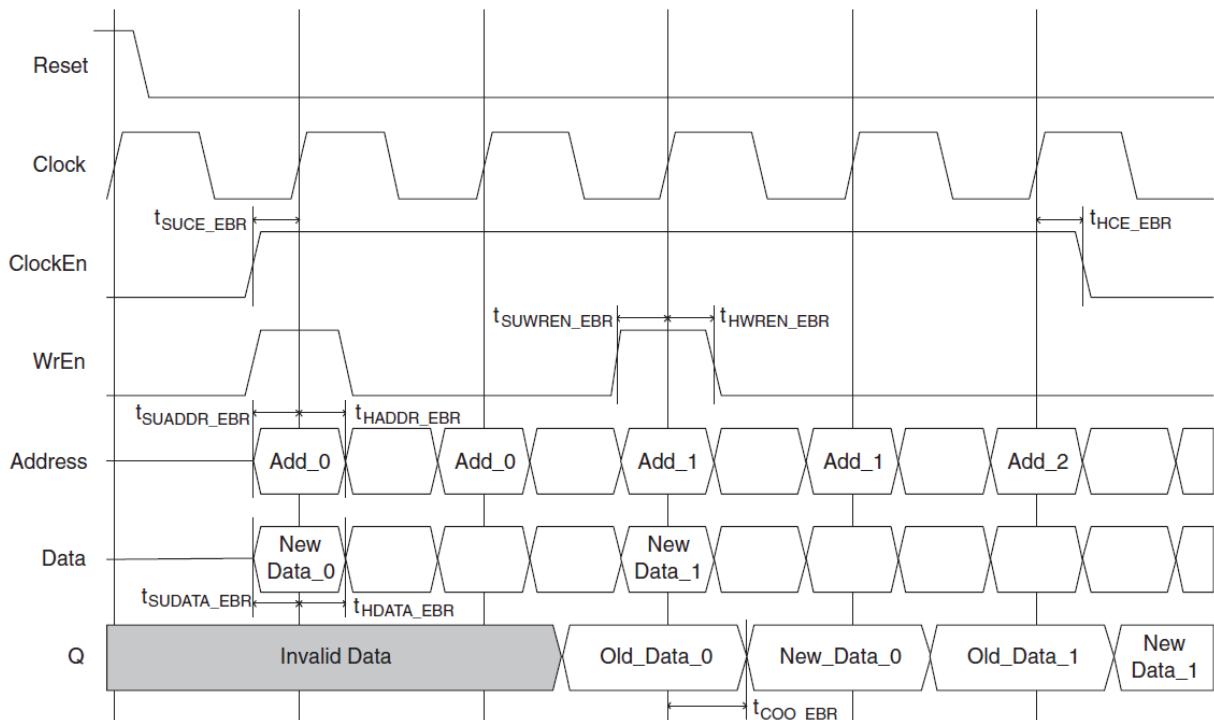
**Figure 12-10. Single Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode, with Output Registers**



**Figure 12-11. Single Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode, without Output Registers**



**Figure 12-12. Single Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode, with Output Registers**



## True Dual-Port RAM (RAM\_DP\_TRUE) – EBR Based

The EBR blocks in the ECP5 devices can be configured as True-Dual Port RAM or RAM\_DP\_TRUE. Clarity Designer allows users to generate the Verilog-HDL, VHDL or EDIF netlists for the memory size as per design requirements.

Clarity Designer generates the memory module, as shown in Figure 12-13.

**Figure 12-13. True Dual-Port Memory Module Generated by Clarity Designer**

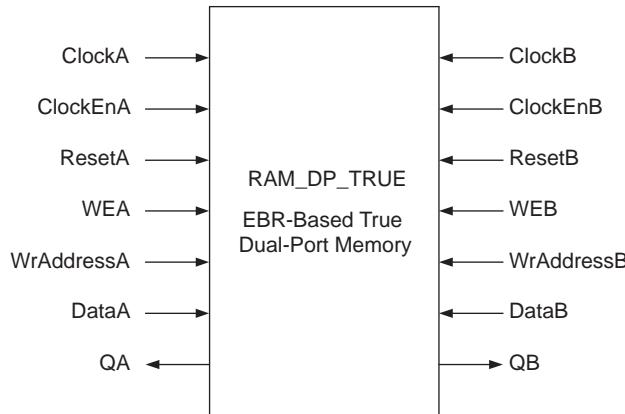
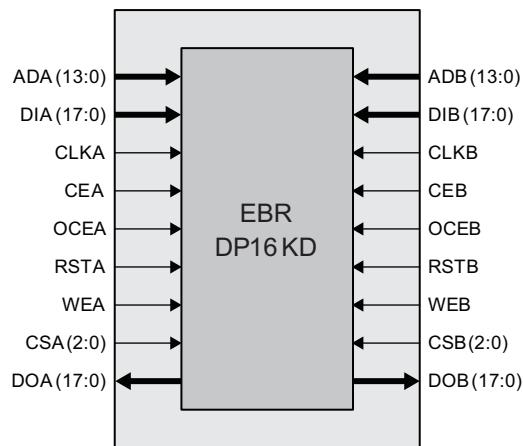


Figure 12-14 provides the primitive that can be instantiated for the True Dual Port RAM. Primitive name is DP16KD and it can be directly instantiated in the code. Check the details on the port and port names under the primitives available under cae\_library/synthesis folder in Lattice Diamond software installation folder.

It is to be noted that each EBR can accommodate 18K bits of memory; if the memory required is larger than 18K, then cascading can be used, using the CS port (CSA and CSB in this case). Check Appendix A for CSDECODE section on how to do cascading of multiple EBR primitives.

**Figure 12-14. True Dual Port RAM Primitive for ECP5 Devices**



The various ports and their definitions for True Dual-Port RAM are listed in Table 12-5. The table lists the corresponding ports for the module generated by Clarity Designer and for the EBR RAM\_DP\_TRUE primitive.

**Table 12-5. EBR-Based True Dual-Port Memory Port Definitions**

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
ClockA, ClockB	CLKA, CLKB	Clock for PortA and PortB
ClockEnA, ClockEnB	CEA, CEB	Clock Enables for Port CLKA and CLKB
AddressA, AddressB	ADA[w:0], ADB[x:0]	Address Bus port A and port B
DataA, DataB	DIA[y:0], DIB[z:0]	Input Data port A and port B
QA, QB	DOA[y:0], DOB[z:0]	Output Data port A and port B
WEA, WEB	WEA, WEB	Write enable port A and port B
ResetA, ResetB	RSTA, RSTB	Reset for PortA and PortB
-	CSA[2:0], CSB[2:0]	Chip Selects for each port

Each EBR block consists of 18,432 bits of RAM. The values for address (w and x) and data (y and z) of each EBR block are listed in Table 12-6.

**Table 12-6. Dual Port Memory Sizes for 18K Memory for ECP5**

Dual Port Memory Size	Input Data Port A	Input Data Port B	Output Data Port A	Output Data Port B	Address Port A	Address Port B
16384 x 1	DataInA	DataInB	QA	QB	AddressA(13:0)	AddressB(13:0)
8192 x 2	DataInA(1:0)	DataInB(1:0)	QA(1:0)	QB(1:0)	AddressA(12:0)	AddressB(12:0)
4096 x 4	DataInA(3:0)	DataInB(3:0)	QA(3:0)	QB(3:0)	AddressA(11:0)	AddressB(11:0)
2049 x 9	DataInA(8:0)	DataInB(8:0)	QA(8:0)	QB(8:0)	AddressA(10:0)	AddressB(10:0)
1024 x 18	DataInA(17:0)	DataInB(17:0)	QA(17:0)	QB(17:0)	AddressA(9:0)	AddressB(9:0)

Table 12-7 shows the various attributes available for True Dual-Port Memory (RAM\_DQ). Some of these attributes are user-selectable through the Clarity Designer GUI.

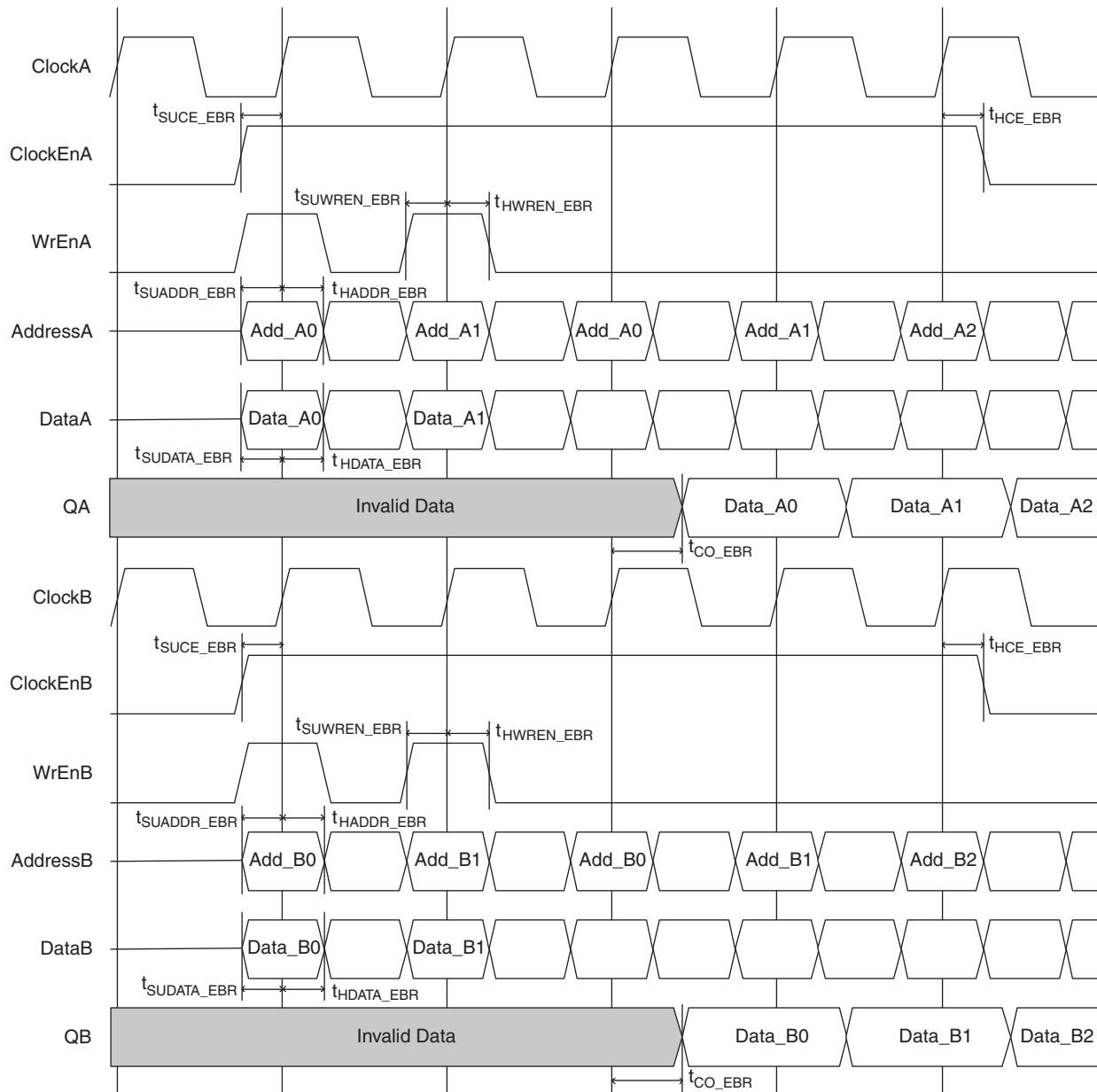
**Table 12-7. True Dual-Port RAM Attributes for ECP5 Devices**

Configuration Tab Attributes	Description	Values	Default Value
Port A Address Depth	Port A Address depth of the read and write port	2 – <Max that can fit in the device>	512
Port A Data Width	Port A Data word width of the Read and write port	1 – 256	36
Port B Address Depth	Port B Address depth of the read and write port	2 – <Max that can fit in the device>	512
Port B Data Width	Port B Data word width of the Read and write port	1 – 256	36
Port A Enable Output Register	Port A Data Out port (QA) can be registered or not using this selection.	TRUE, FALSE	TRUE
Port A Enable Output ClockEn	Port A Clock Enable for the output clock (this option requires Enabling Output Register)	TRUE, FALSE	FALSE
Port B Enable Output Register	Port B Data Out port (QB) can be registered or not using this selection.	TRUE, FALSE	TRUE
Port B Enable Output ClockEn	Port B Clock Enable for the output clock (this option requires Enabling Output Register)	TRUE, FALSE	FALSE
Byte Enables	Allows users to select Byte Enable options	TRUE, FALSE	FALSE
Byte Size	Byte Size selection when Byte Enable option is selected	9, 8	9
Reset Mode	Selection for the Reset to be Synchronous or Asynchronous to the Clock	Async, Sync	Sync
Reset Release	Selection for the release of Asynchronous Reset to be Synchronous or Asynchronous to the Clock	ASYNC, SYNC	SYNC
Optimization	Design optimizations to configure EBR for Speed or Area	Area, Speed	Speed
Initialization	Allows users to initialize their memories to all 1's, 0's or providing a custom initialization by providing a memory file	0's, 1's, File	0's
Memory File	When Memory file is selected, used can browse to the mem file for custom initialization of RAM.		
Memory File Format	This option allows users to select if the memory file is formatted as Binary, Hex or address Hex. (Check Appendix for details on different formats)	Binary, Hex, Addressed Hex	Binary
Enable ECC	Option allows users to enable Error Correction Codes. This option is not available for memory that are wider than 64 bits.	TRUE, FALSE	FALSE
Advanced Tab Attributes	Description	Values	Default Value
Port A Write Mode	Option to select different write modes for Port A. Check Timing waveforms for the behavior of RAMs in these modes	NORMAL, WRITE THROUGH, READ BEFORE WRITE	NORMAL
Port B Write Mode	Option to select different write modes for Port B. Check Timing waveforms for the behavior of RAMs in these modes	NORMAL, WRITE THROUGH, READ BEFORE WRITE	NORMAL

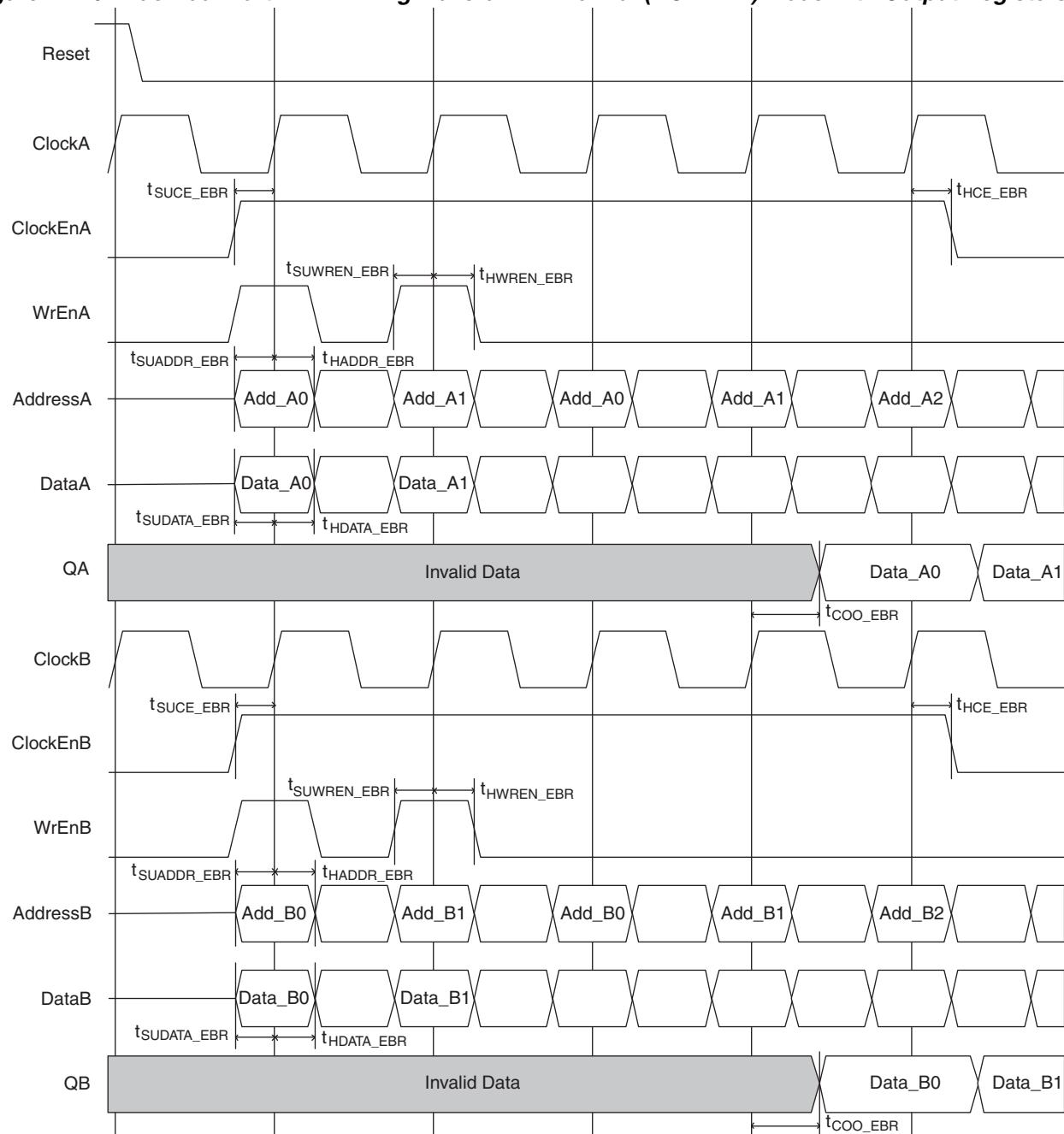
The True Dual Port RAM (RAM\_DP\_TRUE) can be configured as NORMAL, WRITE THROUGH or READBEFOREWRITE modes. Each of these modes affects what data comes out of the port Q of the memory during the write operation followed by the read operation at the same memory location. The detailed discussions of the WRITE modes and the constraints of the True Dual Port can be found in Appendix A.

Additionally, users can select to enable the output registers for RAM\_DP\_TRUE. Waveforms in the following figures show the internal timing waveforms for the True Dual Port RAM (RAM\_DP\_TRUE) with these options.

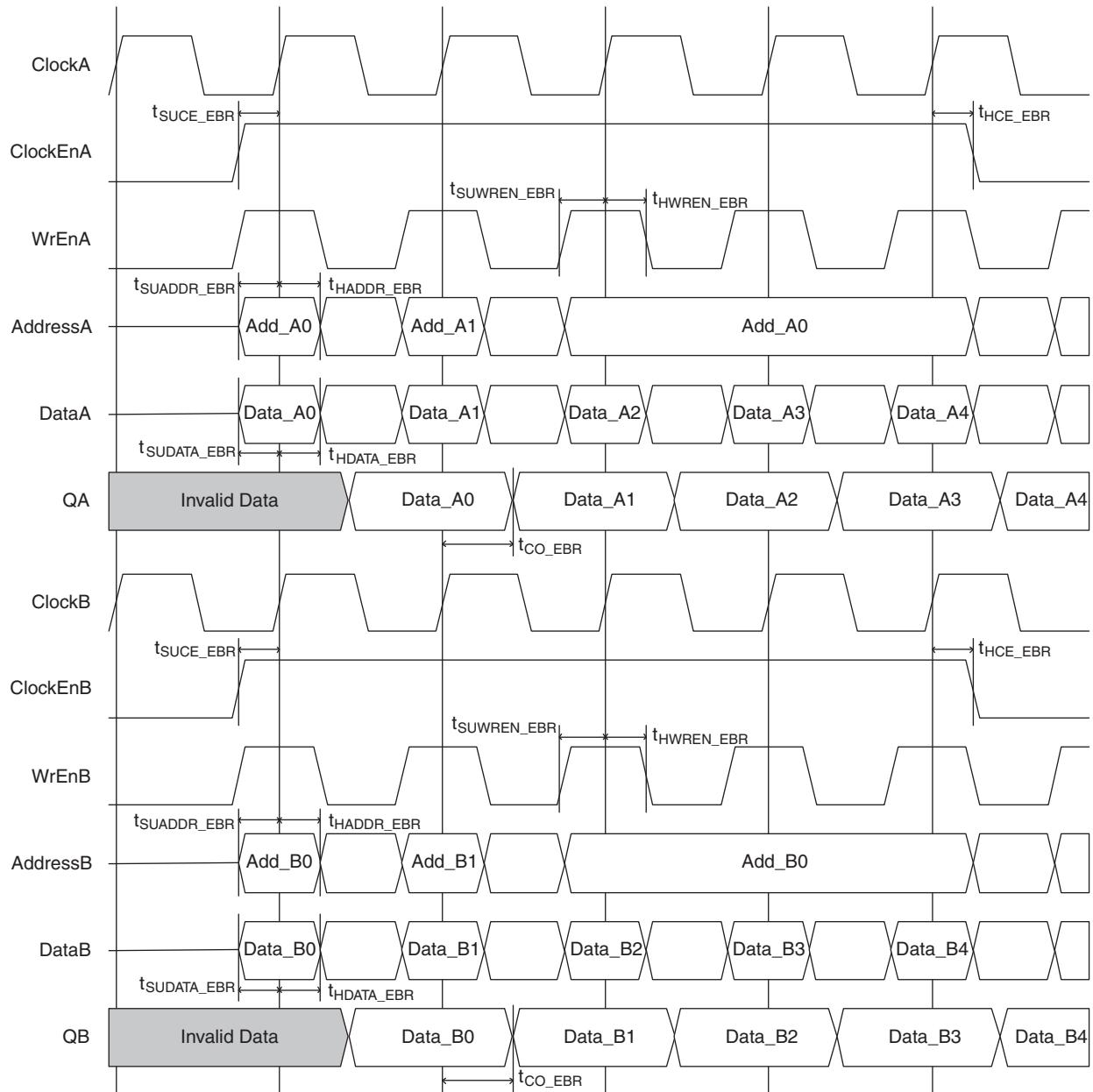
**Figure 12-15. True Dual Port RAM Timing Waveform in Normal (NORMAL) Mode, without Output Registers**



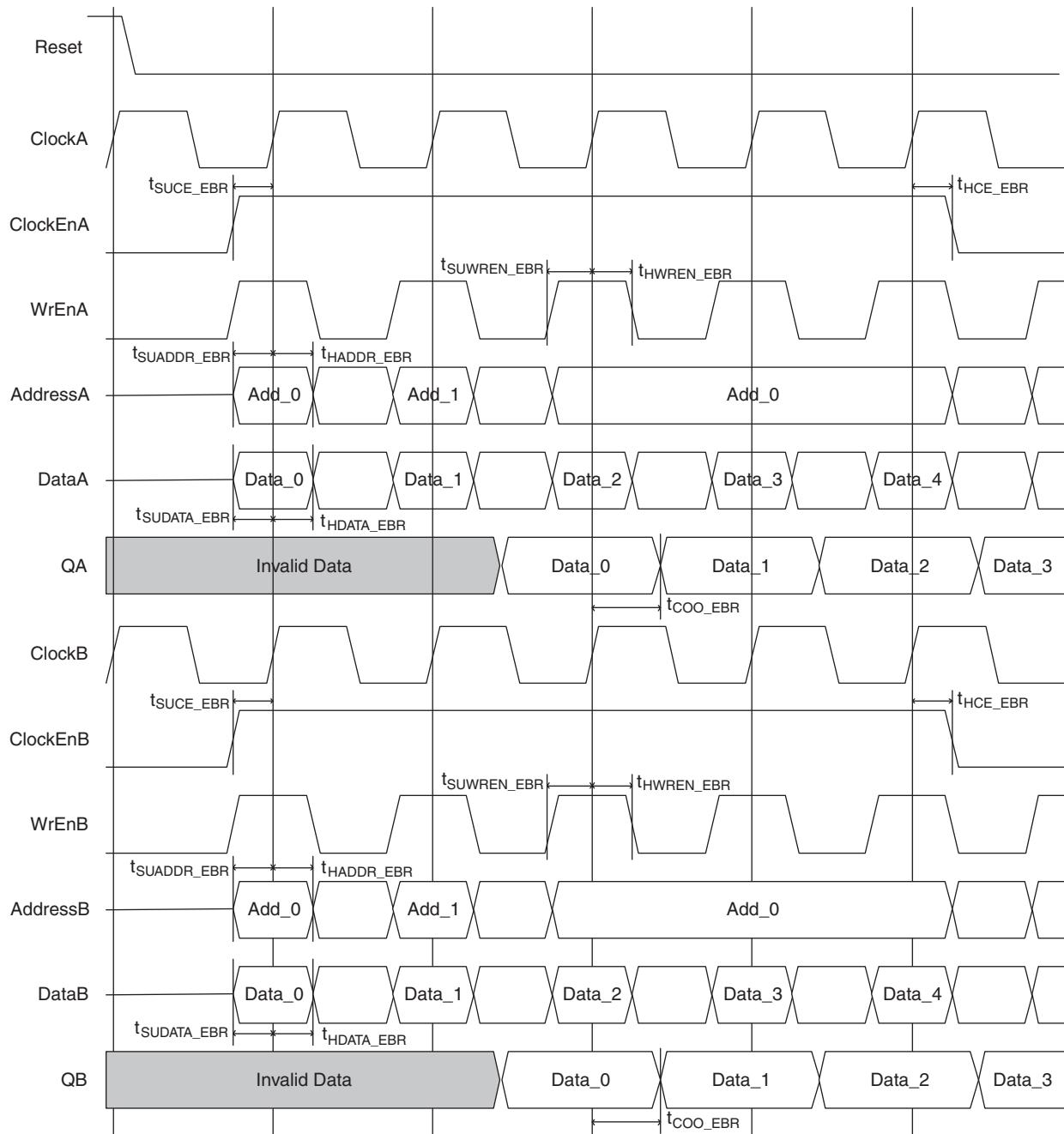
**Figure 12-16. True Dual Port RAM Timing Waveform in Normal (NORMAL) Mode with Output Registers**



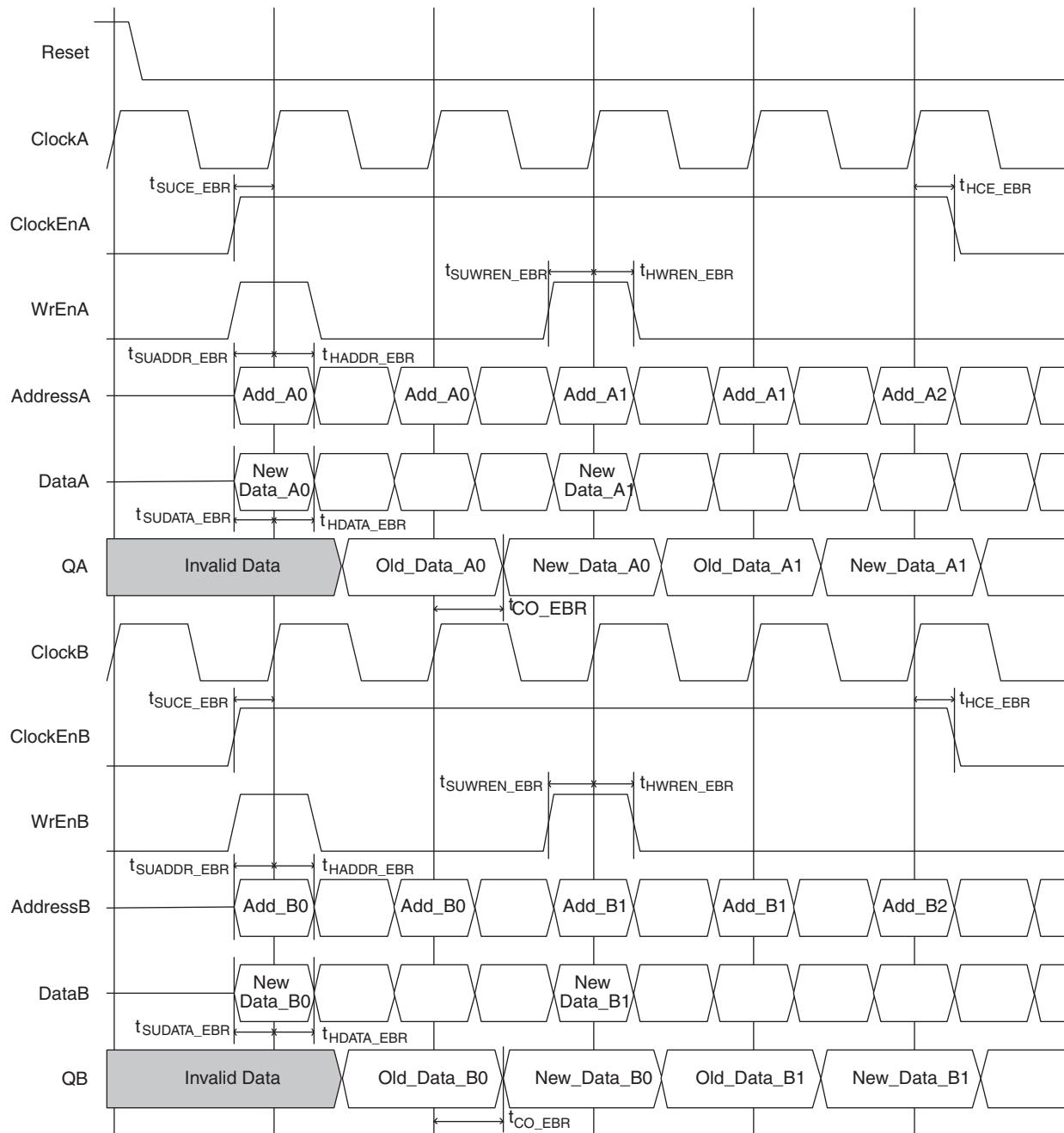
**Figure 12-17. True Dual Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode, without Output Registers**



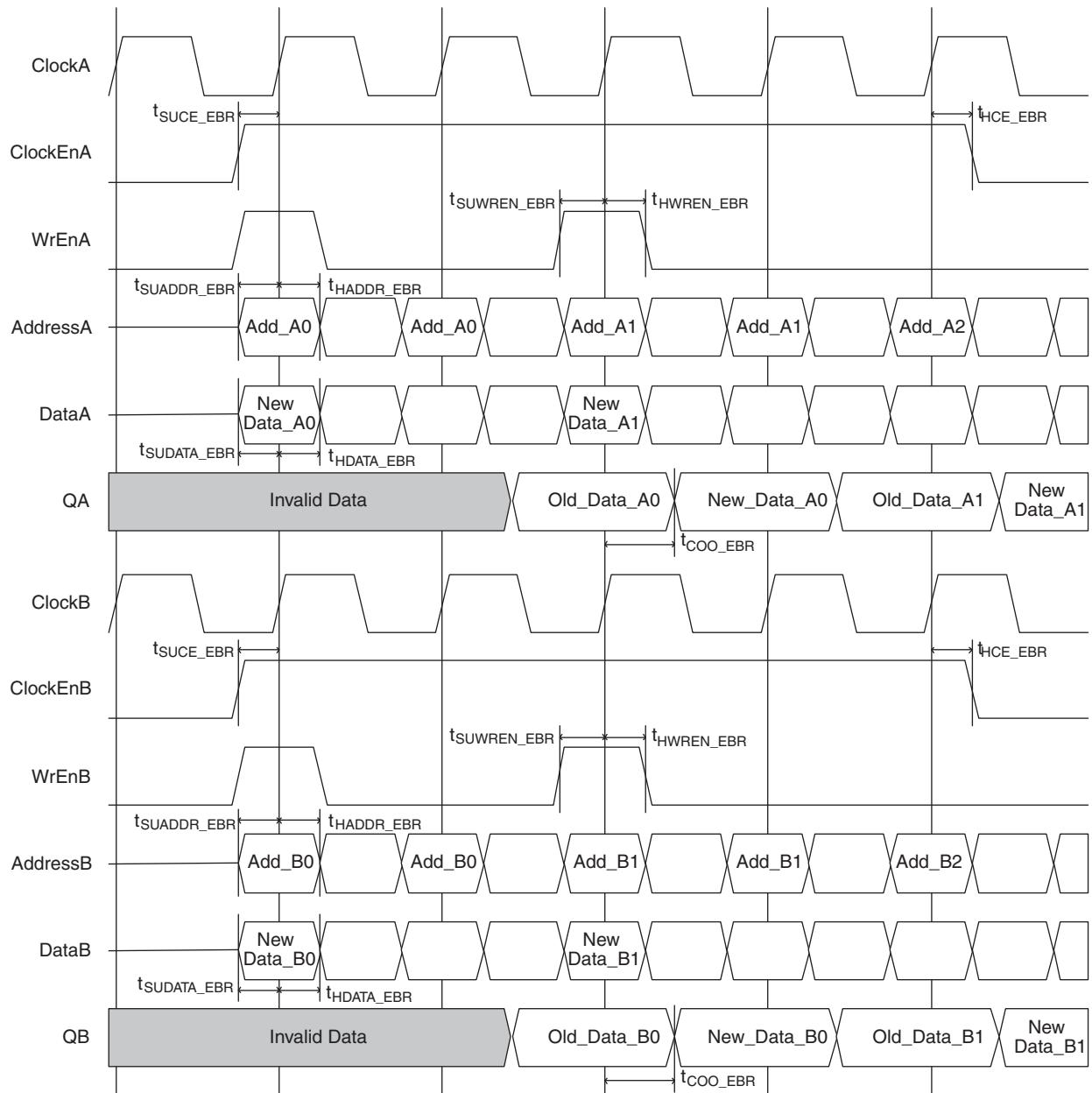
**Figure 12-18. True Dual Port RAM Timing Waveform in Write Through (WRITETHROUGH) Mode, with Output Registers**



**Figure 12-19. True Dual Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode, without Output Registers**



**Figure 12-20. True Dual Port RAM Timing Waveform in Read Before Write (READBEFOREWRITE) Mode, with Output Registers**



## Pseudo Dual-Port RAM (RAM\_DP) – EBR Based

ECP5 FPGAs supports all the features of Pseudo-Dual Port Memory Module or RAM\_DP. Clarity Designer allows users to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirement.

Clarity Designer generates the memory module shown in Figure 12-21.

**Figure 12-21. Pseudo Dual-Port Memory Module Generated by Clarity Designer**

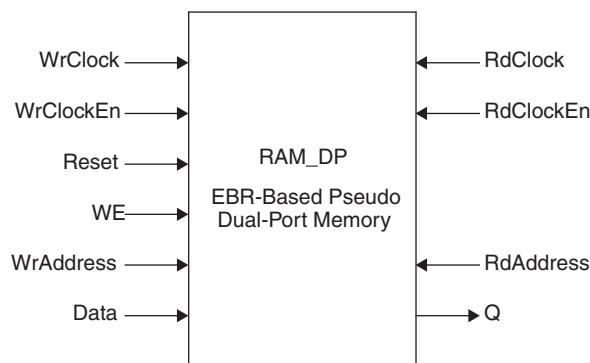
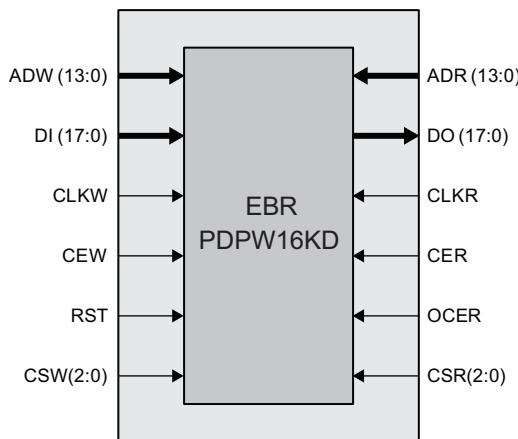


Figure 12-22 provides the primitive that can be instantiated for the Pseudo-Dual Port RAM. Primitive name is PDPW16KD and it can be directly instantiated in the code. Check the details on the port and port names under the primitives available under cae\_library/synthesis folder in Lattice Diamond software installation folder.

It is to be noted that each EBR can accommodate 18K bits of memory; if the memory required is larger than 18K, then cascading can be used, using the CS port (CSA and CSB in this case). Check Appendix A for CSDECODE section on how to do cascading of multiple EBR primitives.

**Figure 12-22. Pseudo-Dual Port RAM Primitive for ECP5 Devices**



The various ports and their definitions for Pseudo Dual-Port memory are listed in Table 12-8. The table lists the corresponding ports for the module generated by Clarity Designer and for the EBR RAM\_DP primitive.

**Table 12-8. EBR-Based Pseudo Dual-Port Memory Port Definitions**

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
RdAddress	ADR[w:0]	Read Address
WrAddress	ADW[x:0]	Write Address
RdClock	CLKR	Read Clock
WrClock	CLKW	Write Clock
RdClockEn	CER	Read Clock Enable
WrClockEn	CEW	Write Clock Enable
Q	DO[y:0]	Read Data
Data	DI[z:0]	Write Data
WE	WE	Write Enable
Reset	RST	Reset
—	CS[2:0]	Chip Select

Each EBR block consists of 18,432 bits of RAM. The values for address (w and x) and data (y and z) of each EBR block are listed in Table 12-9.

**Table 12-9. Pseudo-Dual Port Memory Sizes for 18K Memory for ECP5 Devices**

Dual Port Memory Size	Input Data Write Port	Output Data Read Port	Address Write Port	Address Read Port
16384 x 1	Data	Q	WrAddress(13:0)	RdAddress(13:0)
8192 x 2	Data(1:0)	Q(1:0)	WrAddress(12:0)	RdAddress(12:0)
4096 x 4	Data(3:0)	Q(3:0)	WrAddress(11:0)	RdAddress(11:0)
2049 x 9	Data(8:0)	Q(8:0)	WrAddress(10:0)	RdAddress(10:0)
1024 x 18	Data(17:0)	Q(17:0)	WrAddress(9:0)	RdAddress(9:0)
512 x 36	Data(35:0)	Q(35:0)	WrAddress(8:0)	RdAddress(8:0)

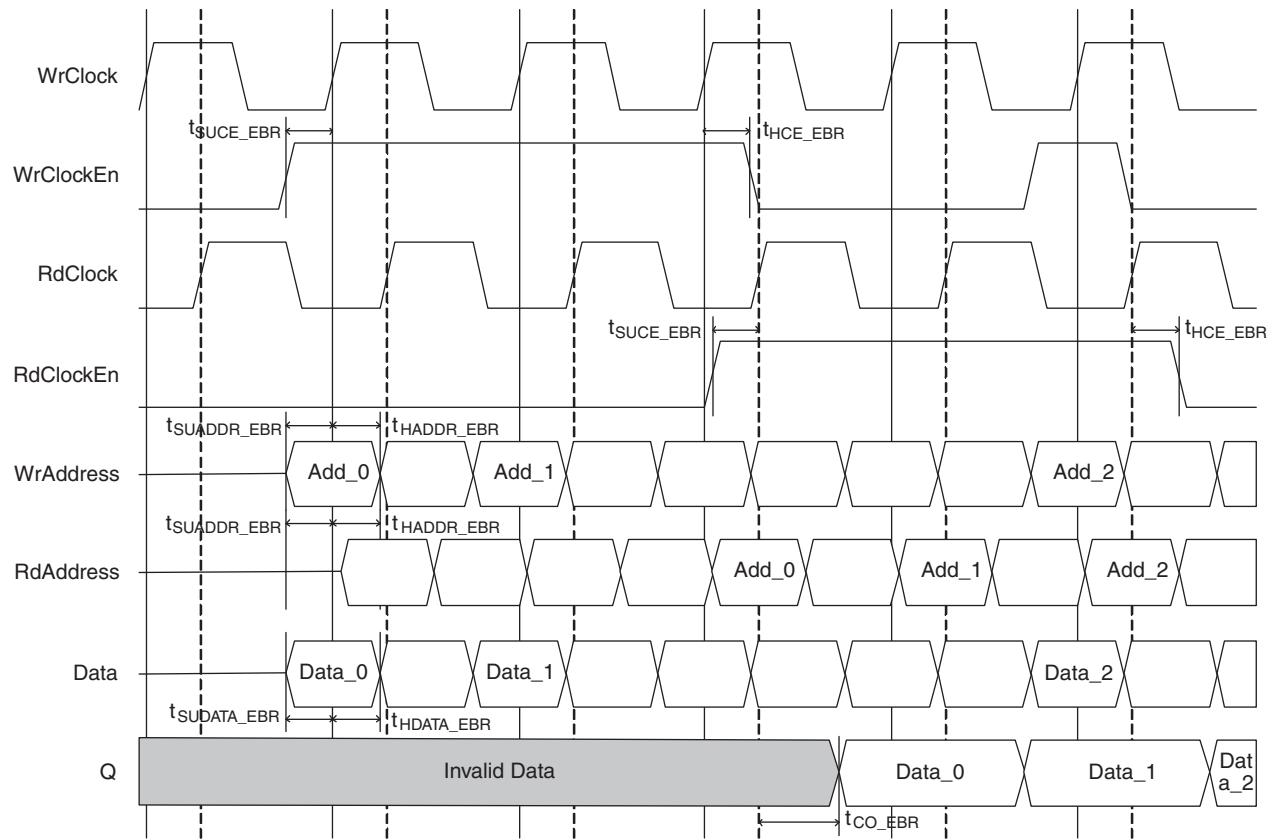
Table 12-10 shows the various attributes available for the Pseudo Dual-Port Memory (RAM\_DP). Some of these attributes are user-selectable through the Clarity Designer GUI.

**Table 12-10. Pseudo Dual-Port RAM Attributes for ECP5 Devices**

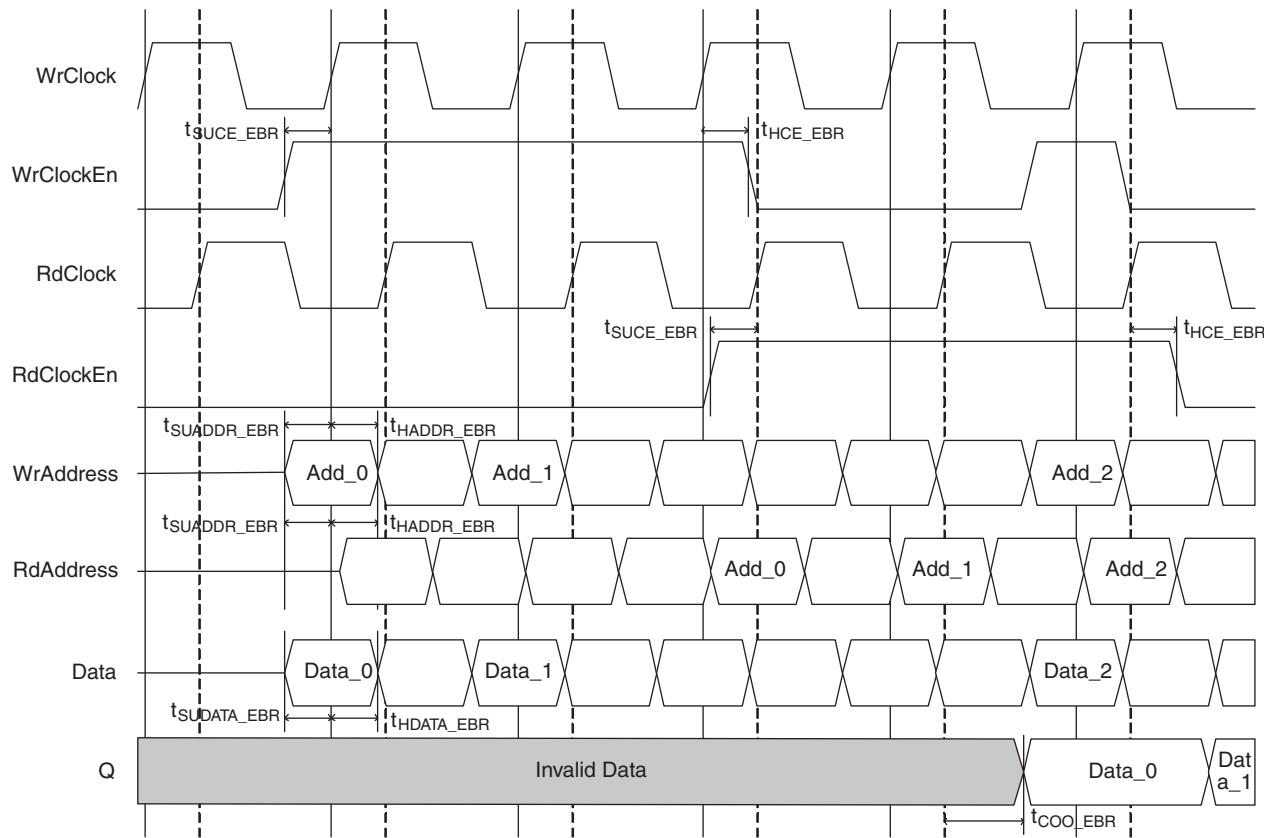
Configuration Tab Attributes	Description	Values	Default Value
Read Port Address Depth	Read Port Address depth of the read and write port	2 - 65536	512
Read Port Data Width	Read Port Data word width of the Read and write port	1 – 256	36
Write Port Address Depth	Write Port Address depth of the read and write port	2 - 65536	512
Write Port Data Width	Write Port Data word width of the Read and write port	1 – 256	36
Enable Output Register	Data Out port (Q) can be registered or not using this selection.	TRUE, FALSE	TRUE
Enable Output ClockEn	Clock Enable for the output clock (this option requires Enabling Output Register)	TRUE, FALSE	FALSE
Byte Enables	Allows users to select Byte Enable options	TRUE, FALSE	FALSE
Byte Size	Byte Size selection when Byte Enable option is selected	9, 8	9
Reset Mode	Selection for the Reset to be Synchronous or Asynchronous to the Clock	Async, Sync	Sync
Reset Release	Selection for the release of Asynchronous Reset to be Synchronous or Asynchronous to the Clock	ASYNC, SYNC	SYNC
Optimization	Design optimizations to configure EBR for Speed or Area	Area, Speed	Speed
Initialization	Allows users to initialize their memories to all 1's, 0's or providing a custom initialization by providing a memory file	0's, 1's, File	0's
Memory File	When Memory file is selected, used can browse to the mem file for custom initialization of RAM.		
Memory File Format	This option allows users to select if the memory file is formatted as Binary, Hex or address Hex. (Check Appendix for details on different formats)	Binary, Hex, Addressed Hex	Binary
Enable ECC	Option allows users to enable Error Correction Codes. This option is not available for memory that are wider than 64 bits.	TRUE, FALSE	FALSE

Users have the option to enable the output registers for Pseudo-Dual Port RAM (RAM\_DP). Waveforms in the following figures show the internal timing waveforms for Pseudo-Dual Port RAM (RAM\_DP) with these options.

**Figure 12-23. PSEUDO DUAL PORT RAM Timing Diagram - without Output Registers**



**Figure 12-24. PSEUDO DUAL PORT RAM Timing Diagram - with Output Registers**

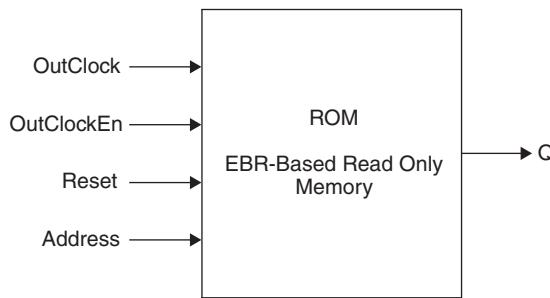


### Read Only Memory (ROM) – EBR Based

ECP5 FPGAs supports all the features of ROM Memory Module or ROM. Clarity Designer allows users to generate the Verilog-HDL or VHDL along EDIF netlist for the memory size as per design requirement. Users are required to provide the ROM memory content in a form of an initialization file.

Clarity Designer generates the memory module shown in Figure 12-25.

**Figure 12-25. ROM - Read Only Memory Module Generated by Clarity Designer**



The various ports and their definitions are listed in Table 12-11. The table lists the corresponding ports for the module generated by Clarity Designer and for the ROM primitive.

**Table 12-11. EBR-Based ROM Port Definitions**

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description
Address	AD[x:0]	Read Address
OutClock	CLK	Clock
OutClockEn	CE	Clock Enable
Reset	RST	Reset
—	CS[2:0]	Chip Select

When generating ROM using Clarity Designer, the designer must provide the initialization file to pre-initialize the contents of the ROM. These files are the \*.mem files and they can be of binary, hex or addressed hex formats. The initialization files are discussed in detail in the Initializing Memory section of this document.

Each EBR block consists of 18,432 bits of RAM. The values for x's (for address) and y's (data) for each EBR block for the devices included in Table 12-12.

**Table 12-12. ROM Memory Sizes for 16K Memory for ECP5 Devices**

Dual Port Memory Size	Output Data Read Port	Address Port
16384 x 1	Q	Address(13:0)
8192 x 2	Q(1:0)	Address(12:0)
4096 x 4	Q(3:0)	Address(11:0)
2049 x 9	Q(8:0)	Address(10:0)
1024 x 18	Q(17:0)	Address(9:0)
512 x 36	Q(35:0)	Address(9:0)

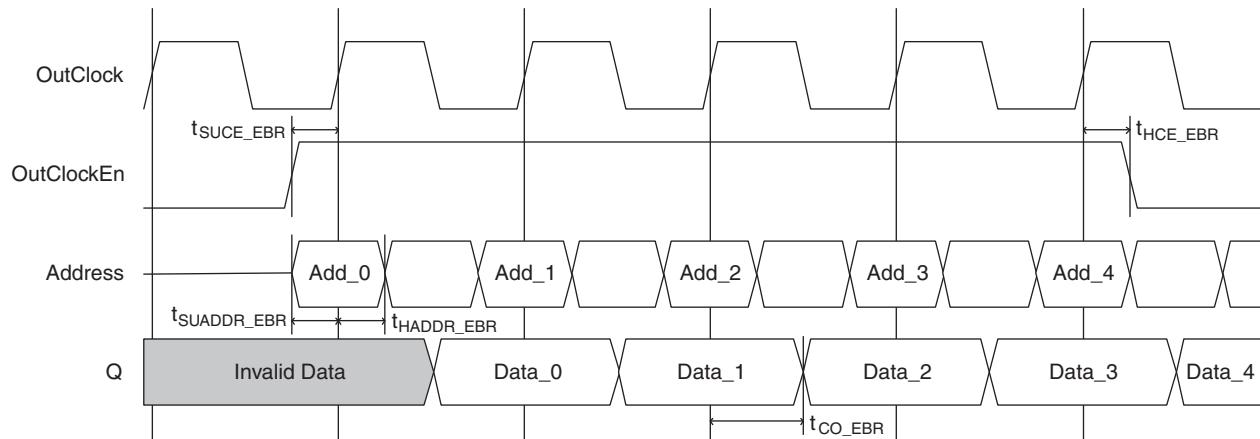
Table 12-13 shows the various attributes available for the Read Only Memory (ROM). Some of these attributes are user-selectable through the Clarity Designer GUI. For detailed attribute definitions, refer to Appendix A.

**Table 12-13. ROM Attributes for ECP5 Devices**

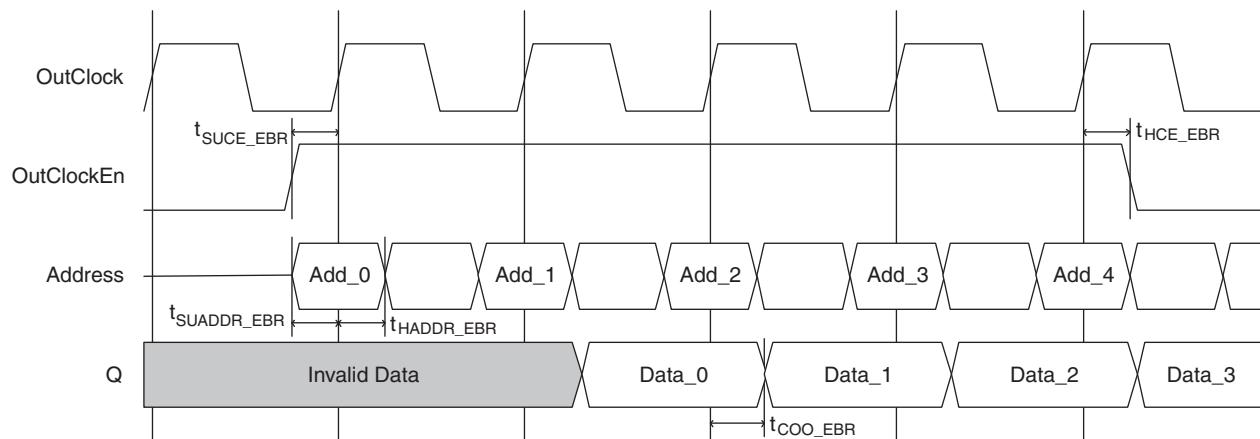
Configuration Tab Attributes	Description	Values	Default Value
Address Depth	Address depth of the read and write port	2 – 65536	512
Data Width	Data word width of the Read and write port	1 – 256	36
Enable Output Register	Data Out port (Q) can be registered or not using this selection.	TRUE, FALSE	TRUE
Optimization	Design optimizations to configure EBR for Speed or Area	Area, Speed	Speed
Reset Mode	Selection for the Reset to be Synchronous or Asynchronous to the Clock	Async, Sync	Sync
Reset Release	Selection for the release of Asynchronous Reset to be Synchronous or Asynchronous to the Clock	ASYNC, SYNC	SYNC
Initialization	Allows users to initialize their memories to all 1's, 0's or providing a custom initialization by providing a memory file	0's, 1's, File	0's
Memory File	When Memory file is selected, used can browse to the mem file for custom initialization of RAM.		
Memory File Format	This option allows users to select if the memory file is formatted as Binary, Hex or address Hex. (Check Appendix for details on different formats)	Binary, Hex, Addressed Hex	Binary
Enable ECC	Option allows users to enable Error Correction Codes. This option is not available for memory that are wider than 64 bits.	TRUE, FALSE	FALSE

Users have the option to enable the output registers for Read Only Memory (ROM). Figure 12-26 and Figure 12-27 show the internal timing waveforms for ROM with these options.

**Figure 12-26. IROM Timing Waveform - without Output Registers**



**Figure 12-27. ROM Timing Waveform - with Output Registers**



## First In First Out (FIFO) Memory

ECP5 devices support two different types of FIFOs:

- Single Clock FIFO (FIFO)
- Dual Clock FIFO (FIFO\_DC)

The FIFOs in ECP5 devices are emulated such that they are built using Dual Port RAMs with additional logic around them. The additional logic (address counters, flag logic and pointer comparators) are implemented using PFUs around the Dual Port RAMs.

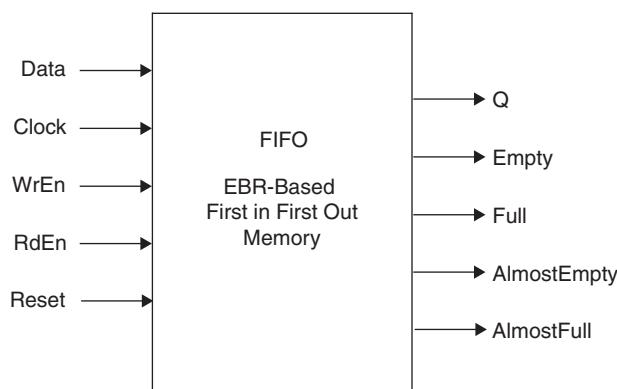
The EBR blocks in ECP5 devices can be configured as LUT based or EBR based, as well as Single Clock First-In First-Out Memory (FIFO) or Dual-Clock First-In First-Out Memory (FIFO\_DC). Clarity Designer allows users to generate the Verilog-HDL or VHDL netlist for various memory sizes depending on design requirements.

Clarity Designer generated FIFO modules and their operation are discussed in detail in the following pages.

### Single Clock FIFO (FIFO) - EBR and LUT

Figure 12-28 shows the module that gets generated by the Clarity Designer for FIFO

**Figure 12-28. FIFO Module Generated by Clarity Designer**



The various ports and their definitions for the FIFO are listed in Table 12-14.

**Table 12-14. Port Names and Definitions for FIFO**

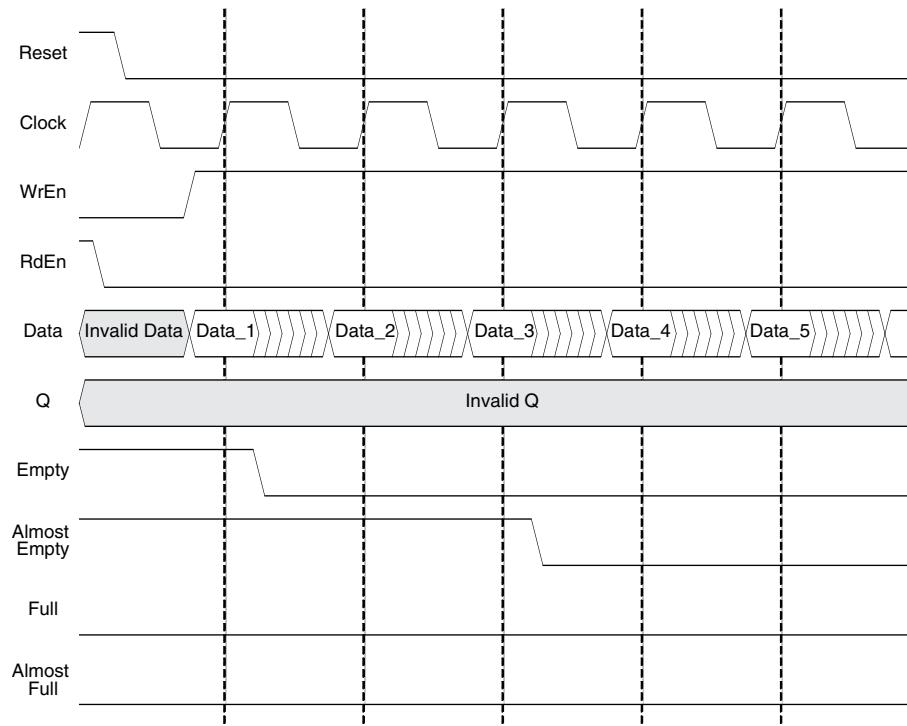
Port Name in Generated Module	Description	Active State
Data	Input Data	-
Clock	Clock	Rising Clock Edge
WrEn	Write Enable	Active High
RdEn	Read Enable	Active High
Reset <sup>2</sup>	Reset	Active High
Q	Data Output	—
Empty	Empty Flag	Active High
Full	Full Flag	Active High
AlmostEmpty	Almost Empty Flag	Active High
AlmostFull	Almost Full Flag	Active High
ERROR <sup>1</sup>	Error Check Code	Active High

1. Denotes optional port

2. Reset resets only the optional output registers, pointer circuitry and flags (except Empty, which gets set to '1') of the FIFO. It does not reset the input registers or the contents of memory.

Let us first discuss the non-pipelined or the FIFO without output registers. Figure 12-29 shows the operation of the FIFO when it is empty and the data begins to be written into it.

**Figure 12-29. FIFO Without Output Registers, Start of data Write Cycle**

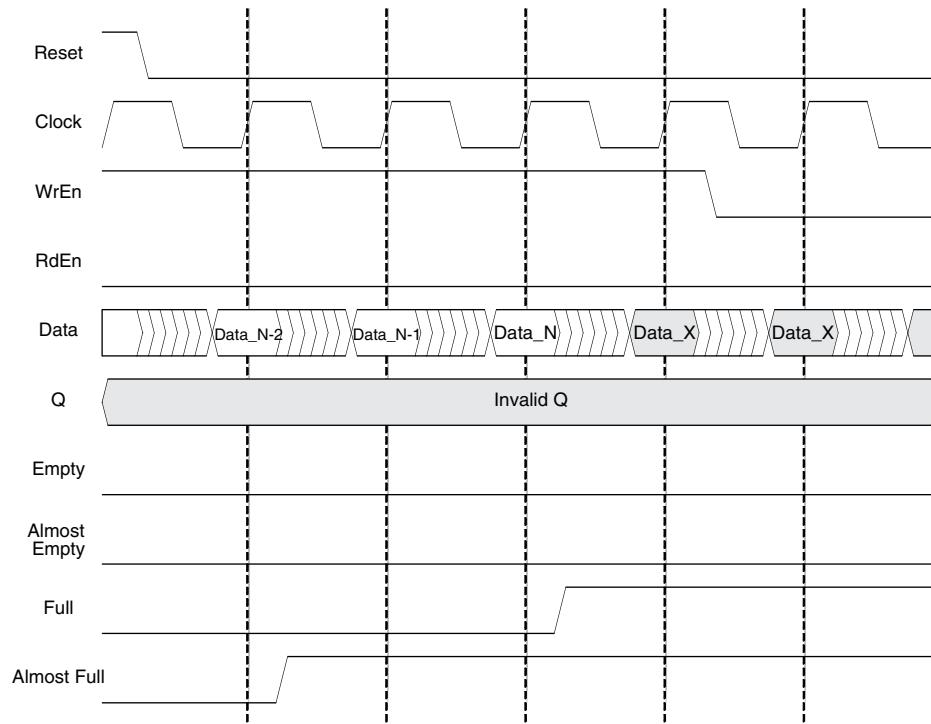


The WrEn signal must be high to start writing into the FIFO. The Empty and Almost Empty flags are high to begin and Full and Almost Full are low.

When the first data is written into the FIFO, the Empty flag de-asserts (or goes low) since the FIFO is no longer empty. In this figure we assume that the Almost Empty flag setting is 3 (address location 3). So, the Almost Empty flag is de-asserted when the third address location is filled.

Assume that we continue to write into the FIFO to fill it. When the FIFO is filled, the Almost Full and Full flags are asserted. Figure 12-30 shows the behavior of these flags. In this figure we assume that the FIFO depth is 'N'.

**Figure 12-30. FIFO Without Output Registers, End of Data Write Cycle**

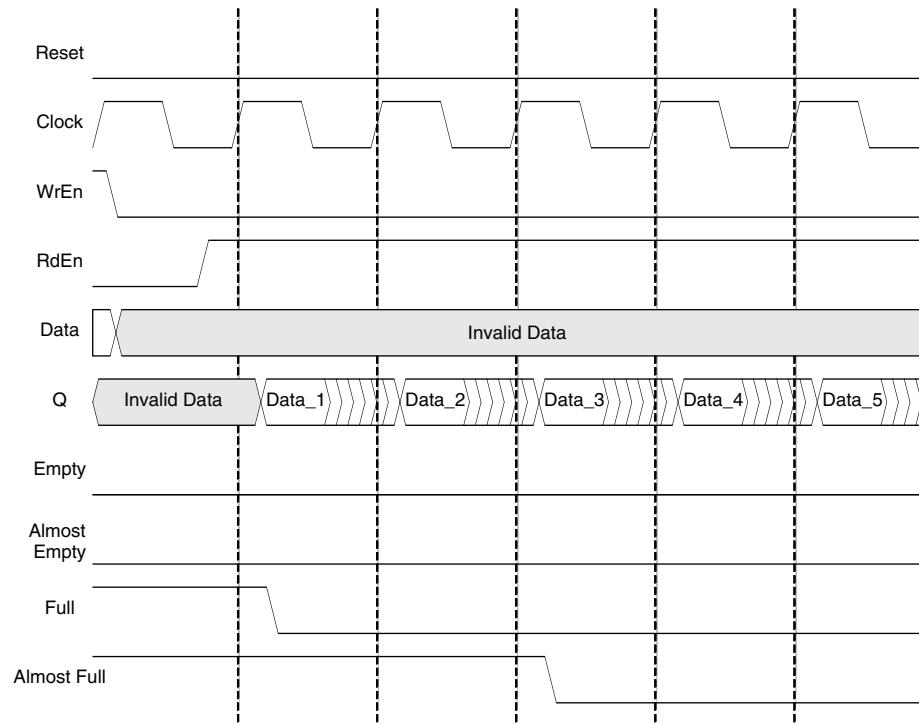


In Figure 12-30, the Almost Full flag is two locations before the FIFO is filled. The Almost Full flag is asserted when the N-2 location is written, and the Full flag is asserted when the last word is written into the FIFO.

Data\_X data inputs are not written since the FIFO is full (Full flag is high).

Now let us look at the waveforms when the contents of the FIFO are read out. Figure 12-31 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags are de-asserted, as shown.

**Figure 12-31. FIFO Without Output Registers, Start of Data Read Cycle**



Similarly, as the data is read out and FIFO is emptied, the Almost Empty and Empty flags are asserted (see Figure 12-32).

**Figure 12-32. FIFO Without Output Registers, End of Data Read Cycle**

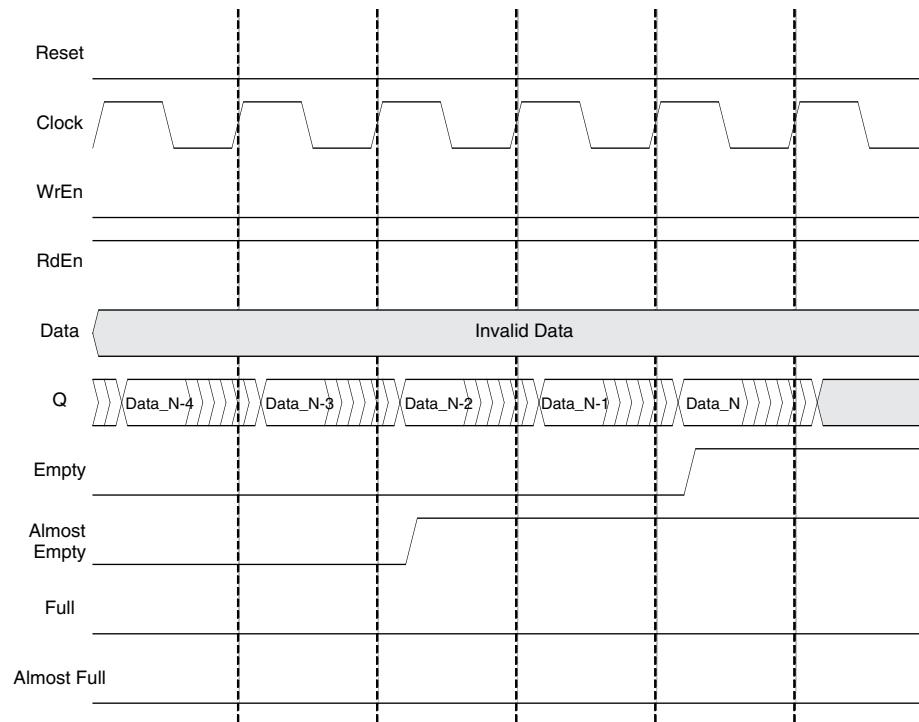
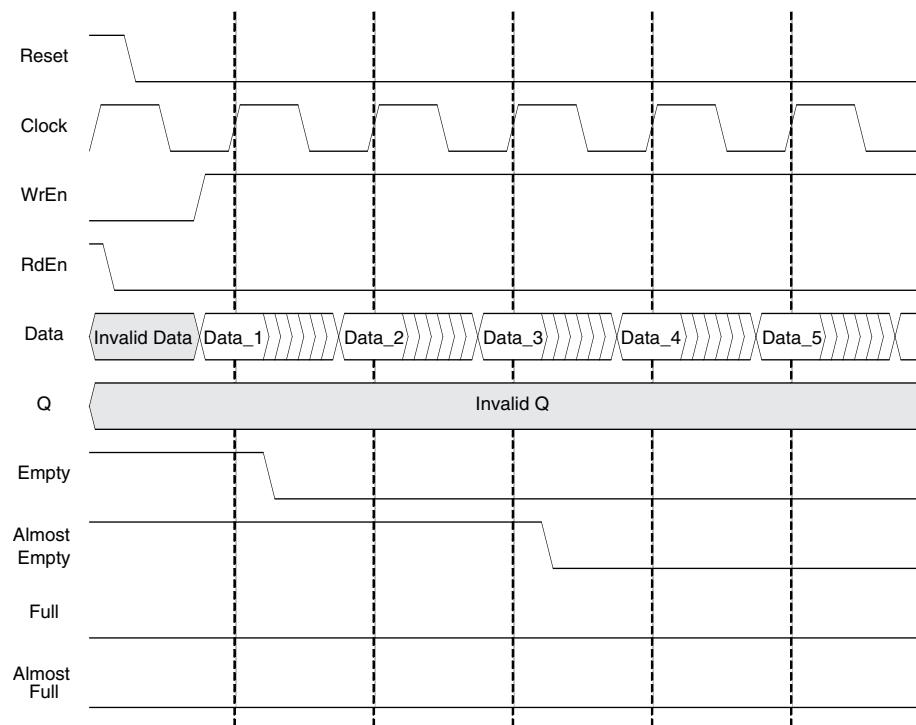


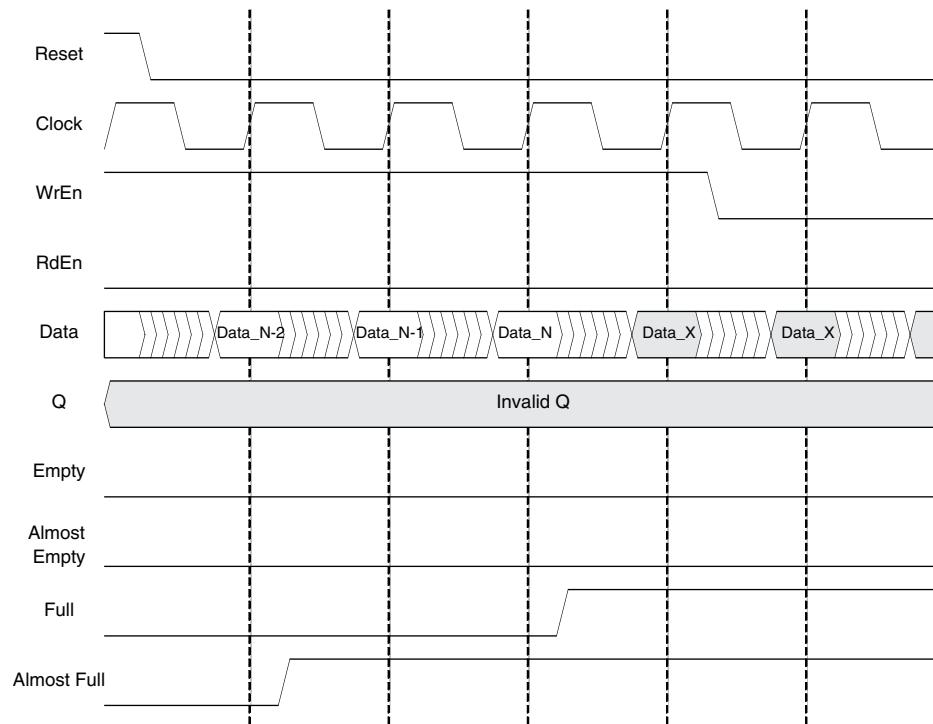
Figure 12-28 to Figure 12-32 show the behavior of non-pipelined FIFO or FIFO without output registers. When the registers are pipelined, the output data is delayed by one clock cycle. There is also an option for output registers to be enabled by the RdEn signal.

Figure 12-33 to Figure 12-36 show similar waveforms for the FIFO with an output register and an output register enable with RdEn. Note that flags are asserted and de-asserted with similar timing to the FIFO without output registers. Only the data out 'Q' gets delayed by one clock cycle.

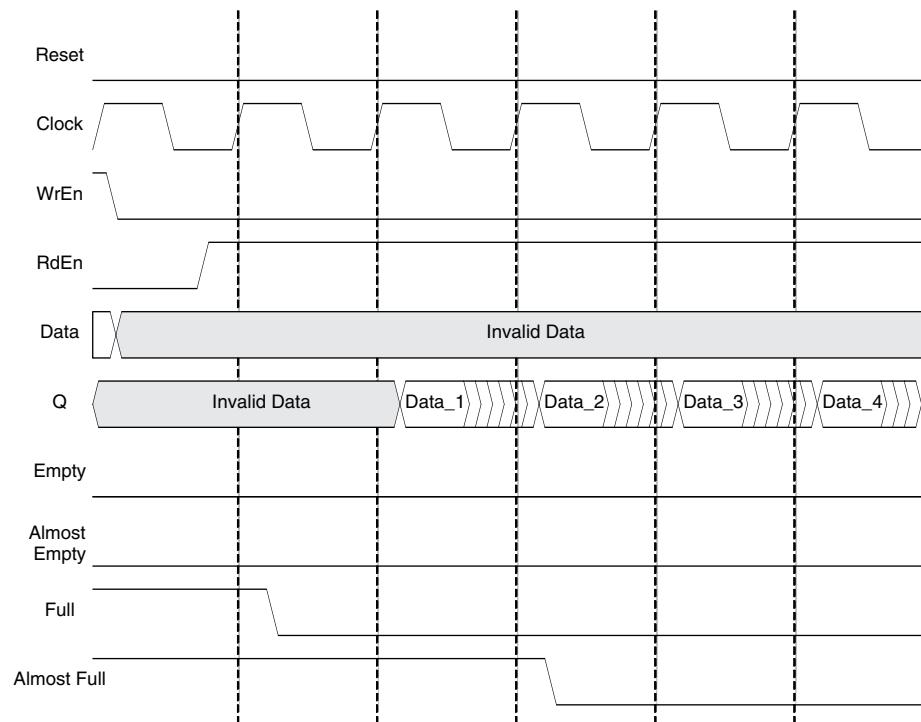
**Figure 12-33. FIFO with Output Registers, Start of Data Write Cycle**



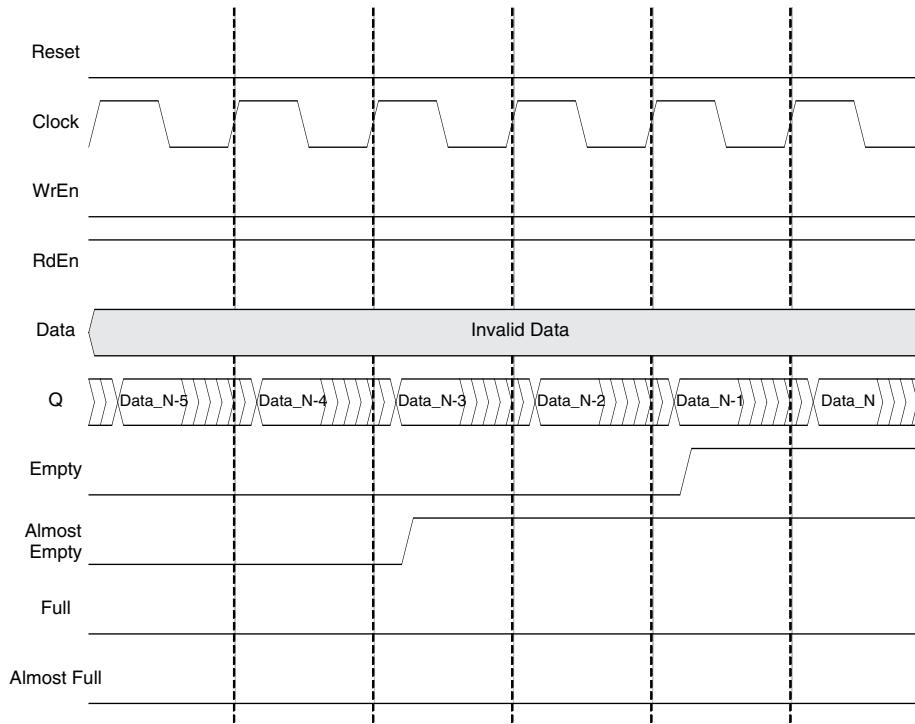
**Figure 12-34. FIFO with Output Registers, End of Data Write Cycle**



**Figure 12-35. FIFO with Output Registers, Start of Data Read Cycle**

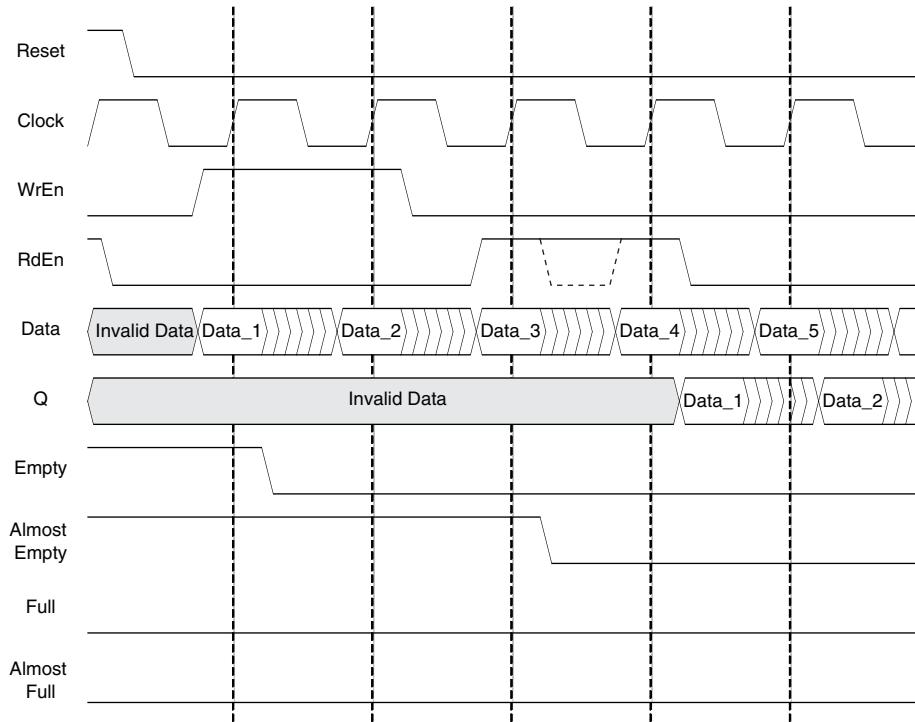


**Figure 12-36. FIFO with Output Registers, End of Data Read Cycle**



If the option enable output register with RdEn is selected, it still delays the data out by one clock cycle (as compared to the non-pipelined FIFO). The RdEn should also be high during that clock cycle, otherwise the data takes an extra clock cycle when the RdEn goes true.

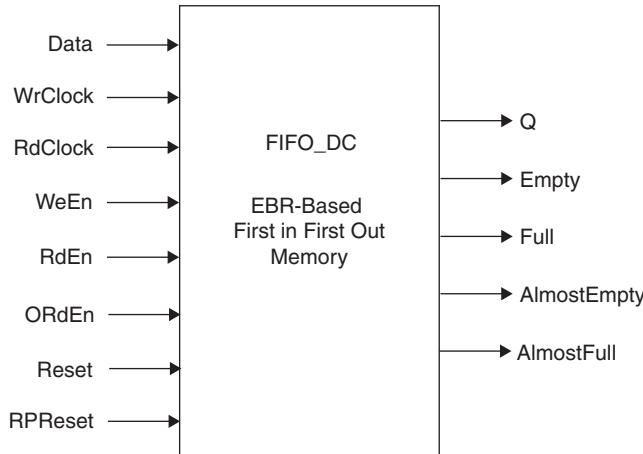
**Figure 12-37. FIFO with Output Registers and RdEn on Output Registers**



## Dual Clock First-In-First-Out (FIFO\_DC) – EBR or LUT Based

Following figure shows the module that gets generated by the Clarity Designer for FIFO

**Figure 12-38. FIFO\_DC Module generated by the Clarity Designer**



The various ports and their definitions for the FIFO\_DC are listed in Table 12-15.

**Table 12-15. Port Names and Definitions for FIFO\_DC**

Port Name in Generated Module	Description	Active State
Data	Input Data	-
WrClock	Write Port Clock	Rising Clock Edge
RdClock	Read Port Clock	Rising Clock Edge
WrEn	Write Enable	Active High
RdEn	Read Enable	Active High
*ORDEn <sup>2</sup>	Output Read Enable	Active High
Reset <sup>3</sup>	Reset	Active High
RPRReset <sup>4</sup>	Read Pointer Reset	Active High
Q	Data Output	-
Empty	Empty Flag	Active High
Full	Full Flag	Active High
AlmostEmpty	Almost Empty Flag	Active High
AlmostFull	Almost Full Flag	Active High
ERROR <sup>1</sup>	Error Check Code	Active High

1. Denotes optional port

2. ORdEn can be used as clock enable for the optional output registers. This allows the full pipeline of data to be output, including the last word.

3. Reset resets only the optional output registers, pointer circuitry and flags of the FIFO. It does not reset the input registers or the contents of memory.

4. RPRReset resets only the read pointer. See additional discussion below.

### FIFO\_DC Flags

As an emulated FIFO, FIFO\_DC requires the flags to be implemented in the FPGA logic around the block RAM. Because of the two clocks, the flags are required to change clock domains from read clock to write clock and vice versa. This adds latency to the flags either during assertion or de-assertion. Latency can be avoided only in one of the cases (either assertion or de-assertion) or distributed among the two.

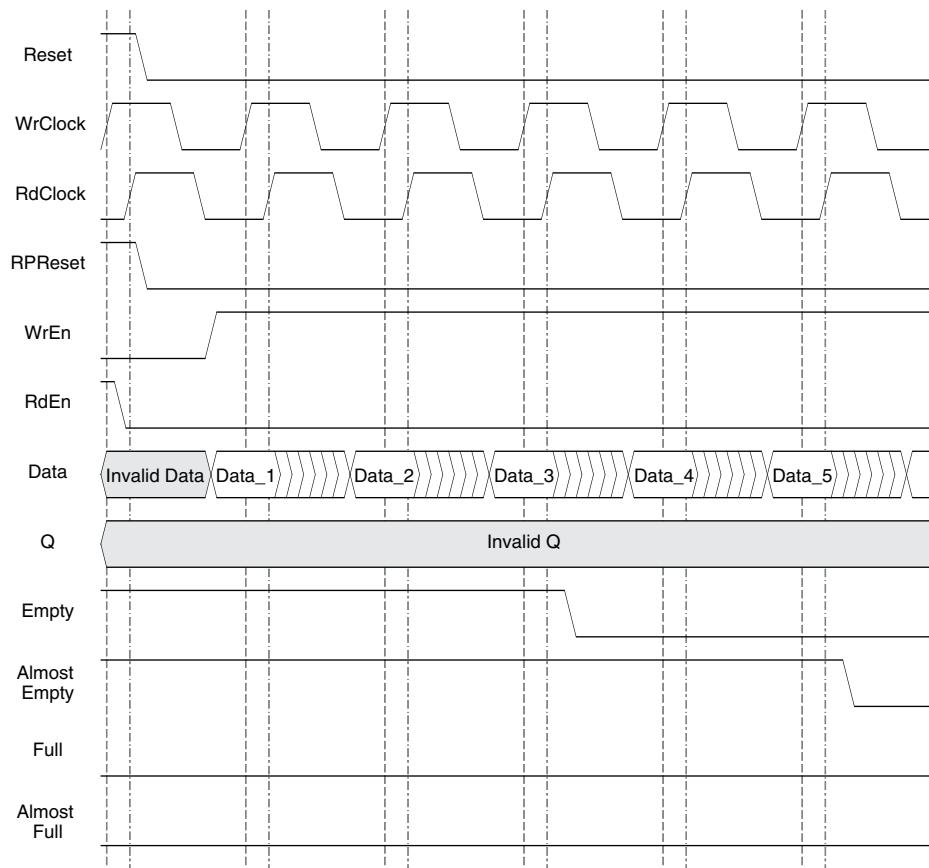
In the emulated FIFO\_DC, there is no latency during assertion of the flags. Thus, when the flag goes true, there is no latency. However, due to the design of the flag logic running on two clock domains, there is latency during de-assertion.

Let us assume that we start to write into the FIFO\_DC to fill it. The write operation is controlled by WrClock and WrEn. However, it takes extra RdClock cycles for the de-assertion of the Empty and Almost Empty flags.

De-assertion of Full and Almost Full although result of reading out the data from the FIFO\_DC, takes extra WrClock cycles after reading the data for these flags to come out.

With this in mind, let us look at the waveforms for FIFO\_DC without output registers. Figure 12-39 shows the operation of the FIFO\_DC when it is empty and the data begins to be written into it.

**Figure 12-39. FIFO\_DC Without Output Registers, Start of Data Write Cycle**

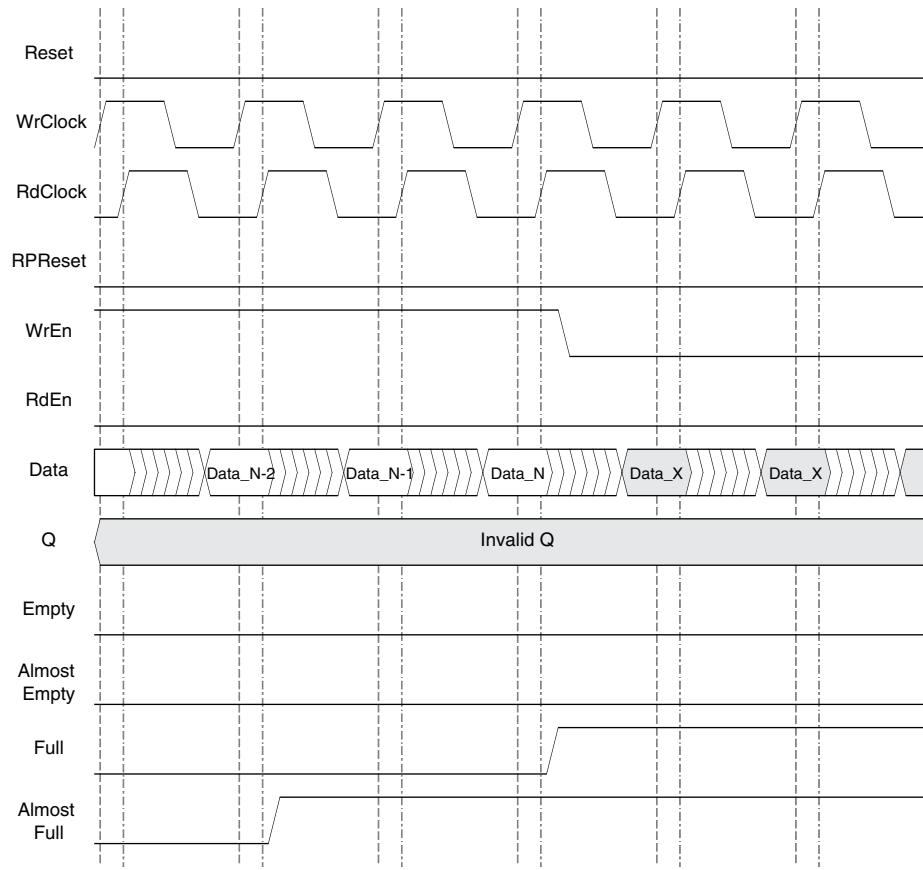


The WrEn signal has to be high to start writing into the FIFO\_DC. The Empty and Almost Empty flags are high to begin and Full and Almost Full are low.

When the first data is written into the FIFO\_DC, the Empty flag de-asserts (or goes low), as the FIFO\_DC is no longer empty. In this figure we assume that the Almost Empty setting flag setting is 3 (address location 3). The Almost Empty flag is de-asserted when the third address location is filled.

Now let us assume that we continue to write into the FIFO\_DC to fill it. When the FIFO\_DC is filled, the Almost Full and Full Flags are asserted. Figure 12-40 shows the behavior of these flags. In this figure we assume that FIFO\_DC depth is 'N'.

**Figure 12-40. FIFO\_DC Without Output Registers, End of Data Write Cycle**



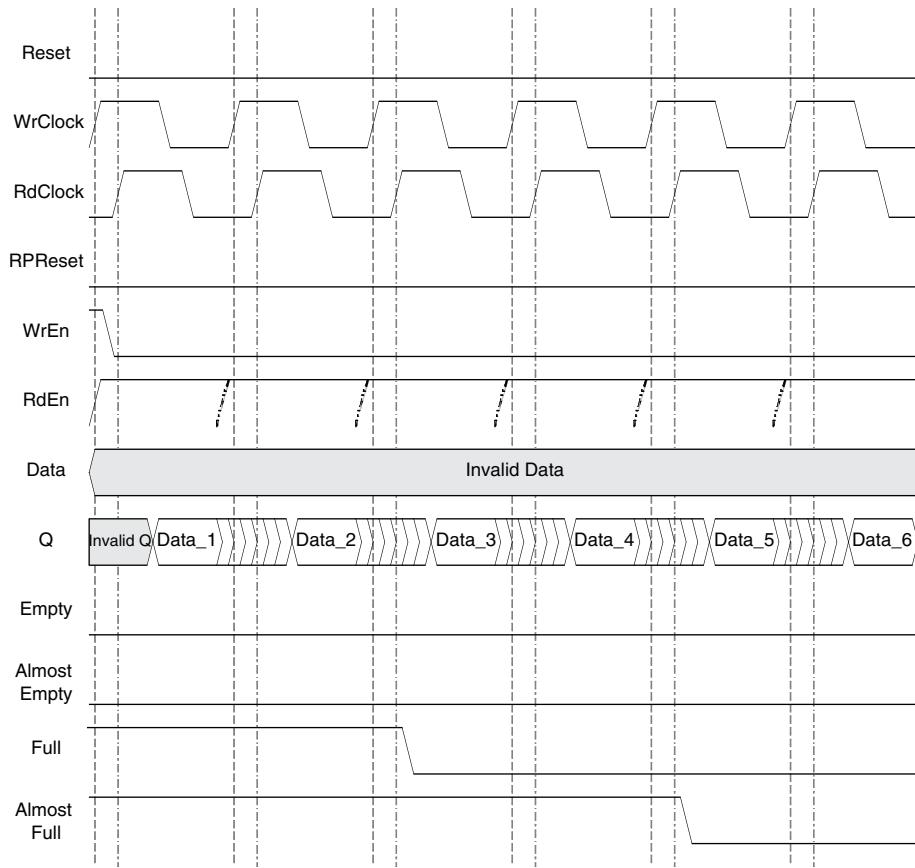
In Figure 12-40, the Almost Full flag is two locations before the FIFO\_DC is filled. The Almost Full flag is asserted when the N-2 location is written, and the Full flag is asserted when the last word is written into the FIFO\_DC.

Data\_X data inputs are not written since the FIFO\_DC is full (Full flag is high).

Note that the assertion of these flags is immediate and there is no latency when they go true.

Now let us look at the waveforms when the contents of the FIFO\_DC are read out. Figure 12-41 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags are de-asserted as shown. Note that the de-assertion is delayed by two clock cycles.

**Figure 12-41. FIFO\_DC Without Output Registers, Start of Data Read Cycle**



Similarly, as the data is read out and FIFO\_DC is emptied, the Almost Empty and Empty flags are asserted (see Figure 12-42).

**Figure 12-42. FIFO\_DC Without Output Registers, End of Data Read Cycle**

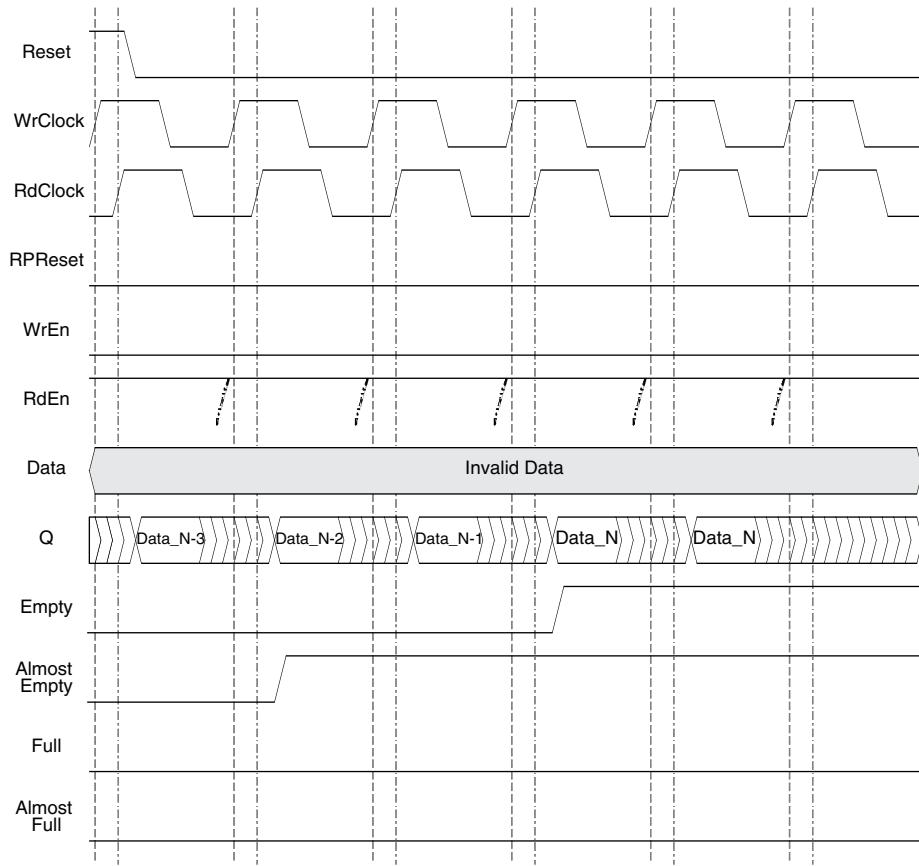
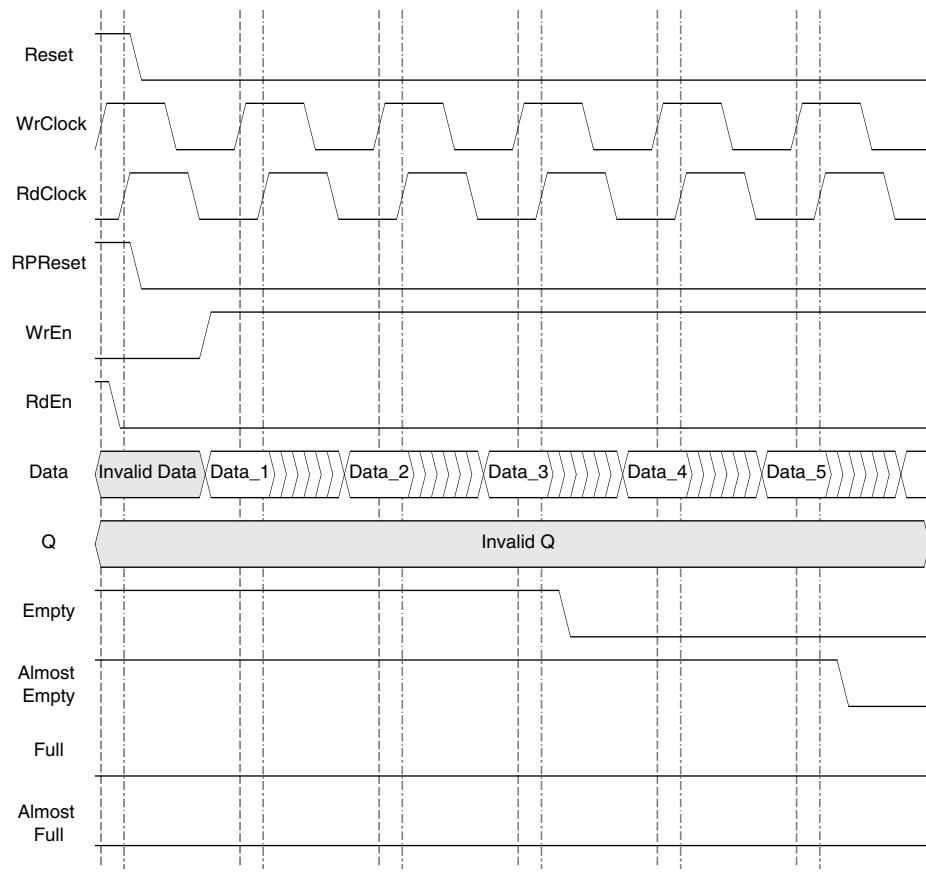


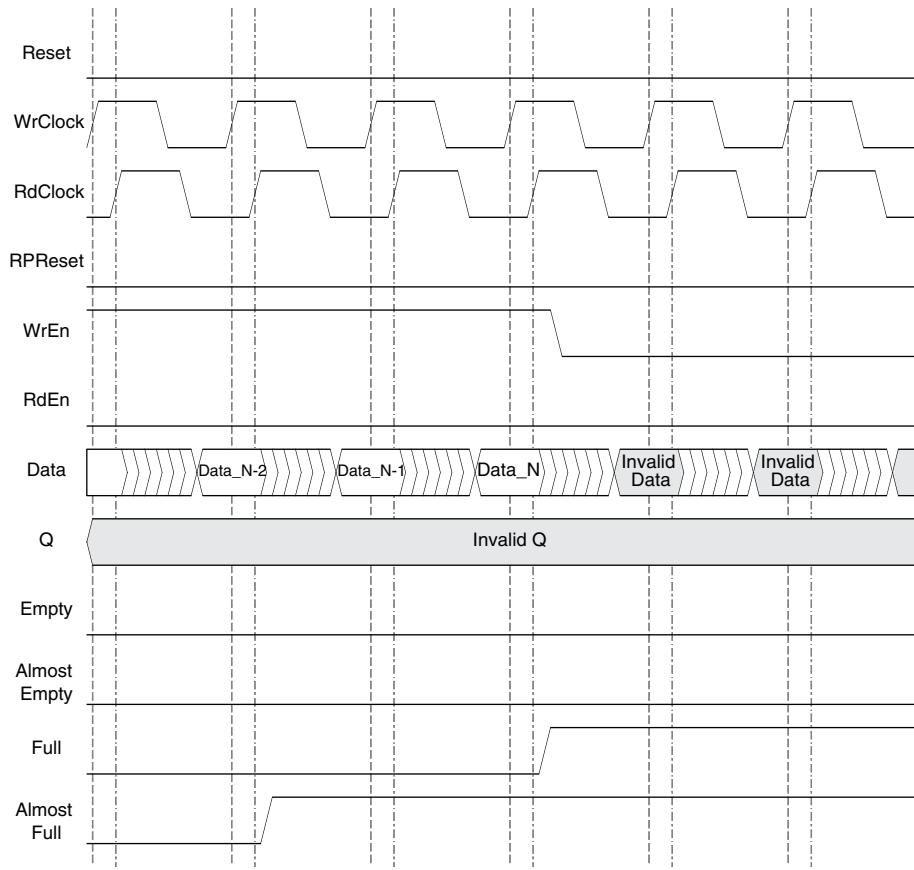
Figure 12-39 to Figure 12-42 show the behavior of the non-pipelined FIFO\_DC or FIFO\_DC without output registers. When we pipeline the registers, the output data is delayed by one clock cycle. There is an extra option for output registers to be enabled by the RdEn signal.

Figure 12-43 to Figure 12-46 show similar waveforms for the FIFO\_DC with output register and without output register enable with RdEn. Note that flags are asserted and de-asserted with timing similar to the FIFO\_DC without output registers. However, only the data out 'Q' is delayed by one clock cycle.

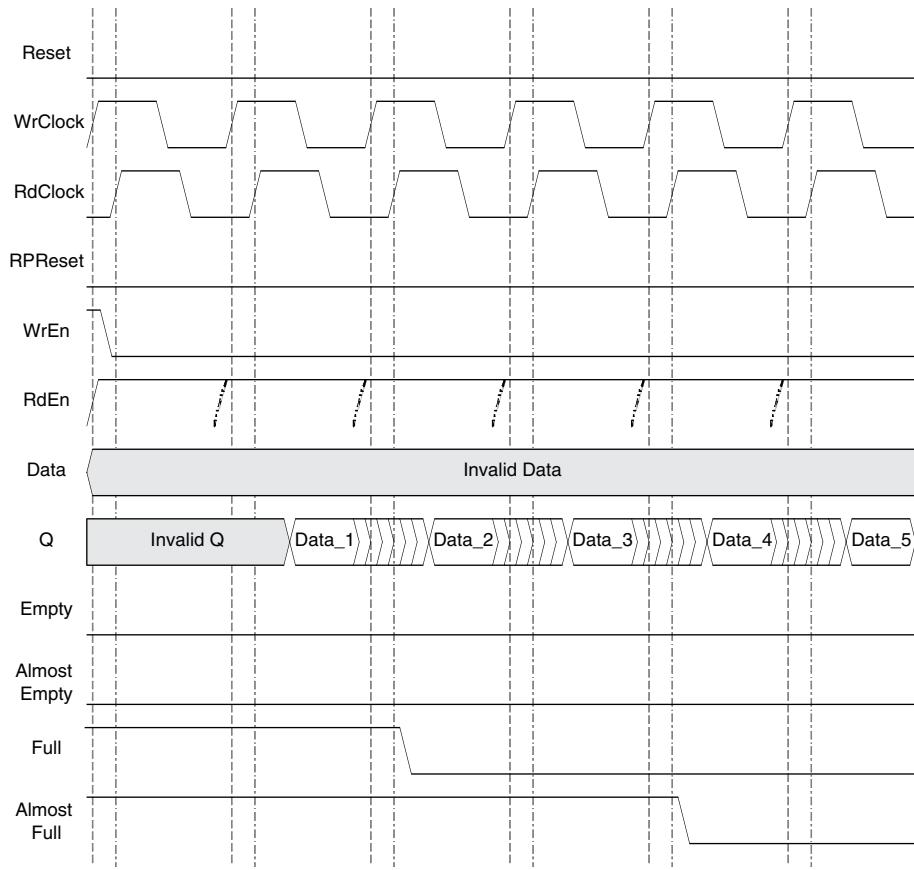
**Figure 12-43. FIFO\_DC with Output Registers, Start of Data Write Cycle**



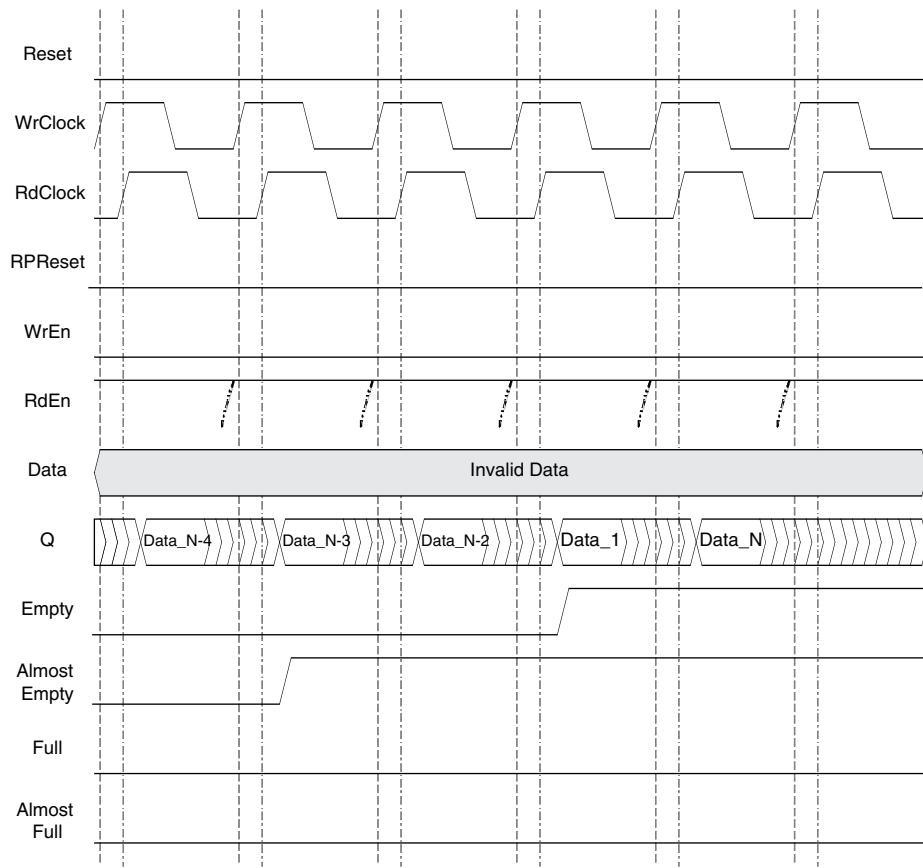
**Figure 12-44. FIFO\_DC with Output Registers, End of Data Write Cycle**



**Figure 12-45. FIFO\_DC with Output Registers, Start of Data Read Cycle**

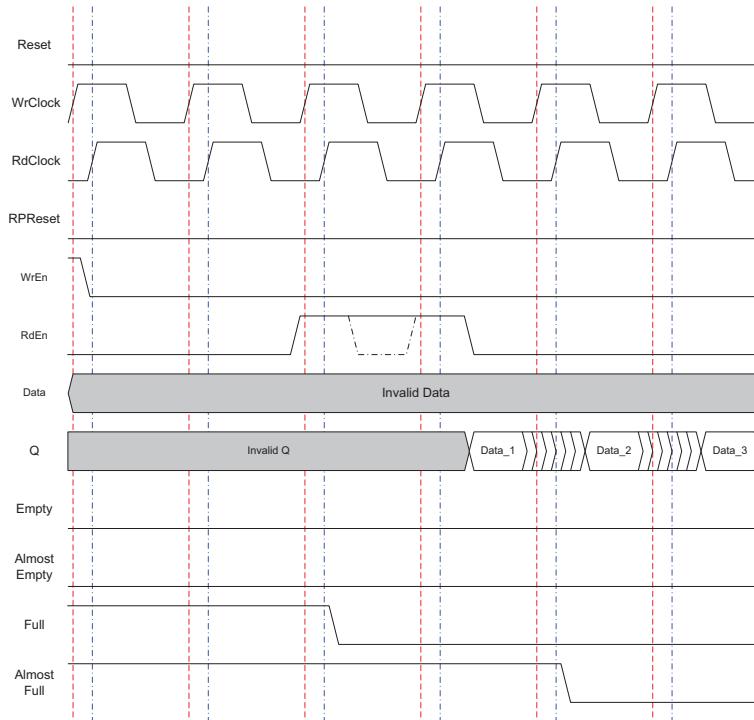


**Figure 12-46. FIFO\_DC with Output Registers, End of Data Read Cycle**



If the designer selects the option enable output register with RdEn, it still delays the data out by one clock cycle (as compared to the non-pipelined FIFO\_DC). RdEn should also be high during that clock cycle, otherwise the data takes an extra clock cycle when the RdEn is goes true.

**Figure 12-47. FIFO\_DC with Output Registers and RdEn on Output Registers**



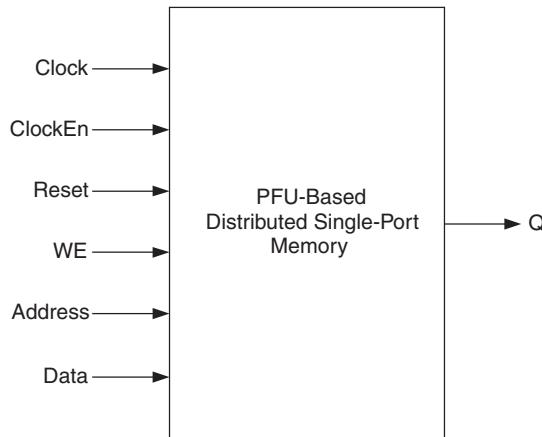
The FIFOs available in ECP5 devices have new features as compared to the previous device families. Some of these options are as follow.

## Distributed Single-Port RAM (Distributed\_SPRAM) – PFU-Based

PFU-based Distributed Single-Port RAM is created using the 4-input LUTs available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 12-48 shows the Distributed Single-Port RAM module as generated by Clarity Designer.

**Figure 12-48. Distributed Single-Port RAM Module Generated by Clarity Designer**



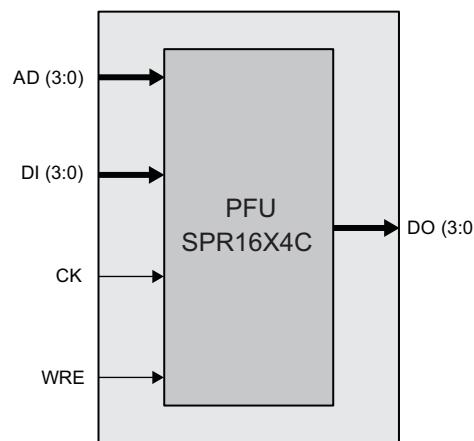
The generated module makes use of the 4-input LUTs available in the PFU. Additional logic such as a clock or reset is generated by utilizing the resources available in the PFU.

Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by Clarity Designer when the user wants to enable the output registers in the Clarity Designer configuration.

Figure 12-49 provides the primitive that can be instantiated for the Single Port Distributed RAM. Primitive name is SPR16X4C and it can be directly instantiated in the code. Check the details on the port and port names under the primitives available under cae\_library/synthesis folder in Lattice Diamond software installation folder.

It is to be noted that each EBR can accommodate 64 bits of memory; if the memory required is larger than 64 bits, then cascading can be used. Further, the ports can be registered by using external PFU registers.

**Figure 12-49. Single Port Distributed RAM Primitive for ECP5 Devices**



The various ports and their definitions listed in Table 12-16. The table lists the corresponding ports for the module generated by Clarity Designer and for the primitive.

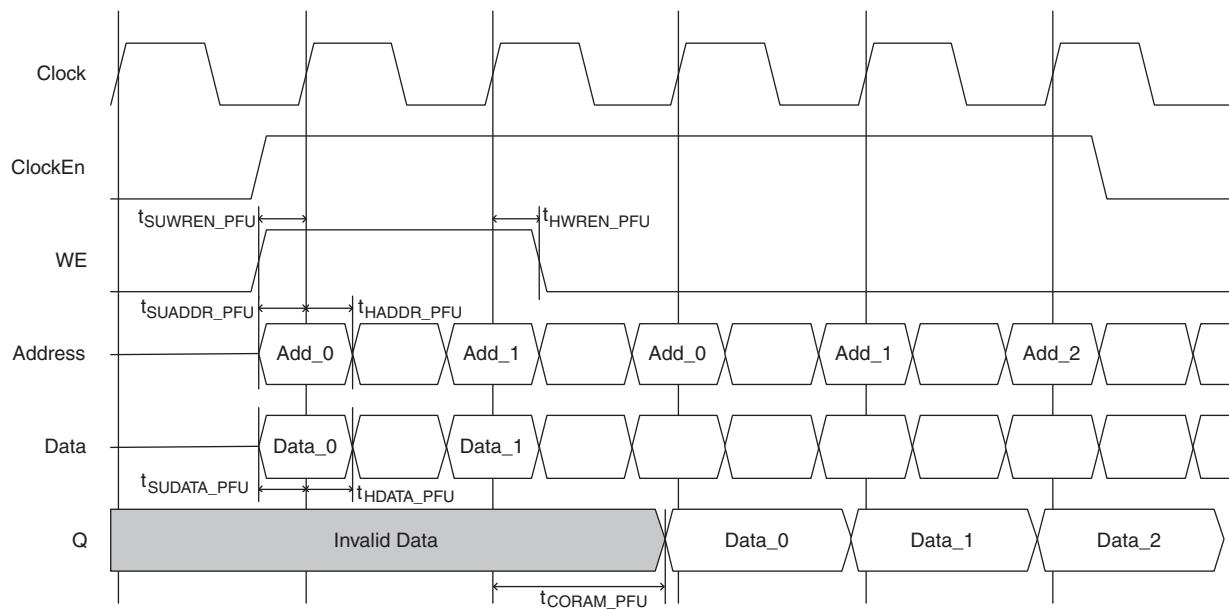
**Table 12-16. PFU-Based Distributed Single Port RAM Port Definitions**

Port Name in Generated Module	Port Name in the EBR block Primitive	Description
Clock	CK	Clock
ClockEn	—	Clock Enable
Reset	—	Reset
WE	WRE	Write Enable
Address	AD[3:0]	Address
Data	DI[1:0]	Data In
Q	DO[1:0]	Data Out

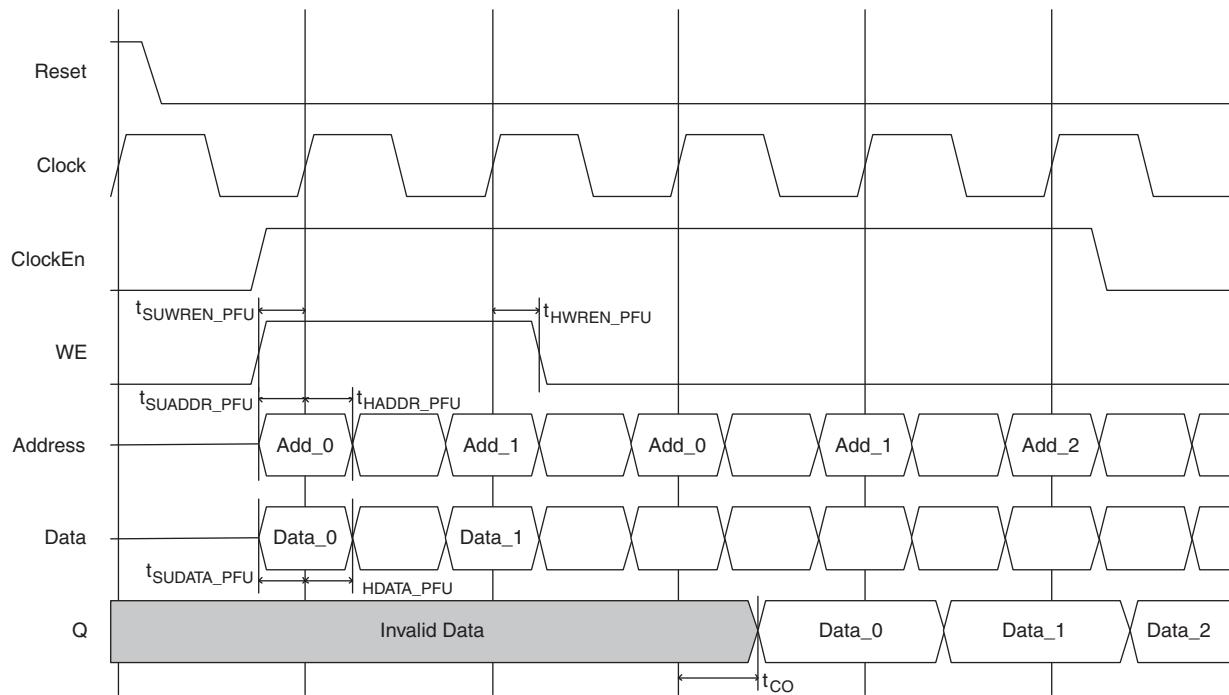
Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn), are not available in the hardware primitive. These are generated by Clarity Designer when the user wants to enable the output registers in their Clarity Designer configuration.

The various ports and their definitions for the memory are included in Table 12-12. The table lists the corresponding ports for the module generated by Clarity Designer and for the primitive.

**Figure 12-50. PFU Based Distributed Single Port RAM Timing Waveform - without Output Registers**



**Figure 12-51. PFU Based Distributed Single Port RAM Timing Waveform - with Output Registers**

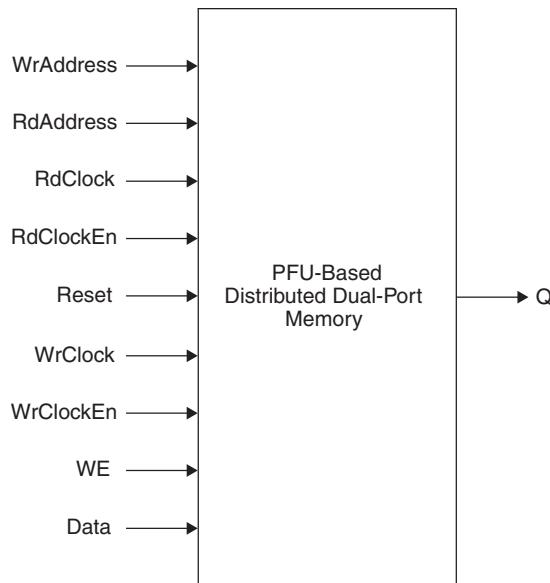


## Distributed Dual-Port RAM (Distributed\_DPRAM) – PFU-Based

PFU-based Distributed Dual-Port RAM is also created using the 4-input LUTs available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 12-52 shows the Distributed Single-Port RAM module as generated by Clarity Designer.

**Figure 12-52. Distributed Dual-Port RAM Module Generated by Clarity Designer**



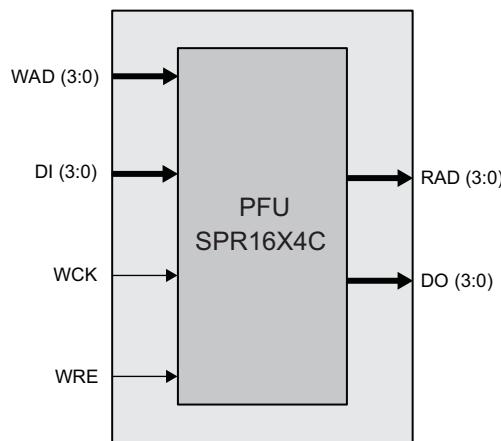
The generated module makes use of the 4-input LUTs available in the PFU. Additional logic such as a clock or reset is generated by utilizing the resources available in the PFU.

Ports such as the Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by Clarity Designer when the user wants the to enable the output registers in the Clarity Designer configuration.

Figure 12-53 provides the primitive that can be instantiated for the Dual Port Distributed RAM. Primitive name is DPR16X4C and it can be directly instantiated in the code. Check the details on the port and port names under the primitives available under cae\_library/synthesis folder in Lattice Diamond software installation folder.

It is to be noted that each EBR can accommodate 64 bits of memory; if the memory required is larger than 64 bits, then cascading can be used. Further, the ports can be registered by using external PFU registers.

**Figure 12-53. Dual Port Distributed RAM Primitive for ECP5 Devices**



The various ports and their definitions are listed in Table 12-17. The table lists the corresponding ports for the module generated by Clarity Designer and for the primitive.

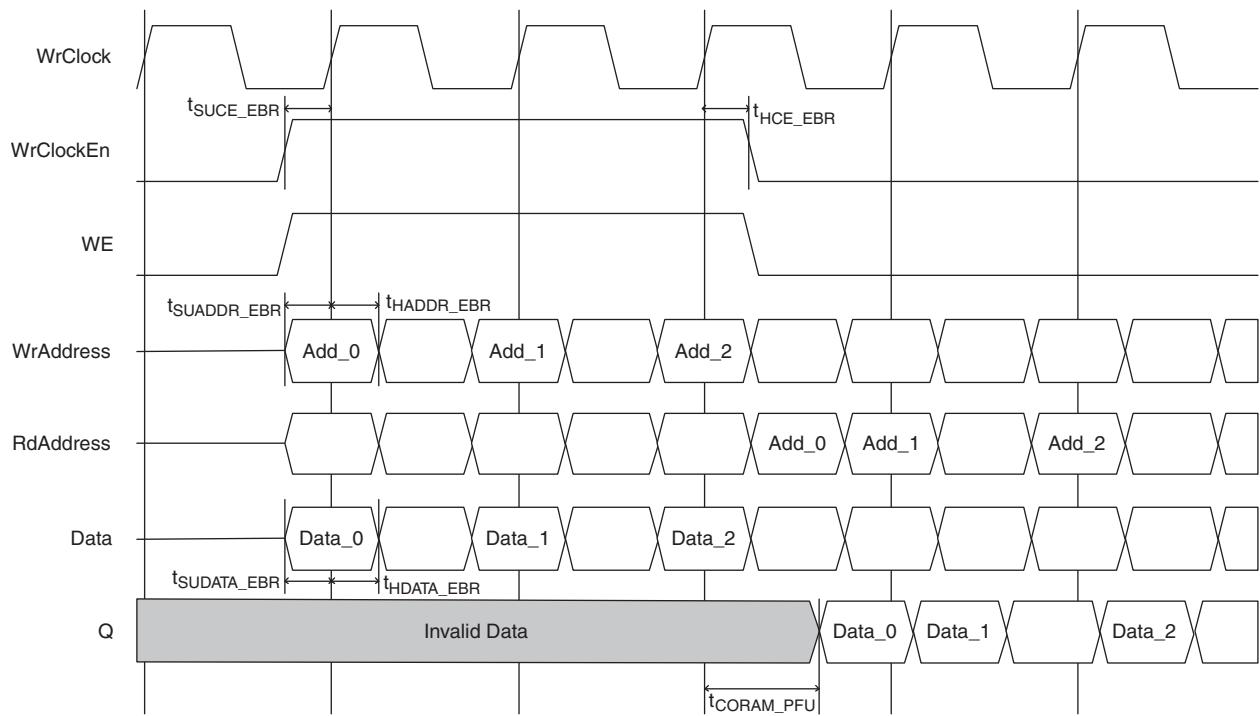
**Table 12-17. PFU-Based Distributed Dual-Port RAM Port Definitions**

Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
WrAddress	WAD[23:0]	Write Address
RdAddress	RAD[3:0]	Read Address
RdClock	—	Read Clock
RdClockEn	—	Read Clock Enable
WrClock	WCK	Write Clock
WrClockEn	—	Write Clock Enable
WE	WRE	Write Enable
Data	DI[1:0]	Data Input
Q	RDO[1:0]	Data Out

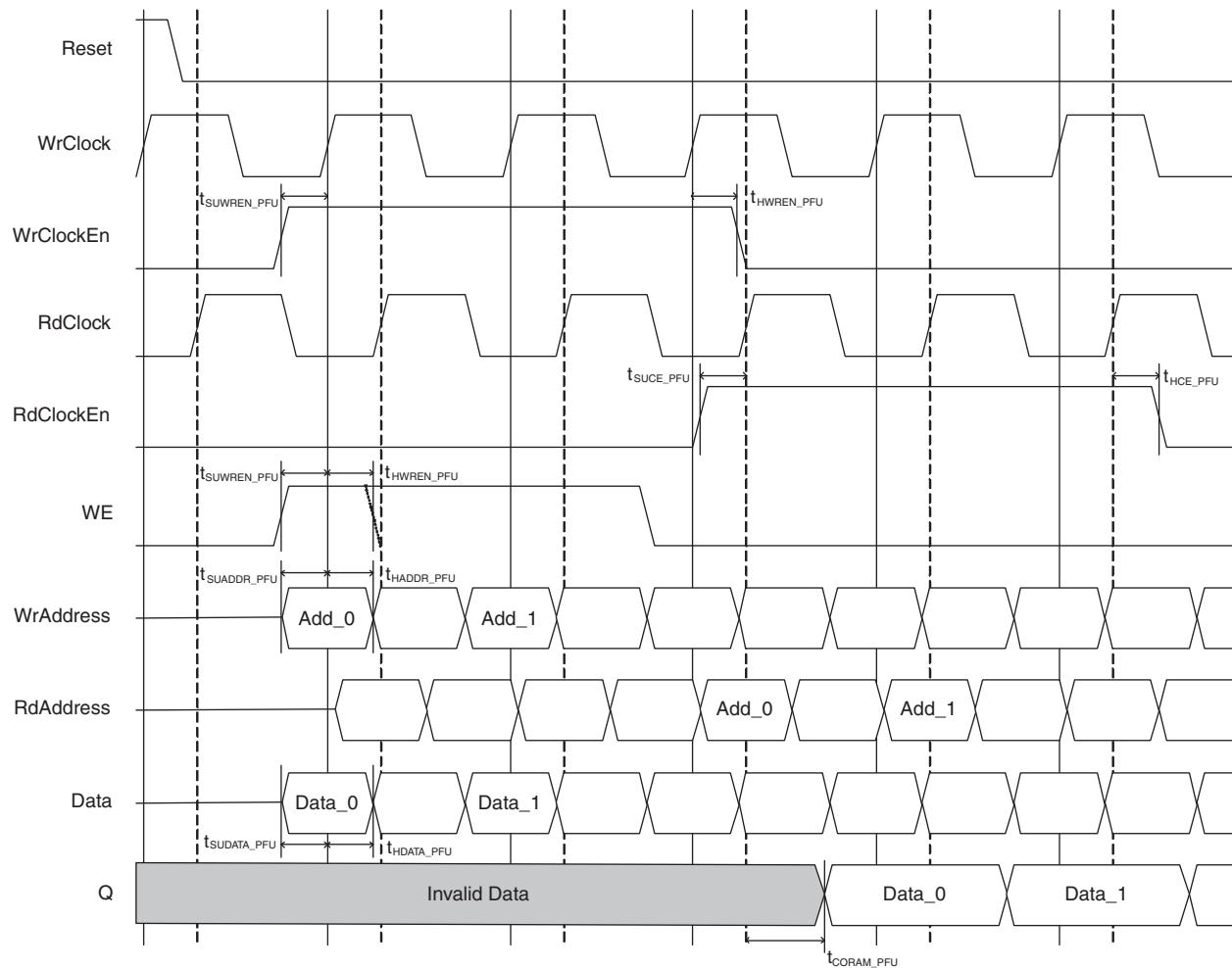
Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by Clarity Designer when the user wants to enable the output registers in the Clarity Designer configuration.

Users have the option of enabling the output registers for Distributed Dual Port RAM (Distributed\_DPRAM). Figure 12-54 and Figure 12-55 show the internal timing waveforms for the Distributed Dual Port RAM (Distributed\_DPRAM) with these options.

**Figure 12-54. PFU Based Distributed Dual Port RAM Timing Waveform - without Output Registers**



**Figure 12-55. PFU Based Distributed Dual Port RAM Timing Waveform - with Output Registers**

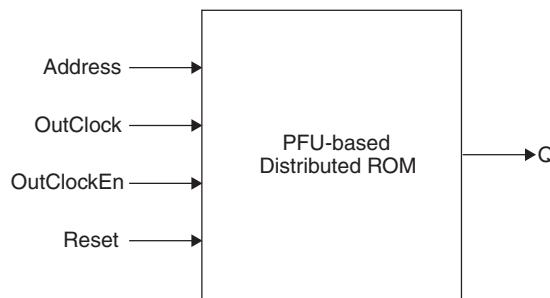


## Distributed ROM (Distributed\_ROM) – PFU-Based

PFU-based Distributed ROM is also created using the 4-input LUTs available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 12-56 shows the Distributed ROM module generated by Clarity Designer.

**Figure 12-56. Distributed ROM Generated by Clarity Designer**

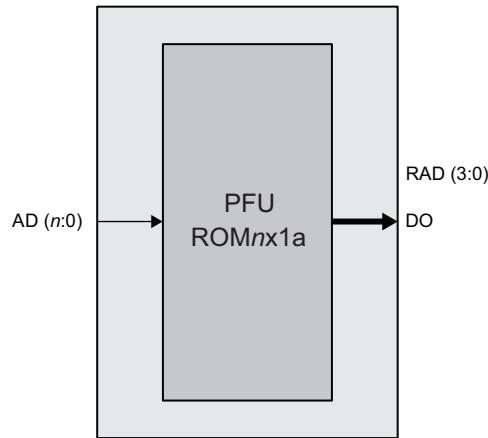


The generated module makes use of the 4-input LUTs available in the PFU. Additional logic such as a clock or reset is generated by utilizing the resources available in the PFU.

Ports such as Out Clock (OutClock) and Out Clock Enable (OutClockEn) are not available in the hardware primitive. These are generated by Clarity Designer when the user wants to enable the output registers in the Clarity Designer configuration.

Figure 12-57 provides the primitive that can be instantiated for the Distributed ROM. Primitive name is DPRnX1a and it can be directly instantiated in the code. 'n' can be 3, 4, 5, 6 or 7 depending upon the size of the ROM. Check the details on the port and port names under the primitives available under cae\_library/synthesis folder in Lattice Diamond software installation folder.

**Figure 12-57. Distributed ROM Primitive for ECP5 Devices**



If the memory required is larger than what can fit in the primitive bits, then cascading can be used. Further, the ports can be registered by using external PFU registers.

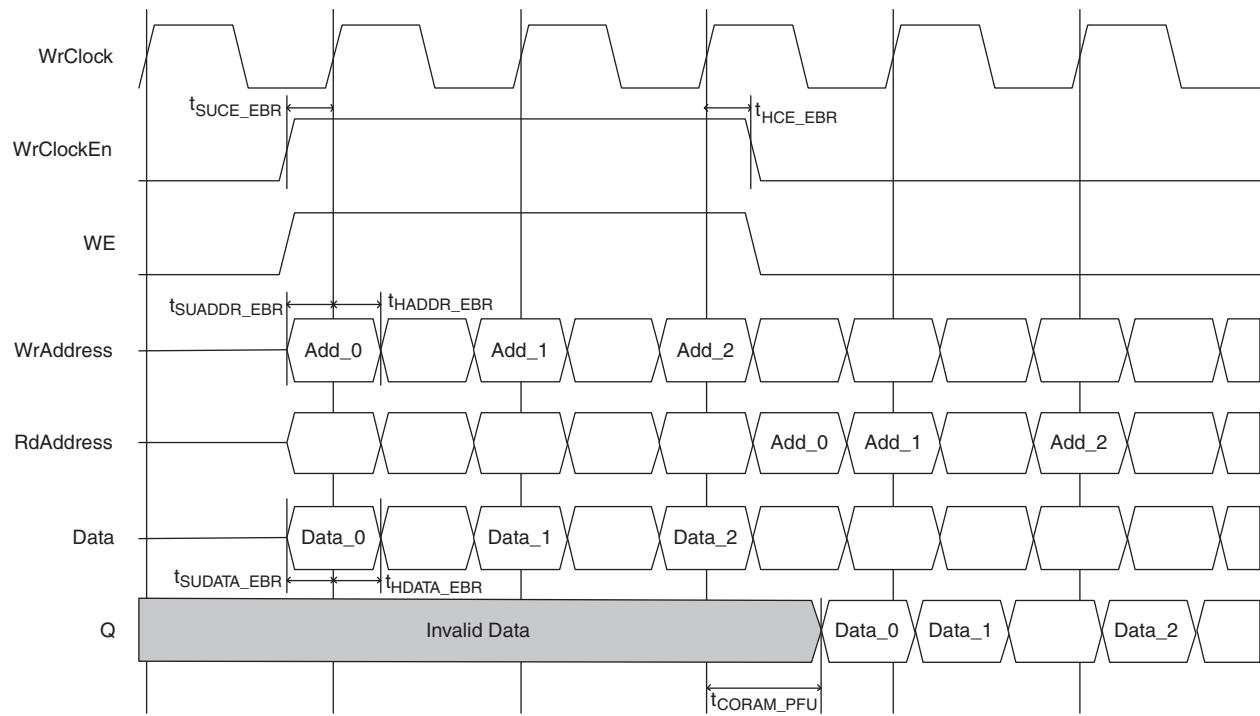
The various ports and their definitions are listed in Table 12-18. The table lists the corresponding ports for the module generated by Clarity Designer and for the primitive.

**Table 12-18. PFU-Based Distributed ROM Port Definitions**

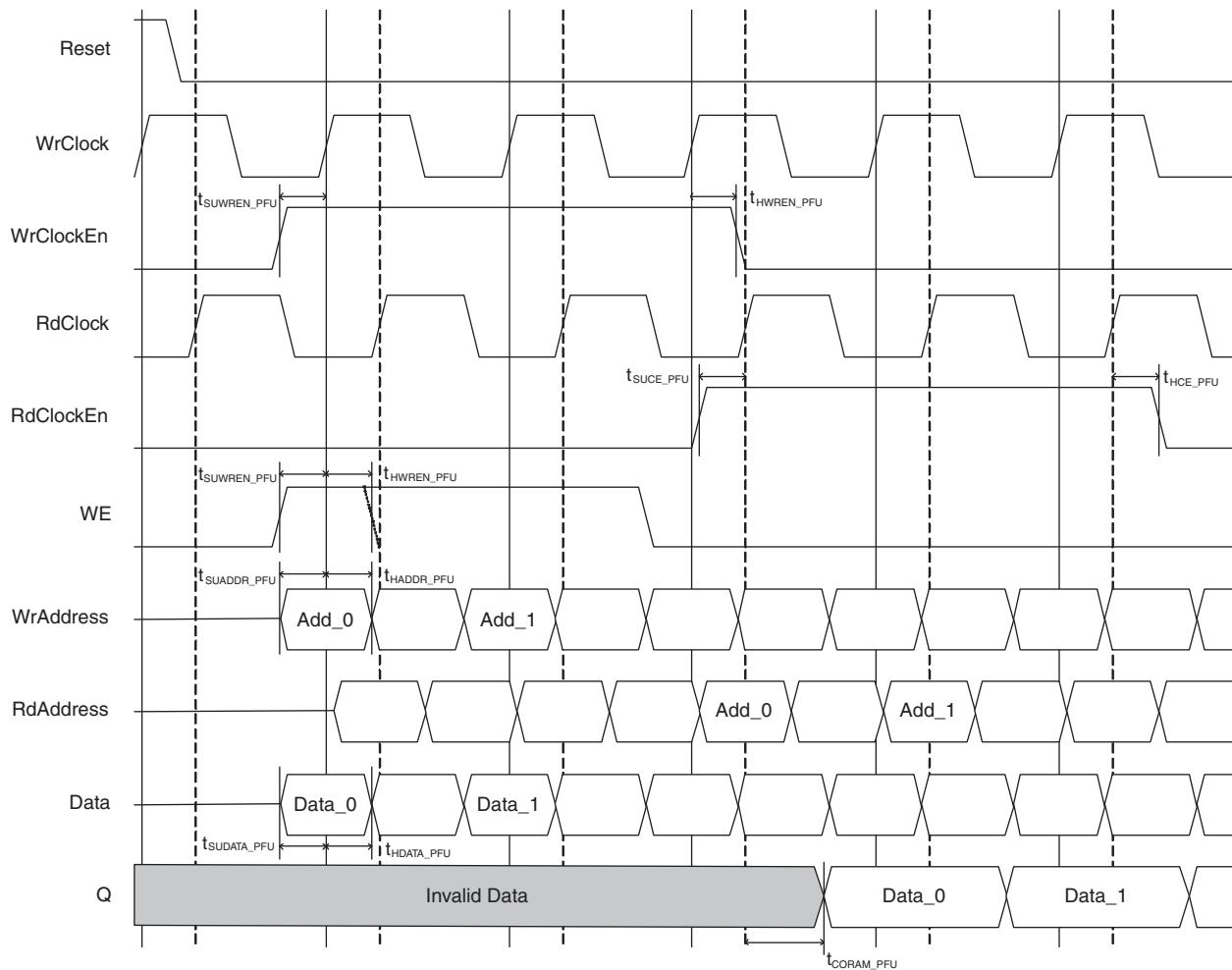
Port Name in the Generated Module	Port Name in the EBR Block Primitive	Description
Address	AD[3:0]	Address
OutClock	—	Out Clock
OutClockEn	—	Out Clock Enable
Reset	—	Reset
Q	DO	Data Out

Users have the option to enable the output registers for Distributed ROM (Distributed\_ROM). Figure 12-58 and Figure 12-59 show the internal timing waveforms for the Distributed ROM with these options.

**Figure 12-58. PFU Based Distributed Dual Port RAM Timing Waveform - without Output Registers**



**Figure 12-59. PFU Based Distributed Dual Port RAM Timing Waveform - with Output Registers**



## Initializing Memory

In each memory mode, it is possible to specify the power-on state of each bit in the memory array. This allows the memory to be used as ROM if desired. Each bit in the memory array can have a value of 0 or 1.

### Initialization File Formats

The initialization file is an ASCII file, which the designer can create or edit using any ASCII editor. Clarity Designer supports three memory file formats:

1. Binary file
2. Hex File
3. Addressed Hex

The file name for the memory initialization file is \*.mem (<file\_name>.mem). Each row includes the value to be stored in a particular memory location. The number of characters (or the number of columns) represents the number of bits for each address (or the width of the memory module).

The memory initialization can be static or dynamic. In case of static initialization, the memory values are stored in the bitstream. Dynamic initialization of memories, involve memory values stored in the external flash and can be

updated by user logic knowing the EBR address locations. The size of the bitstream (bit or rbt file) will be larger due to static values stored in them.

The initialization file is primarily used for configuring the ROMs. RAMs can also use the initialization file to preload memory contents.

### Binary File

The binary file is a text file of 0s and 1s. The rows indicate the number of words and the columns indicate the width of the memory.

Memory Size 20x32

```
00100000010000000010000001000000  
00000001000000010000000100000001  
00000010000000100000001000000010  
00000011000000110000001100000011  
00000100000001000000010000000100  
00000101000001010000010100000101  
00000110000001100000011000000110  
00000111000001110000011100000111  
00001000010010000000100001001000  
00001001010010010000100101001001  
00001010010010100000101001001010  
00001011010010110000101101001011  
00001100000011000000110000001100  
00001101001011010000110100101101  
000011100011110000011100011110  
0000111100111110000111100111111  
00010000000100000001000000010000  
00010001000100010001000100010001  
00010010000100100001001000010010  
00010011000100110001001100010011
```

### Hex File

The hex file is a text file of hexadecimal characters arranged in a similar row-column arrangement. The number of rows in the file is the same as the number of address locations, with each row indicating the content of the memory location.

Memory Size 8x16

```
A001  
0B03  
1004  
CE06  
0007  
040A  
0017  
02A4
```

### Addressed Hex

Addressed hex consists of lines of addresses and data. Each line starts with an address, followed by a colon, and any number of data. The format of the file is “address: data data data data” where the address and data are hexadecimal numbers.

```
A0 : 03 F3 3E 4F  
B2 : 3B 9F
```

In the example above, the first line shows 03 at address A0, F3 at address A1, 3E at address A2, and 4F at address A3. The second line shows 3B at address B2 and 9F at address B3.

There is no limitation on the address and data values. The value range is automatically checked based on the values of `addr_width` and `data_width`. If there is an error in an address or data value, an error message is printed. It is not necessary to specify data at all address locations. If data is not specified at a certain address, the data at that location is initialized to 0. SCUBA makes memory initialization possible both through the synthesis and simulation flows.

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
March 2014	01.0	Initial release.

## Appendix A. Attribute Definitions

### DATA\_WIDTH

Data width is associated with the RAM and FIFO elements. The DATA\_WIDTH attribute defines the number of bits in each word. It uses the values defined in the RAM size tables in each memory module.

### REGMODE

REGMODE, or the Register mode attribute, is used to enable pipelining in the memory. This attribute is associated with the RAM and FIFO elements. The REGMODE attribute takes the NOREG or OUTREG mode parameter that disables and enables the output pipeline registers.

### CSDECODE

CSDECODE or the Chip Select Decode attributes are associated with block RAM elements. CS, or Chip Select, is the port available in the EBR primitive that is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade eight memories easily.

CSDECODE takes the following parameters: "000", "001", "010", "011", "100", "101", "110", and "111". CSDECODE values determine the decoding value of CS[2:0]. CSDECODE\_W is chip select decode for write and CSDECODE\_R is chip select decode for read for Pseudo Dual Port RAM. CSDECODE\_A and CSDECODE\_B are used for True Dual-Port RAM elements and refer to the A and B ports.

### WRITEMODE

The WRITEMODE attribute is associated with the block RAM elements. It takes the NORMAL, WRITETHROUGH, and READBEFOREWRITE mode parameters.

In NORMAL mode, the output data does not change or get updated, during the write operation. This mode is supported for all data widths.

In WRITETHROUGH mode, the output data is updated with the input data during the write cycle. This mode is supported for all data widths.

WRITEMODE\_A and WRITEMODE\_B are used for Dual-Port RAM elements and refer to the A and B ports in True Dual-Port RAM.

In READBEFOREWRITE mode, the output data port is updated with the existing data stored in the write address, during a write cycle. This mode is supported for x9, x18 and x36 data widths.

For all modes of the True Dual-Port module, simultaneous read access from one port and write access from the other port to the same memory address is not recommended. The read data may be unknown in this situation.

Also, simultaneous write access to the same address from both ports is not recommended. When this occurs, the data stored in the address becomes undetermined when one port tries to write a 'H' and the other tries to write a 'L'. It is recommended that control logic be implemented to identify this situation if it occurs and do one of the following:

- Implement status signals to flag the read data as possibly invalid, or
- Implement control logic to prevent simultaneous access from both ports.

## Introduction

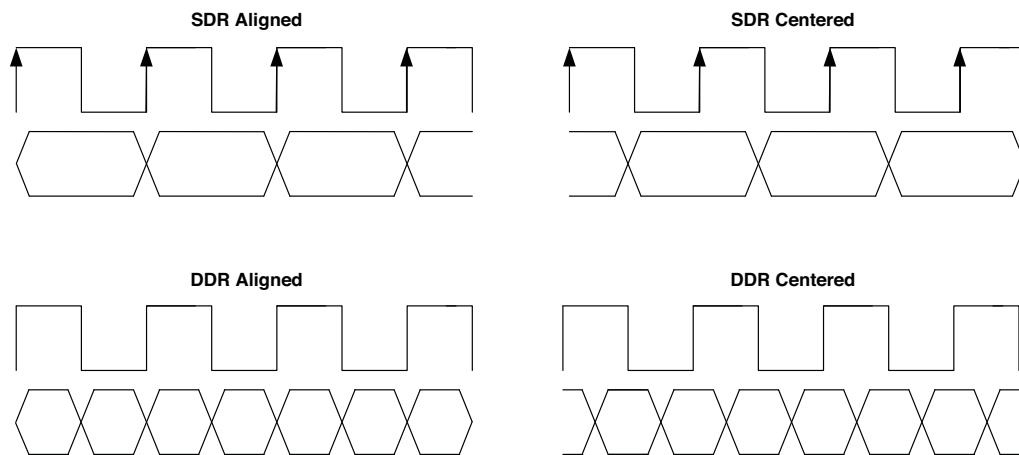
ECP5™ devices support high-speed I/O interfaces, including Double Data Rate (DDR) and Single Data Rate (SDR) interfaces, using the logic built into the Programmable I/O (PIO). SDR applications capture data on one edge of a clock while DDR interfaces capture data on both the rising and falling edges of the clock, thus doubling the performance. ECP5 device I/Os also have dedicated circuitry that is used along with the DDR I/O to support DDR2, DDR3, DDR3L, LPDDR2 and LPDDR3 SDRAM memory interfaces.

This document discusses how to utilize the capabilities of the ECP5 devices to implement high-speed generic DDR interface and the DDR memory interfaces. Refer to the Implementing DDR Memory Interfaces section of this document for more information.

## External Interface Description

This technical note uses two types of external interface definitions, centered and aligned. A centered external interface means that, at the device pins, the clock is centered in the data opening. An aligned external interface means that, at the device pins, the clock and data transition are aligned. This is also sometimes called edge-on-edge. Figure 13-1 shows the external interface waveform for SDR and DDR.

**Figure 13-1. External Interface Definitions**



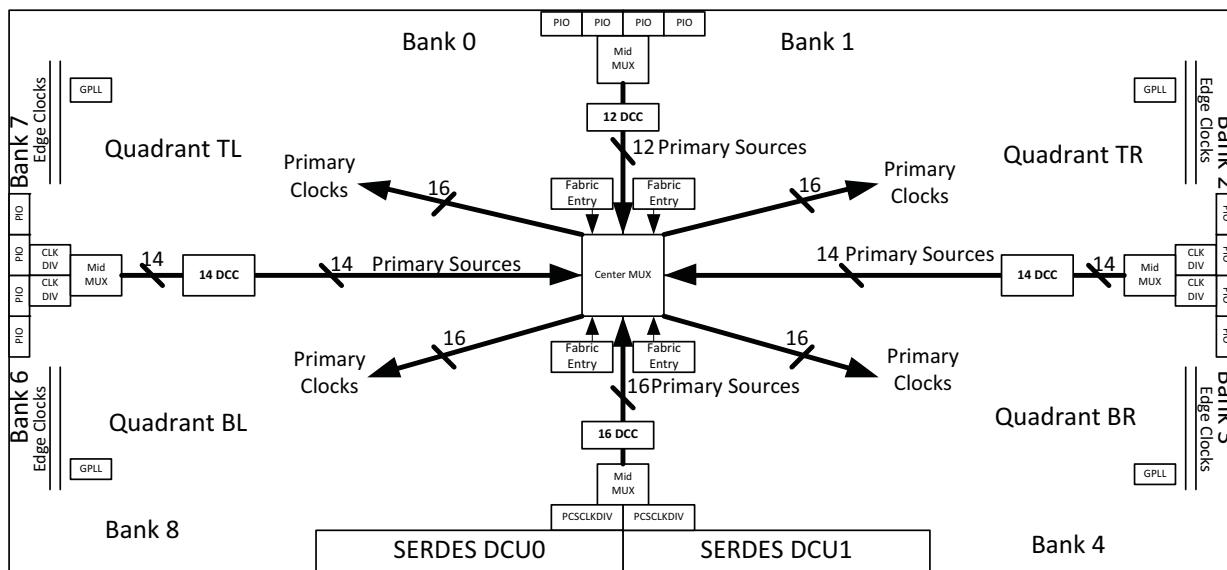
The interfaces described are referenced as centered or aligned interfaces. An aligned interface will need to adjust the clock location to satisfy the capture flip-flop setup and hold times. A centered interface will need to balance the clock and data delay to the first flip-flop to maintain the setup and hold already provided.

## High-Speed I/O Interface Building Blocks

ECP5 devices contain dedicated functions for building high-speed interfaces. This section describes when and how to use these functions. A complete description of the library elements, including descriptions and attributes, is provided at the end of this document.

Figure 13-2 shows a high-level diagram of the clocking resources available in the ECP5 devices for building high-speed I/O interfaces.

Figure 13-2. ECP5 Device Clocking Diagram



A complete description of the ECP5 device family clocking resources and clock routing restrictions are available in TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#).

Below is a brief description of each of the major elements used for building various high-speed interfaces. The [DDR Software Primitives and Attributes](#) section of this document describes the library elements for these components.

## Edge Clocks

Edge clocks (ECLK) are high-speed, low-skew I/O dedicated clocks. They are arranged in groups of two per I/O bank on the left and right sides of the device. Each of these edge clocks can be used to implement a high speed interface. There is an Edge Clock Bridge (ECLKBRIDGECS) that will allow users to build large interfaces by bridging the edge clocks from one bank to the other on the same side or from one side to the other side.

## Primary Clocks

Primary clocks (PCLK) refer to the system clock of the design. The SCLK ports of the DDR primitives are connected to the system clock of the design.

## DQS Lane

A DQS Lane uses the embedded circuit for memory interfaces. Each DQS Lane provides a clock pair (DQSP and DQSN) for the DQS strobe and up to 12 to 16 ports for DQ data and DM data mask signals. The number of DQS Lanes on the device is different for each device size. ECP5 devices support DQS lanes on the left and right sides of the device.

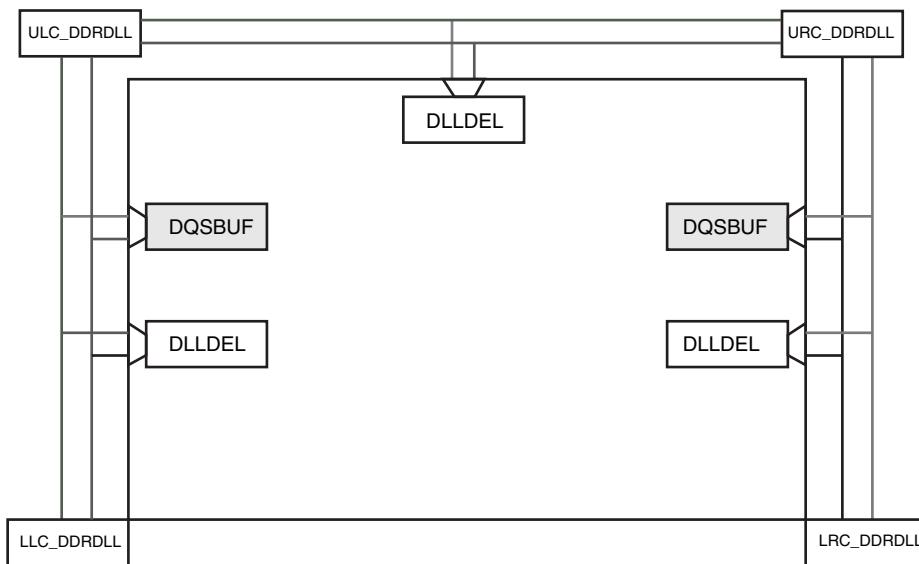
## PLL

The PLL provides frequency synthesis, with additional static and dynamic phase adjustment, as well. Four output ports are provided, CLKOP, CLKOS, CLKOS2 and CLKOS3. All four outputs have the same set of dividers. There is one PLL per corner on the biggest device, totaling to four PLLs on each device.

## DDRDLL

The DDRDLL is a dedicated DLL for creating the 90° clock delay. The DDRDLL outputs delay codes that are used in the DQSBUF elements to delay the DQS input or in the DLLDEL module to delay the input clock. There is one DDRDLL at each corner of the device, totaling to four DDRDLLs on each device. The DDRDLL on the top corners of the device can drive delay codes to two adjacent edges of the device, providing a possible two DDRDLL codes for an edge.

**Figure 13-3. DDRDLL Connectivity**



## DQSBUF

There is one DQSBUF for each DQS lane (every 12 to 16 I/Os depending on the selected device). The DQS input is used when interfacing to DDR memories. It generates the delay on the DQS pin of the DQS lane, to provide a 90° phase shift on DQS to clock the DDR data at the center. The delay is set by a delay code generated in the DDRDLL component. Each DQSBUF can receive delay codes from two different DDRDLs hence two different DDR memory interfaces can be built on one side of the device.

Each of the DQSBUF modes has an additional feature that allows the user to adjust the delay from the delay set by the DDRDLL code, by using the MOVE and DIRECTION inputs controlled by the user logic. The LOADN will reset the delay back to the DDRDLL code.

## DLLDEL

DLLDEL provides phase shift on the receive side clocks to each ECLK. It functions similar to the DQSBUF, shifting the clock input by delay set by the DDRDLL delay code, before the clock drives the clock tree. The DLLDEL element has the ability to further adjust the delay from the delay set by the DDRDLL code, by using the MOVE and DIRECTION inputs controlled by the user logic. The LOADN will reset the delay back to the DDRDLL code.

## Input DDR (IDDR)

The input DDR function can be used in either 1X (2:1), 2X (4:1) or 7:1 gearing modes. In the 1X mode, the IDDR module inputs a single DDR data input and SCLK (primary clock) and provides a 2-bit wide data synchronized to the SCLK (primary clock) to the FPGA fabric.

The 2X gearing is used for interfaces with data rate higher than 400Mbps which would require higher than 200 MHz system clock. There the IDDR element inputs a single DDR data input and DQS clock (for DDR memory interface) or Edge Clock ECLK (for all other high speed interfaces) and provides a 4-bit wide parallel data synchronized to SCLK (primary clock) to the FPGA fabric.

In the 7:1 mode, mostly used in video applications required 7:1 interface, the IDDR element will input a single DDR data input and ECLK and output a 7 bit wide parallel data synchronized to SCLK (primary clock) to the FPGA fabric.

### Output DDR (ODDR)

The output DDR function can also be supported in 1X (2:1), 2X (4:1) or 7:1 gearing modes. In the 1X mode, the ODDR element receives 2-bit wide data from the FPGA fabric and generates a single DDR data output and Clock output.

Similar to input interfaces the 2X gearing is used for data rate higher than 400Mbps which would require higher than 200 MHz system clock. Here the ODDR element receives 4-bit wide data from the FPGA fabric and generates a single DDR data output and Clock output. The 2X element will use high speed edge clock (ECLK) to clock the data out for generic high speed interfaces and DQS clock for DDR memory interfaces.

In 7:1 mode, the ODDR element receives 7-bit wide data from FPGA fabric and generates a single DDR data output and Clock output. The 7:1 element will send out data using high speed edge clock.

### Edge Clock Dividers (CLKDIV)

Clock dividers are provided to create the divided down clocks used with the I/O Mux/DeMux gearing logic (SCLK inputs to the DDR) and drives to the Primary Clock routing to the fabric. There are two clock dividers on each side of the device.

### Input/Output DELAY

There are two different types of input/output data delay available. Both DELAYF and DELAYG provide a fixed value of delay to compensate for clock injection delay. The DELAYF element also allows the delay value to be set by the user using 128 steps of delay. Each delay step generates ~25ps of delay. In DELAYF, user can overwrite the DELAY setting dynamically using the MOVE and DIRECTION control inputs. The LOADN will reset the delay back to the default value.

## Building Generic High Speed Interfaces

This section describes in detail on how the high speed interfaces that can be built using the building blocks described in the section above. The Clarity Designer tool in Lattice Diamond design software will build these interfaces based on external interface requirements.

### Types of High-Speed DDR Interfaces

This section describes the different types of high-speed DDR interfaces available in ECP5 devices. Table 13-1 lists these interfaces. Descriptions for each interface in Table 13-1 are provided below the table,

**Table 13-1. Generic High-Speed I/O DDR Interfaces.**

Mode	Interface Name	Description
Receive SDR	GIREG_RX.SCLK	SDR Input register using SCLK
Receive DDRX1 Aligned	GDDRX1_RX.SCLK.Aligned	DDR 1x Input using SCLK. Data is edge-to-edge with incoming clock. DLLDEL will be used to shift the incoming clock
Receive DDRX1 Centered	GDDRX1_RX.SCLK.Centered	DDR x1 Input using SCLK. Clock is already centered in data window.
Receive DDRX2 Aligned	GDDRX2_RX.ECLK.Aligned	DDR x2 Input using ECLK. Data is edge-to-edge with incoming clock. Generic DDR X2 using Edge Clock. DLLDEL will be used to shift the incoming clock
Receive DDRX2 Centered	GDDRX2_RX.ECLK.Centered	DDR x2 Input using ECLK. Clock is already centered in data window.
Receive DDRX2 MIPI	GDDRX2_RX.MIPI	DDR2 Input using ECLK interfaces to MIPI interface. This will use additional IMIPI module for the interface.
Receive DDRX71	GDDRX71_RX.ECLK	DDR 7:1 input using ECLK.
Transmit SDR	GOREG_TX.SCLK	SDR Output using SCLK. Clock is forwarded through ODDR.
TX DDRX1 Aligned	GDDRX1_TX.SCLK.Aligned	DDR x1 Output using SCLK. Data is edge-on-edge using same clock through ODDR.
TX DDRX1 Centered	GDDRX1_TX.SCLK.Centered	DDR x1 Output using SCLK. Clock is centered using PLL with different SCLK.
TX DDRX2 Aligned	GDDRX2_TX.ECLK.Aligned	DDR x2 Output that is edge-on-edge using ECLK.
TX DDRX2 Centered	GDDRX2_TX.ECLK.Centered	DDR x2 Output that is pre-centered PLL generated 90° phase, and output on ECLKs.
TX DDRX71	GDDRX71_TX.ECLK	DDR 7:1 output using ECLK. Data and CLK are aligned on first of 7 bits.

Note: The following describes the naming conventions used for each of the interfaces:

- **G – Generic**
- **IREG – SDR Input I/O Register**
- **OREG – SDR Output I/O Register**
- **DDRX1 – DDR 1x gearing I/O Register**
- **DDRX2 – DDR 2x gearing I/O Registers**
- **\_RX – Receive Interface**
- **\_TX – Transmit Interface**
- **.ECLK – Uses ECLK (edge clock) clocking resource**
- **.SCLK – Uses SCLK (primary clock) clocking resource**
- **.Centered – Clock is centered to the data when coming into the device**

## High-Speed DDR Interface Details

This section describes each of the generic high-speed interfaces in detail, including the clocking to be used for each interface. For detailed information about the ECP5 device clocking structure, refer to TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#). The various interface rules listed under each interface should be followed to build these interfaces successfully. Refer to the [Timing Analysis for High Speed DDR Interfaces](#) section of this document for more information about the timing analysis on these interfaces.

Some of these interfaces may require a soft IP in order utilize all the features available in the hardware. These soft IP cores are available in Clarity Designer and are described in this section. Some of the soft IPs are optional and can be selected in the Clarity Designer. Some of these are mandatory for the module to function as expected and will automatically be generated when building the interface through Clarity Designer.

### GDDRX1\_RX.SCLK.Centered

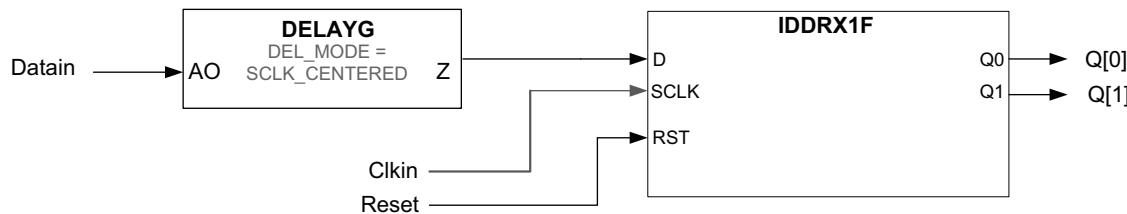
This a Generic 1X gearing Receive interface using SCLK. The clock is coming in centered to the Data. This interface must be used for speeds below 200 MHz.

This DDR interface uses the following modules:

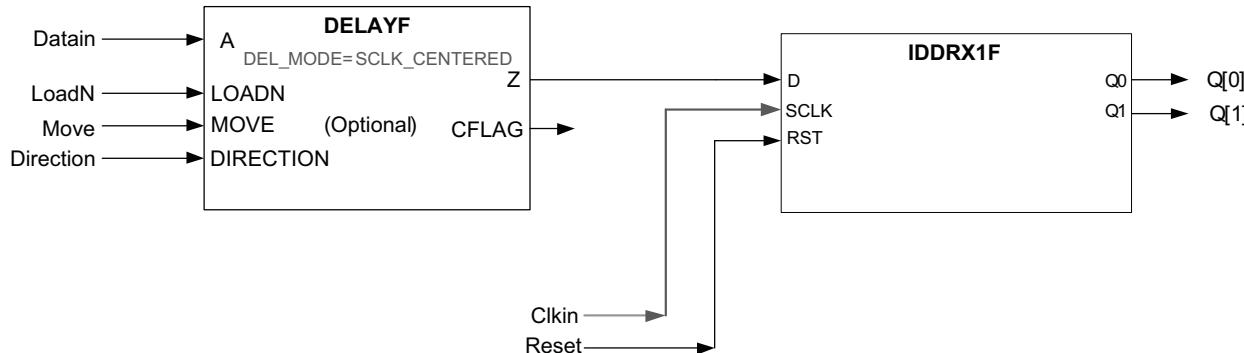
- IDDRX1F element to capture the data
- The incoming clock is routed through the Primary (SCLK) clock tree
- Static data delay element DELAYG is used to delay the incoming data enough to remove the clock injection time.
- Optionally the user can choose to use Dynamic Data delay adjustment using DELAYF element to control the delay on the DATA dynamically. DELAYF will also allow user to override the input delay set. The type of delay required can be selected through Clarity Designer.
- DEL\_MODE attribute is used with DELAYG and DELAYF element to indicate the interface type so that the correct delay value can be set in the delay element.

The following figures show the static delay and dynamic delay options for this interface.

**Figure 13-4. GDDRX1\_RX.SCLK.Centered Interface (Static Delay)**



**Figure 13-5. GDDRX1\_RX.SCLK.Centered Interface (Dynamic Data delay)**



## Interface Requirements

- The clock input must use a PCLK input so that it can be routed directly to the primary clock tree.
- The user must set the timing preferences as per section “Timing Analysis Requirement”

## GDDRX1\_RX.SCLK.Aligned

This a Generic 1X gearing Receive interface using SCLK. The clock is coming in edge aligned to the Data. This interface must be used for speeds below 250 MHz.

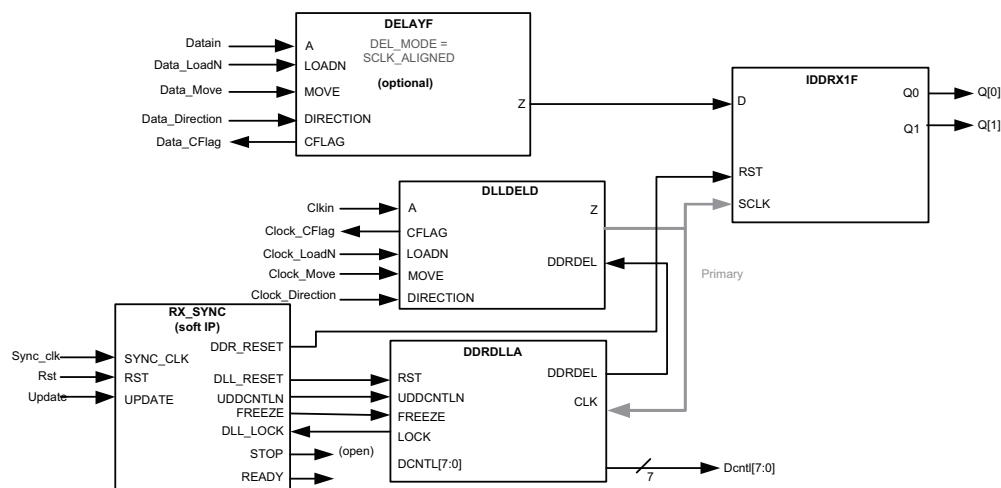
This DDR interface uses the following modules:

- IDDRX1F element to capture the data
- DDRDLLA/DLLDELD blocks are used to phase shift the incoming clock going to primary clock tree (SCLK)
- Static data delay element DELAYF is used to delay the incoming data enough to remove the clock injection time.
- Optionally the user can choose to use Dynamic Data delay adjustment using DELAYF element to control the delay on the DATA dynamically. DELAYF will also allow user to override the input delay set. The type of delay required can be selected through Clarity Designer
- DEL\_MODE attribute is used with DELAYG and DELAYF element to indicate the interface type so that the correct delay value can be set in the delay element.
- Dynamic Margin adjustment in the DDRDLLA module can be optionally used to adjust the DDRDLLA delay dynamically

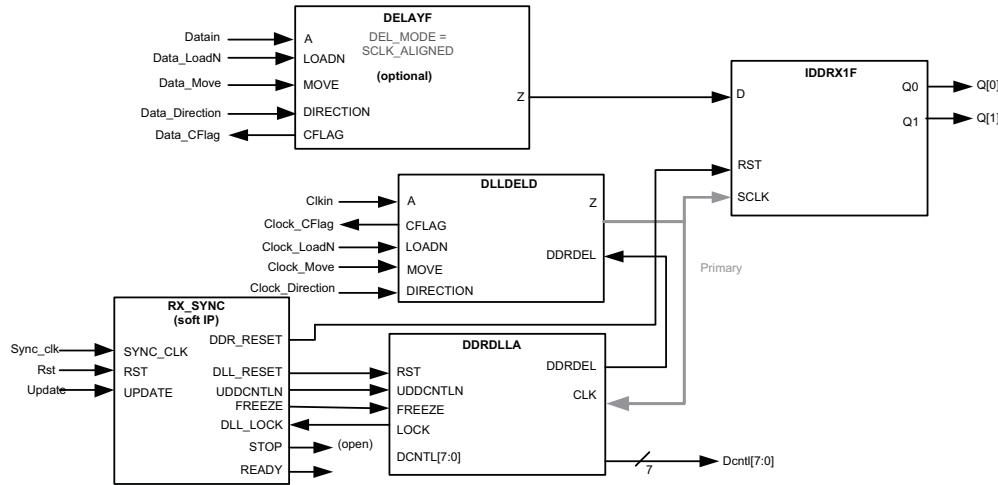
The output of the DLLDELD module is also used as the clock input to the DDRDLLA which sends the delay values to the DLLDELD module. The Receiver Synchronization (RX\_SYNC) soft IP is required for the aligned interfaces to prevent stability issues that may occur due to this loop at startup. The soft IP will prevent any updates to the DLLDELD at start until the DDRDLLA is locked. Once locked the DLLDELD is updated and FREEZE on the DDRDLL is removed. This soft IP will be automatically generated by Clarity Designer.

The following figures show the static delay and dynamic delay options for this interface.

**Figure 13-6. GDDRX1\_RX.SCLK.Aligned Interface (Static Delay)**



**Figure 13-7. GDDRX1\_RX.SCLK Aligned Interface (Dynamic Data/Clock Delay)**



### Interface Requirements

- The clock input must use a PCLK input so that it can be routed directly to the DLLDELD input
- The user must set the timing preferences as per section “Timing Analysis Requirement”

### GDDRX2\_RX.ECLK.Centered

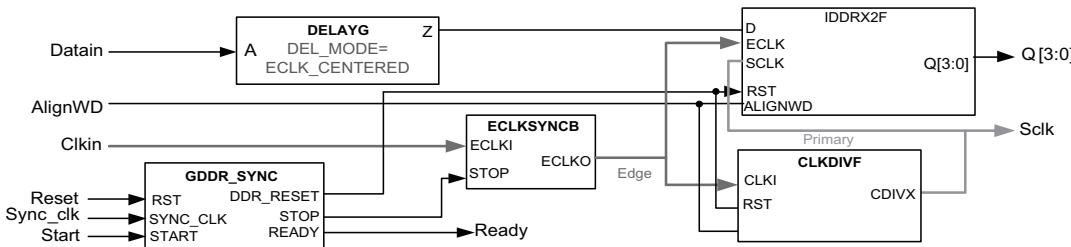
Generic Receive DDR with the 2X gearing using Edge Clock Tree (ECLK). Input clock is centered to the input Data. This interface must be used for speeds above 200 MHz.

This DDR interface uses the following modules:

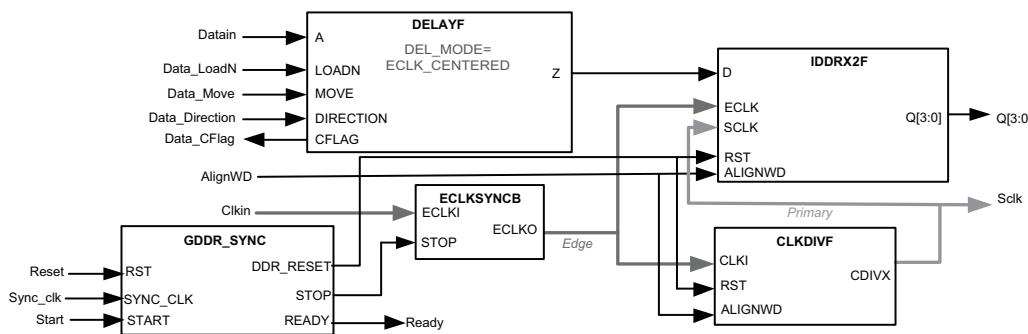
- IDDRX2F element for X2 mode to capture the data
- The incoming clock is routed to the Edge clock (ECLK) clock tree through the ECLKSYNCB module
- CLKDIVF module is used to divide the incoming clock by 2 to generate the SCLK
- Static data delay element DELAYG to delay the incoming data enough to remove the clock injection time
- Optionally the user can choose to use Dynamic Data delay adjustment using DELAYF element to control the delay on the DATA dynamically. DELAYF will also allow user to override the input delay set. The type of delay required can be selected through Clarity Designer
- DEL\_MODE attribute is used with DELAYG and DELAYF element to indicate the interface type so that the correct delay value can be set in the delay element
- The ECLKBRIDGE can be optionally enabled if the data bus will be crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through Clarity Designer.
- The startup synchronization soft IP (GDDRX\_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.

The following figures show the static delay and dynamic delay options for this interface.

**Figure 13-8. GDDRX2\_RX.ECLK.Centered Interface (Static Delay)**



**Figure 13-9. GDDRX2\_RX.ECLK.Centered Interface (Dynamic Data delay)**



### Interface Requirements

- The clock input must use a PCLK input so that it can be routed directly to the edge clock tree.
- ECLK must use the Edge clock tree and the SCLK out of the CLKDIVF must use the Primary clock tree, software will error out if these dedicated clock routes are not used
- “USE PRIMARY” preference may be assigned to the SCLK net
- The user must set the timing preferences as per section “Timing Analysis Requirement”

### GDDRX2\_RX.ECLK.Aligned

Generic Receive DDR with the 2X gearing with ECLK. Input Clock is coming in edge aligned to the Data. This interface must be used for speeds above 20 MHz.

This DDR interface uses the following modules:

- IDDRX2F element for 2X mode to capture the data
- DDRDLLA/DLLDELD blocks are used to phase shift the incoming clock routed to the Edge clock (ECLK) clock tree through the ECLKSYNCB module
- CLKDIVF module is used to divide the incoming clock by 2
- Static data delay element DELAYG to delay the incoming data enough to remove the clock injection time
- Optionally the user can choose to use Dynamic Data delay adjustment using DELAYF element to control the delay on the DATA dynamically. DELAYF will also allow user to override the input delay set. The type of delay required can be selected through Clarity Designer
- DEL\_MODE attribute is used with DELAYG and DELAYF element to indicate the interface type so that the correct delay value can be set in the delay element
- The ECLKBRIDGE can be optionally enabled if the data bus will be crossing over between the left and right sides

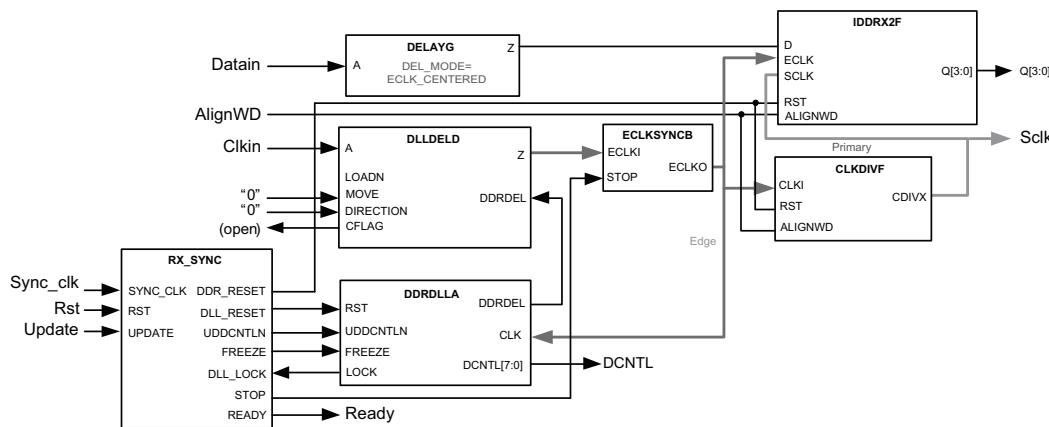
of the device. If ECLKBRIDGE is enabled then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through Clarity Designer.

- Dynamic Margin adjustment in the DDRDLLA module can be optionally used to adjust the DDRDLLA delay dynamically

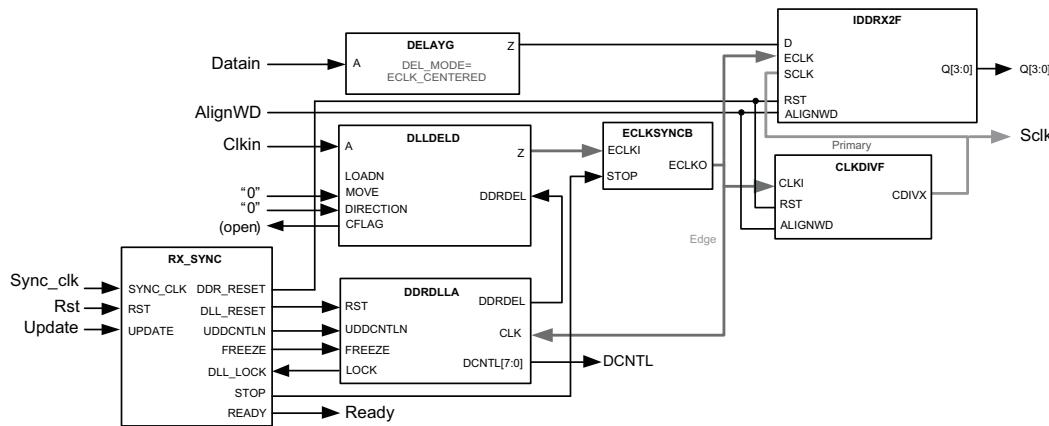
The output of the DLLDELD module is also used as the clock input to the DDRDLLA which sends the delay values to the DLLDELD module. The Receiver Synchronization (RX\_SYNC) soft IP is required for the aligned interfaces to prevent stability issues that may occur due to this loop at startup. The soft IP will prevent any updates to the DLLDELD at start until the DDRDLLA is locked. Once locked the DLLDELD is updated and FREEZE on the DDRDLL is removed. This soft IP will be automatically generated by Clarity Designer.

The following figures show the static delay and dynamic delay options for this interface.

**Figure 13-10. GDDR2\_RX.ECLK.Aligned Interface (Static Delay)**



**Figure 13-11. GDDR2\_RX.ECLK.Aligned Interface (Dynamic Data/Clock Delay)**



### Interface Requirements

- The clock input must use a dedicated PCLK input so that it can be routed directly to the DLLDELD module
- ECLK must use the Edge clock tree and the SCLK out of the CLKDIVD must use the Primary clock tree, software will error out if these dedicated clock routes are not used
- “USE PRIMARY” preference may be assigned to the SCLK net

- The user must set the timing preferences as per section “Timing Analysis Requirement”

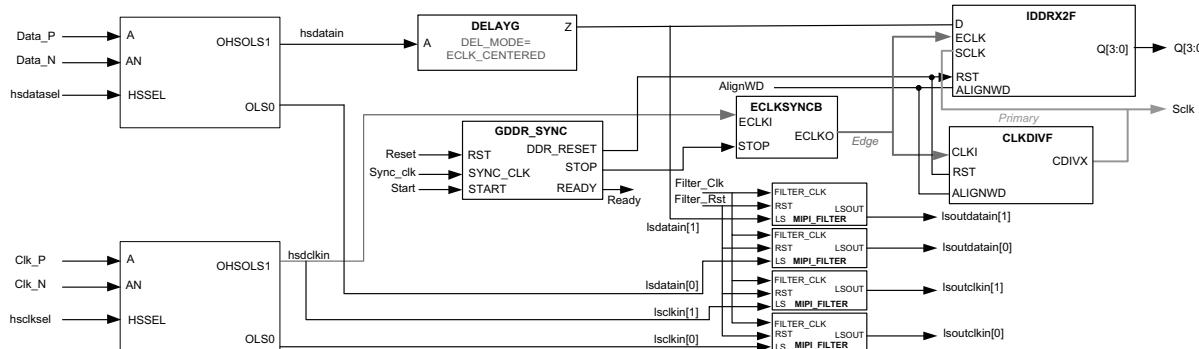
### **GDDRX2\_RX.MIPI**

Generic Receive DDR for MIPI interfaces using the X2 gearing with ECLK. Clock is coming in centered to the Data. This interface must be used for speeds above 200 MHz.

This DDR interface uses the following modules:

- IMIPI element use to receive the MIPI data and clock
- The HSSEL of the IMIPI is used to switch between the High speed and Low Speed modes
- The HSSEL of IMIPI should be driven by a soft IP
- When in high speed mode
  - The OHSOLS1 of the element is active
  - The OHSOLS1 of the data IMIPI element is connected to the Data input GDDRX2\_RX.ECLK.Centered Interface
  - The OHSOLS1 of the clock IMIPI is connected to the ECLK input GDDRX2\_RX.ECLK.Centered Interface
  - This is then treated similar to the GDDRX2\_RX.ECLK.Centered Interface
- When in low speed mode:
  - Both the outputs of IMIPI are active since it is not a 2-bit interface
  - The OHSLS1 is the bit 1 and OLS0 is the bit 0 of the interface.
  - Each of the data input and clock input is connected through a 20ns filter soft IP to the core
- The ECLKBRIDGE can be optionally enabled if the data bus will be crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through Clarity Designer.

**Figure 13-12. GDDRX2\_RX.MIPI**



### **Interface Requirements**

- The clock input must use a PCLK input so that it can be routed directly to the edge clock tree.
- ECLK must use the Edge clock tree and the SCLK out of the CLKDIVF must use the Primary clock tree, software will error out if these dedicated clock routes are not used
- “USE PRIMARY” preference may be assigned to the SCLK net
- The user must set the timing preferences as per section “Timing Analysis Requirement”

### **GDDRX71\_RX.ECLK**

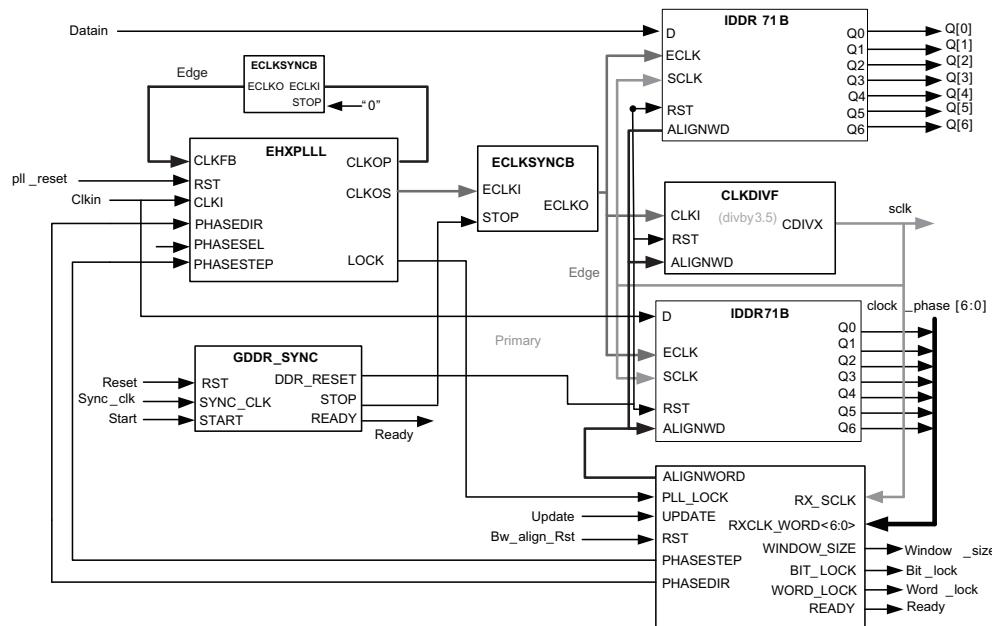
This interface is used to implement 7:1 LVDS Receiver interface using the 1 to 7 gearing with ECLK. Slow speed clock coming in is multiplied 3.5X using a PLL. This clock is used to capture the data at the receiver IDDRX71 module.

This DDR interface uses the following modules:

- IDDRX71B element is used to capture the data
- EHXPLLK will multiply the input clock by 3.5 and phase shift the incoming clock based on the dynamic phase shift input.
- This clock is routed to the Edge clock (ECLK) clock tree through the ECLKSYNCB module
- CLKDIVF module is used to divide the ECLK by 3.5 and is routed to the primary clock tree used as the SCLK input
- A second IDDRX71B element is used with data connected to clock input to generate 7 bit clock phase that can be used for word alignment
- The startup synchronization soft IP (GDDRX\_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- An optional Bit and Word alignment soft IP(BW\_ALIGN) can be enabled in Clarity Designer. The Bit alignment module will rotate PLL's 16 phases to center Edge clock to middle of data eye and the word alignment module will use ALIGNWD function of CLKDIVD and IDDRX71B to achieve 7-bit word alignment
- The ECLKBRIDGE can be optionally enabled if the data bus will be crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through Clarity Designer.

The following figures show the interface with the soft IP modules.

**Figure 13-13. GDDRX71\_RX.ECLK Interface**



### Interface Requirements

- The clock input must use a dedicated PLL input pin so it is routed directly to the PLL
- CLKOP output of the PLL must be used as feedback using another edge clock tree to compensate for ECLK tree delay used by CLKOS. Hence this interface will use two ECLK trees.
- ECLK must use the Edge clock tree and the SCLK out of the CLKDIVF must use the Primary clock tree.

- “Use Primary” preference may be assigned to the SCLK out of the CLKDIVF module

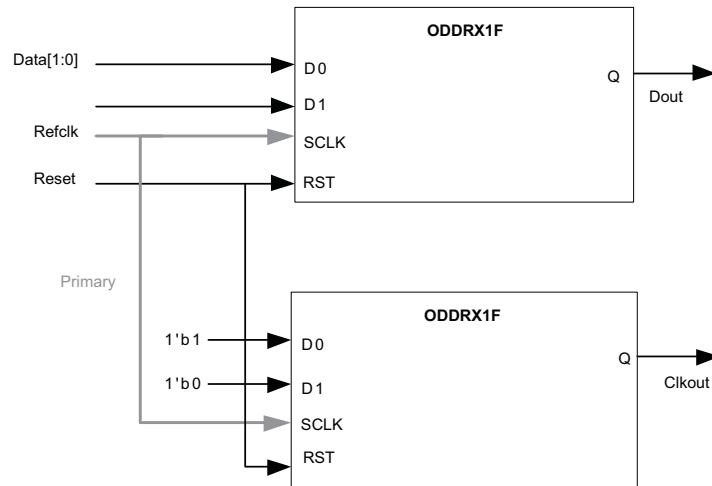
### **GDDRX1\_TX.SCLK.Aligned**

This interface is used to implement Generic Transmit DDR with 1X gearing using primary clock (SCLK). The Clock output is aligned to the Data output.

This DDR interface uses the following modules:

- ODDRX1F element is used to generate the data output
- The primary clock (SCLK) is used as the clock for both data and clock generation
- Optionally the user can choose to use the DELAYG or DELAYF element to delay the output data
- The output data can be optionally tristated using either a Tristate input going through an I/O register.

**Figure 13-14. GDDRX1\_TX.SCLK.Aligned Interface**



### **Interface Requirements**

- The clock to the output DDR modules must be routed on the primary clock tree

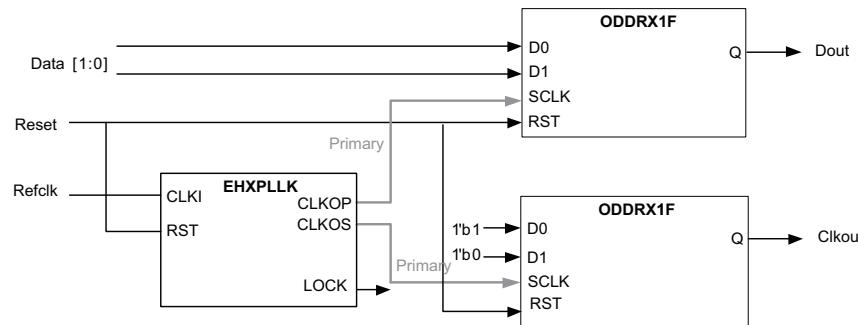
### **GDDRX1\_TX.SCLK.Centered**

Generic Transmit DDR using X1 gearing with SCLK. Clock output is centered to the Data output.

This DDR interface uses the following modules:

- ODDRX1F element is used to generate the data output
- The EHXPLL module is used to generate the clocks for the data and clock ODDRX1F modules. The clock used to generate the clock output is delayed 90 to center to data at the output.
- Both these clocks are routed on primary clock tree
- Optionally the user can choose to use the DELAYG or DELAYF element to delay the output data
- The output data can be optionally tristated using either a Tristate input going through an I/O register.

**Figure 13-15. GDDRX1\_TX.SCLK.Centered Interface**



### Interface Requirements

- The clock to the output DDR modules must be routed on the primary clock tree

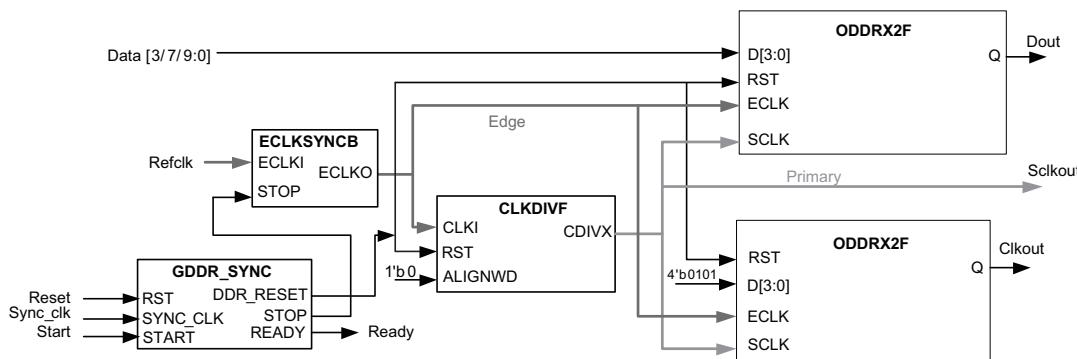
### GDDRX2\_TX.ECLK.Aligned

The interface is used to generate Generic Transmit DDR with 2X gearing using high speed edge clock (ECLK). The Clock output is edge aligned to the Data output.

This DDR interface uses the following modules:

- ODDRX2Ffor 2X gearing is used to generate the output data
- The high speed ECLK is routed to the edge clock tree through the ECLKSYNCB module
- The SCLK is routed on the primary clock tree and is generated from the ECLK using the CLKDIVF module
- The same ECLK and SCLK are used for both Data and Clock generation.
- The startup synchronization soft IP (GDDRX\_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- The ECLKBRIDGE can be optionally enabled if the data bus will be crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through Clarity Designer.
- Optionally the user can choose to use the DELAYG or DELAYF element to delay the data output
- The output data can be optionally tristated using either a Tristate input going through an I/O register.

**Figure 13-16. GDDRX2\_TX.ECLK.Aligned Interface**



### Interface Requirements

- The SCLK input to the output DDR modules must be routed on the primary clock tree and the ECLK input is routed on the edge clock tree
- “USE PRIMARY” preference may be assigned to the SCLK net
- The user must set the timing preferences as per section “Timing Analysis Requirement”

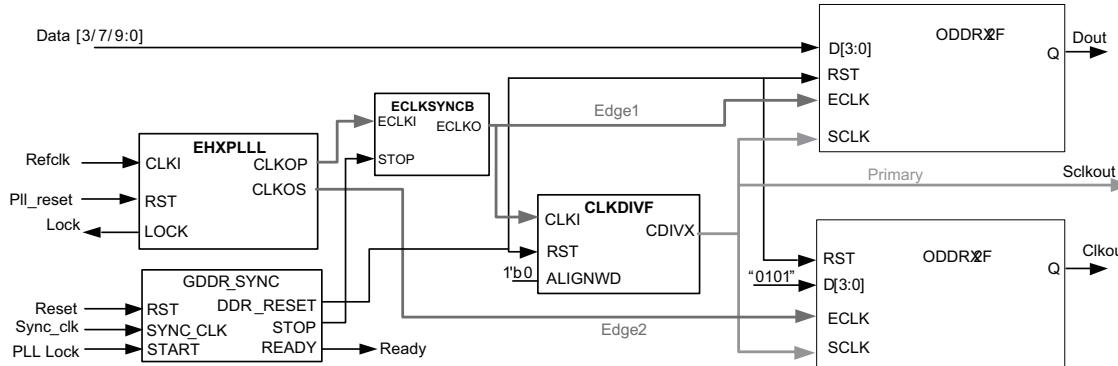
### **GDDRX2\_TX.ECLK.Centered**

This interface is used to implement Generic Transmit DDR with 2X gearing using edge clock (ECLK). The Clock output is centered to the Data output.

This DDR interface uses the following modules:

- ODDRX2F for X2 gearing is used to generate the data output
- The high speed ECLK is routed to the edge clock tree through the ECLKSYNCB module
- The SCLK is routed on the primary clock tree and is generated from the ECLK using the CLKDIVF module
- The same ECLK and SCLK are used for both Data and Clock generation.
- The EHXPLL element is used to generate the clocks for the data and clock ODDR modules. The clock used to generate the clock output is delayed 90 to center to data at the output.
- The startup synchronization soft IP (GDDRX\_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- Optionally the user can choose to use the DELAYG or DELAYF element to delay the data output
- The output data can be optionally tristated using either a Tristate input going through an I/O register.
- The ECLKBRIDGE can be optionally enabled if the data bus will be crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through Clarity Designer.

**Figure 13-17. GDDRX2\_TX.ECLK.Centered Interface**



### **Interface Requirements**

- The SCLK input to the output DDR modules must be routed on the primary clock tree and the ECLK input is routed on the edge clock tree
- “USE PRIMARY” preference may be assigned to the SCLK net
- The user must set the timing preferences as per section “Timing Analysis Requirement”

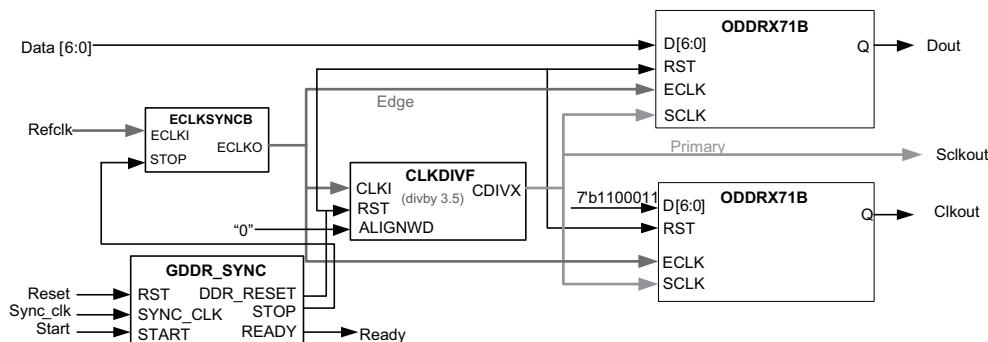
## GDDRX71\_TX.ECLK

This interface is used to implement transmit side of the 7:1 LVDS interface DDR using the 7 to 1 gearing with ECLK. The clock output is aligned to the data output.

This DDR interface uses the following modules:

- ODDRX71B is used to generate the data output
- The high speed ECLK is routed to the edge clock tree through the ECLKSYNCB module
- The SCLK is routed on the primary clock tree and is generated from the ECLK using the CLKDIVD module
- The same ECLK and SCLK are used for both Data and Clock generation.
- The startup synchronization soft IP (GDDRX\_SYNC) is required for this interface to tolerate the skew between the ECLKSYNCB Stop input and the Reset to the DDR and CLKDIV modules.
- The ECLKBRIDGE can be optionally enabled if the data bus will be crossing over between the left and right sides of the device. If ECLKBRIDGE is enabled then the ECLKBRIDGECS element should be used in the interface before the ECLKSYNCB element. This element can be enabled through Clarity Designer.

**Figure 13-18. GDDRX71\_TX.ECLK Interface**



### Interface Requirements

- The SCLK input to the output DDR modules must be routed on the primary clock tree and the ECLK input is routed on the edge clock tree
- “USE PRIMARY” preference may be assigned to the SCLK net
- The user must set the timing preferences as per section “Timing Analysis Requirement”

## Generic DDR Design Guidelines

This section describes the various design guidelines used for building generic high speed DDR interfaces in ECP5 devices. In addition to these guidelines, it is also required to follow the Interface Rules described for each type of interface, you will need to find the interface you are building in the section above “High Speed Interface Details”.

### Using the High Speed Edge Clock Bridge

The High Speed Edge Clock Bridge is available to wide data busses to bridge the edge clock from one side to the other. To enable this bridge the user would need to instantiate the “ECLKBRIDGECS” element in the HDL design. When using the ECLKBRIDGE, both the ECLK1 or ECLK0 on that side (spanning both the banks will be used). This will reduce the number of interfaces that can be built on a given side. See TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#) for details.

## Receive Interface Guidelines

- Differential DDR interface can be implemented on the Left and Right sides of the device
- There are 4 different edge clocks available per side (two per bank).
- Each of the edge clocks can be used to generate either a centered or aligned interface.
- Each side has two CLKDIV modules which would mean you can implement two different GDDR2 RX interface per side since each 2X gearing would require CLKDIV module to generate a slower SCLK.
- There is DDRDLLA located on each corner of the device, total of 4 in a device. LLC and LRC DDRDLLs only drive code to one side whereas the ULC and URC DDRDLLs drive code to two sides turning corners
- Each DQSBUF/DLLDEL has access to two DDRDLLs hence two different RX rates are available per side, 4 are available on the entire device.
- The Receive clock input should be placed on a dedicated PCLK input pin. The PCLK pin has direct access to the edge clock tree for centered interface and it also has direct connection to the DLLDELD when implementing an aligned interface.
- When implementing IDDRX71 interface, the complementary PAD is not available for other functions since the IDDRX71 used the I/O registers of the complementary PAD as well.
- It is recommended that clock input be located on the same side as data pins
- The top side of the device does not have Edge clocks hence can only be used to receive lower speed interfaces (<200 MHz) that use 1x gearing. Top side of the device can be used for single ended interfaces only.
- Interfaces using the x1 gearing will use the primary clock resource. You can use as many interfaces as the number of primary clocks supported in the device.
- In addition to dedicate PCLK pins, ECP5 devices have GR\_PCLK pins, these use shortest general route path to get to the primary clock tree. These pins are not recommended for use with DDR interfaces. They can be used for SDR or other generic FPGA designs.

## Transmit interface Guidelines

- Use PADA and PADB for all TX using true LVDS interfaces.
- When implementing Transmit Centered interface, two ECLKs are required. One to generate the Data Output and the other to generate the CLK Output.
- When implementing Transmit Aligned interface only one ECLK is required for both Data output and Clock output.
- Each side has two CLKDIV modules which would mean you can implement two different GDDR2 TX interface per side since each 2X gearing would require CLKDIV module to generate a slower SCLK.
- The top side of the device does not have Edge clocks hence can only be used to receive lower speed interfaces (<200 MHz) that use 1x gearing. Top side of the device can be used for single ended interfaces only.
- Interfaces using the x1 gearing will use the primary clock resource. You can use as many interfaces as the number of primary clocks supported in the device.

## Clocking Guidelines for Generic DDR Interface

- The edge clock and primary clock resources are used when implementing a 2X receive or transmit interface.
- Only the primary clock (PCLK) resources are used when implementing x1 receive or transmit interfaces.
- Each edge clock can only span up to one side (Left or Right) of the device, hence all the data bits of the in the x2 interface must be locked to one side of the device. If wide bus implementation is required then the ECLKBRIDGE element must be used to bridge the edge clock to the other side. ECLKBRIDGE can be used to bridge the Left and Right Side ECLKs.

- When implementing x1 interfaces, the bus can span Left, Right or Top sides as primary clocks can access DDR registers on all sides.
- Bottom side only supports SERDES function hence does not have any edge clocks or DDR registers except on the LFE-85 device, some I/Os support 1X DDR registers similar to the Top side.
- The ECLK to DDR registers can be accessed through dedicated PCLK pins, GPLL outputs, DDRDLL outputs. See TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#) for details.
- Primary clock to DDR registers can be accessed through dedicated PCLK pins, GPLL outputs and CLKDIV outputs. See TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#) for details.
- None of the clocks going to the DDR registers can come from internal general routing.
- DQS clocking is used for DDR memory interface implementation. DQS clock spans every 12 to 16 I/O's include the DQS pins. Refer to the "DQ-DQS Grouping" section for pinout assignment rules when using DQS clocking.

## Timing Analysis for High Speed DDR Interfaces

It is recommended that the user run Static Timing Analysis in the software for each of the high speed interfaces. This section describes the timing preferences to used for each type of interface and the expected trace results. The preferences can either be entered directly in the .lpf file or through the Design Planner graphical user interface.

The External Switching Characteristics section of DS1044, [ECP5 Family Data Sheet](#) should be used along with this section. The data sheet specifies the actual values for these constraints for each of the interfaces.

### Frequency Constraints

It is required that the user explicitly specify FREQUENCY (or PERIOD) PORT preferences to all input clocks in the design. This preference may not be required if the clock is generated out of a PLL or DLL or is input to a PLL or DLL.

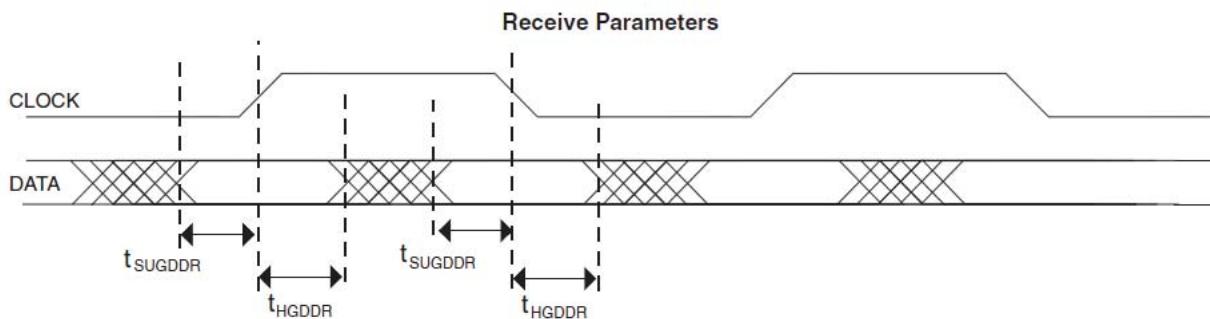
### DDR Input Setup and Hold Time Constraints

All of the Receive (RX) interfaces, both x1 and x2 can be constrained with setup and hold preference.

#### *Receive Centered Interface*

Figure 13-19 below shows the Data and Clock relationship for a Receive Centered Interface. The clock is centered to the data, so it comes into the devices with a setup and hold time.

**Figure 13-19. RX Centered Interface Timing**



Note:  $t_{SUGDDR}$  = Setup Time,  $t_{HGDDR}$  = Hold Time

In this case the user must specify in the software preference the amount of setup and hold time available. These parameters are listed in Figure 13-19 as  $t_{SU\_GDDR}X1/2$  and  $t_{HO\_GDDR}X1/2$ . These can be directly provided using the INPUT\_SETUP and HOLD preference as –

INPUT\_SETUP PORT "DATA" <tSU\_GDDRX1/2> ns HOLD <tHO\_GDDRX1/2> ns CLKPORT "CLOCK";

where:

Data = Input Data Port

Clock = Input Clock Port

The external Switching Characteristics section of DS1044, [ECP5 Family Data Sheet](#) specifies the MIN setup and hold time required for each of the high speed interfaces running at MAX speed. These values can be picked up from the data sheet if the interface is running at MAX speed.

Example:

For GDDRX2\_RX.ECLK.Centered Interface running at max speed of 400 MHz, the preference would be -

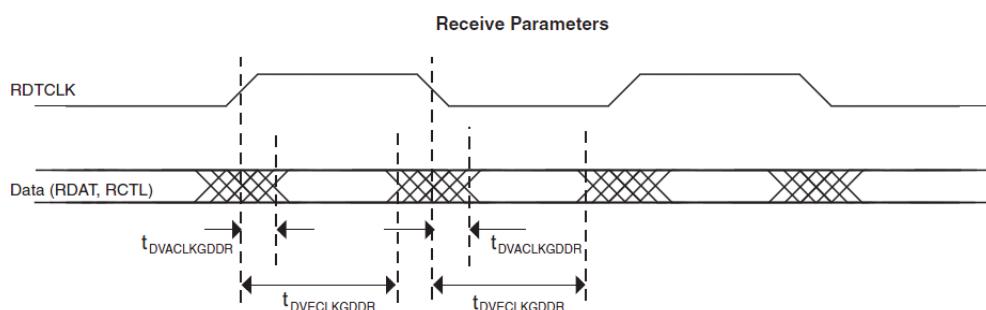
INPUT\_SETUP PORT "datain" 0.320000 ns HOLD 0.320000 ns CLKPORT "clk";

Note: Please check DS1044, [ECP5 Family Data Sheet](#) for the latest tSUDDR and tHOGDDR numbers.

### **Receive Aligned Interface**

Figure 13-20 below shows the Data and Clock relationship for a Receive Aligned Interface. The clock is aligned edge to edge the data.

**Figure 13-20. RX Aligned Interface Timing**



Note: tDVA\_GDDRX1/2 = Data Valid after CLK, tDVE\_GDDRX1/2 = Data Hold After CLK

In this case the worst case data may occur after the clock edge hence has a negative setup time when entering the device. In this case the worst case setup is specified by the tDVACLKGDDR after the clock edge and the worst case hold time is specified as tDVECLKGDDR. For this case the setup and hold time can be specified as -

INPUT\_SETUP PORT "din" <-tDVA\_GDDRX1/2> ns HOLD <tDVE\_GDDRX1/2> ns CLKPORT "clk";

Note - Negative number is used for SETUP time as the data occurs after the clock edge in this case.

The External Switching Characteristics section of DS1044, [ECP5 Family Data Sheet](#) specifies the MIN tDVA\_GDDRX1/2 and tDVE\_GDDRX1/2 values required for each of the high speed interfaces running at MAX speed. These values can be picked up from the data sheet if the interface is running at MAX speed. The data sheet numbers for this preference is listed in ns + ½ UI (Unit Interface). 1 UI is equal to ½ the Clock Period. Hence these numbers will need to be calculated from the CLK Period used.

Preference Example:

For GDDRX2\_RX.ECLK.Aligned interface running at max speed of 400 MHz (UI = 1.25ns)

$t_{DVA\_GDDR2} = -0.344\text{ns} + \frac{1}{2}\text{ UI} = 0.281\text{ns}$ ,  $t_{DVE\_GDDR2} = 0.344\text{ns} + \frac{1}{2}\text{ UI} = 0.969\text{ ns}$

The preference for this case would be -

INPUT\_SETUP PORT "datain" -0.2810000 ns HOLD 0.969 ns CLKPORT "clk";

Note: Please check DS1044, [ECP5 Family Data Sheet](#) for the latest  $t_{DVA\_GDDR1/X2}$  and  $t_{DVE\_GDDR1/X2}$  numbers.

### **Receive Dynamic Interfaces**

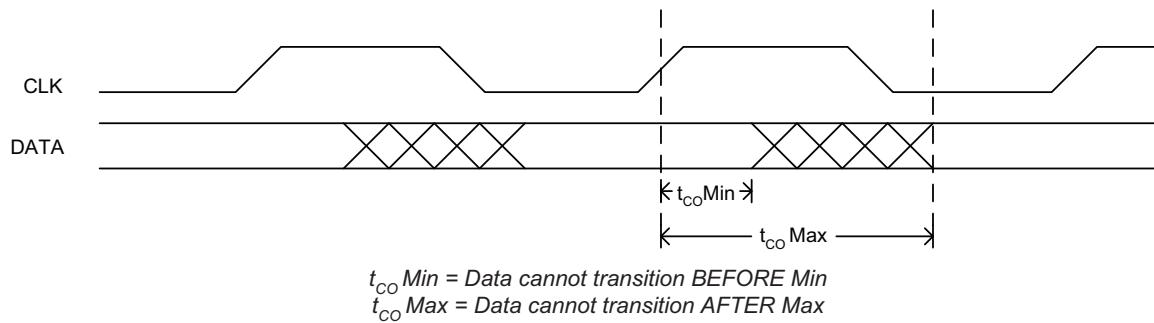
Static Timing Analysis will not show timing for all the Dynamic interfaces cases as the either the Clock or Data delay will be dynamically updated at run time.

### **DDR Clock to Out Constraints for Transmit Interfaces**

All of the Transmit (TX) interfaces both x1 and x2 can be constrained with Clock to out constraint to detect the relationship between the Clock and Data when leaving the device.

Figure 13-21 shows how the clock to out is constrained in the software. Min  $t_{CO}$  is the minimum time after the clock edge transition that the data will not transition. Max  $t_{CO}$  is the maximum time after clock transition before which the data will transition. So any data transition must occur between the  $t_{CO}$  Min and  $t_{CO}$  Max values.

**Figure 13-21.  $t_{CO}$  Min and Max Timing Analysis**



### **Transmit Centered Interfaces**

In this case the transmit clock is expected to be centered to the data when leaving the device. Figure 13-22 shows the timing for a centered transmit interface.

$t_{DVBGDDR}$  = Data valid before clock

$t_{DVAGDDR}$  = Data valid after clock

$t_U$  = Data transition

**Figure 13-22. Transmit Centered Interface Timing**

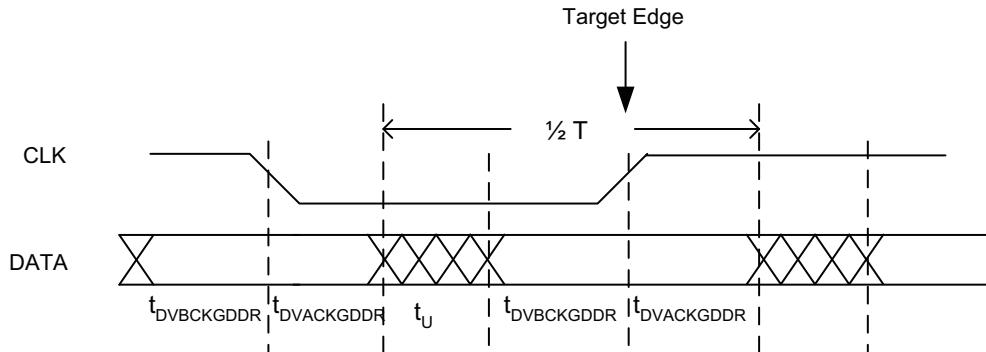


Figure 13-22 shows that max value after which the data cannot transition is  $-t_{VB\_GDDR}$ . The min value before which the data cannot transition is  $-(t_U + t_{VB\_GDDR})$ . Negative sign is used because in this particular case where clock is forwarded centered aligned to the data these two conditions occurs before the clock edge.

The DS1044, [ECP5 Family Data Sheet](#) specifies the  $t_{DVB\_GDDRX1/X2}$  and  $t_{DVA\_GDDRX1/X2}$  values at maximum speed. But we do not have the  $t_U$  value hence min  $t_{CO}$  can be calculated using the following equation.

$$t_{CO} \text{ Min} = - (t_{VB\_GDDRX1/X2} + t_U)$$

$$\frac{1}{2} T = t_{DVA\_GDDRX1/X2} + t_{VB\_GDDRX1/X2} + t_U$$

$$- (t_{VB\_GDDRX1/X2} + t_U) = \frac{1}{2}T - t_{DVA\_GDDRX1/X2}$$

$$t_{CO} \text{ Min} = \frac{1}{2}T - t_{DVA\_GDDRX1/X2}$$

The clock to out time in the software can be specified as –

*CLOCK\_TO\_OUT PORT "dataout" MAX <-t<sub>DVB\_GDDRX1/X2</sub>> MIN <t<sub>DVA\_GDDRX1/X2</sub> -1/2 Clock Period> CLKPORT "clk" CLKOUT PORT "clkout";*

where:

Data = Data Output Port

Clock = Forwarded Clock Output Port

clk = Input Clock Port

The values for  $t_{DVCKGDDR}$  and  $t_{DVACKGDDR}$  can be picked up from the External Switching Characteristics section of DS1044, [ECP5 Family Data Sheet](#) for the MAX speed.

Preference Example:

For GDDRX1\_TX.SCLK.Centered interface running at 250 MHz,  $t_{DVB\_GDDRX1} = t_{DVA\_GDDRX1} = 0.67\text{ns}$ , the preference would be -

*CLOCK\_TO\_OUT PORT "dataout" MAX -0.670000 ns MIN -1.330000 ns CLKPORT "clk" CLKOUT PORT "clkout";*

Note: Please check DS1044, [ECP5 Family Data Sheet](#) for the latest  $t_{DVAGDDR}$  and  $t_{DVBGDDR}$  numbers.

### **Transmit Aligned Interfaces**

In this case the clock and data are aligned when leaving the device. Figure 13-23 below shows the timing diagram for this interface.

$t_{DIAGDDR}$  = Data valid after clock.

$t_{DIBGDDR}$  = Data valid before clock.

**Figure 13-23. Transmit Aligned Interface Timing**

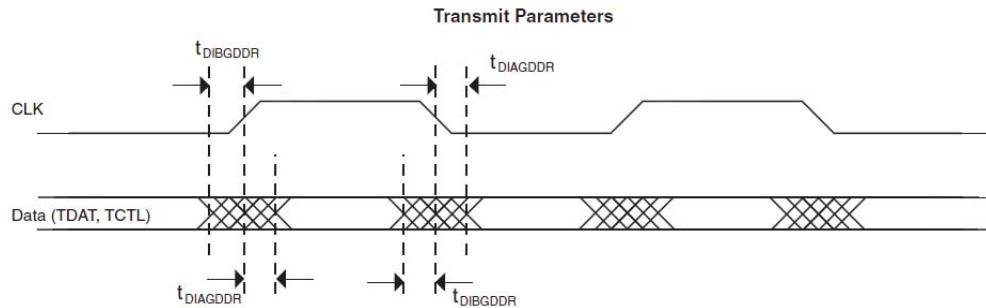


Figure 13-23 shows that max value after which the data cannot transition is  $t_{DIA\_GDDRX1/X2}$ . The min value before which the data cannot transition is  $-t_{DIB\_GDDRX1/X2}$ . Negative sign is used for the min value is because in this particular case the min condition occurs before the clock edge.

The clock to out time in the software can be specified as –

`CLOCK_TO_OUT PORT "dataout" MAX < $t_{DIA\_GDDRX1/X2}$ > MIN <- $t_{DIB\_GDDRX1/X2}$ > CLKPORT "clk" CLKOUT PORT "clk";`

where:

Data = Data Output Port

Clock = Forwarded Clock Output Port

clk = Input Clock Port

Both  $t_{DIA\_GDDRX1/X2}$  and  $t_{DIB\_GDDRX1/X2}$  numbers are available in the External Switching Characteristics section of DS1044, [ECP5 Family Data Sheet](#) for maximum speed.

Preference Example:

For GDDRX2\_TX.Aligned case running at 400 MHz,  $t_{DIA\_GDDRX2}= t_{DIB\_GDDRX2}=0.16\text{ns}$ . The preference would be -

`CLOCK_TO_OUT PORT "dataout" MAX 0.16 ns MIN -0.16ns CLKPORT "clk" CLKOUT PORT "clkout";`

Note: Please check DS1044, [ECP5 Family Data Sheet](#) for the latest  $t_{DIA\_GDDX1/X2}$  and  $t_{DIB\_GDDRX1/X2}$  numbers

## ECP5 Memory Interfaces

All of the DDR SDRAM interface transfers data at both the rising and falling edges of the clock. The I/O DDR registers in the ECP5 device can be used to support DDR2, DDR3, DDR3L, LPDDR2 and LPDDR3 memory interfaces.

These memory interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. The DQS strobe is a differential signal except for DDR2 you can choose between single-ended or differential DQS strobe.

Figure 13-24 shows typical DDR memory signals. DDR2, DDR3 and DDR3L memory interfaces are typically implemented with either four or eight DQ data bits per DQS. So, a 16-bit DDR memory interface will use two or four DQS signals, and each DQS is associated with four or eight DQ bits, respectively. Both the DQ and DQS are bi-direc-

tional ports and are used to read and write to the memory. LPDDR2 and LPDDR3 memory are the same but will only support 8 DQ data bits per DQS strobe.

When reading data from the external memory device, data coming into the FPGA controller is edge-aligned with respect to the DQS signal. This DQS strobe signal needs to be phase shifted 90° before the FPGA logic can sample the read data. When writing to a DDR memory, the memory controller (FPGA) must shift the DQS by 90° to center-align with the data signals (DQ). A clock signal is also provided to the memory. This clock is provided as differential clock (CK and CK#) to minimize duty cycle variations. The memory also uses these clock signals to generate the DQS signal during a read via a DLL inside the memory. The figures below show DQ and DQS timing relationships for read and write cycles.

During read, the DQS signal is low for some duration after it comes out of tristate. This state is called Preamble.

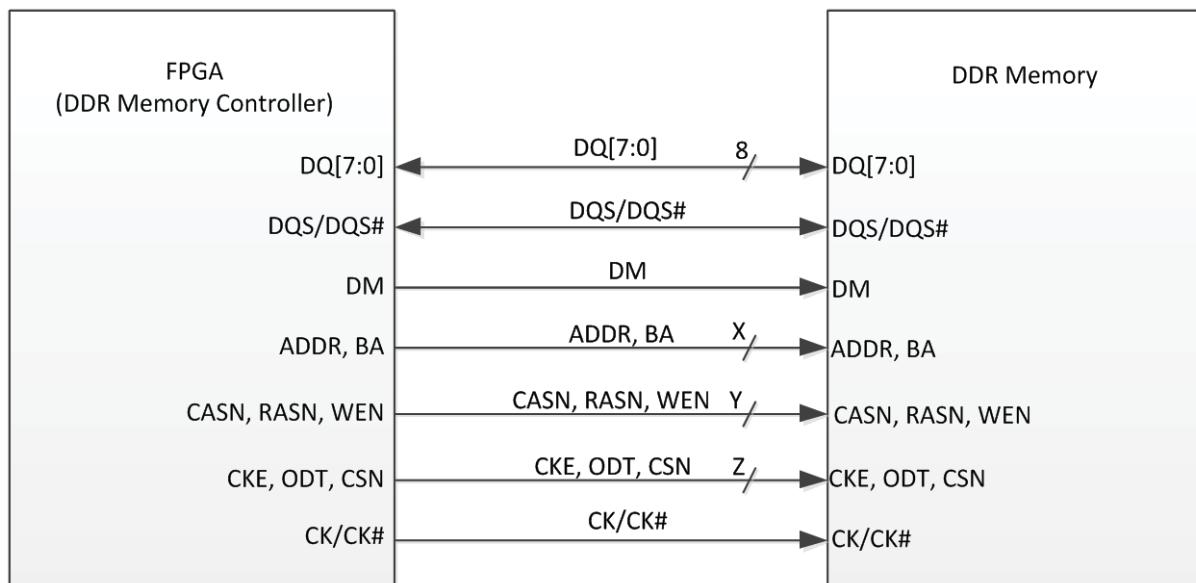
The state when the DQS is low before it goes into tristate is the Postamble state. This is the state after the last valid data transition.

DDR memories also require 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. Figure 13-24 shows a typical 8-bit interface that has eight associated DQ data bits per DQS strobe signal.

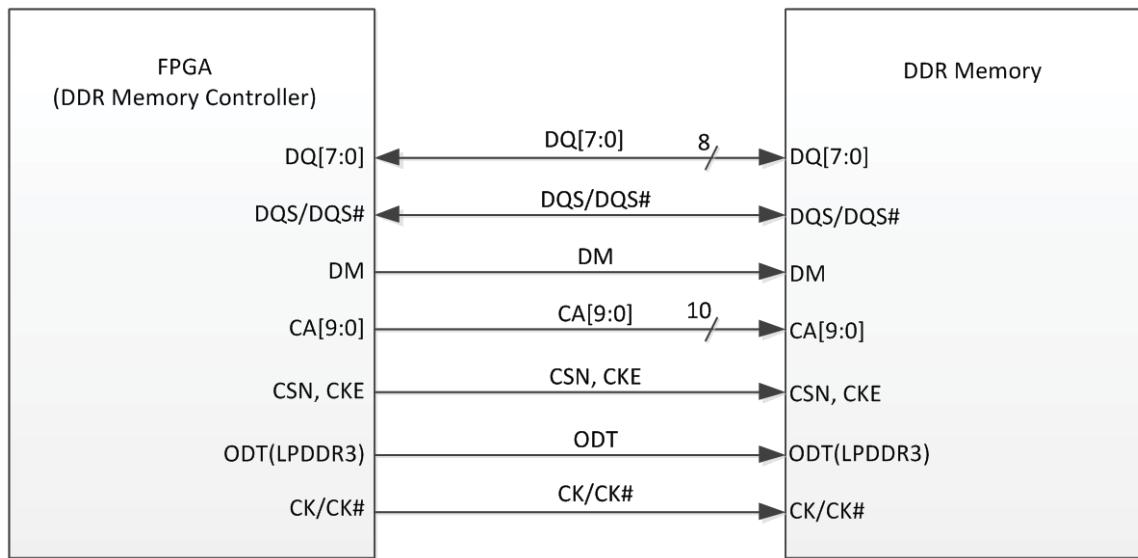
The DDR3 memory module uses fly-by routing topology for the address, command, control and clock signals. This requires the memory controller to support read and write leveling to adjust for leveled delay on read and write data transfers. LPDDR3 does not use fly-by routing but write leveling may be supported if user emulates the fly-by routing using board traces. You can see more information in the DDR pin placement and layout guidelines section of this document.

One major difference between DDR2/DDR3 and LPDDR2/LPDDR3 is lack of DLL in LPDDR2/LPDDR3 memory device. This would mean in LPDDR2 and LPDDR3, the clock to data output delay from memory device is not compensated by DLL as in traditional DDR2/DDR3, thus the delay is much larger and has larger spread. Theoretically, there is no low frequency limitation on LPDDR2/LPDDR3, although most manufactures place a low limit of 10 MHz. Please note FPGA memory controller side, DLL is still needed to manage write and read phase shift, for all memory interfaces including LPDDR2 and LPDDR3.

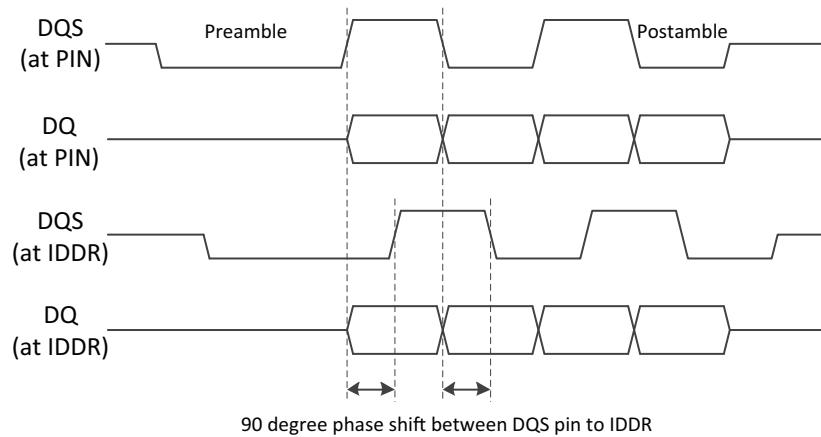
**Figure 13-24. Typical DDR2/DDR3/DDR3L Memory Interface**



**Figure 13-25. Typical LPDDR2/LPDDR3 Memory Interface**



**Figure 13-26. DQ-DQS During Read**



**Figure 13-27. DQ-DQS During Write**

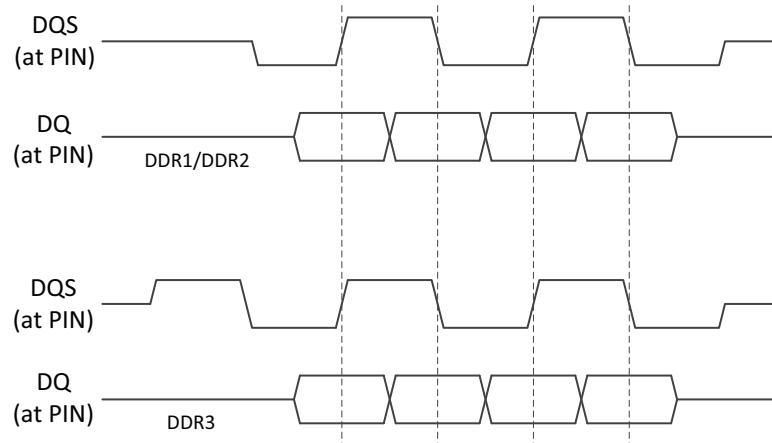


Table below shows the different DDR memory configurations and features supported by ECP5 device.

**Table 13-2. DDR Memory Configurations Support**

DDR Memory	Data Width	VCCIO	DQ	DQS	Modules Types	Rank	Chip Selects	Write Leveling	CMD/ADDR Timing	Fmax
DDR2	8 to 72 bits	1.8V	SSTL18	SSTL18D or SSTL18	UDIMM, SODIMM, RDIMM	Single, Dual	1, 2	No	2T	400 MHz
	8 to 72 bits	1.8V	SSTL18	SSTL18D or SSTL18	Embedded	Single, Dual	1, 2	No	2T <sup>3</sup>	400 MHz
DDR3	8 to 72 bits	1.5V	SSTL15	SSTL15D	UDIMM, SODIMM, RDIMM	Single, Dual	1, 2	Yes	2T <sup>3</sup>	400 MHz
	8 to 72 bits	1.5V	SSTL15	SSTL15D	Embedded	Single, Dual	1, 2	Yes <sup>1</sup>	2T <sup>3</sup>	400 MHz
DDR3L	8 to 72 bits	1.35V	SSTL135	SSTL135D	UDIMM, SODIMM, RDIMM	Single, Dual	1, 2	Yes	2T <sup>3</sup>	400 MHz
	8 to 72 bits	1.35V	SSTL135	SSTL135D	Embedded	Single, Dual	1, 2	Yes <sup>1</sup>	2T <sup>3</sup>	400 MHz
LPDDR2	16 and 32 bits	1.2V	HSUL12	HSUL12D	Embedded (Single Channel)	Single	1	No	ODDRX2	400 MHz
LPDDR3	16 and 32 bits	1.2V	HSUL12	HSUL12D	Embedded (Single Channel)	Single	1	Yes <sup>2</sup>	ODDRZ2	400 MHz

1. If Fly-by Wiring is implemented
2. Fly-by wiring is emulated using board traces (Guidelines are in the section below)
3. CSN will use 1T timing

## DDR Memory Interface Requirements

As described in the overview section, all the DDR memory interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. When reading data from the external memory device, data coming into the ECP5 device is edge-aligned with respect to the DQS signal. Therefore, the ECP5 device needs to shift the incoming DQS (90° phase shift) before using it to sample the read data.

To implement the write portion of a DDR memory interface, parallel single data rate data must be multiplexed depending on the IDDR and ODDR register gearing mode together with data transitioning on both edges of the clock. In addition, during a write cycle, the ECP5 devices generate a DQS signal that is center aligned with the DQ, the data signal. This is accomplished by ensuring a DQS strobe is 90° shifted relative to the DQ data. The ECP5 devices provide the solutions to achieve the following design challenges to implement DDR memory write functions:

- DQ/DM needs to be center-aligned to DQS.
- DQS needs to be edge-aligned to CK. In DDR3 interfaces where fly-by routing is used, write leveling should be used to compensate for skews between CK and DQS.
- The DDR output data must be multiplexed into a single outgoing DDR data stream.
- Generate ADDR/CMD signal edge-aligned to CK falling edge to maximize the tIS and tIH timing parameters.
- Differential CK signals (CK and CK#) need to be generated.
- The controller must meet the DDR interface specification for the tDSS and tDSH parameters, defined as DQS falling edge setup and hold time to/from CK rising edge, respectively. The skews, if caused by the fly-by topology, are compensated by write-leveling.
- In case of LPDDR2 and LPDDR3, the CA[9:0] bus needs to be 90 degree from the CLKP/CLKN signal.
- For DDR3 memory, the memory controller also needs to handle the write leveling required by the interface when the fly-by topology is applied.

## Features for Memory Interface Implementation

The ECP5 devices contain a variety of features to simplify implementation of the read and write operations of a DDR interface:

- DQS Clock Tree spanning the DQS group
- DDRDLL used to generate the 90 degree delay codes
- DLL-compensated DQS delay elements
- Input FIFO for read data clock domain transfer
- Dedicated DDR Memory input and output registers
- Dynamic Margin Control Circuit to adjust Read and Write delays
- Input/Output Data Delay used to compensate for DQS clock tree delay

### DQS Grouping

In DDR interfaces with eight DQ pads associated to one DQS pad, each DQS group generally consists of at least 10 I/Os (one DQS, eight DQ and one DM) for an 8-bit DDR2 memory interface or 11 I/Os (two DQS, eight DQ, one DM) to implement a complete 8-bit DDR3/DDR3L/LPDDR2/LPDDR3 memory interface. In case of LPDDR2/3, two additional DQS groups are required to generate the CA[9:0] (with 10 I/Os) and Control/CLKP/CLKN (with 5 I/Os for LPDDR3 and 4 I/O on LPDDR2) outputs.

In ECP5 devices, a DQS group consists of 12 to 16 I/Os depending on the device and package selected to accommodate these DDR interface needs. ECP5 devices support DQS signals on the left and right sides of the device.

Each DQS signal spans across 12 to 16 I/Os. Any 10 (for DDR2) or 11 (for DDR2/DDR3/LPDDR2/LPDDR3) of these 16 I/Os spanned by the DQS can be used to implement an 8-bit Data side interface. For LPDDR2/LPDDR23, any group with 10 I/Os is required for CA[9:0] bus and another group with 5 I/Os for LPDDR3 and 4 I/Os for LPDDR2 is required to generate the Control and CLKP/CLKN outputs. In addition to the DQS grouping, the user must also assign the reference voltage (VREF) input to an I/O in that bank required to implement the referenced I/O standard.

**Figure 13-28. DQ-DQS Grouping**

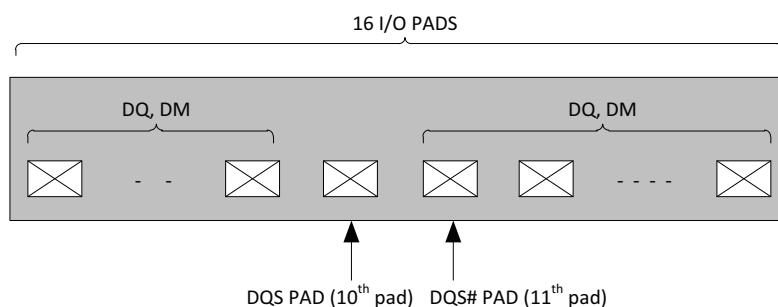


Figure 13-28 shows a typical DQ-DQS group for ECP5 devices. The 10th I/O Pad of this 16 I/O group is the dedicated DQS pin. All the 9 pads before the DQS and 6 pads after the DQS are covered by this DQS bus span. If a differential DQS pair is required then the 11th pad is used by the DQS# signal. The user can assign any other I/O pads to be DQ data or DM pins. Therefore, for example, to implement a 32-bit wide memory interface you would need to use four such DQ-DQS groups when eight-to-one DQ-DQS association is used.

In case of LPDDR2 and LPDDR2, two additional DQS groups are required to assign the CA[9:0] and the control signals.

In case of DDR2/DDR3/DDR3L, additional I/O pads are required to implement address, command and control. They do not have to be I/Os in DQS groups but need to use 1X gearing generic output DDR capable pads.

Each of the dedicated DQS pins is internally connected to the DQS phase shift circuitry. The pin out sheets included as part of DS1044, [ECP5 Family Data Sheet](#) shows pin locations for each of the DQS groups.

### DLL-Compensated DQS Delay Elements

The DQS to and from the memory is connected to the DQS delay element inside the ECP5 device. The DQS delay block receives the delay control code, DDRDEL, from the on-chip DDRDLL. The code generated by DDRDLL is connected to the DQSBUF circuit to perform 90° read phase shift and 90° write phase shift. DDRDLL requires the frequency reference from PLL, normally going through the edge clock tree.

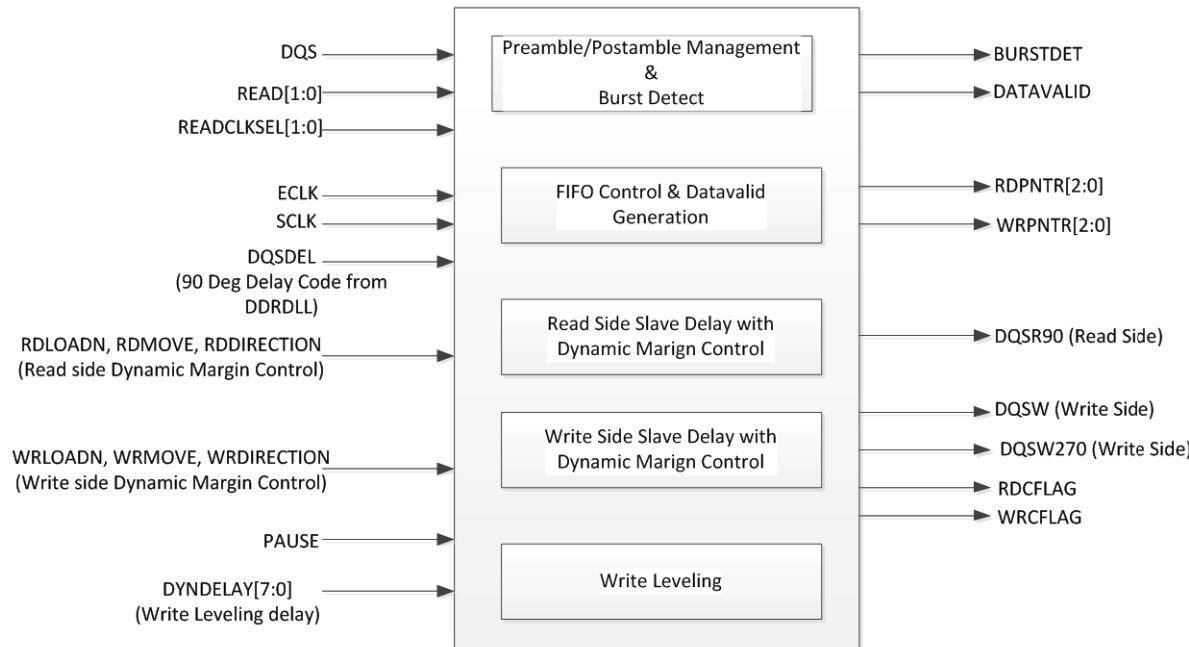
ECP5 devices support one DDRDLL modules in each corner of the device. The DQSBUF modules that receive the DDRDEL code from DDRDLL can either receive the code from the top or bottom DDRDLL on that side. Hence each side can support up to two different rate DDR memory interfaces.

**Table 13-3. DDRDLL Connectivity**

DDRDLL Location	Left DQSBUFs	Right DQSBUFs
DDRDLL_TR		X
DDRDLL_TL	X	
DDRDLL_BR		X
DDRDLL_BL	X	

The DQS received from the memory is delayed in the DQS delay element in the DQSBUF block, and this delayed DQS is used to clock the first set stage DDR input registers.

**Figure 13-29. DQSBUF Block Functions**



### Data Valid Module

The DQSBUF block generates a DATAVALID signal. This signal indicates the timing that the IDDR module drives the valid read data transmitting to the FPGA fabric. DATAVALID is a level-sensitive active high signal and indicates that the read data output from the IDDR module is valid while it is asserted high. The DATAVALID signal can stay

asserted during the IDDR module outputs back-to-back valid read data until all consecutive read operations are completed.

### **READ Pulse Positioning Optimization**

The memory controller is required to provide the READ1/0 signal to the DQSBUF block to position the internal READ pulse and generate the DATAVALID output that indicates the proper timing window of the valid read data. The internal READ pulse is also used to get a clean internal DQS signal between the preamble and postamble periods. The clean DQS is 90° shifted internally to be used to capture the read data.

Due to the DQS round trip delay that includes PCB routing and I/O pad delays, proper positioning of the READ pulse is crucial for successful read operations. The ECP5 device DQSBUF block provides the dynamic READ pulse positioning function which allows the memory controller to locate the READ pulse to an appropriate timing window for the read operations by monitoring the positioning result.

The READCLKSEL2/1/0 and BURSTDET signals are used to accomplish the READ pulse positioning function for a corresponding DQSBUF block. The READ1/0, READCLKSEL2/1/0 signals are driven by the user logic and are part of the DDR memory controller.

The READ1/0 signal needs to be asserted high a certain amount of time before the read preamble starts. The suggested READ1/0 signal assertion timing and the required duration of assertion are listed in Table 13-4. When the internal READ pulse is properly positioned, BURSTDET will be asserted high and guarantee that the generated DATAVALID signal properly indicates the valid read data time window. The READ1/0 signal must stay asserted as long as the number of SCLK cycles that is equal to one fourth of the total burst length as listed in the Table 13-4.

**Table 13-4. READ Training Signals and Initial Read Assertion Position**

Gearing Mode	READ Control	DQSBUF Block	Initial READ Assertion Position*	READ Width in SCLK
X2 Gearing (All DDR Memory Interfaces)	READ1 READ0 READCLKSEL2 READCLKSEL1 READCLKSEL0	DQSBUFM	At least 5.5T before preamble	Total Burst Length / 4

Note: Subject to change after validation tests. The number shown does not include DQS round trip delay.  
1T = 1 tCK memory clock cycle

Once the memory controller initially positions the internal READ pulse using the READ1/0 and READCLKSEL2/1/0 signals, BURSTDET can be used to monitor the positioning result to optimize the READ pulse position. The BURSTDET signal provides a feedback mechanism to inform the memory controller whether the READ pulse has reached to the optimal position for the read operations or not with the current READ1/0 and READCLKSEL2/1/0 values. When it reaches to the optimal position, BURSTDET is asserted High after a read operation. Otherwise, BURSTDET will remain Low. A minimum burst length of eight on the memory bus must be used in the training process. This can be done either with two consecutive BL4 (BL=4) read accesses or one BL8 read access. Any even number of BL4, or any multiplication of BL8(BL=8) can also be used. This read pulse training process must be performed during the initial training and can also be periodically calibrated during the normal operations.

The BURSTDET signal is asserted after the last DQS transition is completed during a read operation and lasts until the next read cycle is started. Once a read operation is started, the memory controller should wait until the DATAVALID signal from DQSBUFM is asserted and then sample the BURSTDET signal at the next cycle to monitor the READ pulse positioning result. If there is no assertion on BURSTDET, it means that the READ pulse has not been located to the optimal position yet. Then, the memory controller needs to shift the READ1/0 signal and/or increase the READCLKSEL2/1/0 value until it detects a BURSTDET assertion. It is recommended that at least 128 read operations be performed repetitively at a READ pulse position during the initialization for getting jitter immunity. 16 read operations can be performed in a periodic calibration if used during the normal operation. The memory controller can determine the proper position alignment when there is no failure on BURSTDET assertions during these multiple trials.

There are a few steps to reposition the internal READ pulse:

**Step 1:**

The memory controller sets READ1/0 to an initial position before starting the read pulse training. READ1/0 must be asserted for the number of SCLK cycles that is equal to one-fourth of the current read burst length as listed in Table 13-4. Each READ bit (READ1 or READ0) in the system clock domain is translated to an 1T time slot of the memory clock domain as shown in Figure 13-30.

**Step 2:**

Once READ1/0 positions the READ pulse, READCLKSEL2/1/0 can be used to shift the READ pulse by 1/4T per step. With the total eight possible combinations from “000” to “111”, READCLKSEL2/1/0 covers the READ pulse shift up to a whole 2T timing window. If BURSTDET is asserted with a certain READCLKSEL2/1/0 value, it indicates that the READ pulse has been located to the optimal position. If no BURSTDET is asserted during this step, the READ pulse needs to be moved to the next timing window.

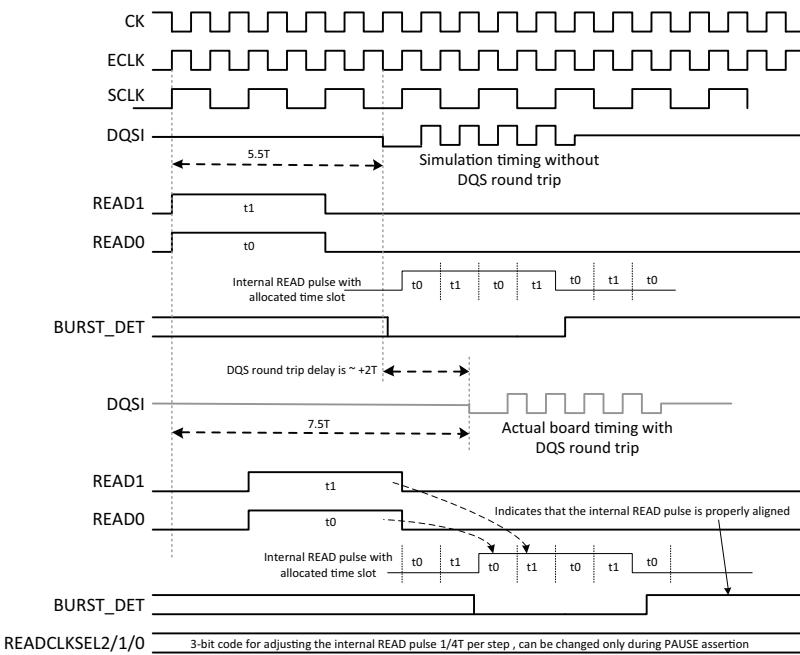
**Step 3:**

To shift the READ pulse timing window, READ1/0 can be moved to the next cycle. If a READ bit is asserted in the next cycle while the other READ bit remains in the current cycle, only the corresponding time slot on the READ pulse will move to the next allotted slot. Therefore, only READ0 must be moved to the next cycle if the READ pulse needs to be shifted only by 1T. However, if only READ1 is moved to the next SCLK cycle, then the READ pulse will have two short pulses in wrong timing. Since READCLKSEL2/1/0 covers a whole 2T timing, it is recommended that both READ1 and READ0 get moved to the next cycle together to shift the READ pulse to the next 2T window as the example shown in Figure 13-30.

Repeat **Step 2** and **Step 3** until BURSTDET is asserted.

Figure 13-30 shows an example of a burst length 8 (BL8) read operation. The bottom side of the diagram indicates the case that the incoming DQS (DQSI) gets slightly more than 2T delay after a round trip. Due to this round trip delay, both READ1 and READ0 need to be shifted to the next SCLK cycle so that the internal READ pulse gets a 2T shift. Then, the READCLKSEL2/1/0 signals can be used to fine tune the READ signal position throughout the training process. When any of READCLKSEL2/1/0 is changed at any time after a system reset, the PAUSE input to DQSBUFM must be asserted before 4T of the change and remain asserted for another 4T after the change to avoid glitches and malfunction.

**Figure 13-30. READ signal Training Process**



Note that the DYNDelay[7:0] signal, margin control signals (WR/RDMOVE, WR/RDLOADN) and DDRDLL update signal (UDDCNTLN) also have the same PAUSE requirement.

#### Dynamic Margin Control on DQSBUF

The ECP5 family includes dynamic margin control signals in the DQSBUF module will allow user to dynamically adjust the read or write side DQS delays generated in the DDRDLL.

Once the margin control mode is enabled by de-asserting WRLOADN (=1) or RDLOADN (=1), the DQSBUF's phase shift control to make a center aligned interface is no longer controlled by the DDRDLL component. It becomes a user's responsibility to complete the margin control training to maximize the valid window and then continuously monitor the DDRDLL delay code (DCNTL7~DCNTL0) and controls the DQSBUF delays accordingly using the WR/RDMOVE and WR/RDDIRECTION signals to compensate the PVT variations.

#### Read Data Clock Domain Transfer Using Input FIFO

Each IDDR module in the ECP5 device has a dedicated input FIFO to provide a safe clock domain transfer from the DQS domain to the ECLK or SCLK domain. The input FIFO is 8-level deep with 3-bit write and read pointers. It transfers the read data from the non-continuous DQS domain to the continuous ECLK. The FIFO is written by the DQS strobe and read back by ECLK which has the identical frequency rate as DQS.

The input FIFO also performs the read leveling function. When each DQS strobe signal and its associated DQ data signals arrive at slightly different time with others to the FPGA, the input FIFO allows the skewed read data to be captured and transferred properly.

Each DQS group has one FIFO control in the DQSBUF block. It distributes the FIFO read/write pointers, WRPNTR [2:0] and RDPNTR [2:0], to each memory IDDR module in the same DQS group. Safe domain crossing between ECLK and SCLK is guaranteed by the ECP5 device hardware design.

#### DDR Input and Output Registers (IDDR/ODDR)

ECP5 devices provide dedicated input DDR (IDDR) and output DDR (ODDR) functions supporting 4:1(X2) gearing modes that are used to implement the DDR memory functions. These automatically handle the transfer of data from ECLK domain to the SCLK (FPGA clock) domain.

## Memory Interface Implementation

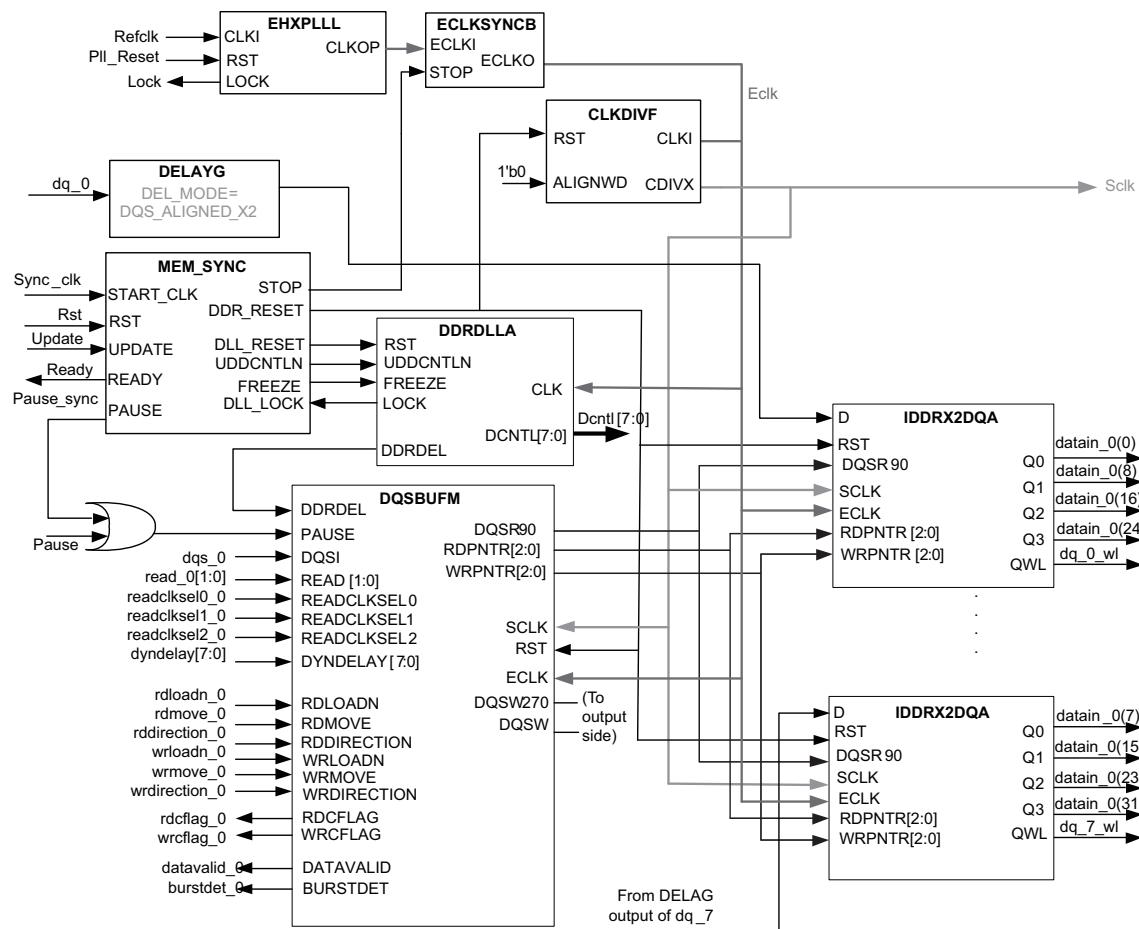
The following sections explain the DDR2, DDR3/DDR3L, LPDDR2, LPDDR3 memory interfaces implementation using the X2 gearing mode. ECP5 devices support these memory interfaces generation through the Clarity Designer tool. Clarity Designer will generate one module that include the Read and Write side implementation shown below.

All of the memory interfaces use DQS clock, one ECLK and one SCLK to implement the read and write side operations. ECLK must always be routed on the edge clock tree and SCLK on the primary clock tree.

### Read Implementation

The read side implementation is shown in Figure 13-31.

**Figure 13-31. DDR2, DDR3/DDR3L, LPDDR2 and LPDDR3 Read side Implementation**



The read side is implemented using the following software elements.

- **DDRX2DQA** element to capture the data
- **DDR DLLA** is used to generate the delay code for **DQSBUFM** to get the 90 degree phase shift on the DQS input (**DQSR90**)
- The incoming DQS clock (**DQSI**) is routed through the **DQSBUFM** module to the DQS clock tree
- The **DQSBUFM** receives the delay code from **DDR DLLA** and generates the delayed DQS signal to **DDRX2DQA**
- The **DQSBUFM** is used to generate the Read and Write pointers that is used to transfer data from the DQS to

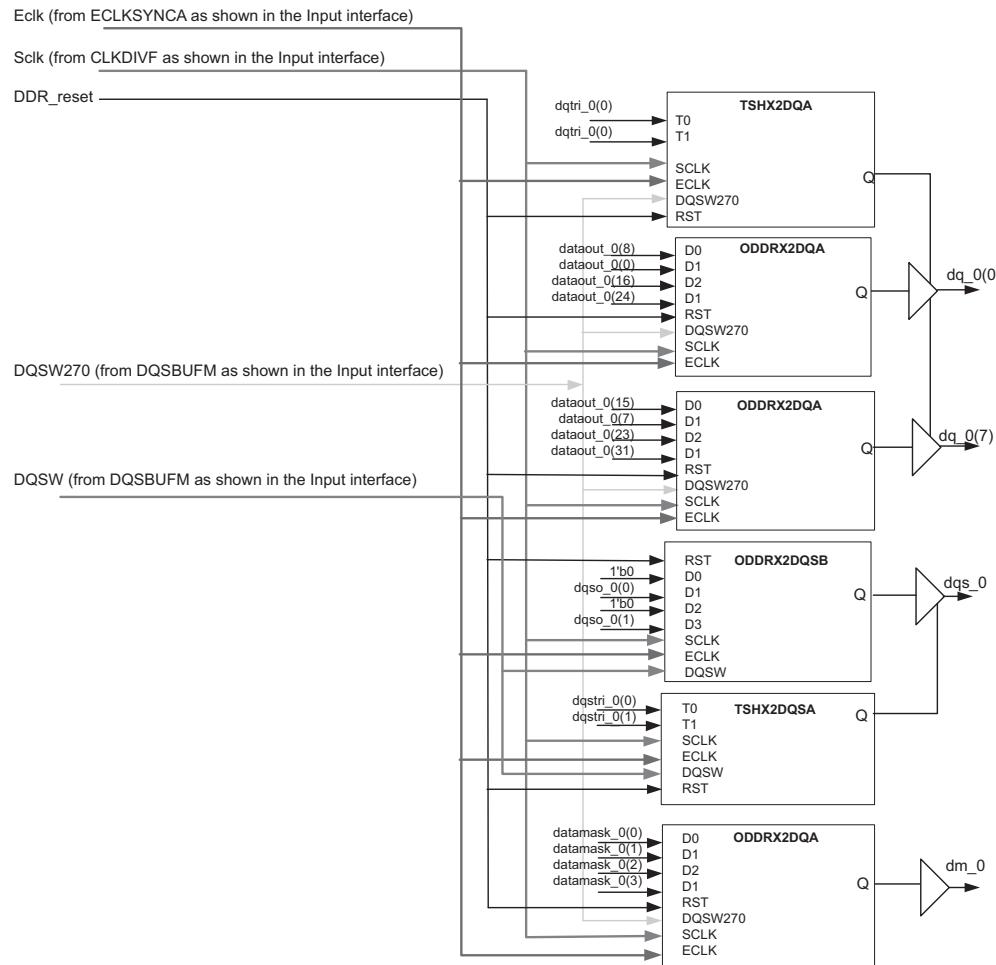
ECLK inside the IDDRX2DQA module

- Read 1, 0 and ReadClksel\_2, 1, 0 signals of DQSBUFM are used by the user logic to obtain the optimal READ pulse position and driven by the user logic to generate a clean DQS output signal based on the trained READ pulse with respect to preamble and postamble
- The dynamic delay control ports are available on the DQSBUFM module when user selects the “enable dynamic margin control” option
- DYNDELAY[7:0] of DQSBUFM is used to perform write leveling. If write leveling is not used, it is connected to “0”.
- Port QWL of IDDRX2DQA is used for DDR3/DDR3L and LPDDR3 to support write leveling. It is used to deliver the write leveling monitor signals from the memory device to the FPGA user logic.
- MEM\_SYNC soft IP must always be included in the interface. It is required to avoid issues on DDR memory bus and update code in operation without interrupting interface operation. When a DDR memory interface IP is generated from Clarity Designer, the MEM\_SYNC soft IP block is also generated and included.
- The Pause\_sync output of the MEM\_SYNC soft IP is used to request DQSBUFM pause for the DDRDLL update and goes to an output port of the Clarity Designer module. The input port Pause\_data goes to DQSBUFM. It is required by user logic, to OR the Pause\_sync output of the MEM\_SYNC module with the user pause to drive the Pause\_data input of the DQSBUFM. This OR would need to be implemented outside of the Clarity Designer module in user’s design.
- CLKDIVF set to divide by 2 function is used to generate the SCLK from the ECLK
- When DDR data bus is required to cross two sides, an ECLKBRIDGECS should be enabled in Clarity Designer. When using ECLKBRIDGECS, there will be two DDRDLLs in the design one for each side. Also the DQSBUFMs used on the second side should be connected to its DDRDLL. Clarity Designer will automatically generate all the required DDRDLLs and DQSBUFMs

#### **Write Implementation (DQ, DQS and DM)**

Figure 13-32 shows the DDR2, DDR3/DDR3L, LPDDR2 and LPDDR3 memory interface write side implementation to generate DQ, DQS and DM outputs.

**Figure 13-32. DDR2, DDR3/DDR3L, LPDDR2 and LPDDR3 Write Side (DQ, DQS and DM)**



This interface uses the following modules:

- ODDRX2DQA to generate the data DQ and DM signals. TSHX2DQA is used to generate the tristate control for the DQ output
- ODDRX2DQSB to generate DQS output. TSHX2DQSA is used to generate the DQS tristate control.
- DQSW270 which is the 270 degree delayed DQS signal is used to generate the DQ and DM outputs
- DQSW is 90 degrees shifted from the DQSW270 is used to generate DQS output
- The DQSW270 and DQSW clocks are generated in the DQSBUFM module shown on the Read side Implementation figure.
- When write leveling is enabled, the dynamic delay for write leveling (DYNDELAY[7:0]) is applied both to DQSW and DQSW270 so that the DQ and DQS phase relationship is maintained.
- ECLK and SCLK are used inside the ODDRX2 module before data is transferred to the DQSW270 and DQSW clocks. The ECLK is generated by the EHXPLL module and the SCLK is generated by the CLKDIVF module, both shown in the Read side implementation.

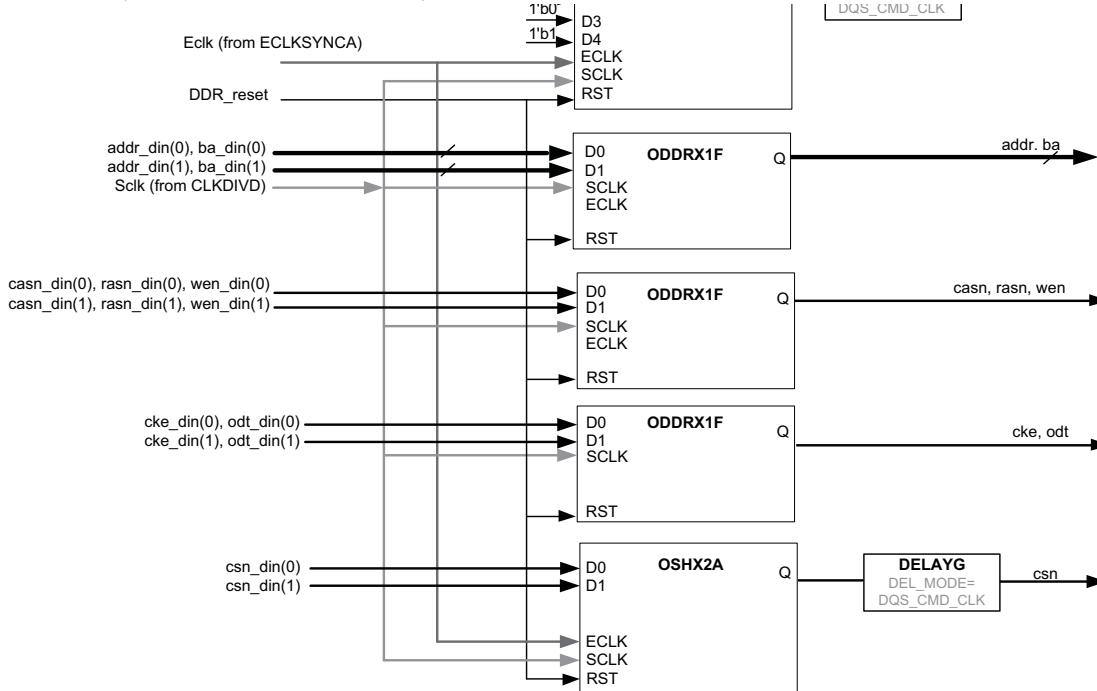
\* Figure 13-32 shows one tristate. The software generates one tristate element for each DQ port

### Write Implementation (DDR2, DDR3/DDR3L Address, Command and Clock)

DDR2, DDR3, DDR3L write side interface side to generate the Clock, Address and Command uses the following modules:

- ODDRX2F with inputs tied to constants to generate the DDRCLK output.
- The ADDR, BA, CASN, RASN, WEN, CKE and ODT command and address signals are generated using the ODDRX1F. CSN output is generated using OSHX2A.
- Both ECLK and SCLK is used in these elements. This is same ECLK and SLCK generated in the Read side.

**Figure 13-33. DDR2, DDR3/DDR3L Address, Command and Clock Generation**

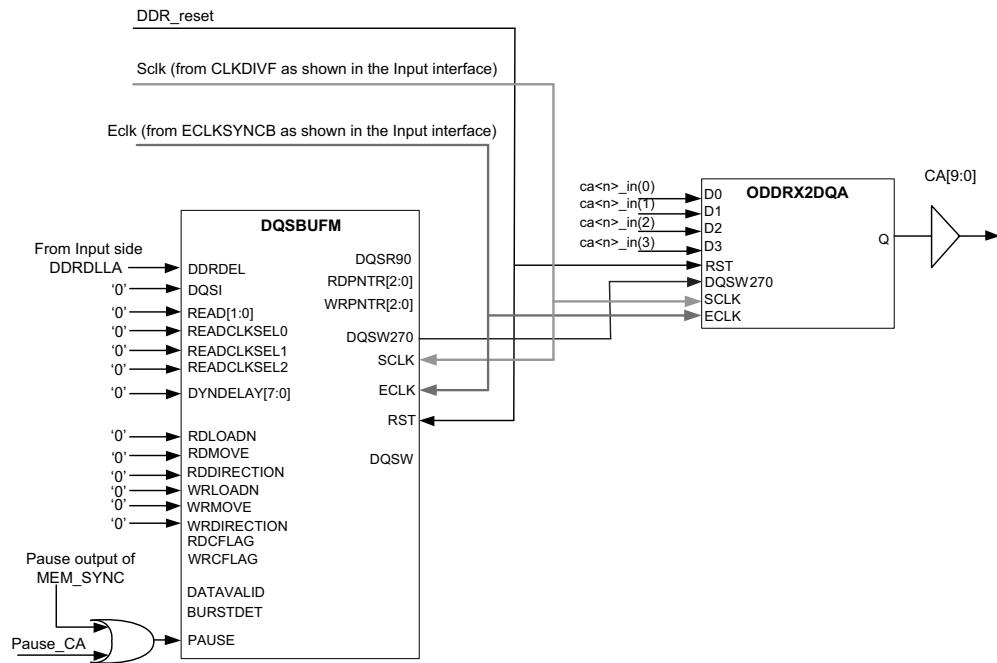


### Write Implementation (LPDDR2 and LPDDR3 Address, Command and Clock)

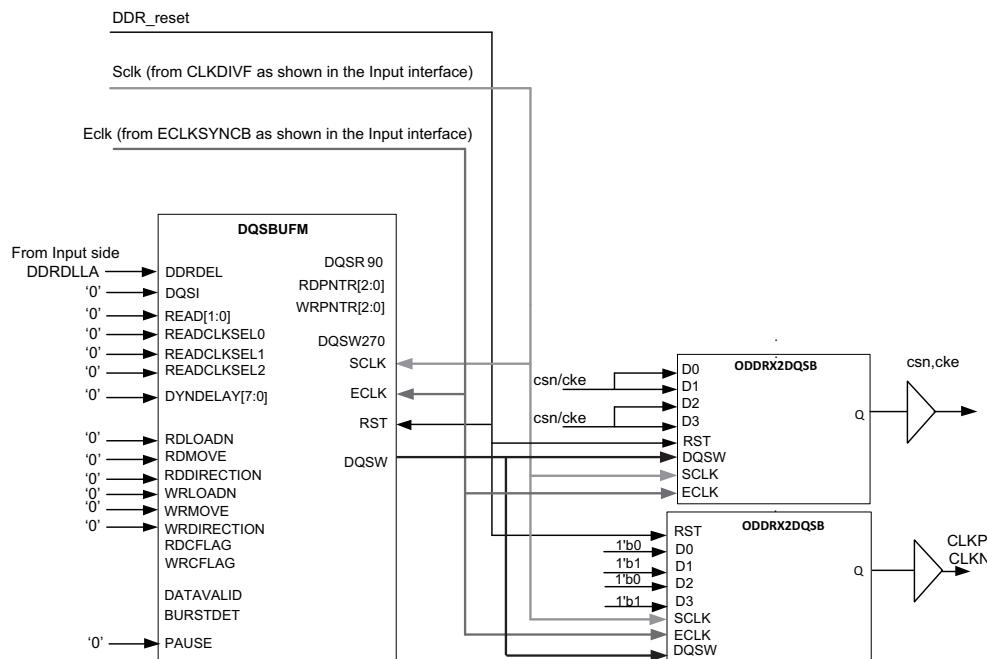
LPDDR2 and LPDDR3 output side interface side to generate the Clock, Address and Command uses the following modules:

- Two DQSBUFM are required generate the LPDDR2 and LPDDR3 address/command and clock signals. One of the DQSBUFM is used for CA output and the other for CKE, CSN, ODT and CLKP/CLKN (note ODT is only for LPDDR3)
- ODDRX2DQA module to generate the CA[9:0] outputs
- ODDRX2DQSB with D0 and D1 tied together and D2 and D3 tied together to generate CSN, CKE signals
- ODDRX2DQSB with inputs tied to “0” and “1” is used to generate the CLKP/CLKN outputs
- On LPDDR3, even the ODT will have D0 and D1 tied together and D2 and D3 tied together and will use same DQSBUFM
- The DQSBUFM used for CA will require a separate input. Ideally user must take Pause\_sync output of the MEM\_SYNC module and make an OR gate with user CA pause to drive the PAUSE input port of DQSBUFM. The Pause\_CA pause is connected to the user's CA training logic PAUSE required. If CA training is not used then user CA pause should be tied to GND.
- Both ECLK and SCLK is used in these elements. This is same ECLK and SLCK generated in the Input Read side module shown above.

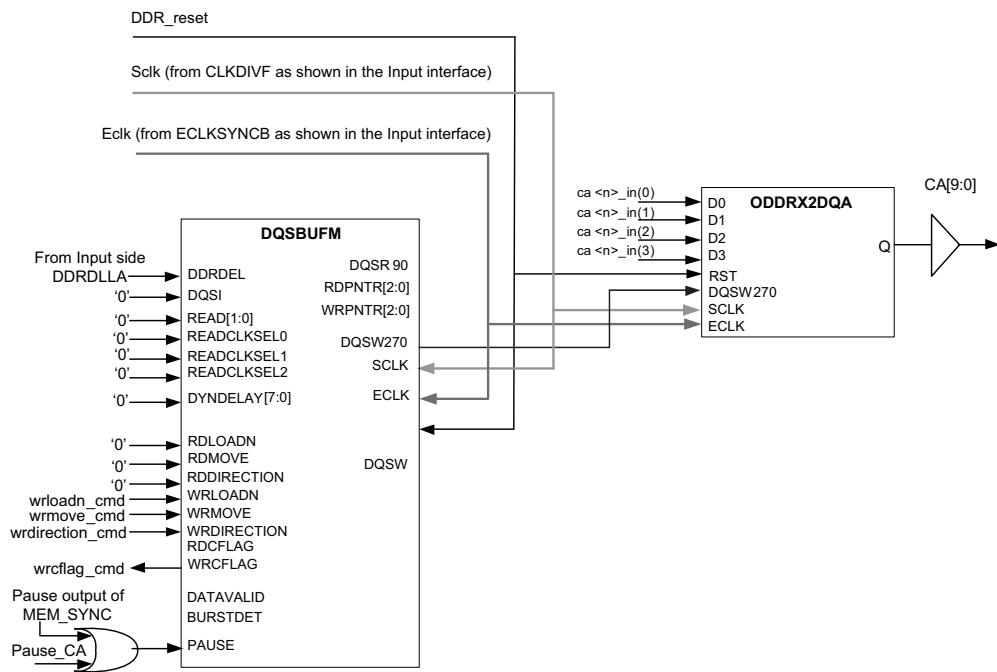
**Figure 13-34. LPDDR2 Output for CA generation**



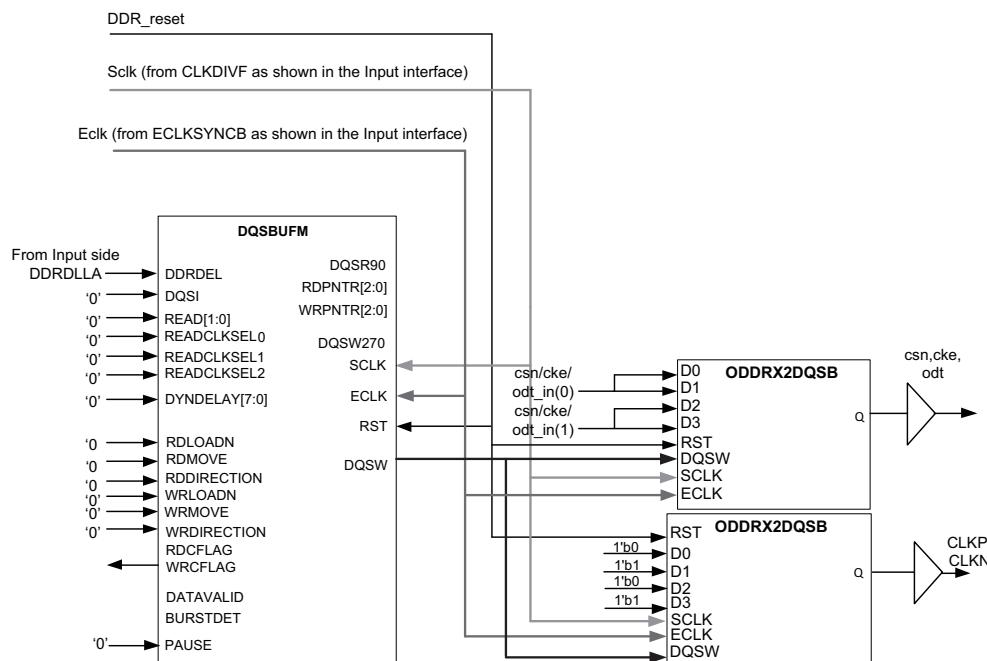
**Figure 13-35. LPDDR2 Output for CSN, CKE and CLOCK generation**



**Figure 13-36. LPDDR3 Output side for CA Generation**



**Figure 13-37. LPDDR3 Output side for CSN, CKE, ODT and CLOCK generation**



## **DDR Memory Interface Design Rules and Guidelines**

Listed below are some rules and guidelines to keep in mind when implementing DDR memory interfaces in ECP5 devices. ECP5 devices have dedicated DQS banks with the associated DQ pads.

- The left and right sides of an ECP5 device share an identical I/O structure. All of the memory interfaces can be implemented on these sides.
- The Top side of the device does not support DQSBUF blocks hence will not support DDR memory interfaces. Although some of the ADDR/CMD generation that uses ODDRX1 modules can be placed on the top side of the device for DDR2, DDR3 and DDR3L.
- DDRDLLA primitive should be instantiated for all DDR memory interfaces. Each DDRDLLA generates 90° digital delay code for all the connected DQS delay blocks based on the reference clock input to the DDRDLLA. Therefore, all the DDR memory interfaces under the same DDRDLLA coverage must run at the same frequency.
- There are four DDRDLLs on each device, one DDRDLL in each corner of the device. Each DQSBUF module can receive delay codes from either of the DDRDLLs in each top and bottom corner of the device. This would be an exception for the smallest device where there are only two DDRDLLs on the device.
- When a DDR memory interface is added to the side where another DDR memory interface is running at a different frequency, another available DDRDLLA for the side must be instantiated and used for the new interface.
- The reference clock input to the PLL used in the DDR memory interface implementation must be located to the dedicated PLL pin or a PCLK pin. The dedicated PLL input pin is preferred due to less skew.
- Each DDR memory interface must use its corresponding I/O standard.
  - For the DDR2 memory interface, the interface signal should use the SSTL18 I/O standards.
  - For the DDR3 memory interface, these signals should be connected to the SSTL15 standards.
  - For the DDR3L memory interface, these signals should be connected to the SSTL135 standards.
  - For the LPDDR2 and LPDDR3 memory interface, the interface signal should use the HSUL12 standard.
  - DDR3, DDR3L, LPDDR2 and LPDDR3 memory interfaces also requires differential DQS signals. The use of differential DQS is optional for DDR2.

Table 13-5 shows the IO\_TYPE setup for each of the DDR memory interfaces.

**Table 13-5. I/O Standards for DDR Memory**

	<b>DDR2</b>	<b>DDR3</b>	<b>DDR3L</b>	<b>LPDDR2</b>	<b>LPDDR3</b>
DQ	SSTL18_I, SSTL18_II	SSTL15_I, SSTL15_II	SSTL135_I, SSTL135_II	HSUL12	HSUL12
DQS	SSTL18_I, SSTL18_II, SSTL18D_I, SSTL18D_II	SSTL15D_I, SSTL15D_II	SSTL135D_I, SSTL135D_II	HSUL12D	HSUL12D
Cmd/Addr	SSTL18_I, SSTL18_II	SSTL15_I, SSTL15_II	SSTL135_I, SSTL135_II	HSUL12	HSUL12
CK	SSTL18D_I, SSTL18D_II	SSTL15D_I, SSTL15D_II	SSTL135D_I, SSTL135D_II	HSUL12D	HSUL12D

- When implementing the DDR interface, the VREF1 of the bank is used to provide the reference voltage for the interface pins.

## DDR2/DDR3 Memory Interface Termination Guidelines

(These updates are still preliminary. We will need to update again once the whole validation processes are completed.)

Proper termination of a DDR memory interface is an important part of implementation that ensures reliable data transactions at high speed. Below is the general termination guideline for the ECP5 device DDR memory interface.

### Termination for DQ, DQS and DM

- Do not locate any termination on the memory side. The memory side termination on DQ, DQS and DM is dynamically controlled by the DDR3 SDRAM's ODT function.
- Do not locate any termination on the FPGA side. The ECP5 device has internal termination on DQ and DQS, which is dynamically controlled. Use the TERMINATION preference for DQ and DQS pads to enable the internal parallel termination to VCCIO/2. The TERMINATION preference has the OFF, 50-, 60-, and 75-ohm options. (Recommended setting for each interface TBD)

### Termination for CK

DDR memory clocks require differential termination because they use a differential signaling. Use SSTL15D in DDR3 or SSTL18D in DDR2 to drive the clock signals. You can locate an effective 100-ohm termination resistance on the memory side to achieve the differential termination using the following guideline:

- Locate a 100-ohm resistor between the positive and negative clock signal, OR
- Connect one end of an Rtt resistor to the positive pin and one end of another Rtt to the negative pin of a CK pair, then connect the other ends of two Rtt resistors together and return to VDD or GND through a Ctt capacitance. Note that the JEDEC CK termination scheme defined in the DIMM specifications uses 36-ohm for Rtt with 0.1 $\mu$ F Ctt for DIMM for DDR3 DIMMs returning to VDD. 50-ohm Rtt can also be used for non-DIMM applications.
- Use of series termination resistors at the FPGA side is not recommended.

When fly-by wiring is used in DDR3, the CK termination resistor should be located after the last DDR3 SDRAM device.

### Termination for Address, Commands and Controls

- Parallel termination to VTT on address, command and control lines is typically required at the DDR2/DDR3 and DDR3L memory side:
- Locate a 50-ohm parallel-to-VTT resistor (or a best known resistance obtained from your SI simulation) to each address, command and control line on the memory side.
- Series termination resistors can be optionally used on the address, command and control signals to suppress overshoot/undershoot and to help decrease overall SSO noise level. 22-ohm or 15-ohm series termination is recommended when used.
- When fly-by wiring is used in DDR3, the address, command and control termination resistors should be located after the last DDR3 SDRAM device.)

No termination is required on the LPDDR2 and LPDDR3 CA bus and control lines. They use point-to-point connections.

### Termination for DDR3/DDR3L DIMM

The DDR3 DIMMs incorporate internal termination following the requirements defined by the JEDEC DIMM specification. For this reason, the user termination requirement for the DDR3 DIMM is slightly different from that of DDR3 SDRAM devices:

- Do not locate any termination on the memory side. The memory side termination on DQ, DQS and DM is dynamically controlled by the DDR3 SDRAM's ODT function.

- Do not locate differential termination on CK at the memory side because the DIMM already has termination on the module.
- Do not locate parallel termination to VTT on address, command and control signals at the memory side because the DIMM already has termination on the module.
- Follow the termination for DQ, DQS and DM guideline above for the FPGA side termination.

## DDR Memory Interface Pinout Guidelines

The ECP5 device contains dedicated I/O functions for supporting DDR memory interfaces. The following pinout rules must be followed to properly use the dedicated I/O functions.

- The DQS-DQ association rule must be followed.
- All associated DQs (8 or 4) to a DQS must be in the same DQS group.
- A data mask (DM) must be part of the corresponding DQS group.  
Example: DM[0] must be in the DQS-16 group that has DQ[7:0], DQS[0].
- A DQS pad must be allocated to a dedicated DQS True (+) pad.
- A DQS# pad is auto-placed when a differential SSTL type (SSTL15D in DDR3, SSTL18D in DDR2, SSTL135D in DDR3L and HSUL12D in LPDDR2/LPDDR3) is selected.
- Do not assign any signal to a DQS# pad if used as differential strobe. The software automatically places DQS# when a differential I/O type is applied.
- DQS/DQS# pads can be used for other DDR functions. For example, DQS# can be used as a DQ pad for a non-differential DQS interface such as DDR2 with single ended strobe. However, a DQS signal must use the DQS/DQS# pads only.
- Data group signals (DQ, DQS, DM) can use any of the left and right sides of the ECP5 device as long as they keep the DQS-DQ association rule.
- It is recommended that the CK/CK# outputs be located on the same side where the DQ and DQS pads are located to minimize the skew.
- Place the address, command and control signals either on the same side as where the DQ and DQS pads are located or they can be placed on the top side for DDR2, DDR3 and DDR3L memory interfaces.
- In DDR3 interface the RST# can be located anywhere an output is available as long as the same I/O Standard as the memory interface is applicable.
- The input reference clock to the PLL must be assigned to use dedicated clock routing. The dedicated PLL input pads are recommended while PCLK inputs can also be used.
- VREF1 of the bank where the DQ, DQS and DM pads are located must be available to be used as a reference voltage input.
  - Do not assign an I/O signal to VREF1 in the preference file. VREF1 for the bank has to be available for the DDR memory interface.
  - Unused VREF1 can be taken as a general purpose I/O in the bank where no DQ/DQS pad is located.

## Pin Placement Considerations for Improved Noise Immunity

In addition to the general pinout guidelines, you will need to pay attention to additional pinout considerations to minimize simultaneous switching noise (SSN) impact. The following considerations are generally necessary to control SSN within the required level:

- a. Properly terminated interface
- b. SSN optimized PCB layout
- c. SSN considered I/O pad assignment
- d. Use of pseudo power pads

The guidelines listed below address the I/O pad assignment and pseudo power pad usage. Unlike the pinout guidelines, they are not absolute requirements. However, it is recommended that the pin placement follow the guidelines as much as possible to increase the SSO/SSI immunity.

- Place the DQS groups for data implementation starting from the middle of the (right or left) edge of the ECP5 device. Allow a corner DQS group to be used as a data group only when necessary to implement the required width.
- Locate a spacer DQS group between the data DQS groups if possible. A DQS group becomes a spacer DQS group if the I/O pads inside the group are not used as data pads (DQ, DQS, DM).
  - In DDR2, DDR3 and DDR3, the pads in a spacer group can be used for address, command, control or CK pads as well as for user logic or the pseudo power pads.
  - It would provide better noise immunity if no more than two data DQS groups are consecutively placed. If more data DQS groups need to be placed consecutively, use the pseudo power pads as many as possible to isolate each DQS group more effectively from others.
- It is recommended that you locate a few pseudo VCCIO/ground (GND) pads inside a spacer DQS group and at least one pseudo VCCIO in the data DQS group. An I/O pad becomes a pseudo power pad when it is configured to OUTPUT with its maximum driving strength (i.e., SSTL15, 10mA for DDR3) and connected to the external VCCIO or ground power source on the PCB.
  - Your design needs to drive the pseudo power I/O pads according to the external connection. (i.e., you assign them as OUTPUT and let your design drive '1' for pseudo VCCIO pads and '0' for pseudo GND pads in your RTL coding.)
  - Locating two to four pseudo power pads in a spacer DQS group should be sufficient to provide suppressing the SSN impact.
  - Locate a pseudo power pad in a location where it can provide the best balanced and isolated separation.
- You may have one or more remaining pads in a data DQS group which are not assigned as a data pad in a DDR memory interface. Assign them to pseudo VCCIO or pseudo GND. Preferred location is in the middle of the group (right next to a DQS pad pair) if the DQS group is isolated by a spacer DQS group. If consecutively placed, locating the pseudo power pads to the edge of the group may be more effective. Note that you may not have this extra pad if the DQS group has 12 pins only and includes a VREF pad for the bank.

The additional guidelines below are not as effective as the ones listed above. However, following them is still recommended to improve the SSN immunity further:

- Assign the DM (data mask) pad in a data DQS group close to the other side of DQS pads where a pseudo power pad is located. If the data DQS group includes VREF1, locate DM to the other side of VREF with respect to DQS. It can be used as an isolator due to its almost static nature in most applications.
- Other DQS groups (neither data nor spacer group) can be used for accommodating DDR memory interface's address, command, control and clock pads.
- You can assign more unused I/O pads to pseudo power if you want to increase the SSN immunity. Note that the SSN immunity does not get increased at the same rate as the increased number of pseudo power pads. The first few pseudo power pad placements described above are more crucial. Keep the total pseudo power pad ratio (VCCIO vs. GND) between 2:1 to 3:1.
- It is a good idea to shield the VREF pad by locating pseudo power pads around it if extra pins are available in the bank where the VREF1 pad is not located.

## Using Clarity Designer to Build and Plan High Speed DDR Interfaces

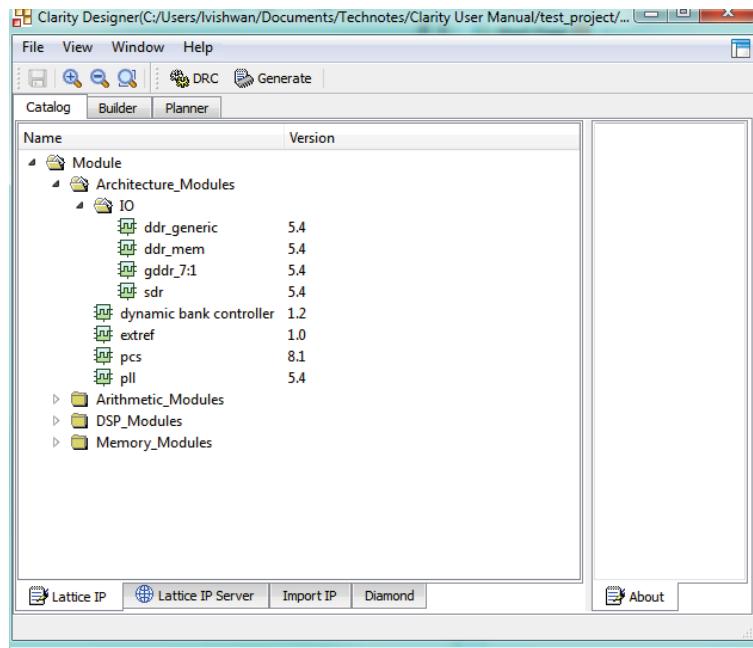
The Clarity Designer tool is used to configure, build and plan placement all DDR interfaces. This section will cover how Clarity Designer is used to configure and plan placement for the DDR interfaces. In addition to building and planning Clarity Designer can be used to build top level modules by connecting the modules together in the Builder.

For step by step assistance with Clarity, please refer to the [Clarity Designer User Manual](#).

Clarity Designer can be opened from the Tools Menu in Project Navigator. Figure 13-38 shows a Clarity Design Project which includes following three tabs:

- Catalog – Used to Configure the DDR Modules
- Builder – Used to Build the HDL file that includes all the DDR modules configured
- Planner – Used to Plan the Placement of the various DDR Interfaces

**Figure 13-38. Clarity Design Main Window**



*Note: It is recommend that all the DDR modules required for the current design be generated in the same Clarity Design project. This will allow for the Design Rule Checking as well as Resource Conflict Checking for all the modules at the same time.*

### Configuring DDR Modules in Clarity Designer

The catalog section of Clarity Design lists all the DDR architecture modules available on ECP5.

All the DDR modules are located under Architecture Modules – IO. This includes:

- **SDR** – Select to build SDR Modules
- **DDR\_GENERIC** – Select to build any DDR Generic Receive and Transmit Interfaces
- **GDDR\_7:1** – Select to build 7:1 LVDS Receiver and Transmit Interface
- **DDR\_MEM** – Select to build DDR Memory Interfaces

To see the detailed block diagram for each interface generated by Clarity Designer see the [High-Speed DDR Interface Details](#) section.

## Configuring SDR Modules

To build and SDR interface, select **SDR** option under Architecture Modules – IO in the Catalog tab of Clarity Designer. Enter the name of the module. Figure 13-39 shows the type of interface selected as **SDR** and module name entered. This module can then be configured by clicking the **Customize** button.

**Figure 13-39. SDR Option Selected in the Catalog Tab of Clarity Designer**

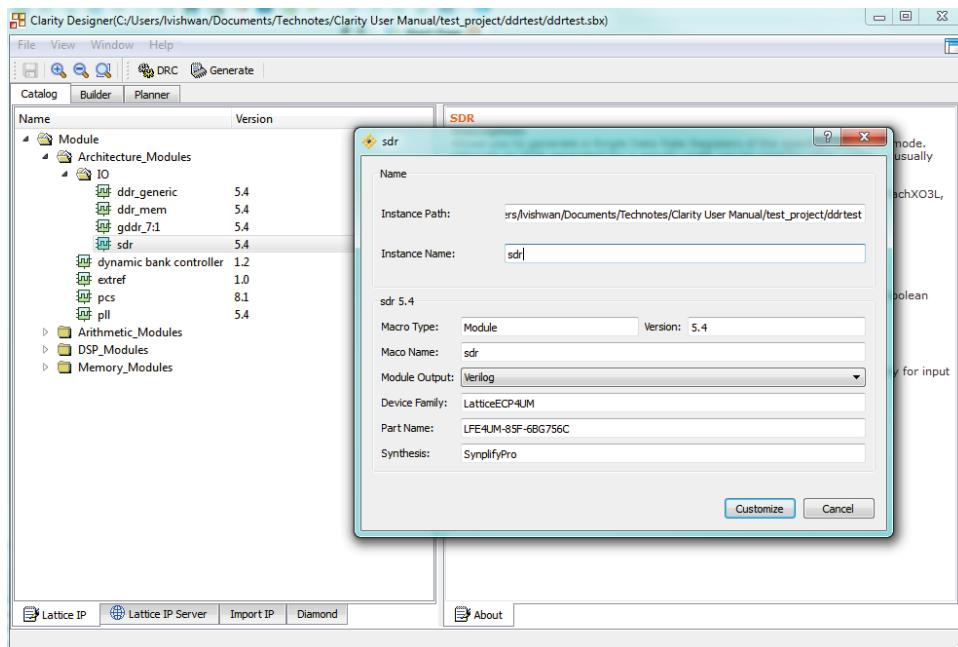


Figure 13-40 shows the Configuration Tab for SDR module. User can make selections and then click **Configure**.

**Figure 13-40. SDR Configuration Tab**

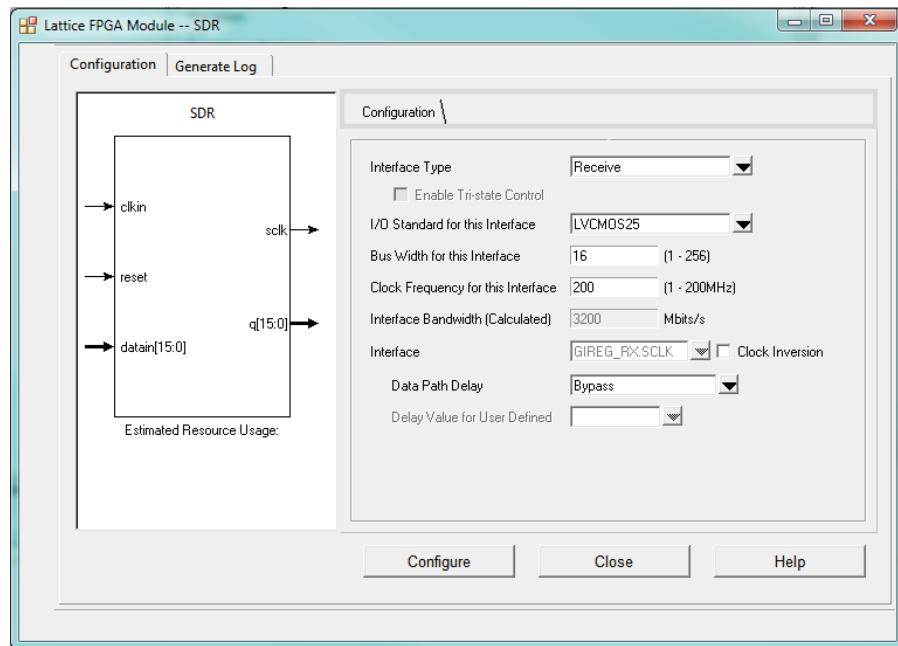


Table 13-6 explains the various configurations options available for SDR modules.

**Table 13-6. SDR Configuration Parameters**

GUI Option	Description	Values	Default
Interface Type	Type of Interface (Transmit or Receive)	Transmit, Receive	Receive
I/O Standard for this Interface	I/O Standard to be used for the interface	All Valid IO_TYPES	LVCMOS25
Bus Width for this Interface	Bus size for the interface	1 – 256	16
Clock Frequency for this Interface	Interface Speed	1 - 200	200
Bandwidth (Calculated)	This is the calculated from the Clock frequency entered	(Calculated)	(calculated)
Interface	Interface selected based on previous entries	Transmit: GOREG_TX.SCLK Receive: GIREG_RX.SCLK (default)	GIREG_RX.SCLK
Clock Inversion	Option to invert the clock Input to the IO Register	DISABLED, ENABLED	DISABLED
Data Path Delay <sup>1</sup>	Data input can be optionally delayed using the DELAY block	If Interface Type= Receive then:  Bypass, Static Default Dynamic Default Static User Defined Dynamic User Defined  If Interface Type= Transmit then:  Bypass, Static User Defined Dynamic User Defined	Bypass

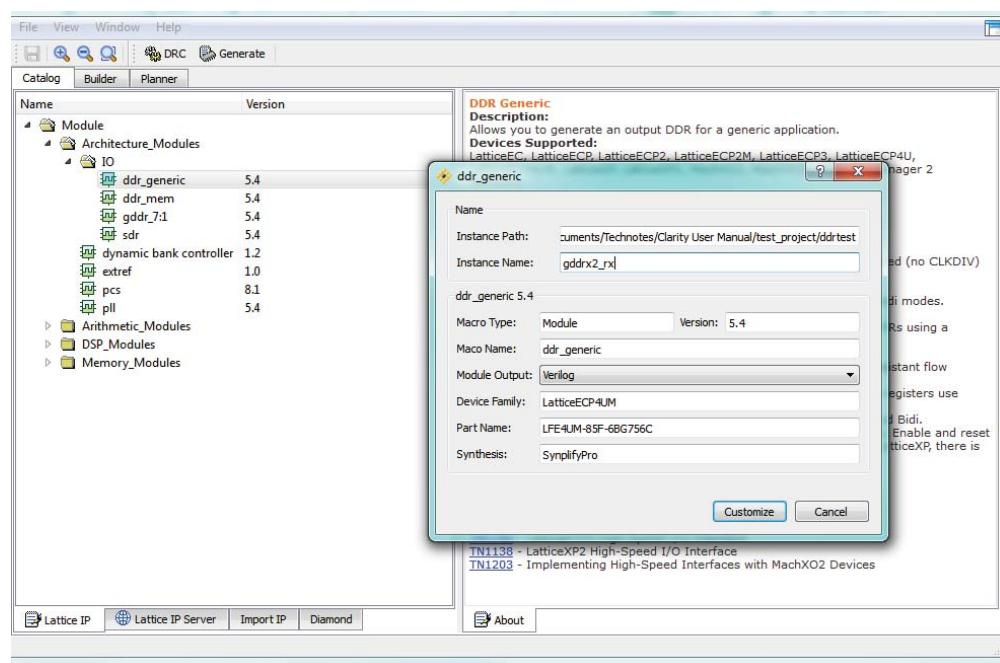
GUI Option	Description	Values	Default
FDEL for User Defined <sup>2</sup>	If Delay type selected above is "user defined", delay values can be entered with this parameter	0 to 127	0

1. Notes: When Data Path Delay value is
  - a.Bypass: No delay cell is used
  - b.Static Default: Static delay element DELAYG is used with attribute DEL\_MODE set to SCLK\_ZEROHOLD
  - c.Static User Defined: Static delay element DELAYG is used with attribute DEL\_MODE set to USER\_DEFINED
  - d.Dynamic Default: Dynamic delay element DELAYF is used with attribute DEL\_MODE set to SCLK\_ZEROHOLD
  - e.Dynamic User Defined: Dynamic delay element DELAYF is used with attribute DEL\_MODE = USER\_DEFINED
2. FDEL is a fine delay value that is additive. The delay value for a FDEL can be found in DS1044, [ECP5 Family Data Sheet](#).

## Configuring DDR Generic modules

To build a DDR Generic interface, select the **DDR\_Generic** option under Architecture Modules – IO in the Catalog tab of Clarity Designer. Enter the name of the module. Figure 13-41 shows the type of interface selected as **DDR\_Generic** and module name entered. This module can then be configured by clicking the **Customize** button.

**Figure 13-41. DDR\_Generic Option Selected in the Catalog Tab of Clarity Designer**



When clicking **Customize**, DDR modules have a Pre-Configuration Tab and a Configuration Tab. The Pre-Configuration tab allows the user to enter information about the type of interface to be built. Based on the entries in the Pre-Configuration Tab, the Configuration Tab is populated with the best interface selection. The user can also, if needed, override the selection made for the interface in the Configuration Tab and customize the interface based on the design requirement.

**Figure 13-42. DDR\_Generic Pre-Configuration Tab**

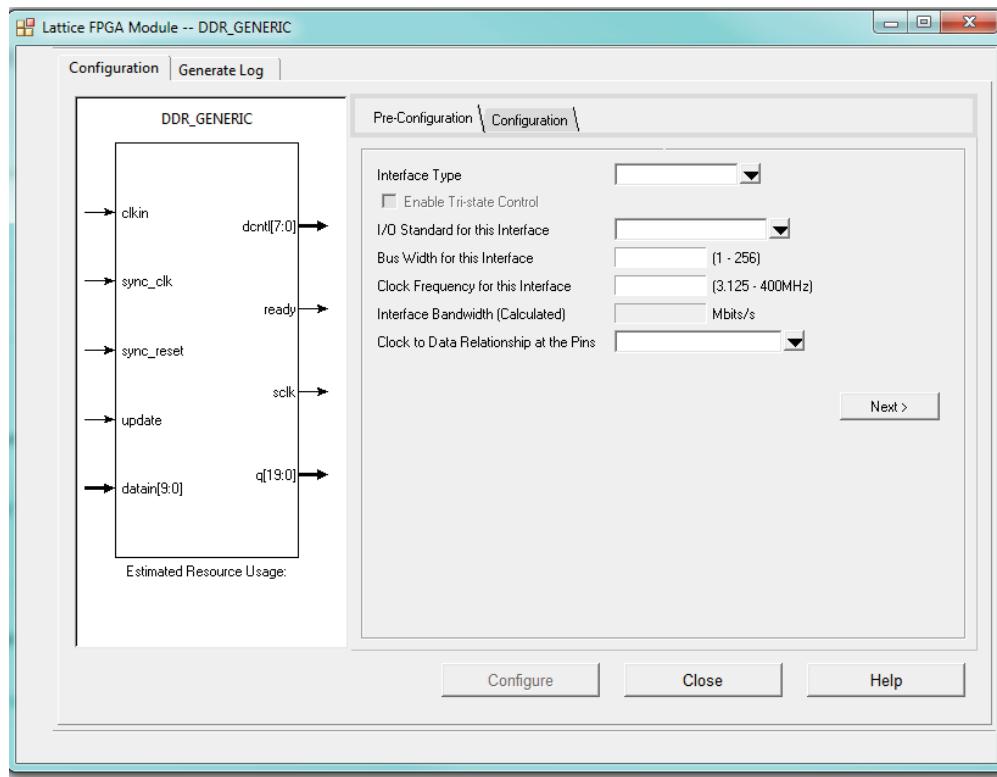


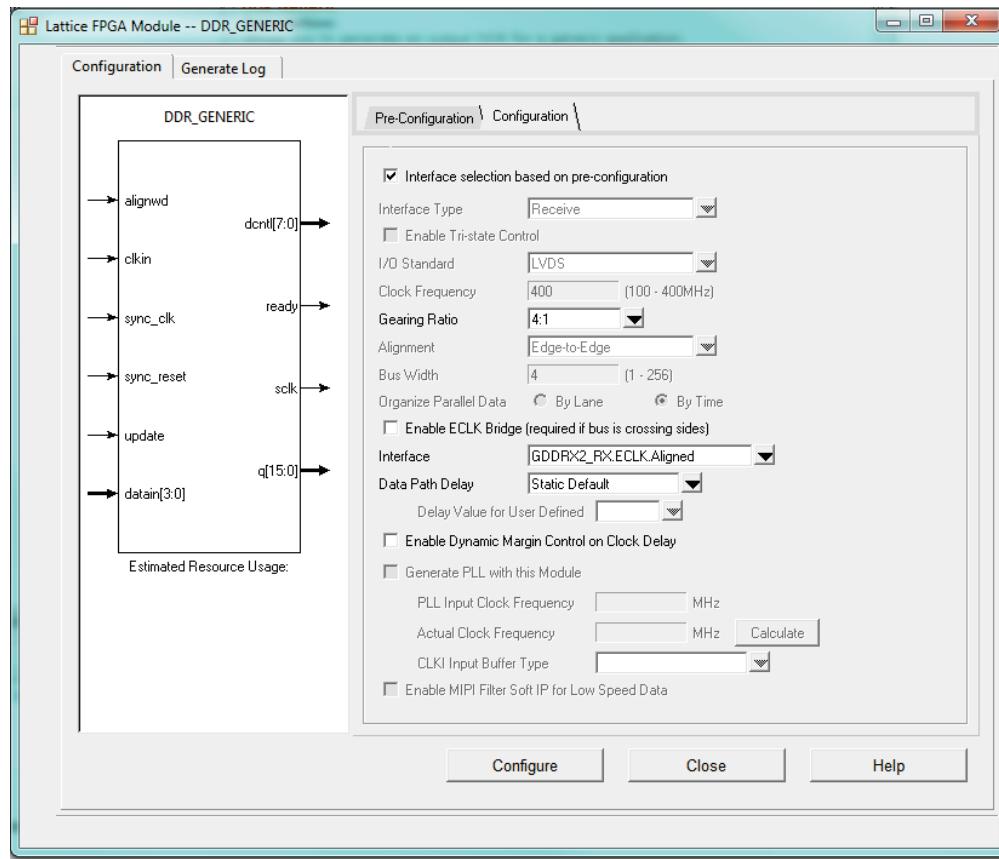
Figure 13-42 shows the Pre-Configuration Tab for DDR generic interfaces. The Table 13-7 explains the various parameters in this tab.

**Table 13-7. DDR\_Generic Pre-Configuration Parameters**

GUI Option	Range
Interface Type	Transmit, Receive, Receive MIPI
I/O Standard for this Interface	List of Legal Input or Output Standards
Enable Tristate Control	Enabled, Disabled
Bus Width for this Interface	1 – 256
Clock Frequency for this Interface	3.125 – 400 MHz 200 – 400 MHz (or Receive MIPI)
Interface Bandwidth (Calculated)	Clock Frequency for * 2 * Bus Width
Clock to Data Relationship at the Pins	Edge-to-Edge, Centered Centered (for Receive MIPI)

Based on the selections made in the Pre-Configuration Tab, the Configuration Tab is populated with the selections. Figure 13-43 shows the Configuration Tab for the selection made in Pre-Configuration Tab.

**Figure 13-43. DDR\_Generic Configuration Tab**



The check box on the top of this tab indicates that the interface is selected based on entries in the Pre-Configuration Tab. The user can choose to change these values by disabling this entry. The best suitable interface is picked based on the selections made in the Pre-Configuration tab.

Table 13-8 explains the various parameters in the Configuration Tab

**Table 13-8. DDR\_Generic Configuration Tab Parameters**

GUI Option	Description	Values	Default Value
Interface selection based on pre-configuration	Indicates interface is selected based on selection made in the Pre-configuration tab. Disabling this checkbox would allow users to make changes if needed.	ENABLED, DISABLED	ENABLED
Interface Type	Type of interface (Receive or Transmit)	Transmit, Receive, Receive MIPI	Receive
Enable Tristate Control	Generate Tristate control for Transmit Interfaces	ENABLED, DISABLED	DISABLED
I/O Standard	I/O Standard used for the interface	All Legal Input & Output standards	LVCMS25
Clock Frequency	Speed of the Interface	100 MHz – 400 MHz (Transmit) 3.125 MHz – 400 MHz (Receive) 200 MHz – 400 MHz (Receive MIPI)	200 MHz

GUI Option	Description	Values	Default Value
Gearing Ratio	DDR register gearing ratio	2:1, 4:1	2:1
Alignment	Clock to Data alignment	Edge-to-Edge or Centered	Centered
Bus Width	Bus width for each interface.	1 – 256	10
Organize Parallel Data	Allows user to select how the data bits of the parallel bus should be arranged. You can choose to set it <b>By Lane</b> where all the parallel data bits from each lane are organized together in the data output. If <b>By Time</b> is chosen instead, a single bit from each of the data lanes is put together in the data output.	By Lane, By Time	By Time
Enable ECLK Bridge (required if bus is crossing sides)	This is required if the data bus is wide and will be crossing sides. Enabling this option will instantiate the ECLK Bridge module in the HDL	Enable, Disable	Disable
Interface	Shows list of all valid high speed Interfaces for the given configuration	See Table below to see interfaces available for a given configuration.	
Data Path Delay	Data input can be optionally delayed using the DELAY block. Default Value is selected based on Interface Type.	If Interface Type = Receive: Static Default, Dynamic Default, Static User Defined, Dynamic User Defined  If interface type = Receive MIPI Static Default Static User Defined  If Interface Type = Transmit: Bypass, Static User Defined, Dynamic User Defined	If Interface Type= Receive: Static Default  If Interface Type= Transmit: Bypass
Delay Value for User Defined	When Data Path Delay of user defined is selected, the user will also need to set the number of delay steps to be used	1 – 127	1
Enable Dynamic Margin Control on Clock Delay	Allows dynamic user control on clock phase shift for Receiver edge to edge aligned interfaces	Enable, Disable	Disable
Generate PLL with this module	When this option is enabled for Transmit interfaces, the PLL used to generate the clocks is included in the generated module	Enable, Disable	Disable
PLL Input Clock Frequency	Frequency of the clock used as PLL Input	10 MHz – 400 MHz	

GUI Option	Description	Values	Default Value
Actual Clock Frequency	Displays the achieved PLL output clock frequency	Actual PLL output Frequency achieved based on interface requirement	
CLKI Input Buffer Type	The I/O Standard for the PLL Reference Clock	List of Legal Input Standards, None (if coming from fabric)	LVCMS25
Enable MIPI Filter Soft IP for Low Speed Data	Generates the MIPI Filter soft IP in module for Interface = Receiver MIPI	Enable, Disable	Disable

The table below shows how the interfaces are selected by Clarity Designer based on the selections made in the Pre-Configuration Tab.

**Table 13-9. Clarity Designer DDR\_Generic Interface Selection**

Interface Type	Gearing Ratio	Alignment	Default Interface
Receive	2:1	Edge-to-Edge	GDDR1_RX.SCLK.Aligned
Receive	2:1	Centered	GDDR1_RX.SCLK.Centered
Receive	4:1	Edge-to-Edge	GDDR2_RX.ECLK.Aligned
Receive	4:1	Centered	GDDR2_RX.ECLK.Centered
Receive MIPI	4:1	Centered	GDDR2_RX.ECLK.MIPI
Transmit	2:1	Edge-to-Edge	GDDR1_TX.SCLK.Aligned
Transmit	2:1	Centered	GDDR1_TX.SCLK.Centered
Transmit	4:1	Edge-to-Edge	GDDR2_TX.ECLK.Aligned
Transmit	4:1	Centered	GDDR2_TX.ECLK.Centered

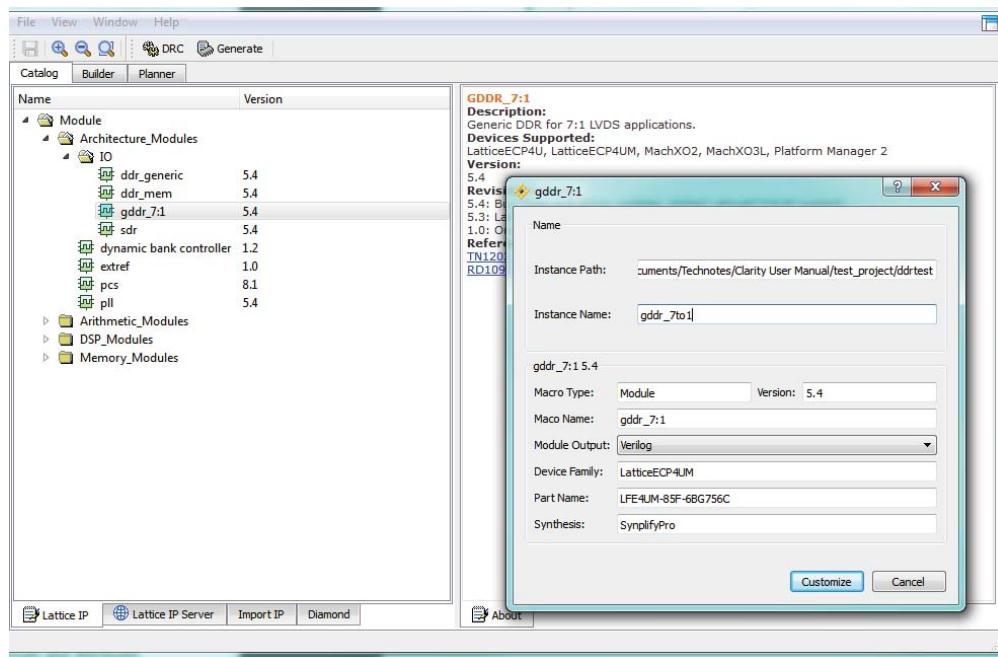
Refer to the [High-Speed DDR Interface Details](#) section to see implementation details for each of these interfaces.

## Configuring 7:1 LVDS Interface Modules

To build a 7:1 LVDS DDR interface, select **GDDR\_7:1** option under Architecture Modules – IO in the Catalog Tab of Clarity Designer. Enter the name of the module.

Figure 13-44 shows the type of interface selected as “GDDR\_7:1” and module name entered. This module can then be configured by clicking the Customize button.

**Figure 13-44. GDDR\_7:1 Option Selected in the Catalog Tab of Clarity Designer**



Clicking **Customize** displays the Configuration Tab where the 7:1 LVDS interface can be configured. Figure 13-45 shows the “Configuration” tab for 7:1 LVDS interfaces.

**Figure 13-45. GDDR\_7:1 LVDS Configuration Tab**

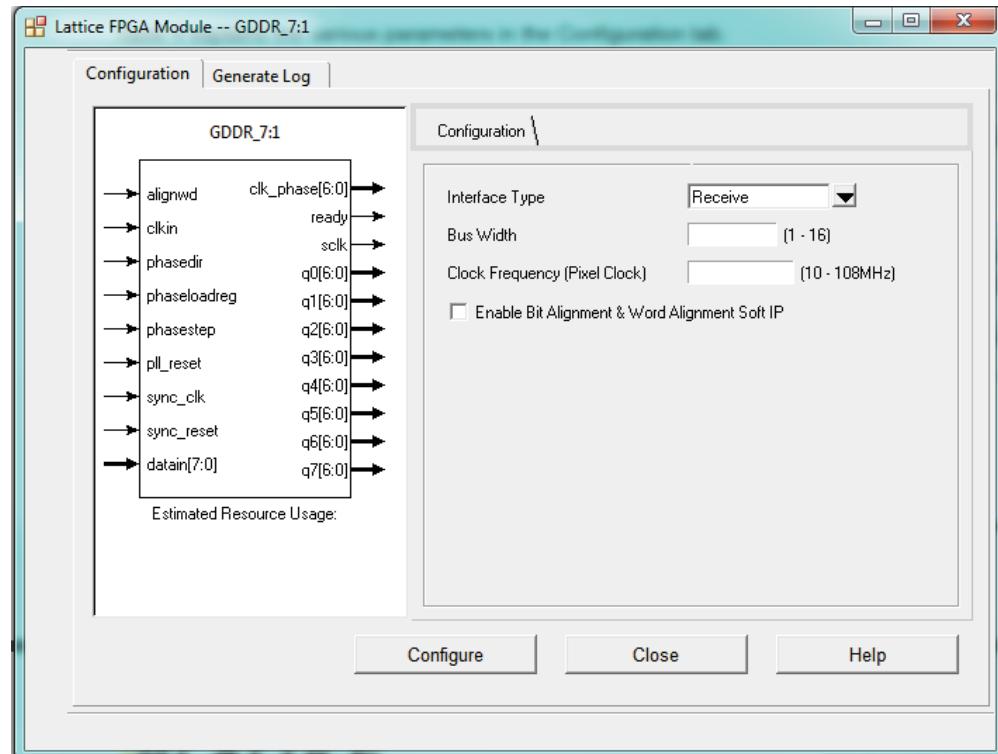


Table 13-10 explains the various parameters in this tab.

**Table 13-10. GDDR\_7:1 LVDS Configuration Parameters**

GUI Option	Description	Values
Interface Type	Type of interface (Receive or Transmit)	Transmit, Receive
Bus Width	Bus width for 1 channel of 7:1 LVDS interface	1 – 16
Clock Frequency	Pixel clock speed	3.125 MHz – 108 MHz
Enable Bit Alignment & Word Alignment	Soft IP included with the module to implement Bit and Word alignment for the parallel data on the 7:1 LVDS Receive Interface	Enable, Disable

## Configuring DDR Memory Interfaces

Clarity Designer is used to configure the PHY portion of the DDR2, DDR3, DDR3L, LPDDR2 and LPDDR3 memory interfaces. For the detailed block diagram for each interface, see the [Memory Interface Implementation](#) section.

To build a DDR Memory interface, select **DDR\_MEM** option under Architecture Modules – IO in the Catalog Tab of Clarity Designer. Enter the name of the module.

Figure 13-46 shows the type of interface selected as “GDDR\_MEM” and module name entered. This module can then be configured by clicking the Customize button.

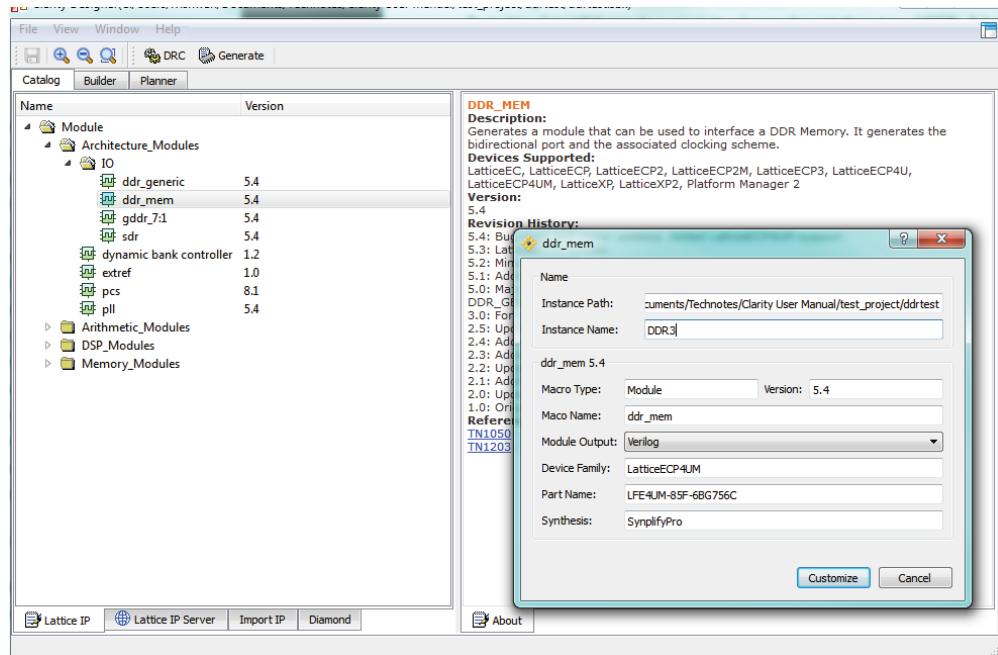
**Figure 13-46. DDR\_MEM Option Selected in the Catalog Tab of Clarity Designer**


Figure 13-47 shows the Configuration Tab for the DDR\_MEM interface.

**Figure 13-47. DDR\_MEM Configuration Tab**

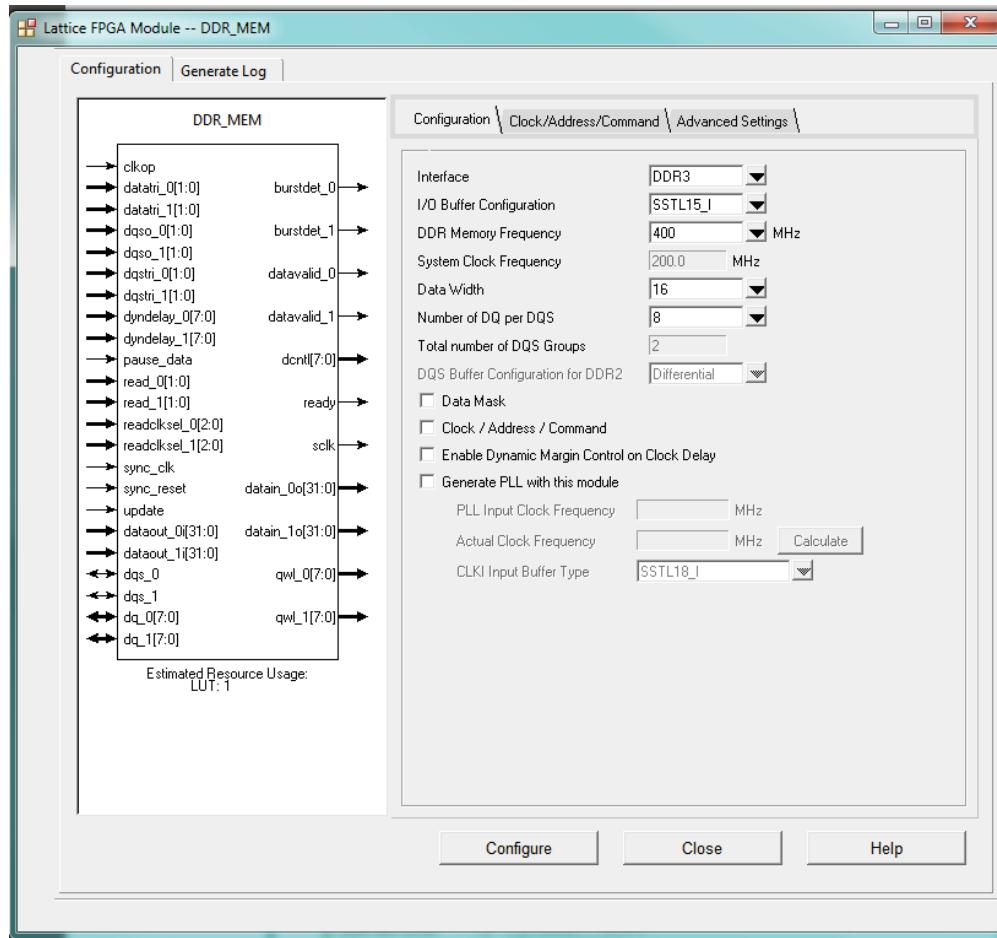


Table 13-11 below describes the various settings shown in the Configuration tab.

**Table 13-11. DDR\_MEM Configuration Tab Parameters**

GUI Option	Description	Range	Default Value
Interface	DDR memory interface type	DDR2, DDR3, DDR3L, LPDDR2, LPDDR3	DDR2
I/O Buffer Configuration	I/O type configuration for DDR pins	DDR2: SSTL18_I, SSTL18_II DDR3: SSTL15_I, SSTL15_II DDR3L: SSTL135_I, SSTL135_II LPDDR2: HSUL12 LPDDR3: HSUL12	DDR2: SSTL18_I DDR3: SSTL15_I DDR3L: SSTL135_I LPDDR2: HSUL2 LPDDR3: HSUL12
DDR Memory Frequency	Target DDR memory interface frequency	DDR2: 125, 200, 267, 333, 400 (MHz) DDR3: 300, 400 DDR3L: 300, 400 LPDDR2: 125, 200, 267, 333, 400 (MHz) LPDDR3: 125, 200, 267, 333, 400 (MHz)	DDR2: 267 MHz DDR3: 400 MHz DDR3L: 400 MHz LPDDR2: 400 MHz LPDDR3: 400 MHz
System Clock Frequency	Calculated system clock frequency. Not user selectable, display only.	SCLK Frequency Value	DDR Memory Frequency/2

<b>GUI Option</b>	<b>Description</b>	<b>Range</b>	<b>Default Value</b>
Data Width	DDR memory interface data width	DDR2, DDR3, DDR3L: 8, 16, 24, 32, 40, 48, 56, 64, 72 LPDDR2, LPDDR3: 16, 32	16
Number of DQ per DQS	Number of associated DQ per DQS pin	DDR2, DDR3, DDR3L: 4, 8 LPDDR2, LPDDR3: 8	8
Total number of DQS Groups	Total number of DQS groups. Not user selectable, display only.	Data width/number of DQ per DQS group	2
DQS Buffer Configuration for DDR2	DDR2 DQS IO buffer type selection	Single-ended, Differential	Single-ended
Clock / Address / Command	Clock/address/command pins added with this option checked	ENABLED, DISABLED	DISABLED
Data Mask	Data mask pins added with this option checked	ENABLED, DISABLED	DISABLED
Enable Dynamic Margin Control on Clock Delay	Dynamic margin control ports added with this option checked	ENABLED, DISABLED	DISABLED
Generate PLL with this module	PLL included with this option checked	ENABLED, DISABLED	DISABLED
PLL Input Clock Frequency	Input reference clock frequency	10 MHz - 400 MHz	-
CLKI Input Buffer Type	The I/O Standard for the PLL Reference Clock	List of Legal Input Standards, None (if coming from fabric)	LVCMS25
Actual DDR Memory Frequency	Calculated actual memory bus frequency. Not user selectable, display only.		

If the user chooses to generate the Clock/Address/Command signals then the settings in the Clock/Address/Command Tab are active and can be set up as required.

Figure 13-48 shows the Clock/Address/Command Tab of the DDR memory Catalog.

**Figure 13-48. DDR\_MEM Clock/Address/Command Tab**

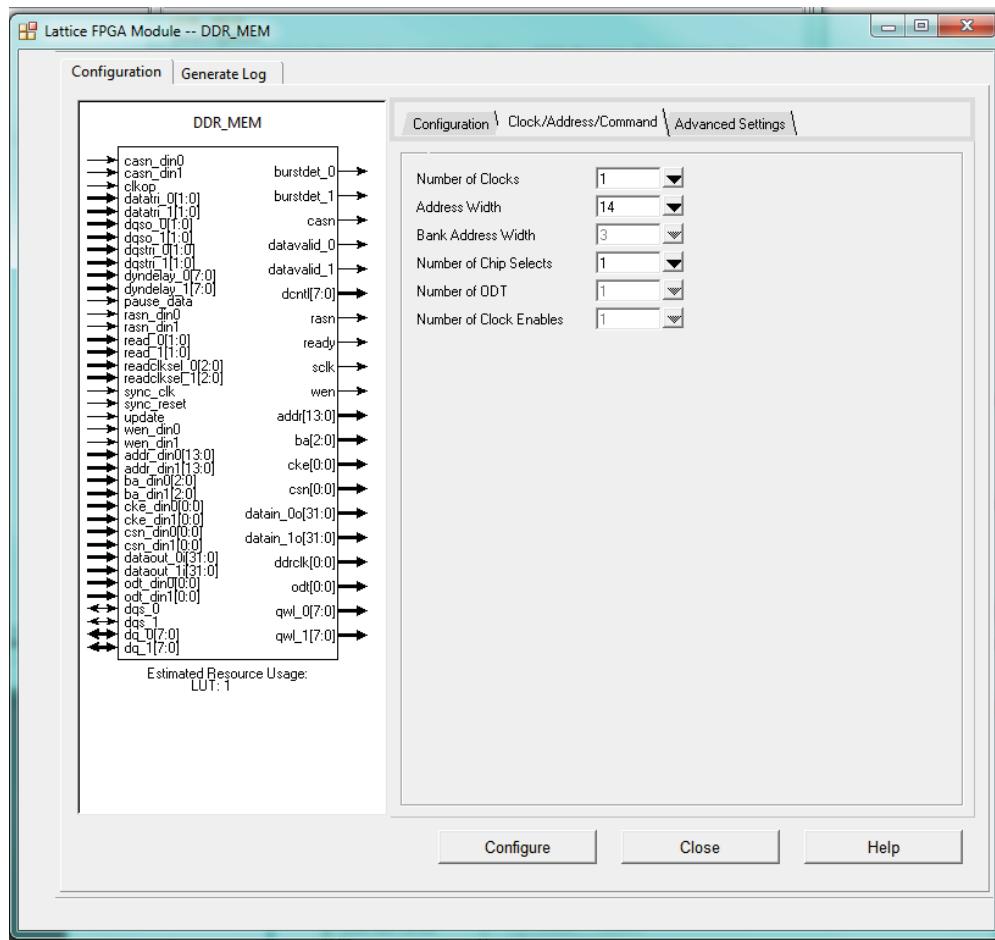


Table 13-12 lists the values that can be used for the Clock/Address/Command settings.

**Table 13-12. DDR\_MEM Clock/Address/Command Parameters**

GUI Option	Range	Default Value
Number of Clocks	DDR2: 1, 2, 4 DDR3: 1, 2, 4 DDR3L: 1, 2, 4 LPDDR2: 1 LPDDR3: 1	DDR3: 1 DDR2: 1 DDR3L: 1 LPDDR2: 1 LPDDR3: 1
Address Width	DDR2: 13 – 16 DDR3: 13 – 16 DDR3L: 13 – 16 LPDDR2: Blank LPDDR3: Blank	DDR2: 13 DDR3: 14 DDR3L: 14 LPDDR2: Blank LPDDR3: Blank
Bank Address Width	DDR2: 2, 3 DDR3: 3 DDR3L: 3 LPDDR2: Blank LPDDR3: Blank	DDR2: 2 DDR3: 3 DDR3L: 3 LPDDR2: Blank LPDDR3: Blank

GUI Option	Range	Default Value
Number of Chip Selects	DDR2: 1, 2, 4 DDR3: 1, 2, 4 DDR3L: 1, 2, 4 LPDDR2: 1 LPDDR3: 1	DDR2: 1 DDR3: 1 DDR3L: 1 LPDDR2: 1 LPDDR3: 1
Number of Clock Enables	= Number of Chip Selects	= Number of Chip Selects
Number of ODT	DDR2, DDR3, DDR3L = Number of Chip Selects LPDDR2: Blank LPDDR3=Number of Chip Selects	DDR2, DDR3, DDR3L = Number of Chip Selects LPDDR2: Blank LPDDR3= Number of chip Selects

There is an additional tab called Advanced Settings for the ECP5 device that can be used to adjust the default DQS Read and Write Delay settings.

**Figure 13-49. DDR\_MEM Advanced Settings Tab**

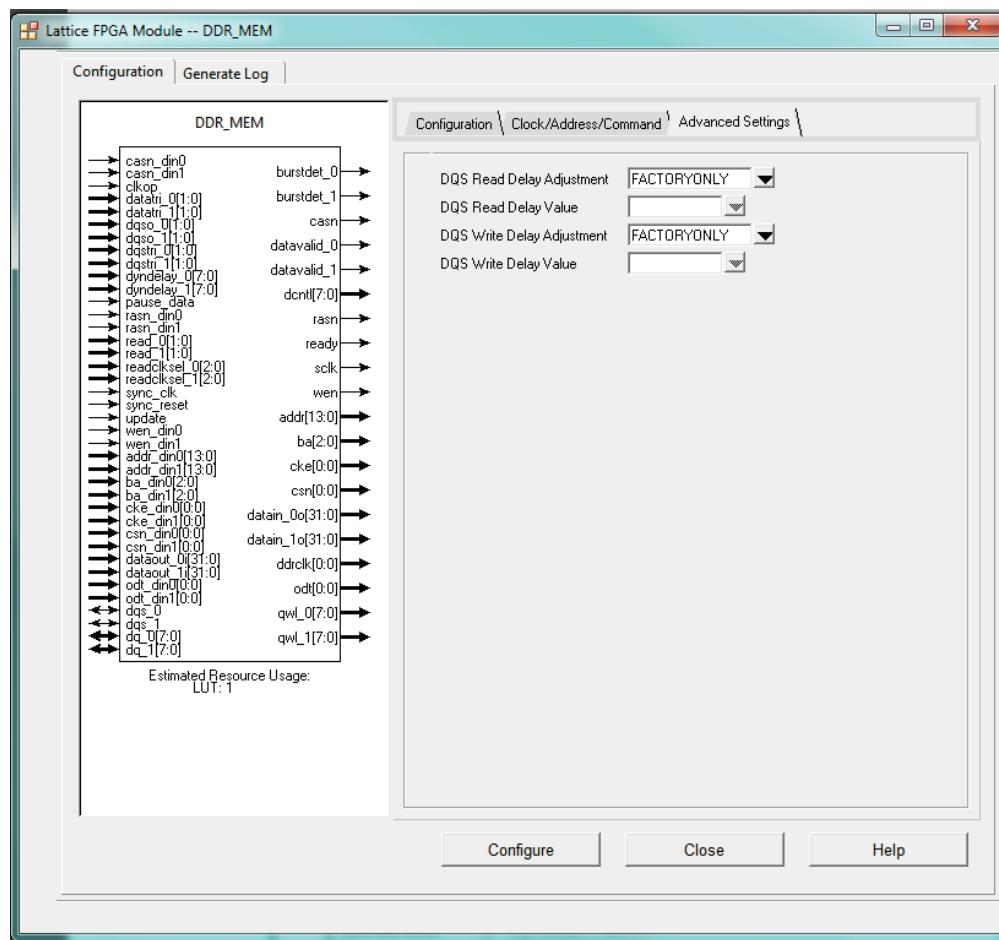


Table 13-13 shows the available values in this tab.

**Table 13-13. DDR\_MEM Advanced Settings Tab Parameters**

GUI Option	Range	Default Value
DQS Read Delay Adjustment	FACTORYONLY, PLUS, MINUS	FACTORYONLY
DQS Read Delay Value	Grey out (if DQS Delay Adjustment = FACTORY-ONLY) 0-255 (if DQS delay adjustment = PLUS) 1-256 (If DQS delay Adjustment = MINUS)	
DQS Write Delay Adjustment	FACTORYONLY, PLUS, MINUS	FACTORYONLY
DQS Write Delay Value	Grey out (if DQS Delay Adjustment = FACTORY-ONLY) 0-255 (if DQS delay adjustment = PLUS) 1-256 (If DQS delay Adjustment = MINUS)	

## Building DDR Interfaces in Clarity Designer

After all the DDR Modules are configured, they can be connected up together in the Builder Tab of Clarity Designer. The connections made in the “Builder” will be carried over to the combined HDL file that will contain all the DDR module instances. If the DDR modules were to share resources, the connections for the sharing can be done here. For example, if a single PLL was shared among the different modules, then the clocks can be connected together in the Builder Tab.

For step by step instructions on using the “Builder”, refer to the [Clarity Designer User Manual](#).

## Planning DDR Interfaces in Clarity Designer

Once the interface are configured and connected, the placement of these modules can be planned in the “Planner” tab of Clarity Designer.

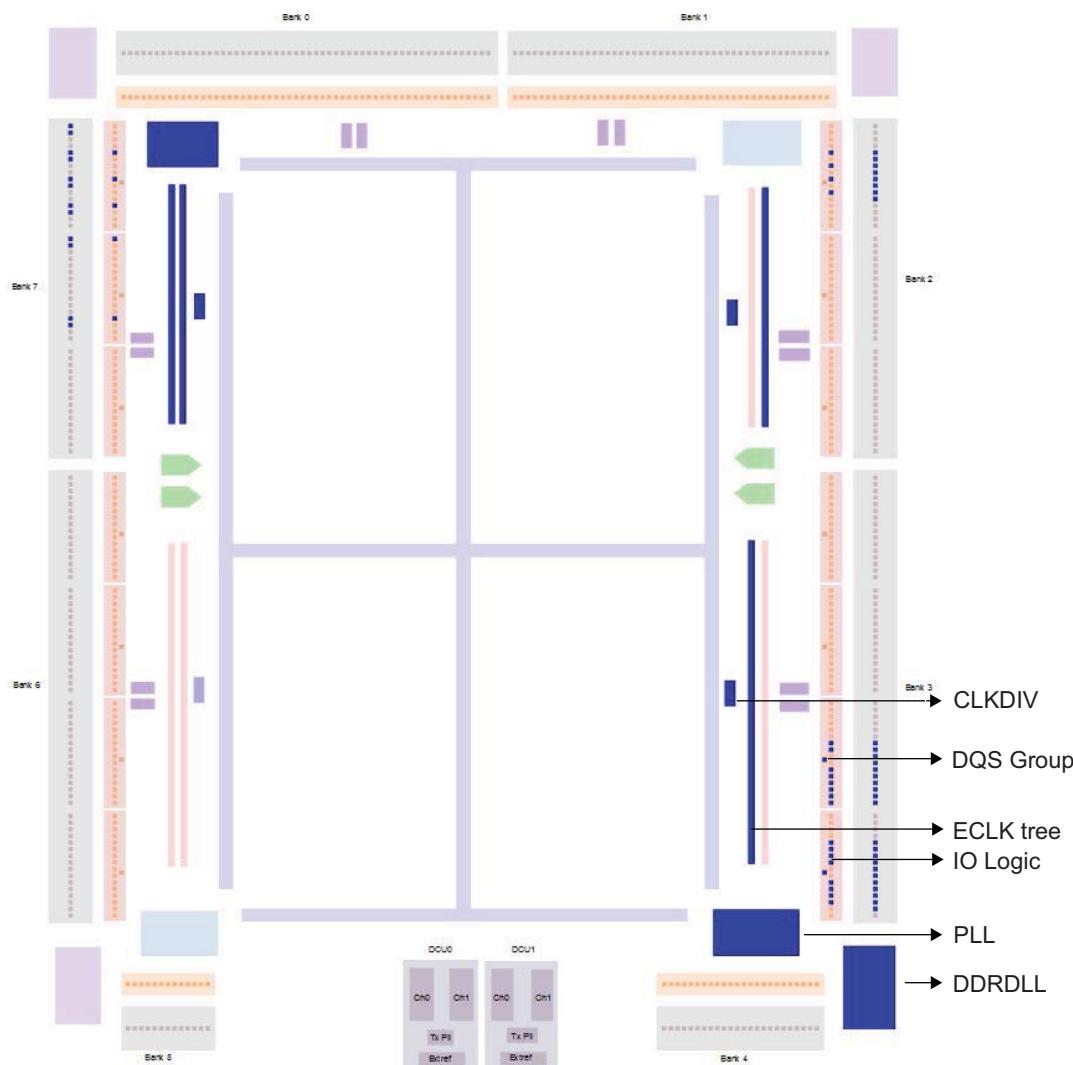
The Planner will allow user to drag & drop each DDR interface into the Chip View. This will automatically lock the pins for that DDR interface at the selected location. The planner takes into account all the clocking & placement requirements, any architecture limitations for each type DDR interface. If any of the placement rules are violated the planner will not place the module if there aren’t enough resources or it will place it at the next available location.

This capability allows user to plan and place all the DDR interfaces before Synthesis. The placement constraints will be carried over through the rest of the flow. Since all the design constraints are taken into account it saves the user a lot of time to not have to run multiple iterations through the tool.

For step by step instructions on using the Planner, refer to the [Clarity Designer User Manual](#).

Figure 13-50 shows DDR modules that are placed using Clarity Design Planner.

**Figure 13-50. DDR Modules Paced Using Clarity Design Planner**



## DDR Software Primitives and Attributes

This section describes the software primitives that can be used to implement all the DDR interfaces. These primitives are divided into ones that are used to implement the DDR data and ones for DDR Strobe signal or the Source Synchronous clock. The DQSBUF primitives are used to generate the signals required to correctly capture the data from the DDR memory.

**Table 13-14. Software Primitives**

Type	Primitive	Description
Input or Output Delay	DELAYF	Dynamic Delay module
	DELAYG	Static Delay module, used to remove clock injection time
DDRDLL	DDRDLLA	DLL used to generate 90 degree phase shift
DLL Delay	DLLDELD	Slave Delay module used for Generic DDR
Generic DDR Data Input	IDDRX1F	Generic DDR 1X gearing registers
	IDDRX2F	Generic DDR 2X gearing registers
	IDDR71B	Generic DDR 7:1 gearing registers, shared by two IOLOGIC blocks
Generic DDR Data Output	ODDRX1F	Generic DDR 1X gearing registers
	ODDRX2F	Generic DDR 2X gearing registers
	ODDR71B	Generic DDR 7:1 gearing registers, shared by two IOLOGIC blocks
DDR Memory DQSBUF Control	DQSBUFM	Used to phase shift DQS Strobe signal and generate control signals for DDR memory
DDR Memory DQ Input	IDDRX2DQA	Used to receive DQ input
DDR Memory DQ Output	ODDRX2DQA	Used to transmit DQ output
DDR Memory DQS Output	ODDRX2DQSB	Used to transmit DQS output
DDR Memory Data Tristate	TSHX2DQA	Used for DQ Tristate output
DDR Memory DQS Tristate	TSHX2DQSA	Used for DQS Tristate output
DDR Memory Address/Command	OSHX2A	Used to generate the Address/Command output

## Input/Output DELAY

The DELAY block can be used to delay the input data from the input pin to the IDDR or IREG or FPGA OR to delay the output data from the ODDR, OREG or FPGA fabric to the output pin. It is useful to adjust for any skews amongst the input or output data bus. It can also be used to generate skew between the bits of output bus to reduce SSO noise.

The DELAY block can be used with IDDR or ODDR modules, SDR module, as well as on the direct input to the FPGA. The DELAY is shared by the input and output paths and hence can only be used either to delay the input data or the output data on a given pin.

The data input to this block can be delayed using:

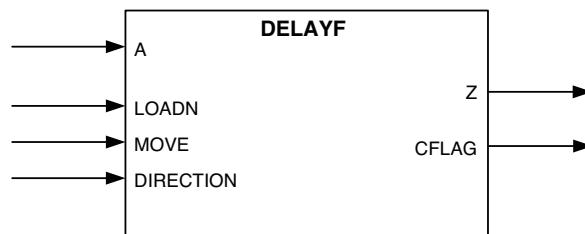
- Pre-determined delay value (for Zero Hold time, delay based on Interface Type)
- Fixed Delay values entered by the user

The data input to this block can also be dynamically updated using counter up and down controls. You can optionally bypass the DELAY block completely as well.

## DELAYF

By default, the DELAYF is configured to factory delay settings based on the clocking structure. Users can overwrite the DELAY setting using the MOVE and DIRECTION control inputs. The LOADN will reset the delay back to the default value.

**Figure 13-51. DELAYF Primitive**



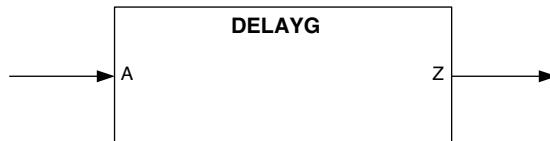
**Table 13-15. DELAYF Port List**

Port	I/O	Description
A	I	Data input from pin or output register block
LOADN	I	'0' on LOADN will reset to default delay setting
MOVE	I	"Pulse" on MOVE will change delay setting. DIRECTION will be sampled at falling edge of MOVE.
DIRECTION	I	'1' to decrease delay and '0' to increase delay
Z	O	Delayed data to input register block or to pin
CFLAG	O	Flag indicating the delay counter has reached the max (when moving up) or min (when moving down) value

## DELAYG

By default, the DELAYG will be configured to factory delay settings based on the clocking structure. Users cannot change the delay when using this module.

**Figure 13-52. DELAYG Primitive**



**Table 13-16. DELAYG Port List**

Port	I/O	Description
A	I	Data input from pin or output register block
Z	O	Delayed data to input register block or to pin

## DELAY Attribute Description

Table 13-17 describes the attributes available for the DELAYF and DELAYG elements. The value of DEL\_MODE is selected based on the interface that will be generated. These values are used to compensate for the clock injection time, hence should be selected based on clocking used. IP Express will automatically assign the correct values for this attribute when Clarity Designer is used to build the interface.

**Table 13-17. DELAYF and DELAYG Attributes**

Attribute	Description	Values <sup>1, 2, 3, 4</sup>	Default	DELAY
DEL_MODE	Sets the delay mode to be used	USER_DEFINED SCLK_ZEROHOLD ECLK_ALIGNED ECLK_CENTERED SCLK_ALIGNED SCLK_CENTERED ECLKBRIDGE_ALIGNED ECLKBRIDGE_CENTER ED DQS_CMD_CLK DQS_ALIGNED_X2	USER_DEFINED D	DELAYG DELAYF
DEL_VALUE	Sets delay value when DEL_MODE is set to USER_DEFINED	0..127	0	DELAYG DELAYF

1. DEL\_MODE must be ECLKBRIDGE\_ALIGNED or ECLKBRIDGE\_CENTERED when it is required to place the data pins of the same high-speed interface on the other side of the device. In addition to setting the DEL\_MODE attribute it is also required to instantiate the ECLKBRIDGECS element in the HDL design.
2. DQS\_CMD\_CLK is only for the DDR Memory CMD and CLK Outputs.
3. DQS\_ALIGNED\_X2 is shared by DQS Generic and the DDR Memory Inputs.

## DDRDLL (Master DLL)

The DDRDLL is used to generate a 90 degree delay for the DQS Strobe Input during a memory interface or for the clock input for a generic DDR interface.

There is one DDRDLL module on each corner of the device. The DDRDLL outputs delay codes that are used in the DQSBUF elements to delay the DQS input, or in the DLLDEL module to delay the input clock. DDRDLL by default will generate 90degree phase shift.

### DDRDLLA

**Figure 13-53. DDRDLLA Primitive**

**Table 13-18. DDRDLLA Port List**

Port	I/O	Description
CLK	I	Reference clock input to the DDRDLL. Should run at the same frequency as the clock to be delayed.
RST	I	Reset input to the DDRDLL
UDDCNTLN	I	Update control to update the delay code. When low, the delay code out of the DDRDLL is updated. Should not be active during a read or a write cycle
FREEZE	I	Releases the DDRDLL input clock
DDRDEL	O	The delay codes from the DDRDLL to be used in DQSBUF or DLLDEL
LOCK	O	Lock output to indicate the DDRDLL has valid delay output
DCNTL [7:0]	O	The delay codes from the DDRDLL available for the user IP.

**Table 13-19. DDRDLL Attributes**

Attribute	Description	Values	Default
FORCE_MAX_DELAY <sup>1</sup>	Bypass DLL locking procedure at low frequency	YES, NO	NO

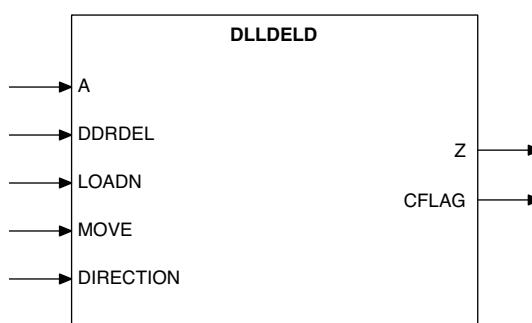
1. When Fin is =<30 MHz. The software sets Force\_max\_delay to YES. DDRDLL will not go through the locking process but will be locked to maximum delay steps under such conditions.

### DLL Delay (DLLDEL)

The DLLDEL receive delay codes from the DDRDLL and generates a delayed clock output.

The DLLDEL receives delay from the DDRDLL. The delayed clock output of the DLLDEL can be connected to the IDDR module.

The delay from the DLLDEL can be dynamically adjusted using counter margin control signals that can shift the delay up or down.DLLDELD

**Figure 13-54. DLLDELD Primitive**

**Table 13-20. DLLDELD Port List**

Port	I/O	Description
A	I	Clock input
DDRDEL	I	Delay inputs from DDRDLL
LOADN	I	Used to reset back to 90° delay.
MOVE	I	Pulse is required to change delay settings. The value on Direction will be sampled at the falling edge of MOVE.
DIRECTION	I	Indicates delay direction. '1' to decrease delay and '0' to increase delay
CFLAG	O	Indicates the delay counter has reached its maximum value when moving up or minimum value when moving down.
Z	O	Delayed clock output

**Table 13-21. DLLDELD Attributes**

Attribute	Description	Values	Default
DEL_ADJ <sup>2</sup>	Sign bit for READ delay adjustment, DDR input	PLUS, MINUS	PLUS
DEL_VAL <sup>2</sup>	Value of delay for input DDR.	0 to 255 (PLUS) 1 to 256 (MINUS)	Note <sup>2</sup>

1. Attributes will only be available through EPIC and ECL Editor. It is recommended that values of this attribute are not updated without consulting Lattice Semiconductor Technical Support.

2. Default value is set based on device characterization to achieve the 90 degree phase shift.

## Generic DDR Input and Output Primitives

The ECP5 device IDDR/ODDR modules support 2:1, 4:1 and 7:1 gearing modes on the left and right sides only. IDDR/ODDR modules on the top (and bottom for non-SERDES parts) will only support 2:1 due to lack of edge clocks. The 2:1 is available on each pin. The 4:1 gearing IDDR/ODDR is available on each pin on the left and right. 7:1 gearing mode is only available per pin pair on the left and right. This means the DDR register of the N side pin will be used to implement 7:1 mode and will not be available to the user. It is assumed that Generic DDR applications using 7:1 model will use a differential input so it would not require the DDR registers of the N side.

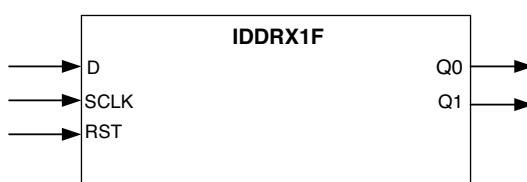
### Input DDR Primitives

The following are the primitives used to implement various Generic DDR Input and Output data.

#### IDDRX1F

This primitive is used to receive Generic DDR with 1X gearing.

**Figure 13-55. IDDRX1F Primitive**



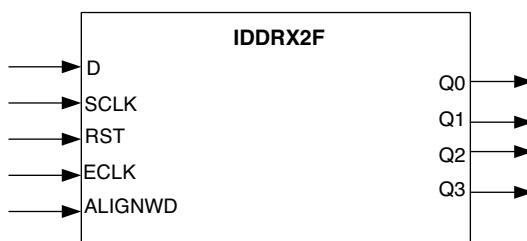
**Table 13-22. IDDRX1F Port List**

Port	I/O	Description
D	I	DDR data input
SCLK	I	Primary clock input
RST	I	Reset to DDR registers
Q0	O	Data at the positive edge of the clock
Q1	O	Data at the negative edge of the clock

#### IDDRX2F

This primitive is used to receive Generic DDR with 2X gearing.

**Figure 13-56. IDDRX2F Primitive**

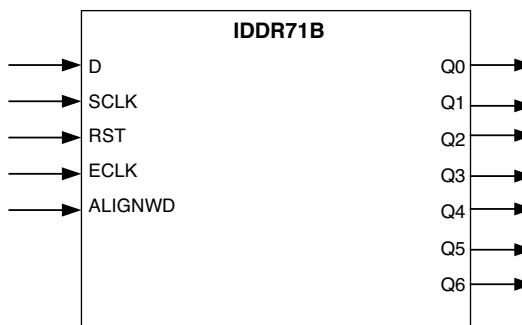


**Table 13-23. IDDRX2F Port List**

Port	I/O	Description
D	I	DDR data input
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide-by-2 of ECLK)
RST	I	Reset to DDR registers
ALIGNWD	I	This signal is used for word alignment. It will shift the word by one bit.
Q0, Q2	O	Data at positive edge of input ECLK
Q1, Q3	O	Data at negative edge of input ECLK

### IDDR71B

This primitive is used for 7:1 LVDS input side implementation.

**Figure 13-57. IDDR71B**

**Table 13-24. IDDR71B Port List**

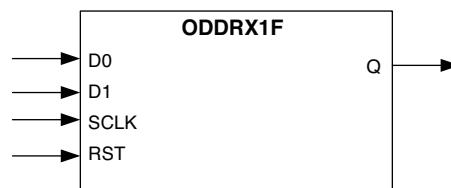
Port	I/O	Description
D	I	DDR data input
ECLK	I	Edge clock
SCLK	I	Primary clock (divide-by-3.5 of ECLK)
RST	I	Reset to DDR registers
ALIGNWD	I	This signal is used for word alignment. It will shift the word by one bit.
Q0 to Q6	O	7 bits of output data.

### Output DDR Primitives

The following are the primitives used to implement various Generic DDR output configurations.

#### ODDRX1F

This primitive is used to transmit Generic DDR with 1X gearing.

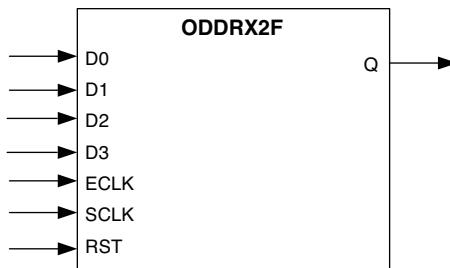
**Figure 13-58. ODDRX1F**


**Table 13-25. ODDRX1F Port List**

Port	I/O	Description
D0, D1	I	Parallel data input to ODDR (D0 is sent out first then D1)
SCLK	I	SCLK input
RST	I	Reset input
Q	O	DDR data output on both edges of SCLK

### ODDRX2F

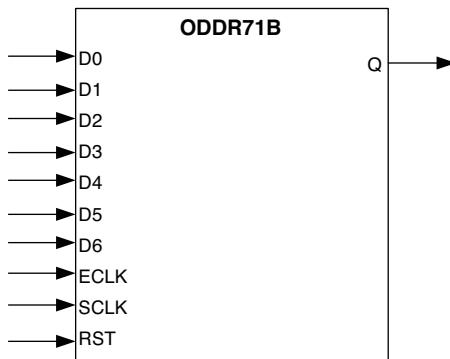
This primitive is used to receive Generic DDR with 2X gearing.

**Figure 13-59. ODDRX2F**

**Table 13-26. ODDRX2F Port List**

Port	I/O	Description
D0, D1, D2, D3	I	Parallel Data input to the ODDR (D0 is sent out first and D3 last)
ECLK	I	ECLK input (2x speed of SCLK)
SCLK	I	SCLK input
RST	I	Reset input
Q	O	DDR data output on both edges of ECLK.

### ODDR71B

This primitive is used for 7:1 LVDS ODDR implementation.

**Figure 13-60. ODDR71B Primitive**


**Table 13-27. ODDR71B Port List**

Port	I/O	Description
D0, D1, D2, D3, D4, D5, D6	I	Parallel data input to the ODDR (D0 is sent out first and D6 last)
ECLK	I	ECLK input (3.5x speed of SCLK)
SCLK	I	SCLK input
RST	I	Reset input
Q	O	DDR data output on both edges of ECLK

## Memory DDR Primitives

This section describes the primitives used to build DDR2, DDR3, DDR3L, LPDDR2 and LPDDR3 memory interfaces.

### DQSBUF (DQS Strobe Control Block)

DQSBUF block is used to delay the incoming DQS signal by 90degrees. DQSBUF receives a delay code from DDRDLL and shifts the signal accordingly. There will be one DQSBUF block for every 16 I/Os. The DQSBUF should be used when the DQS clock tree is used for clocking the IDDR module.

The following describes the functions of the DQSBUF module:

- Receives the delay code from the DDRDLL and generates the 90 delayed DQS signal that is used as a Write clock in the IDDR module
- User can choose to move the delay up or down using the dynamic margin control signals (loadn, move and direction).
- When this margin control circuit is used and LOADN goes high any further delay code changes from the DDRDLL will not be reflected in the delay DQS signal. A soft IP is required to detect the code changes from the DDRDLL and update the MOVE pulse input to the DQSBUF so that the DDRDLL code changes can be tracked.
- If margin control is not used then LOADN should be low to continuously get code from DDRDLL.
- Pause should be asserted prior to changing readclksel, DYNDEL<> or DLL code update.
- Receives READ clock select signals that are used to correctly position the READ signal with respect to the DQS preamble
- Generates a BURSTDET output that can be used to validate the READ pulse positioning.
- Generates the Read and Write pointers required in the IDDR to correctly transfer data between the DQS and ECLK clock domains
- Generates DQS write clocks to be used in ODDR modules to generate DQ and DQS
- Generates the Write Leveling delay required for DDR3 or LPDDR3 interfaces

### DQSBUFM

DQSBUFM element is used for all the DDR Memory interfaces.

Figure 13-61. DQSBUFM Primitive

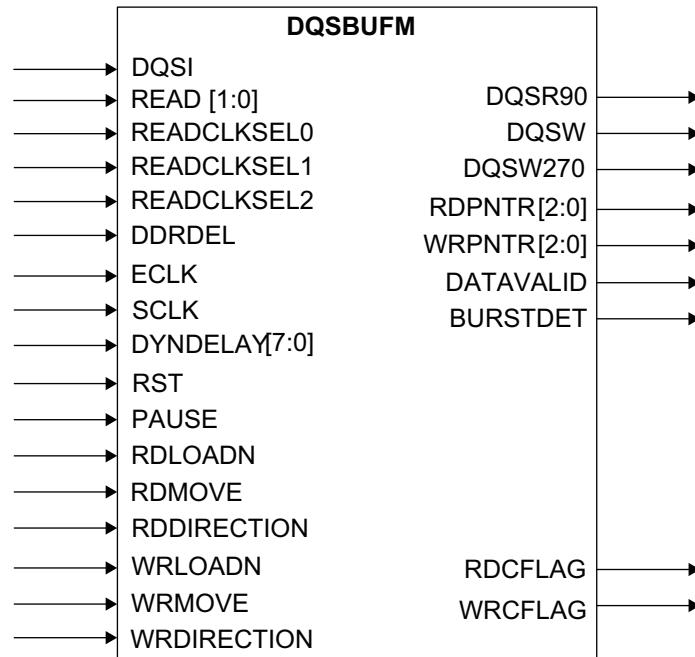


Table 13-28. DQSBUF Port List

Port	I/O	Description
DQSI	I	DQS input from the DQS pin
DDRDEL	I	Delay code from DDRDLL
ECLK	I	Edge clock
SCLK	I	System clock
RST	I	Reset input
READ[1:0]	I	Read input width for DQSBUFM
READCLKSEL0, READCLKSEL1, READCLKSEL2	I	Read clock pulse selection
DYNDELAY[7:0]	I	Dynamic Write leveling delay (only for DDR3)
PAUSE	I	Pause input to stop the DQSW/DQSW270 during write leveling or DDRDLL delay code change.
RDLOADN	I	Used to reset back to 90° delay for read side DQS
RDMOVE	I	Pulse is required to change delay settings. The value on Direction will be sampled at “falling edge” of MOVE. Used to change delay on the read side DQS.
RDDIRECTION	I	Indicates delay direction. ‘1’ decreases the delay count, ‘0’ increases the delay count. Used to change delay on the read side DQS.
RDCFLAG	O	Indicates the delay counter has reached max value for the read side DQS delay.
WRLOADN	I	Used to reset back to 90° delay for write side DQSW270
WRMOVE	I	Pulse is required to change delay settings. The value on Direction will be sampled at the falling edge of MOVE. Used to change delay on the write side DQSW270.
WRDIRECTION	I	Indicates delay direction. ‘1’ decrease delay count, ‘0’ increases the delay count. Used to change delay on the write side DQSW270.
WRCFLAG	O	Indicates the delay counter has reached the maximum value for the write side DQSW270 delay.

Port	I/O	Description
DQSR90	O	90° delay DQS used for read
DQSW270	O	90° delay clock used for DQ write
DQSW	O	Clock used for DQS write
RDPNTR[2:0]	O	Read pointer for IFIFO module
WRPNTR[2:0]	O	Write pointer for IFIFO module
DATAVALID	O	Signal indicating start of valid data
BURSTDET	O	Burst Detect indicator

**Table 13-29. DQSBUFM Attributes**

Attribute	Description	Values	Default	DQSBUF
DQS_LI_DEL_ADJ	Sign bit for READ delay adjustment, DDR input	PLUS, MINUS	PLUS	All
DQS_LI_DEL_VA	Value of delay for input DDR.	0 to 255(PLUS) 1 to 256 (MINUS)	Note <sup>1</sup>	All
DQS_LO_DEL_ADJ	Sign bit for WRITE delay adjustment, DDR output	PLUS, MINUS	PLUS	All
DQS_LO_DEL_VAL	Value of delay for output DDR.	0 to 255 (PLUS) 1 to 256 (MINUS)	Note <sup>1</sup>	All

1. Default value is set based on device characterization to achieve the 90 degree phase shift

## Input and Output Memory DDR Primitives

The ECP5 device IDDR/ODDR modules support 4:1(2X) gearing mode that are used to implement the memory functions.

Table 13-30 shows a summary of all the DDR memory primitives. See the sections below for detailed descriptions.

**Table 13-30. Summary of all DDR Memory Primitives**

DDR Memory	DQ Input	DQ Output	DQ Tristate	DQS Output	DQS Tristate	Addr/Cmd	Clock
DDR2 DDR3 DDR3L	IDDRX1DQA	ODDRX2DQA	TSHX2DQA	ODDRX2DQSB	TSHX2DQSA	ODDRX1F CS_N: OSHX2A	ODDRX2F
LPDDR2	IDDRX2DQA	ODDRX2DQA	TSHX2DQA	ODDRX2DQSB	TSHX2DQSA	ODDRX2DQA CS_N & CKE: ODDRX2DQA <sup>1</sup>	ODDRX2DQSA
LPDDR3	IDDRX2DQA	ODDRX2DQA	TSHX2DQA	ODDRX2DQSB	TSHX2DQSA	ODDRX2DQA CS_N, CKE & ODT: ODDRX2DQA <sup>1</sup>	ODDRX2DQSA

Note: The D0 and D1 inputs are tied together. The D2 and D3 inputs are also tied together.

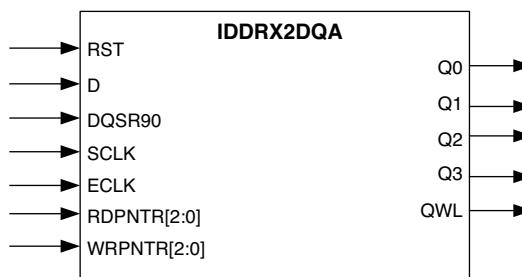
## Memory Input DDR Primitives

The following are the primitives used to implement various memory DDR input configurations.

### IDDRX2DQA

This primitive is used to implement the DDR2 memory input interface at higher speeds and DDR3 memory interface.

**Figure 13-62. IDDRX2DQA Primitive**



**Table 13-31. IDDRX2DQA Port List**

Port	I/O	Description
D	I	DDR data input
RST	I	Reset to DDR registers
DQSR90	I	DQS clock Input
ECLK	I	Fast edge clock
SCLK	I	Primary clock input (divide-by-2 of ECLK)
RDPNTR[2:0]	I	Read pointer from the DQSBUF module used to transfer data to ECLK
WRPNTR[2:0]	I	Write pointer from the DQSBUF module used to transfer data to ECLK
Q0, Q2	O	Data at positive edge of DQS
Q1, Q3	O	Data at negative edge of DQS
QWL	O	Data output used for write leveling

**Table 13-32. Memory Primitive Attributes**

Attribute	Description	Values	Default	Valid Primitives
REGSET	Set the Tristate register to either "SET" or "RESET". By Default value is "SET" so that all output buffers are tristated by default	SET, RESET	SET	TSHX2DQA

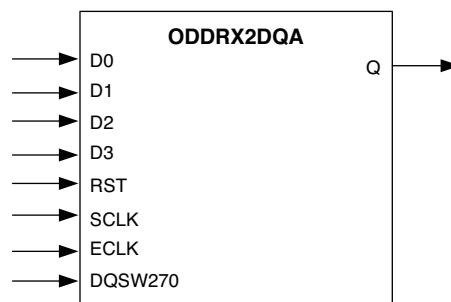
## Memory Output DDR Primitives for DQ Outputs

The following are the primitives used to implement various memory DDR output configurations to generate the DQ outputs.

### ODDRX2DQA

This primitive is used to generate DQ data output for DDR2 with x2 gearing and for DDR3 memory interface.

**Figure 13-63. ODDRX2DQA**



**Table 13-33. ODDRX2DQA Port List**

Port	I/O	Description
D0, D1, D2, D3	I	Data input to the ODDR (D0 is output first, D3 last)
ECLK	I	Fast edge clock input
DQSW270	I	Clock that is 90° ahead of clock used to generate the DQS output
SCLK	I	SCLK input
RST	I	Reset input
Q	O	DDR data output on both edges of DQSW270

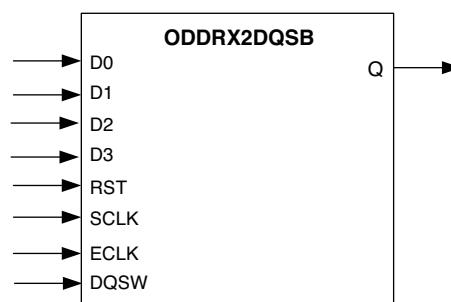
## Memory Output DDR Primitives for DQS Output

Following are the primitives used to implement the DQS outputs to the DDR memory.

### ODDRX2DQSB

This primitive is used to generate DQS clock output for DDR2 and DDR3 memory.

**Figure 13-64. ODDRX2DQSB Primitive**



**Table 13-34. ODDRX2DQSB Port List**

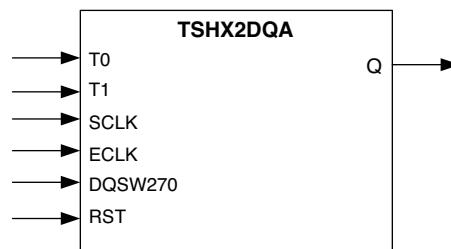
Port	I/O	Description
D0, D1, D2, D3	I	Data input to the ODDR (D0 is output first, D3 last)
ECLK	I	ECLK input
SCLK	I	SCLK input
DQSW	I	DQSW includes write leveling phase shift from ECLK
RST	I	Reset input
Q	O	DDR data output on both edges of DQSW

### Memory Output DDR Primitives for Tristate Output Control

The following are the primitives used to implement tristate control for the outputs to the DDR memory.

#### TSHX2DQA

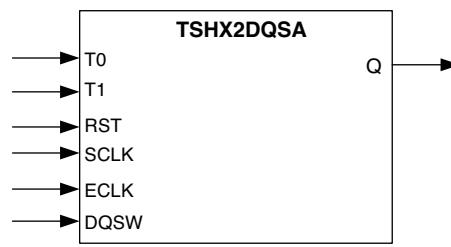
This primitive is used to generate the tristate control for DQ data output.

**Figure 13-65. TSHX2DQA Primitive**

**Table 13-35. TSHX2DQA Port List**

Port	I/O	Description
T0, T1	I	Tristate input (T0 is output first, followed by T1)
ECLK	I	ECLK input (2x speed of SCLK)
DQSW270	I	Clock that is 90° ahead of the clock used to generate the DQS output
SCLK	I	SCLK input
RST	I	Reset input
Q	O	Tristate output

#### TSHX2DQSA

This primitive is used to generate the tristate control for DQS output.

**Figure 13-66. TSHX2DQSA Primitive**


**Table 13-36. TSHX2DQSA Port List**

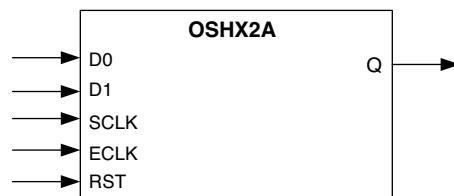
Port	I/O	Description
T0, T1	I	Tristate input (T0 is output first then T1)
ECLK	I	ECLK input (2x speed of SCLK)
SCLK	I	SCLK input
DQSW	I	DQSW includes write leveling phase shift from ECLK
RST	I	Reset Input
Q	O	Tristate output

### Memory Output DDR Primitives for Address and Command

The following are the primitives used to implement the address and command outputs to the DDR memory.

#### OSHX2A

This primitive is used to generate the address and command for DDR3 memory with x2 gearing and write leveling.

**Figure 13-67. OSHX2A Primitive**

**Table 13-37. OSHX2A Port List**

Port	I/O	Description
D0, D1	I	Data input (D0 is output first then D1)
ECLK	I	ECLK input (2x speed of SCLK)
SCLK	I	SCLK input
RST	I	Reset input
Q	O	Address and command output

## Soft IP Modules

The following soft IP Modules are available for use with the Generic DDR interfaces described above. All of the soft IP Modules can be generated using Clarity Designer. Table 13-38 below summarizes the list of soft IPs available and the ones that are optional vs the ones that will be automatically generated with the interface in Clarity Designer.

**Table 13-38. List of Soft IPs supported**

Soft IP Name	Function	Required
RX_SYNC	Used to break up the DDRDLL to DLLDEL clock loop for Aligned Interfaces	Yes
GDDR_SYNC	Needed to tolerate large skew between stop and reset input	Yes
MEM_SYNC	Needed to avoid issues on DDR memory bus and update code in operation without interrupting interface operation.	Yes
7:1 LVDS Bit and Word Alignment (BW_ALIGN)	The soft IP is used to perform bit and word alignment using PLL's dynamic phase shift interface and aligned input of IDDR71C.	Optional
MIPI_FILTER	Implements low pass filter on low speed MIPI data	Optional

Table 13-39 below summarized the soft IPs used in each interface.

**Table 13-39. Soft IP Used in Each Interface**

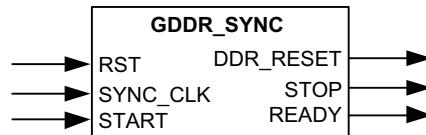
Interface	Soft IP
GDDRX1_RX.SCLK.Centered	None
GDDRX1_RX.SCLK.Aligned	RX_SYNC
GDDRX2_RX.ECLK.Centered	GDDR_SYNC
GDDRX2_RX.ECLK.Aligned	RX_SYNC
GDDRX2_RX.MIPI	GDDR_SYNC, MIPI_FILTER
GDDRX71_RX.ECLK	GDDR_SYNC, BW_ALIGN
GDDRX1_TX.SCLK.Centered	None
GDDRX1_TX.SCLK.Aligned	None
GDDRX2_TX.ECLK.Centered	GDDR_SYNC
GDDRX2_TX.ECLK.Aligned	GDDR_SYNC
GDDRX71_TX.ECLK	GDDR_S

## Detailed Description of Each Soft IP

### GDDR\_SYNC

This module is needed to startup al RX Centered and all TX interfaces with 2x gearing.

**Figure 13-68. GDDR\_SYNC Ports**



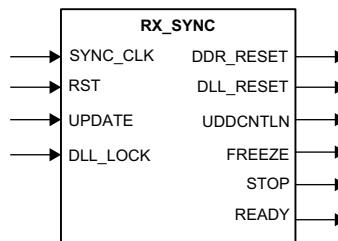
**Table 13-40. GDDR\_SYNC Port List description**

Port	In/Out	Descriptions
SYNC_CLK	IN	Startup clock. This cannot be the RX_CLK or divided version. It can be other low speed continuously running clock. For example, oscillator clock
RST	IN	Active high reset to this sync circuit. When RST=1, STOP=0, DDR_RESET=1, READY=0
START	IN	Start sync process. This is used to wait for PLL lock, then start sync process in 7:1 LVDS interface
STOP	OUT	Connects to ECLKSYNC.STOP
DDR_RESET	OUT	Reset to all IDDRX or ODDRX components and CLKDIV
READY	OUT	Indicate that startup is finished and RX circuit is ready to operate

### RX\_SYNC

This module is needed to startup RX Aligned interfaces with 2x gearing.

**Figure 13-69. RX\_SYNC Ports**

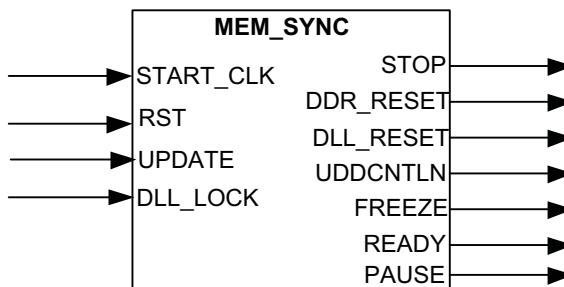


**Table 13-41. RX\_SYNC Port Description**

Port	In/Out	Descriptions
SYNC_CLK	IN	Startup clock. This cannot be the RX_CLK or divided version. It can be other low speed continuously running clock. For example, oscillator clock.
RST	IN	Active high reset to this sync circuit. When RST=1, STOP=0, FREEZE=0, UDDCNTLN=1, DLL_RESET=1, DDR_RESET=1, READY=0.
DLL_LOCK	IN	LOCK output from DDRDLL
UPDATE	IN	UPDATE can be used to re-start sync process. READY will go low and wait for sync process to finish before going high again. This can only be performed when no traffic is present.
STOP	OUT	Connect to ECLKSYNC.STOP.
FREEZE	OUT	Connect to DDRDLL.FREEZE.
UDDCNTLN	OUT	Connect to DDRDLL.UDDCNTLN
DLL_RESET	OUT	Reset to DDRDLL
DDR_RESET	OUT	Reset to all IDDRX components and CLKDIV
READY	OUT	Indicate that startup is finished and RX circuit is ready to operate

### MEM\_SYNC

This module is needed to startup external memory controller interfaces with 2x gearing.

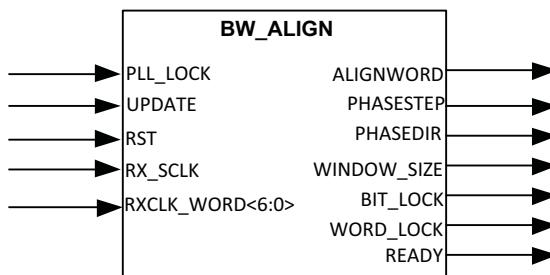
**Figure 13-70. MEM\_SYNC Ports**

**Table 13-42. MEM\_SYNC Port Description**

Port	In/Out	Descriptions
START_CLK	IN	Startup clock. This cannot be the RX_CLK or divided version. It can be other low speed continuously running clock. For example, oscillator clock.
RST	IN	Active high reset to this sync circuit. When RST=1, STOP=0, FREEZE=0, UDDCNTLN=1, DLL_RESET=1, DDR_RESET=1, READY=0 PAUSE =0.
DLL_LOCK	IN	LOCK output from DDRDLL
UPDATE	IN	After ready goes high, user can use UPDATE to update code in DQSBUF, perform training (change read_clk_sel) or write leveling (change dyndelay<>).
PAUSE	OUT	Connect to DQSBUF.PAUSE
STOP	OUT	Connect to ECLKSYNC.STOP.
FREEZE	OUT	Connect to DDRDLL.FREEZE.
UDDCNTLN	OUT	Connect to DDRDLL.UDDCNTLN
DLL_RESET	OUT	Reset to DDRDLL
DDR_RESET	OUT	Reset to all IDDRX, ODDRX, OSRX components, DQSBUF and CLKDIV
READY	OUT	Indicate that startup is finished and RX circuit is ready to operate

## BW\_ALIGN

This module is used to perform 7:1 video RX bit and word alignment. This module is optional and can be enabled in Clarity Designer.

**Figure 13-71. BW\_ALIGN Ports**



**Table 13-43. BW\_ALIGN Port Description**

Port	In/Out	Descriptions
RX_SCLK	IN	Divided RX clock from the 7:1 RX interface, produced by CLKDIV.
RST	IN	Active high reset to this circuit. When RST=1, All outputs=0.
PLL_LOCK	IN	Connect to PLL's LOCK output. Start the alignment procedures after PLL lock goes high.
UPDATE	IN	Start the procedure, or re-start if need to optimize again.
RXCLK_WORD<6:0>	IN	Parallel data output from the 2nd IDDRX71 attached to RX CLK Input.
PHASESTEP	OUT	Rotate phase for PLL
PHASEDIR	OUT	Phase rotation direction for PLL, fixed to forward (0) for this design.
ALIGNWORD	OUT	Connect to IDDRX71.ALIGNWORD, for word rotation.
WINDOW_SIZE	OUT	Final valid window size.
BIT_LOCK	OUT	Status output, bit lock has been achieved.
WORD_LOCK	OUT	Status output, word lock has been achieved.
READY	OUT	Indicate that alignment procedure is finished and RX circuit is ready to operate

With Bit Alignment, the goal is to place edge clock (under PLL dynamic phase shift control) to the center of valid window for the clock word and data words. The PLL phase rotation goes through all 16 phases. The PLL's high speed output is used to sample RX input clock. Transitions are detected on 2nd IDDR71 output which inputs the RX Clock and phases close to transition are identified. The IP will choose the phase most away from transition as the final phase to use.

The low speed clock has two transitions per 7-bit word. It is not the worst case in terms of inter-symbol interference. On the other hand, we do have 8 possible sample points per bit period. Minimum eye-opening of 3/8 UI is needed to achieve lock. Jitter tolerance is around 0.25UI, about 300ps at 756Mb/sec.

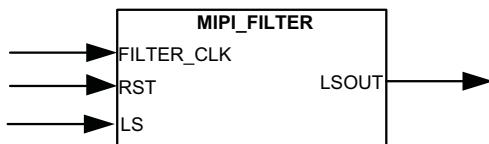
After bit alignment is achieved, word alignment is needed so video data (in 7-bit words) can be processed in core. The IP uses the "ALIGNWD" function of the IDDRX71 primitive for word alignment. Each pulse on ALIGNWD rotates the 7-bit bus by 2 bits. In maximum 7 ALIGNWD operations, the word will loop through all 7 possibilities. The goal is to get "7'b1100011" (7'h63) in the clock word. The clock word is the clock (4 bit 1 and 3'b 0) converted to parallel data, exactly as the video data traffic. For the 7:1 video, the RX input Clock serves as:

- Frequency reference to generate high speed.
- Phase reference as source synchronized RX, since the clock is edge aligned with the data bits.
- Word alignment reference.

**MIPI\_FILTER**

This module is needed to filter low speed signal for MIPI RX. It filters out narrow pulses. It will allow pulse width above 40ns to pass.

**Figure 13-72. MIPI\_FILTER Ports**



**Table 13-44. MIPI\_FILTER Port Description**

Port	In/Out	Descriptions
FILTER_CLK	IN	Clock used to drive digital filter. Min freq=100 MHz. Recommendation is to use internal oscillator at 133 MHz.
LS	IN	Low speed signal from MIPI PHY
RST	IN	Active high reset. When RST=1, LSOUT=0.
LS_OUT	OUT	Filtered output signal
cutoff	Parameter	Parameterize the circuit, default=4, pass signal above 4 cycles. cutoff=round(20ns/period (filter_clk)). Filter_clk=100 MHz, cutoff=4. Filter_clk=133 MHz, cutoff=6. Filter_clk=175 MHz, cutoff=7. Filter_clk=200 MHz. cutoff=8.

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
March 2014	01.0	Initial Release

## Introduction

A key requirement for many of today's high volume FPGA applications is low power consumption. The ECP5™ device provides many dynamic power-saving features for different blocks. This technical note provides users with detail for using the ECP5 device's low power architectural features including power supply considerations and power estimations provided by the Power Calculator tool.

## Power Supply Sequencing and Hot Socketing

ECP5 devices have been designed to ensure predictable behavior during power-up and power-down. During power-up and power-down sequences, the I/Os remain in tri-state until the power supply voltage is high enough (VCCMIN) to ensure reliable operation. In addition, leakage into I/O pins is controlled to within the limits specified in DS1044, [ECP5 Family Data Sheet](#), allowing for easy integration with the rest of the system.

These capabilities, along with lowest-power FPGA with SERDES, makes the ECP5 device the ideal choice for many low-power, high-speed SERDES, multiple power supply and hot-swap applications.

## Recommended Power-up Sequence

Refer to the DC and Switching Characteristics section of DS1044, [ECP5 Family Data Sheet](#) for more information on any power-up sequence for ECP5 family.

## Power Standby Mode

FPGA designers often minimize power consumption by turning off subsystems while configured and operational. Let us look at the different modes of operation for the device.

### Normal operation

Device is fully operational and all circuits are active and consuming highest power consumption.

### Low power operation (Standby)

The ECP5 device have dynamic standby control for the components that are known to consume significant current consumption. The ECP device offers a flexible architecture that allows many on-chip components to be dynamically turned off during the low power operation modes.

Each of the block that has dynamic power control, is listed below, along with the signal that places it in standby mode.

**Table 14-1. Power Save Features of ECP5 Devices**

IP Block	Standby Control Signal
PLL	Through Standby Port.
DLL	Through freeze control. Internal oscillator will stop running to save power.
SERDES	Each RX and TX of each channel can be independently powered down through register bits or CIB signal controls.
IO	Through LVDS output buffer disable (per bank control) and INR disable (per bank control). Valid for Banks # 2, 3, 6, and 7.

The details of major power save features of the device are discussed in the sections that follow.

**Dynamic Bank Controllers**

The ECP5 device has Dynamic Bank Controllers that further help shut down power consuming sources in the IO blocks.

Referenced, differential and LVDS I/O standards consume more power than other I/O standards and are not always required to be active. The active high Bank Controller allows the designer to turn these I/Os off dynamically on a per-bank selection. The Dynamic InRD (input referenced and differential I/Os) is used to turn off referenced and differential inputs. Dynamic LVDS control is used to turn off the LVDS output driver. The Bank Controller can be instantiated using the primitives shown (BCINRD for dynamic InRD, BCLVDSO for dynamic LVDS) are shown in the following sections.

In the ECP5 device, the Dynamic Bank Controller is used to power down banks InRD (Input Referenced and Differential), LVDS Outputs, PUSL (Pull up and Slew Rate) control.

*INRDENI (or Bank Controller for Input & Reference Differential)*

This is used to disable the differential/reference receivers, vref generators and any bank controller reference circuits that consume DC power. It is to be noted that BCINRD can be enabled for all differential and referenced receivers - This applies to all rows of legal I/O combinations inputs except for the PCI, LVTTL or LVCMOS rows.

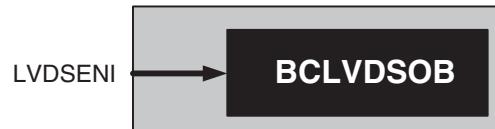
Here is the primitive for the BCINRD.

**Figure 14-1. Input and Reference Differential Bank Controller**



For a design that utilizes the power savings features of the devices, users can instantiate the INRDENI block in their design or in a soft power controller circuit that can turn off the Inputs and referenced differentials. It applies to the Banks # 2, 3, 6 and 7.

**Figure 14-2. LVDS Output Buffers Bank Controller**



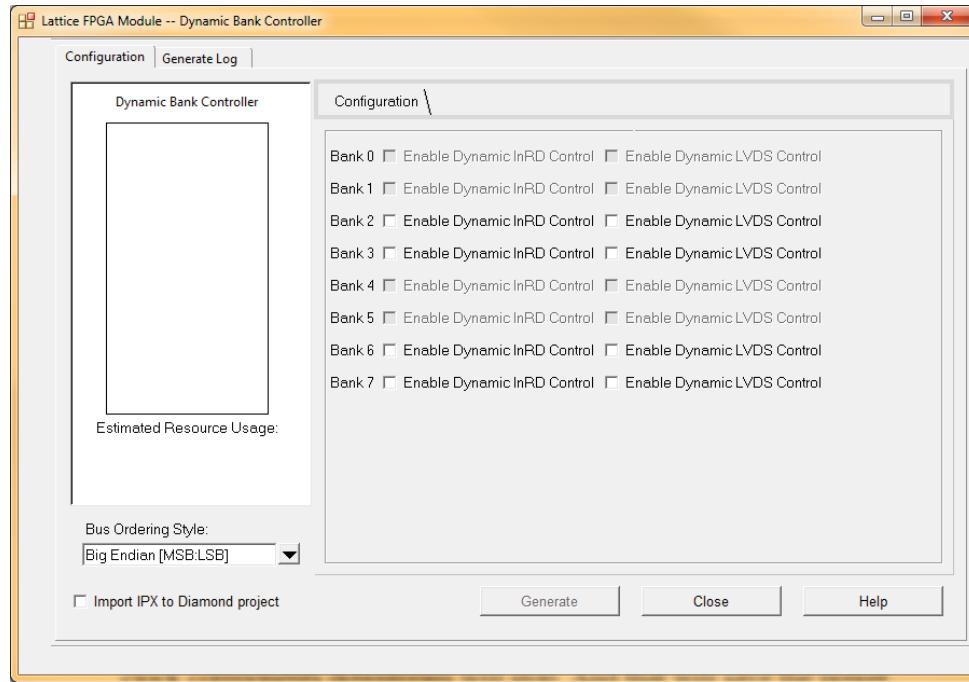
For a design that utilizes the power savings features of the devices, users can instantiate the LVDSENI block in their design or in a soft power controller circuit that can turn off the Inputs and referenced differentials. It applies to the Banks # 2, 3, 6 and 7.

**Generating the Dynamic Bank Controller**

The Dynamic Bank Controller can be generated using Clarity Designer. From the Lattice Diamond environment, launch Clarity Designer and double-click **Dynamic Bank Controller** under Architecture Modules.

Select the options such as module name and HDL language, and then click **Customize**. The screen, shown in Figure 14-3, opens and allows you to generate a Dynamic Bank Controller module.

**Figure 14-3. Generating Dynamic Bank Controller Using Clarity Designer**



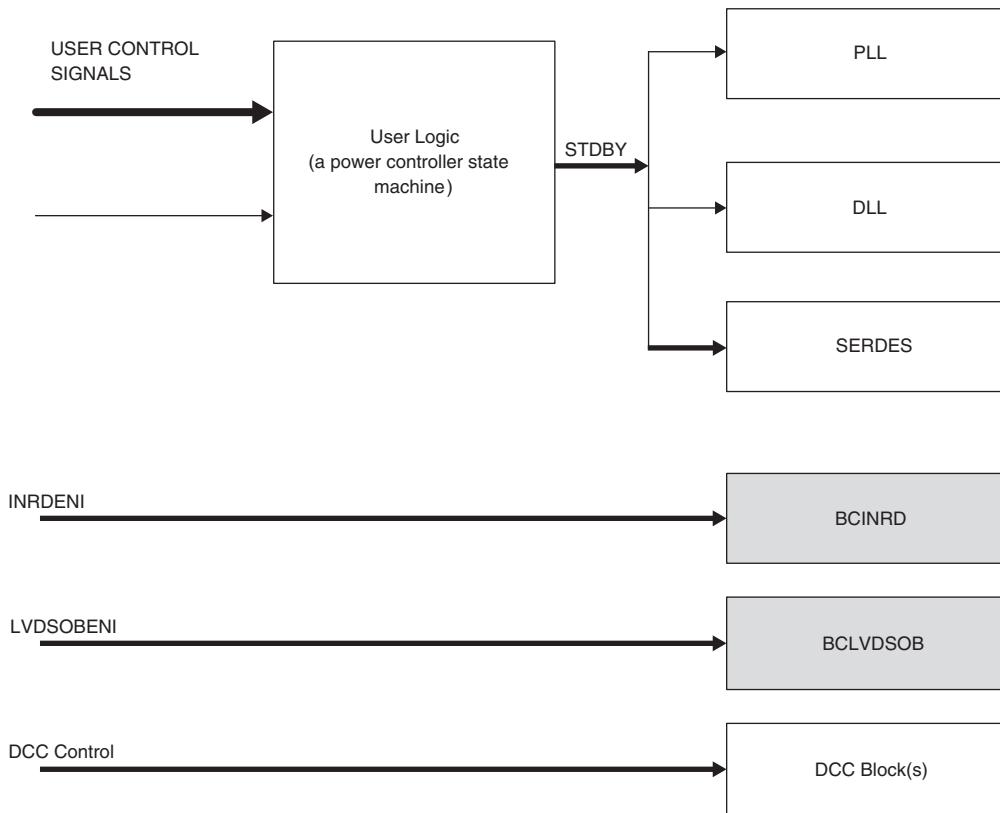
### Soft Power Controller

All those standby operations can be access through fabric soft logic. The ECP5 device does not have a hard standby controller. All standby power management is done through soft logic.

Users can create their own logic that will control placing the components in the standby mode. The various controllers can be instantiated in the user code to place the components in standby mode.

A macro level block diagram for customer scenario is shown in Figure 14-4. This is for example purpose only.

**Figure 14-4. Macro Level Block Diagram for User Defined Power Controller**



## Digital Temperature Readout

ECP5 devices include a Digital Temperature Readout module. There are two DTRs on the chip and users can use that to read the junction temperature at which the chip is operating.

Figure 14-5 provides the primitive that is required to be instantiated in the user design to be able to read the output of the DTR.

**Figure 14-5. DTR Primitive**

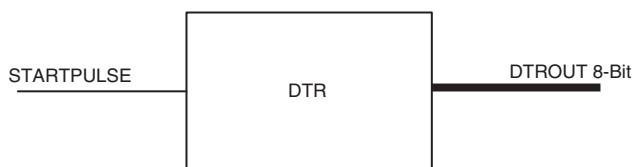


Table 14-2 provides the port definitions for the primitive and the logical capture port.

**Table 14-2. Port Definitions for DTR**

Port Name	Description	Logical Capture Port Name
STARTPULSE	Pulse to instruct DTR to start capturing temperature	cib_start_pulse
DTROUT	8-bit DTR output	tempcode<7:0>

The temperature codes for the ECP5 devices are defined for DTROUT(5:0). DTROUT(6) is reserved and DTROUT(7) is used as data valid bit.

## DTR Timing

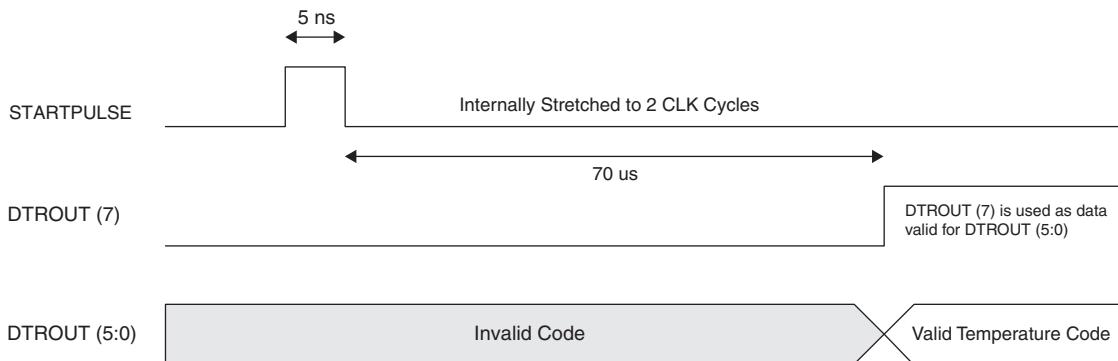
The default output for DTROUT in the functional and timing simulation corresponds to the code for 25 °C temperature. User can also override the value by providing a temperature code via a sim parameter. This is useful in simulation of using the right code in user logic when temperature reaches a particular value.

**Table 14-3. DTR Simulation parameters.**

Primitive Port	Parameter Name	Parameter Value	
START_PULSE			5ns wide start pulse to initiate DTR to measure temperature.
DTROUT(5:0)	DTR_TEMP	0 ... 63 011001 (Default value, equivalent to 25°C)	DTR output that corresponds to the temperature (refer to DTR spec for binary values for different codes).
DTROUT(6)		0	DTROUT(6) is unused and reserved bit. The output value of this bit should match the HW and always set to 0 in simulation.
DTROUT(7)			DTROUT(7) bit is used for getting a data valid signal.

The timing waveform for the DTROUT is shown in Figure 14-6.

**Figure 14-6. Timing Waveform for Valid Temperature with Respect to STARTPULSE**

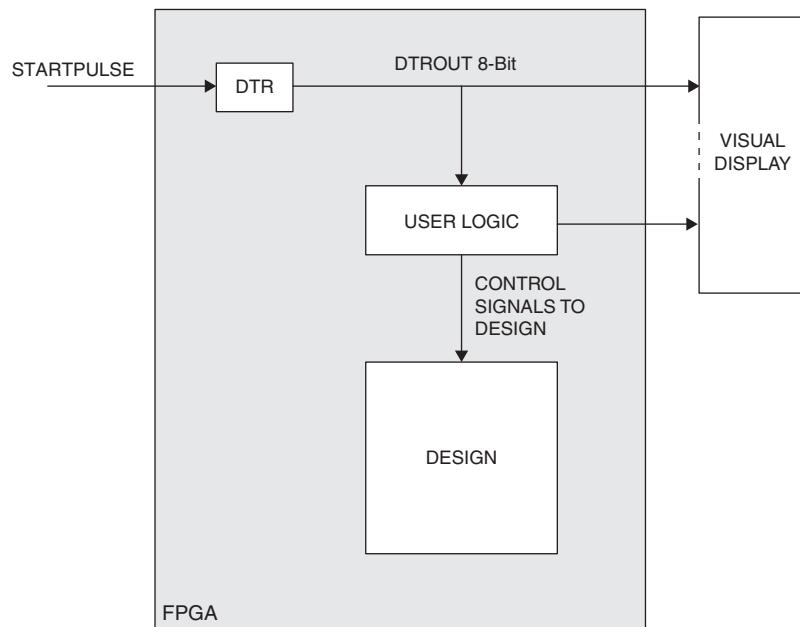


Note: DTROUT (6) is unused.

Figure 14-7 shows one of the scenarios where the DTR can be used. This is for example purposes only.

The User Logic block monitors the 8 bit DTROUT and can have fail safe control. When DTROUT shows that the temperature is high, user can get a visual indicator using Visual Display. Another way is to read DTROUT and then the USER LOGIC (user defined), can monitor the temperature. In case the temperature is high, this block can then generate a control signal for the design to shut down the clock.

Alternatively, the Control signal can also be used as an input to the user defined Power Controller and place the device in standby mode (for example).

**Figure 14-7. Using DTR Usage**


### Equivalent Junction Temperature for DTROUT values

Table 14-4 provides the values of the corresponding Junction Temperature of the device. The table also includes the equivalent decimal values for ease of use.

**Table 14-4. Valid DTROUT Values and Corresponding Junction Temperatures**

Digital Temperature Readout						Decimal Equivalent	Junction Temperature
DTROUT (5)	DTROUT (4)	DTROUT (3)	DTROUT (2)	DTROUT (1)	DTROUT (0)		
0	0	0	0	0	0	0	-58
0	0	0	0	0	1	1	-56
0	0	0	0	1	0	2	-54
0	0	0	0	1	1	3	-52
0	0	0	1	0	0	4	-45
0	0	0	1	0	1	5	-44
0	0	0	1	1	0	6	-43
0	0	0	1	1	1	7	-42
0	0	1	0	0	0	8	-41
0	0	1	0	0	1	9	-40
0	0	1	0	1	0	10	-39
0	0	1	0	1	1	11	-38
0	0	1	1	0	0	12	-37
0	0	1	1	0	1	13	-36
0	0	1	1	1	0	14	-30
0	0	1	1	1	1	15	-20
0	1	0	0	0	0	16	-10
0	1	0	0	0	1	17	-4
0	1	0	0	1	0	18	0
0	1	0	0	1	1	19	4

Digital Temperature Readout						Decimal Equivalent	Junction Temperature
DTROUT (5)	DTROUT (4)	DTROUT (3)	DTROUT (2)	DTROUT (1)	DTROUT (0)		
0	1	0	1	0	0	20	10
0	1	0	1	0	1	21	21
0	1	0	1	1	0	22	22
0	1	0	1	1	1	23	23
0	1	1	0	0	0	24	24
0	1	1	0	0	1	25	25
0	1	1	0	1	0	26	26
0	1	1	0	1	1	27	27
0	1	1	1	0	0	28	28
0	1	1	1	0	1	29	29
0	1	1	1	1	0	30	40
0	1	1	1	1	1	31	50
1	0	0	0	0	0	32	60
1	0	0	0	0	1	33	70
1	0	0	0	1	0	34	76
1	0	0	0	1	1	35	80
1	0	0	1	0	0	36	81
1	0	0	1	0	1	37	82
1	0	0	1	1	0	38	83
1	0	0	1	1	1	39	84
1	0	1	0	0	0	40	85
1	0	1	0	0	1	41	86
1	0	1	0	1	0	42	87
1	0	1	0	1	1	43	88
1	0	1	1	0	0	44	89
1	0	1	1	0	1	45	95
1	0	1	1	1	0	46	96
1	0	1	1	1	1	47	97
1	1	0	0	0	0	48	98
1	1	0	0	0	1	49	99
1	1	0	0	1	0	50	100
1	1	0	0	1	1	51	101
1	1	0	1	0	0	52	102
1	1	0	1	0	1	53	103
1	1	0	1	1	0	54	104
1	1	0	1	1	1	55	105
1	1	1	0	0	0	56	106
1	1	1	0	0	1	57	107
1	1	1	0	1	0	58	108
1	1	1	0	1	1	59	116
1	1	1	1	0	0	60	120
1	1	1	1	0	1	61	124
1	1	1	1	1	0	62	128
1	1	1	1	1	1	63	132

## Power Calculator

Power Calculator is the fastest power simulation tool available in the industry. The tool offers Estimation Mode for “what-if” analysis, and also allows designers to import NCD design files to accurately estimate power for their designs. The background engine performs each calculation quickly and accurately.

When running the Power Calculator tool in Estimation mode, designers provide estimates of the utilization of various components and the tool provides an estimate of the power consumption. This is a good start, especially for what-if analyses and device selection.

Calculation mode is a more accurate approach, where the designer imports the actual device utilization by importing the post place and route netlist design file (or NCD) file.

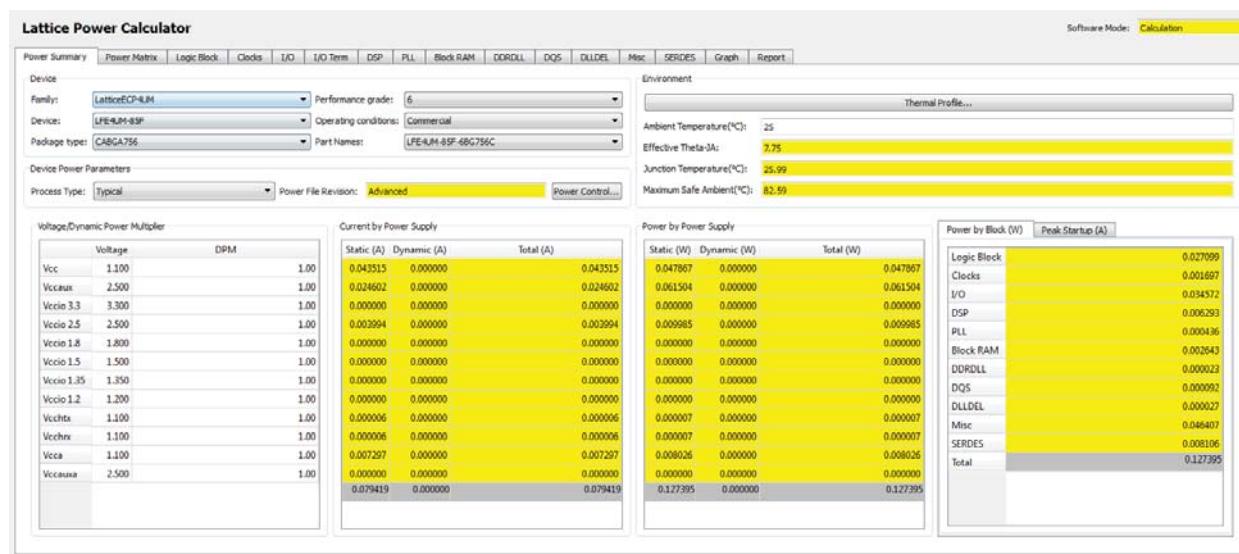
Users can also import a Trace Report (or TWR) file where the frequencies for various clocks are also imported. Note that the Trace Report only includes frequencies of the clocks nets that are constrained in the Preference file.

The default Activity Factor (AF%) for dynamic power calculation is set to 10% in the Power Calculator. Users can change the default AF for the entire project or for each clock net individually. Activity Factor is discussed in more detail later in this document.

## Power Calculator and Power Equations

Please refer to the Diamond® Tutorial under Diamond Startup Page. Once you step through the procedure, you will see the Lattice Power Calculator main window as shown in Figure 14-8.

**Figure 14-8. Power Calculator for ECP5 Devices**



It is important to understand how the options available in Power Calculator affect the power. For example, if the ambient temperature is changed, the junction temperature is affected according to the following equation:

$$T_J = T_A + \Theta_{JA\_EFFECTIVE} * P \quad (1)$$

Where  $T_J$  and  $T_A$  are the junction and ambient temperatures, respectively, and  $P$  is the power.

$\Theta_{JA\_EFFECTIVE}$  is the effective thermal impedance between the die and its environment.

The junction temperature is directly dependent on the ambient temperature. An increase in  $T_A$  will increase  $T_J$  and result in an increase of the static leakage component.

Selecting the Process Type again affects the static leakage (or Static Power). For dynamic power, increasing the frequency of toggling will increase the dynamic component of power.

#### Typical and Worst Case Process Power/ICC

Another factor that affects DC power is process variation. This variation, in turn, causes variation in quiescent power.

Power Calculator takes these factors into account and allows designers to specify either a typical process or a worst case process.

#### Junction Temperature

Junction temperature is the temperature of the die during operation. It is one of the most important factors that affects the device power. For a fixed junction temperature, voltage and device package combination, quiescent power is fixed.

Ambient temperature affects the junction temperature as shown in Equation 1. Devices operating in a high-temperature environment have higher leakage since their junction temperature will be higher. Power Calculator models this ambient to junction temperature dependency. When the user provides an ambient temperature, it is rolled into an algorithm that calculates the junction temperature and power through an iterative process to find the thermal equilibrium of the system (device running with the design) with respect to its environment ( $T_A$ , airflow etc.).

#### Maximum Safe Ambient Temperature

Max. Safe Ambient Temperature is one of the most important numbers displayed in the Summary tab of the Power Calculator. This is the maximum ambient temperature at which the design can run without violating the junction temperature limits for commercial or industrial devices.

Power Calculator uses an algorithm to accurately predict this temperature. The algorithm adjusts itself as the user changes options such as voltage, process, frequency, AF% etc. (or any factor that may affect the power dissipation of the device).

#### Operating Temperature Range

When designing a system, engineers must make sure a device operates at specified temperatures within the system environment. This is particularly important to consider before a system is designed. With Power Calculator, users can predict device thermodynamics and estimate the dynamic power budget. The ability to estimate a device's operating temperature prior to board design also allows the designer to better plan for power budgeting and airflow.

Although total power, ambient temperature, thermal resistance and airflow all contribute to device thermodynamics, the junction temperature (as specified in DS1044, [ECP5 Family Data Sheet](#)) is the key to device operation. The allowed junction temperature range is 0 °C to 85 °C for commercial devices and -40 °C to 105 °C for industrial devices. If the junction temperature of the die is not within these temperature ranges, the performance and reliability of the device's functionality cannot be guaranteed.

#### Dynamic Power Multiplier (DPM)

It is difficult to estimate the temperature dependence of dynamic power due to various ways in which a design can be placed and routed. The user-defined frequency of operation makes this problem even more complex. To help resolve this issue, the Dynamic Power Multiplier provides some guard bands for system and board designers.

The Dynamic Power Multiplier is defaulted to "1" which means the dynamic power is what it is. If the user wishes to add 20% additional dynamic power, the DPM can be set to 1.2 (1 + 20%) and it can be placed against the appropriate power supply. This increases the dynamic power for that supply by 20% and provides users with some guard band (if needed).

### Power Budgeting

Power Calculator provides the power dissipation of a design under a given set of conditions. It also predicts the junction temperature ( $T_J$ ) for the design. Any time this junction temperature is outside the limits specified in DS1044, [ECP5 Family Data Sheet](#), the viability of operating the device at this junction temperature must be re-evaluated.

A commercial device is likely to show speed degradation with a junction temperature above 85 °C and an industrial device at a junction temperature will degrade above 100 °C. It is required that the die temperature be kept below these limits to achieve the guaranteed speed operation.

Operating a device at a higher temperature also means a higher SICC. The difference between the SICC and the total ICC (both Static ICC and Dynamic ICC) at a given temperature provides the dynamic budget available. If the device runs at a dynamic ICC higher than this budget, the total ICC is also higher. This causes the die temperature to rise above the specified operating conditions.

The four factors of power, ambient temperature, thermal resistance and airflow, can also be varied and controlled to reduce the junction temperature of the device. Power Calculator is a powerful tool to help system designers to properly budget the FPGA power that, in turn, helps improve overall system reliability.

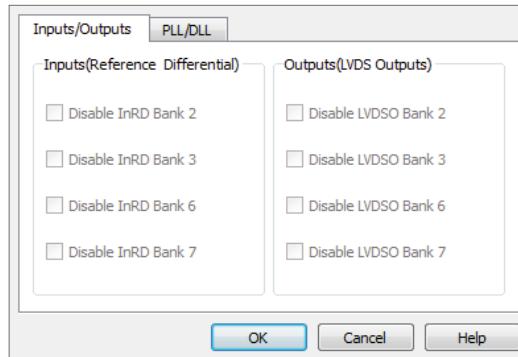
### Dynamic Power Savings

The Power Calculator dynamically estimates the power when the Bank Controller and other power save features are implemented in a design.

The ECP5 device has number of options to control the power, these can range from Bank Controllers to the disabling individual PLLs/ DLLs.

On the Summary tab of Power Calculator, there is a Power Controller button. Clicking the button will launch a window to perform what-if analysis to evaluate power consumption when different dynamic power options are used. The window looks like as shown below.

**Figure 14-9. Dynamic Power Controller for ECP5 Devices**



### Activity Factor Calculation

The Activity Factor % (or AF%) is defined as the percentage of frequency (or time) that a signal is active or toggling the output. Most resources associated with a clock domain are running or toggling at some percentage of the frequency at which the clock is running. Users must provide this value as a percentage under the AF% column in the Power Calculator tool.

Another term for I/Os is the I/O Toggle Rate. The AF% is applicable to the PFU, Routing, and Memory Read Write Ports, etc. The activity of I/Os is determined by the signals provided by the user (in the case of inputs) or as an output of the design (in the case of outputs). The rates at which the I/Os toggle define their activity. The I/O Toggle Rate or the I/O Toggle Frequency is a better measure of their activity.

The Toggle Rate (or TR) in MHz of the output is defined in the following equation:

$$\text{Toggle Rate (MHz)} = 1/2 * f * AF\% \quad (5)$$

Users are required to provide the TR (MHz) value for the I/O instead of providing the frequency and AF% for other resources. AF can be calculated for each routing resource, output or PFU. However, this involves long calculations. The general recommendation for a design occupying roughly 30% to 70% of the device is an AF% between 15% and 25%. This is an average value. The accurate value of an AF depends upon clock frequency, stimulus to the design and the final output.

Power Calculator allows users to import a VCD file from their simulation to accurately assess the activity factor of their design (Edit > Open Simulation File). The VCD file is based on Post-P&R simulation and it is an ASCII file generated by the simulator. All major simulation tools allow that, please check simulation tool documentation on how to generate VCD file. It is to be noted that the AF calculated from the VCD file is based on how accurate the test-bench or stimulus is for the simulation.

## Thermal Impedance and Airflow

A common method for characterizing a packaged device's thermal performance is with "Thermal Resistance",  $\Theta$ . For a semiconductor device, thermal resistance indicates the steady state temperature rise of the die junction above a given reference for each watt of power (heat) dissipated at the die surface. Its units are °C/W.

The most common examples are:

- $\Theta_{JA}$ , Thermal Resistance Junction-to-Ambient (in °C/W)
- $\Theta_{JC}$ , Thermal Resistance Junction-to-Case (also in °C/W)
- $\Theta_{JB}$ , 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 per following equations.

- $T_J = T_A + \Theta_{JA} * P$
- $T_J = T_C + \Theta_{JC} * P$
- $T_J = T_B + \Theta_{JB} * P$

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 heatsink. And  $\Theta_{JB}$  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. Users can also provide the airflow values (in LFM) to obtain a more accurate junction temperature value.

To improve airflow effectiveness, it is important to maximize the amount of air that flows over the device or the surface area of the heat sink. The airflow around the device can be increased by providing an additional fan or increasing the output of the existing fan. If this is not possible, baffling the airflow to direct it across the device may help. This means the addition of sheet metal or objects to provide the mechanical airflow guides to guide air to the target device. Often the addition of simple baffles can eliminate the need for an extra fan. In addition, the order in which air passes over devices can impact the amount of heat dissipated.

## DELPHI Models

DELPHI Models are thermo-mechanical models that can be used to simulate thermal behavior of the ECP5 device when used in a system.

DELPHI Models for the ECP5 device can be downloaded from the web and they are compatible with Flomerics' FloTHERM® and Mentor Graphics' Icepak tools.

## Reducing Power Consumption

One of the most critical challenges for designers today is reducing the system power consumption. A low-order reduction in power consumption goes a long way, especially in modern hand-held devices and electronics. There are several design techniques that can be used to significantly reduce overall system power consumption. Some of these include:

- Using the ECP5 device power saving architecture features like Bank Controller and standby power controls.
- Reducing operating voltage while staying within data sheet limits.
- Operating within the specified package temperature limitations.
- Using optimum clock frequency reduces power consumption, as the dynamic power is directly proportional to the frequency of operation. Designers must determine if some portions of the design can be clocked at a lower rate that will reduce power.
- Reducing the span of the design across the device. A more closely-placed design uses fewer routing resources and therefore less power.
- Reducing the voltage swing of I/Os where possible.
- The unused IOs are powered by the  $V_{CCIO}$  of a bank. Confining the unused IOs in a single bank and then lowering  $V_{CCIO}$  of the bank also helps reduce leakage of the unused IOs.
- Ensuring input logic levels are not left floating but pulled either up or down.
- Ensuring no I/O pull-up/down conflicts with other components on the board.
- Using optimum encoding for the Finite State Machines and counters, where possible. For example, a 16-bit binary counter has, on average, only 12% activity factor and a 7-bit binary counter has an average of 28% activity factor. On the other hand, a 7-bit LFSR counter will toggle at an activity factor of 50%, which causes higher power consumption. A gray code counter, where only one bit changes at each clock edge, will use the least amount of power, as the activity factor is less than 10%.
- Clock gating techniques can be used along with reducing the area of the spread of the design. When used together, these reduce the elements that are toggled by the clock, resulting in lower dynamic power. For details on Clock Gating, refer to TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#).
- Minimizing the operating temperature by the following methods:
  - Use packages that can better dissipate heat, such as ceramic packages.
  - Place heat sinks and thermal planes around the device on the PCB.
  - Use better airflow techniques, such as mechanical airflow guides and fans (both system fans and device mounted fans).
- To achieve the lowest standby power:
  - All clocks and combinatorial logic should be held at a steady state.
  - All inputs should be held at a rail.
  - All outputs should be tri-stated and the Bank Controller should turn off referenced and LVDS outputs.
  - PLLs should be turned off using the standby port.

## Power Calculator Assumptions

The following are the assumptions made by the Power Calculator.

- The Power Calculator tool uses equations with constants based on a room temperature of 25°C. The default temperature of 25 °C can be changed.
- Users can define the ambient temperature ( $T_A$ ) for device junction temperature ( $T_J$ ) calculation based on the power estimation.  $T_J$  is calculated from the user-entered  $T_A$  and the power calculation of typical room temperature.
- I/O power consumption is based on an output loading of 5 pF. Designers have the ability to change this capacitive loading.
- Users can estimate power dissipation and current for each type of power supply ( $V_{CC}$  and  $V_{CCIO}$ ).
- The nominal  $V_{CC}$  is used by default to calculate power consumption. A lower or higher  $V_{CC}$  can be chosen from a list of available values.
- $\Theta_{JA}$  can be changed to better estimate the operating system manually or by entering Airflow in Linear Feet per Minute (LFM) along with a Heat Sink options.
- The default value of the I/O types for ECP5 devices is LVCMOS25, 8 mA.
- The activity factor (AF) is defined as the toggle rate of the registered output. For example, assuming that the input of a flip-flop is changing at every clock cycle, 100% AF of a flip-flop running at 100 MHz is 50 MHz. The default activity factor for logic is 10%.
- Users can import the VCD file from their simulation to get activity factor based on the simulation. It is to be noted that the AF from VCD is as good as the coverage in the simulation.
- Unused IOs are configured as LVCMOS Inputs with weak internal pull ups. These are generally powered off the  $V_{CCIO}$  that is connected to the bank.
- Users have an option to import the Frequency from trace report (TWR) or preference file (LPF).
- The operating junction temperature range for commercial grade devices is 0 °C to 85 °C, Industrial is -40 °C to 100 °C and Automotive is -40 °C to 125 °C. For details, please refer to the DC Electrical Characteristics section in the data sheet.
- For thermal impedance, Power Calc default value is based on a medium size board (~6" x 6", with six to eight layers), with no heatsink and 200 LFM airflow. This can be changed as needed under Thermal Profile section in the tool.
- All virtual devices current and power consumption is calculated for the larger device on which the virtual device is based on.

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
March 2014	01.0	Initial release.

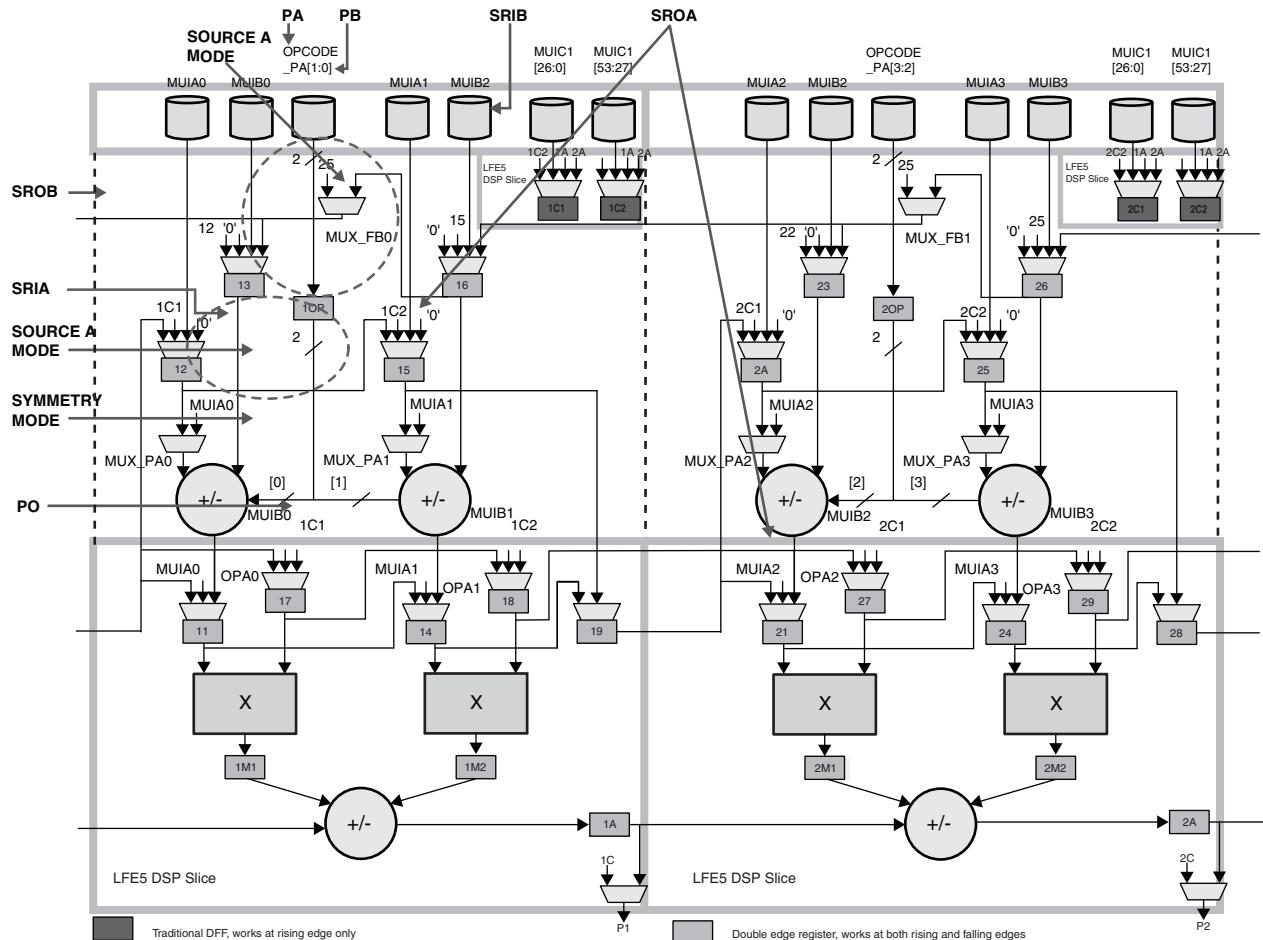
## Introduction

This technical note discusses how to access the features of the ECP5™ sysDSP™ (Digital Signal Processing) slice described in DS1044, [ECP5 Family Data Sheet](#). ECP5 devices are optimized to support high-performance DSP applications, such as wireless base station channel cards, Remote Radio Head (RRH) systems, video and imaging applications, and Fast Fourier Transform (FFT) functions.

## sysDSP Overview

Figure 15-1 shows the ECP5 device DSP Block Diagram at a higher level. As shown each DSP slice has two 18-bit pre-adders, pre-adder registers, two 18-bit multipliers, input registers, pipeline registers, 54-bit ALU, output registers.

**Figure 15-1. ECP5 DSP Block Diagram Overview**



sysDSP slices are located in rows throughout the device. Figure 15-2 shows the simplified block diagram of the sysDSP slices. The programmable resources in a slice include the pre-adders, multipliers, ALU, multiplexers, pipeline registers, shift register chain and cascade chain. If the shift out register A is selected, the cascade match register (Casc) is available. The pre-adders and the multipliers can be configured as 9 bits or 18 bits wide and the ALU

can be configured as 24 bits or 54 bits wide. Multipliers and accumulators can be configured independently and can be used as stand-alone primitives. However, pre-adders must only be used in conjunction with the associated multiplier block. Advanced features of the sysDSP slice are described later in this document.

**Figure 15-2. ECP5 DSP Slice Detailed View**

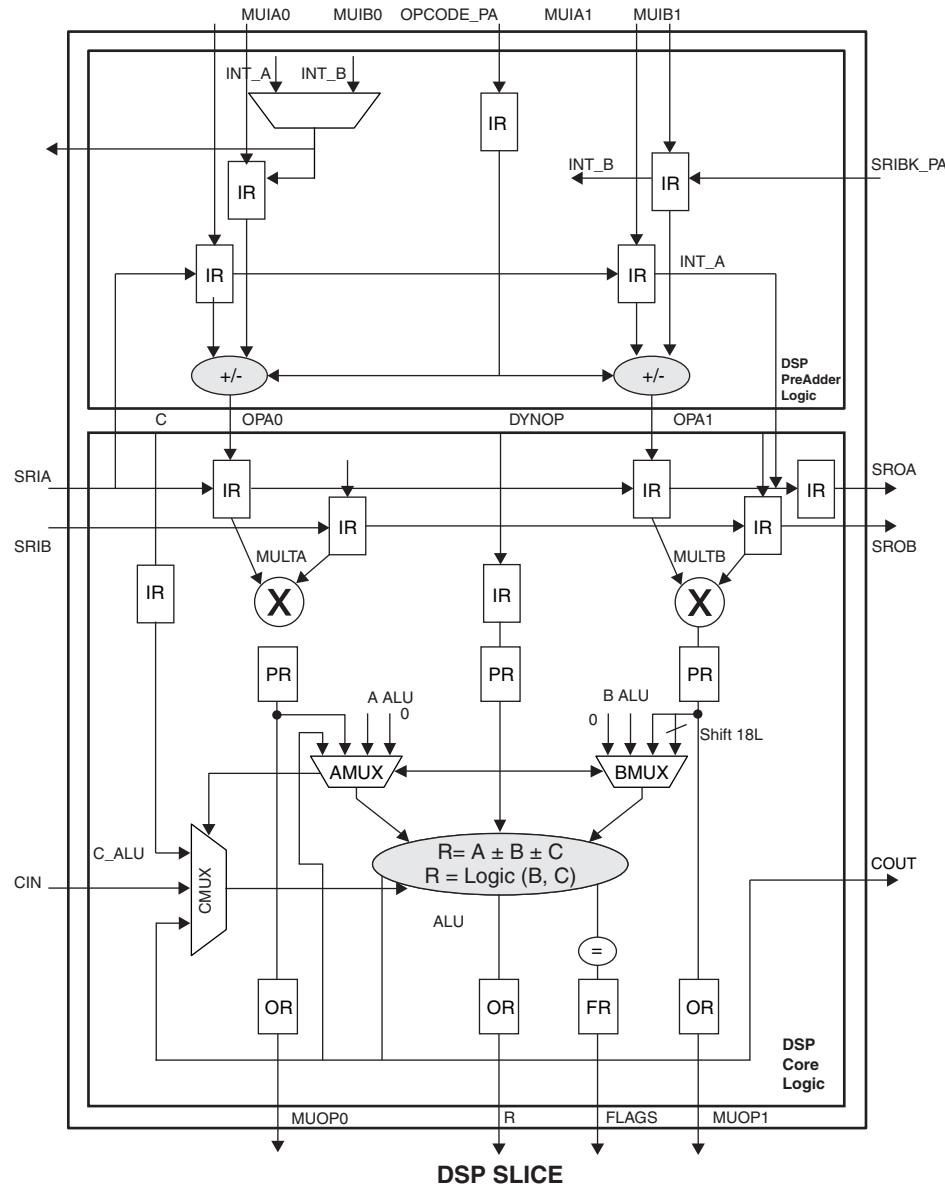


Figure 15-2 shows the individual ECP5 sysDSP slice in greater detail. It shows dual pre-adders with the core ECP5 DSP logic. The built-in pre-adders, multipliers and ALU minimize the amount of external logic required to implement some of the key DSP functions, resulting in efficient resource usage, reduced power consumption, improved performance, and data throughput for DSP applications. The ECP5 sysDSP slice can be configured several ways to suit users' end applications.

The IR shown in a blue outline is an 18-bit register. The ORs and FR share a 72-bit register. If simple multiplier mode is implemented, the register is used as multiplier output. If ALU is implemented, it is used as ALU output.

## Operating modes and features

The DSP Block has three main operating modes:

- One 36x36 Multiplier
  - Basic Multiplier, no add/sub/accumulator/sum blocks.
- Four 18x18 Multipliers
  - Two add/sub/accumulator blocks
  - One summation Block for adding four multipliers
- Eight 9x9 Multipliers
  - Four add/sub/accumulator blocks
  - Two Summation Blocks

Additionally, the device has advanced features such as:

- 18-bit dual multipliers
- 54-bit ternary adder/accumulator
- Additional multiplexer logic to support high-speed option
- Enhanced Pre-Adder Logic
  - 18-bit pre-adder/subtractor in front of each multiplier's sample register
  - Additional multiplexer logic to support high-speed option

In addition to these modes, ECP5 DSP Slice also includes pre-adders and additional shim logic to support:

- 1D Symmetry for Wireless Applications
- 2D Symmetry for Video Applications
- Long Tap FIR Filter Support across multiple DSP Rows
- Full 54-bit Accumulator Support
- Higher Operation of Frequency (400 MHz).

Various components are used in combination to enable the advanced functions of the sysDSP slice, such as:

- Cascading of slices for implementing adder trees in sysDSP slices
- Ternary addition functions implemented through the bypassing of multipliers
- Various rounding techniques that modify the data using the ALU
- ALU flags
- Dynamic multiplexer input selection allows for Time Division Multiplexing (TDM) of the sysDSP slice resources.
- High-speed logic to support the high-speed operating mode.

SOURCEA\_MUX: SOURCEA\_MUX selects between shift (SRIA) or parallel (A) input to the multiplier.

SOURCEB\_MUX: SOURCEB\_MUX selects between shift (SRIB) or one of the parallel inputs (B or C).

AMUX: AMUX selects between multiple 54-bit inputs to the ALU statically or dynamically. The inputs to AMUX are listed in Table 15-1.

---

## Using sysDSP

The DSP slices can be used in a number of ways in ECP5 devices, as described in the sections that follow.

### Primitive Instantiation sysDSP

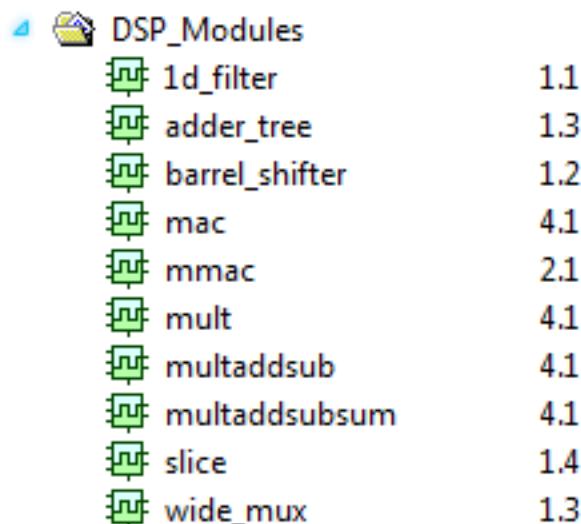
The sysDSP primitives can be directly instantiated in the design. Each of the primitives has a fixed set of attributes that can be customized to meet the design requirements.

An example of the primitive instantiation is given in “Appendix A: Instantiating DSP Primitives in HDL”. You can get the detailed list of the primitives from the synthesis libraries under *cae\_library\synthesis* folder under Diamond® installation.

### Using Clarity Designer to Configure and Generate DSP Modules

Designers can utilize the Clarity Designer to easily specify a variety of DSP modules in their designs. Here is a screenshot of the module selection for the memory modules under Clarity Designer in Lattice Diamond software.

**Figure 15-3. DSP Modules in Clarity Designer**



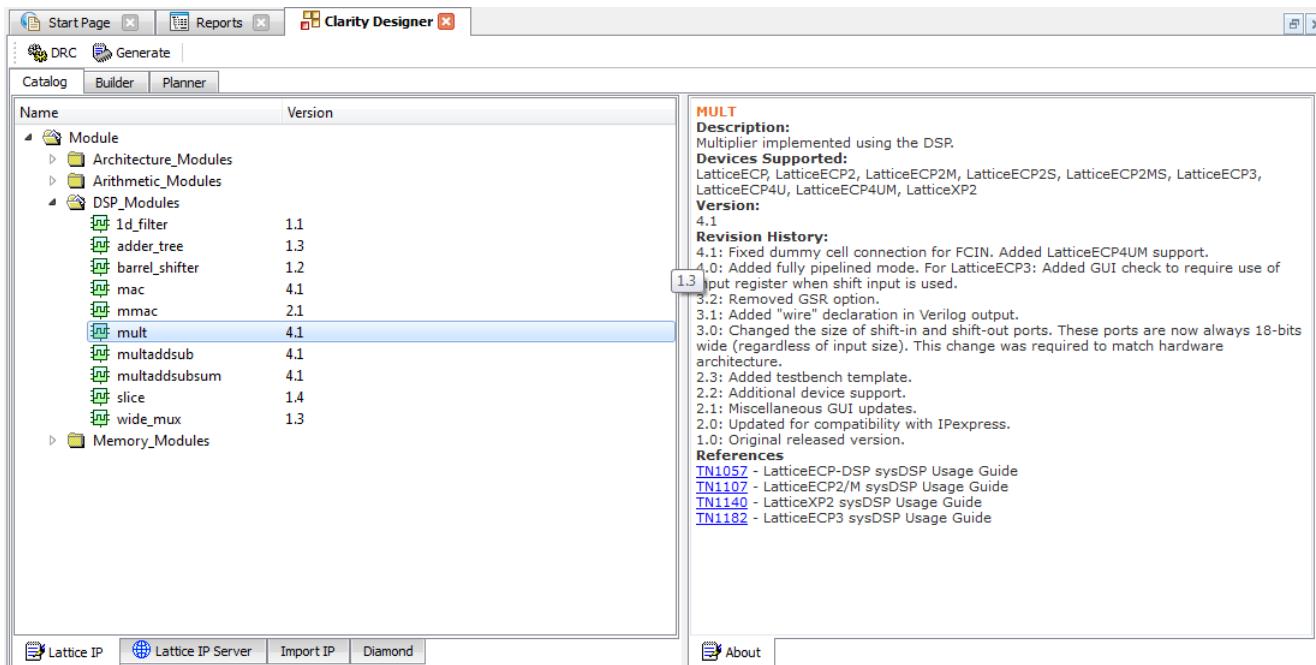
## Clarity Designer Flow

Clarity Designer allows you to generate, create (or open) any of the above modules for ECP5 devices.

From the Lattice Diamond software, select Tools > Clarity Designer.

Alternatively, you can also click on the  button in the toolbar. This opens the Clarity Designer window as shown in Figure 15-4.

**Figure 15-4. Clarity Designer in Lattice Diamond Software**

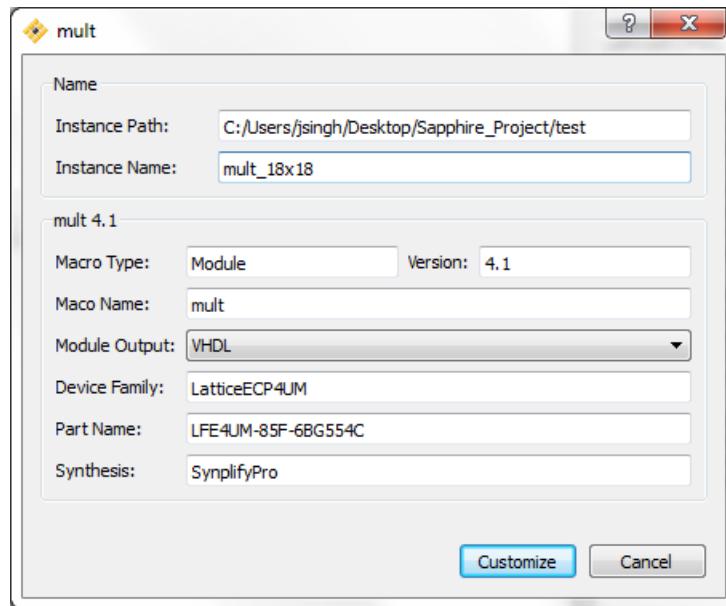


The left section of Clarity Designer window has the Module tree, and all the sysDSP related modules are under DSP\_Modules. The right section of the window provides a brief description of the selected module and links to further documentation.

Let us look at an example of generating an 18x18 multiplier using the Clarity Designer.

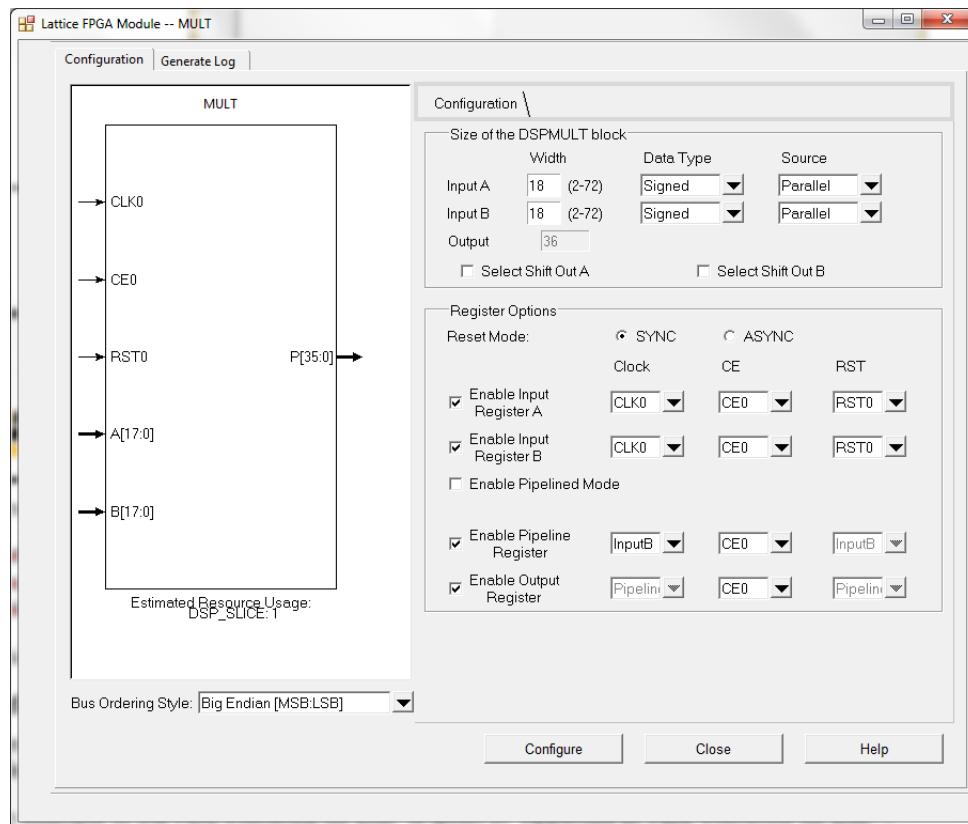
Double-click MULT under the DSP\_Modules. This opens the Clarity Designer window that allows you to specify file name and macro name. Fill out the form, select the preferred language (Verilog or VHDL) and click **Customize**. Fill out the information of the module to generate. This is shown in Figure 15-5.

**Figure 15-5. Generating Distributed 18x18 Multiplier in Clarity Designer in Lattice Diamond Software**



Click **Customize** to open another window, as shown in Figure 15-6, where you can customize the 18x18 Multiplier.

**Figure 15-6. Customizing Multiplier in Clarity Designer in Lattice Diamond Software**



Once all the right options of the module being generated are filled in, click on the Generate button.

This module, once in the Diamond project, can be instantiated within other modules.

## Inferencing sysDSP slice

Designers can write a behavioral code for the DSP function such as multiplier, ALU etc., and the synthesis tool can infer the block into the ECP5 sysDSP functions.

An example of the HDL inference for DSP is given in “Appendix B: HDL Inference for DSP”.

## Targeting the sysDSP Slice by Instantiating Primitives

The sysDSP slice can be targeted by instantiating the sysDSP slice primitive into a design. The advantage of instantiating primitives is that it provides access to all the available ports and parameters. The disadvantage of this flow is that the customization requires extra coding and knowledge by the user. This section details the primitives supported by ECP5 devices. Please refer to “Appendix A: Instantiating DSP Primitives in HDL” that shows an HDL examples on how to instantiate sysDSP primitives.

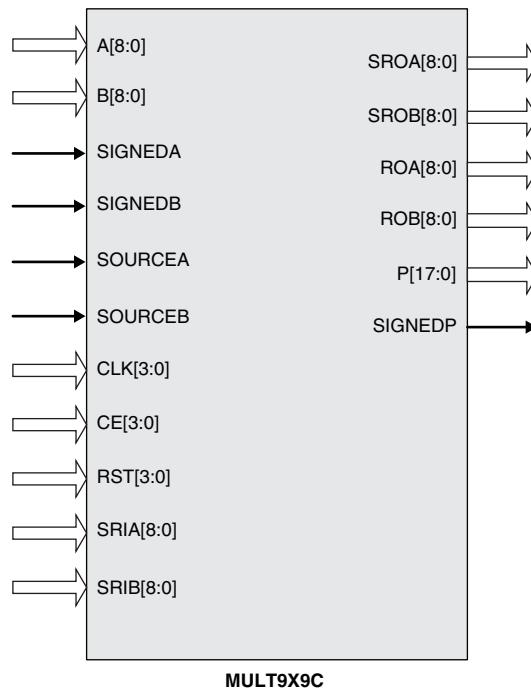
The ECP5 sysDSP supports all the legacy ECP5 device primitives, namely MULT9X9C, MULT18X18C, ALU24A and ALU24B. In addition, several other library primitives have been defined to take advantage of the features of the ECP5 sysDSP slice.

Various primitives available to the designers, along with the port definitions and attributes are discussed in the sections that follow.

### MULT9X9C – Advanced 9X9 DSP Multiplier

The 9x9 multiplier is a widely used module. Figure 15-7 shows the MULT9X9C primitive available in the ECP5 device.

**Figure 15-7. MULT9X9C Primitive**



### MULT9X9C – I/O Port Description

Table 15-1 describes the list of ports available for MULT9X9C primitive.

**Table 15-1. MULT9X9C I/O Port Description**

Port	Input/ Output	Description
A[8:0]	I	Multiplier parallel Input A
B[8:0]	I	Multiplier parallel Input B
SIGNEDA	I	Signed Bit for Input A
SIGNEDB	I	Signed Bit for Input B
SOURCEA	I	Source Selector for Multiplier Input A
SOURCEB	I	Source Selector for Multiplier Input B
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SRIA[8:0]	I	Multiplier shift Input A
SRIB[8:0]	I	Multiplier shift Input B
SROA[8:0]	O	Shift Output A
SROB[8:0]	O	Shift Output B
ROA[8:0]	O	Output A
ROB[8:0]	O	Output B
P[17:0]	O	Product Output
SIGNEDP	O	Signed Bit for the Product Output

### MULT9X9C – Attribute Description

Table 15-2 describes the attributes for MULT9X9C primitive.

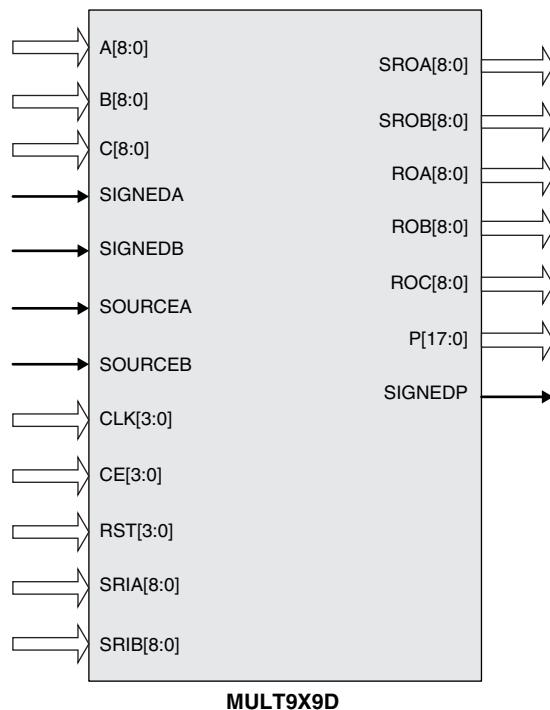
**Table 15-2. Attribute Description for MULT9X9C**

Attribute Name	Values	Default Value	GUI Access
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_PIPELINE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_PIPELINE_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_PIPELINE_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT_RST	RST0, RST1, RST2, RST3	RST0	Y
GSR	ENABLED, DISABLED	ENABLED	N
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y
MULT_BYPASS	ENABLED, DISABLED	DISABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y

## MULT9X9D – Advanced 9x9 DSP Multiplier for Highspeed

This version of 9x9 multiplier has been optimized for high speed. Figure 15-8 shows the MULT9X9D primitive available in ECP5 device.

**Figure 15-8. MULT9X9D Primitive**



### MULT9X9D – I/O Port Description

The Table 15-3 describes the list of ports available for MULT9X9D primitive.

**Table 15-3. MULT9X9D I/O Port Description**

Port	I/O	Description
A[8:0]	I	Multiplier parallel Input A
B[8:0]	I	Multiplier parallel Input B
C[8:0]	I	Multiplier Input C
SIGNEDA	I	Signed Bit for Input A
SIGNEDB	I	Signed Bit for Input B
SOURCEA	I	Source Selector for Multiplier Input A
SOURCEB	I	Source Selector for Multiplier Input B
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SRIA[8:0]	I	Multiplier shift Input A
SRIB[8:0]	I	Multiplier shift Input B
SROA[8:0]	O	Shift Output A
SROB[8:0]	O	Shift Output B
ROA[8:0]	O	Output A
ROB[8:0]	O	Output B
ROC[8:0]	O	Shift Output C – To be used for right side of the slice only
P[17:0]	O	Product Output
SIGNEDP	O	Signed Bit for the Product Output

**MULT9X9D – Attribute Description**

The Table 15-4 describes the attributes for MULT9X9D primitive.

**Table 15-4. Attribute Description for MULT9X9D**

Attribute Name	Values	Default Value	GUI Access
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTC_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTC_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTC_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_PIPELINE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_PIPELINE_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_PIPELINE_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT_RST	RST0, RST1, RST2, RST3	RST0	Y
CLK0_DIV	ENABLED, DISABLED	ENABLED	Y
CLK1_DIV	ENABLED, DISABLED	ENABLED	Y
CLK2_DIV	ENABLED, DISABLED	ENABLED	Y
CLK3_DIV	ENABLED, DISABLED	ENABLED	Y
HIGHSPEED_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
GSR	ENABLED, DISABLED	ENABLED	N
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y
SOURCEB_MODE	B_SHIFT, C_SHIFT, B_C_DYNAMIC, HIGHSPEED	B_SHIFT	Y
MULT_BYPASS	ENABLED, DISABLED	DISABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y

MULT9X9D has an option to select the source for the Multiplier Input B. Table 15-5 lists the details of SOURCEB\_MODE Attribute for MULT18X18D Primitive

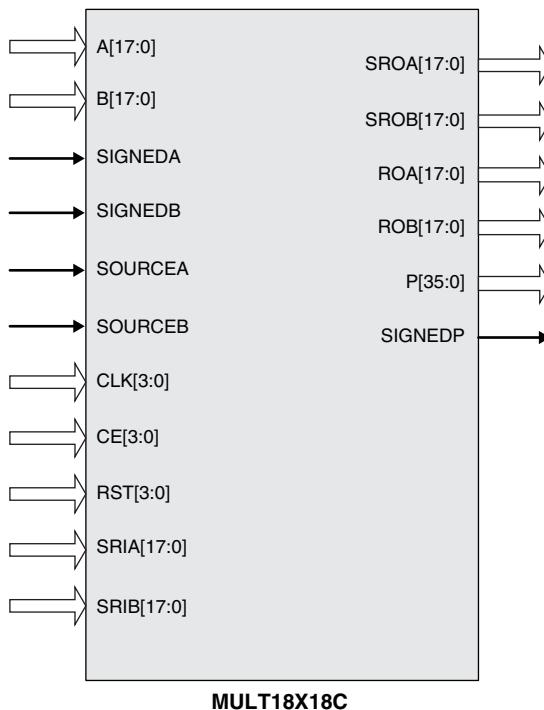
**Table 15-5. SOURCEB\_MODE Attribute for MULT18X18D Primitive**

IP Express Operation	SOURCEB_MODE Attribute	SOURCEB Port	Mc1_b0_mux3	Mc1_0b_mux4
Shift	B_SHIFT	1	00	01
B	B_SHIFT	0	00	00
C	C_SHIFT	0	01	00
B/C Dynamic	B_C_DYNAMIC	Live	10	00
Highspeed BC	HIGHSPEED	0	11	00
Dynamic Shift/B	B_SHIFT	Live	00	10
Dynamic Shift/C	C_SHIFT	Live	01	10

## MULT18X18C – Basic 18X18 DSP Multiplier

The ECP5 device also includes the 18X18 multiplier natively. Figure 15-9 shows the MULT18X18C primitive available in ECP5 device.

**Figure 15-9. MULT18X18C Primitive**



### MULT18X18C – I/O Port Description

Table 15-6 describes the port list for MULT18X18C primitive.

**Table 15-6. MULT18X18C I/O Port Description**

Port	I/O	Description
A[17:0]	I	Multiplier parallel Input A
B[17:0]	I	Multiplier parallel Input B
SIGNEDA	I	Signed Bit for Input A
SIGNEDB	I	Signed Bit for Input B
SOURCEA	I	Source Selector for Multiplier Input A
SOURCEB	I	Source Selector for Multiplier Input B
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SRIA[17:0]	I	Multiplier shift Input A
SRIB[17:0]	I	Multiplier shift Input B
SROA[17:0]	O	Shift Output A
SROB[17:0]	O	Shift Output B
ROA[17:0]	O	Output A
ROB[17:0]	O	Output B
P[35:0]	O	Product Output
SIGNEDP	O	Signed Bit for the Product Output

**MULT18X18C – Attribute Description**

Table 15-7 describes the attributes for MULT18X18C primitive.

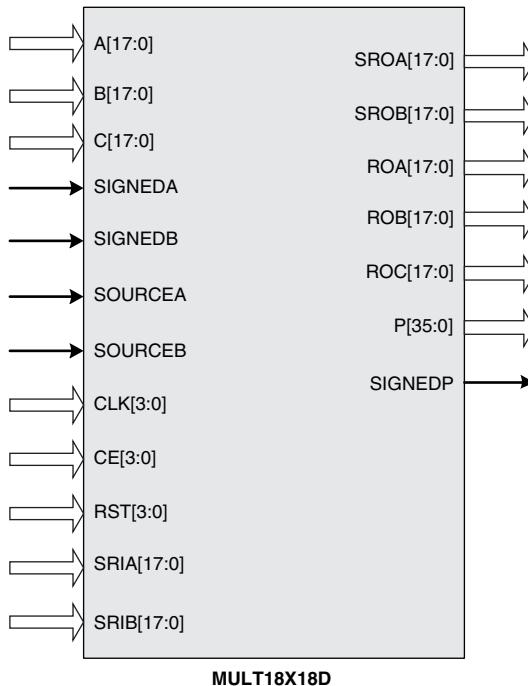
**Table 15-7. Attribute Description for MULT18X18C**

Attribute Name	Values	Default Value	GUI Access
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_PIPELINE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_PIPELINE_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_PIPELINE_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT_RST	RST0, RST1, RST2, RST3	RST0	Y
GSR	ENABLED, DISABLED	ENABLED	N
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y
MULT_BYPASS	ENABLED, DISABLED	DISABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y

## MULT18X18D – Advanced 18X18 DSP Multiplier for High Speed

Similar to its 9X9 counterpart, 18X18 also has a high speed version – MULT18X18D. Figure 15-10 shows the MULT18X18D primitive

**Figure 15-10. MULT18X18D Primitive**



### MULT18X18D – I/O Port Description

Table 15-8 describes the port list for MULT18X18D primitive.

**Table 15-8. MULT18X18D I/O Port Description**

Port	I/O	Description
A[17:0]	I	Multiplier parallel Input A
B[17:0]	I	Multiplier parallel Input B
C[17:0]	I	Multiplier Input C
SIGNEDA	I	Signed Bit for Input A
SIGNEDB	I	Signed Bit for Input B
SOURCEA	I	Source Selector for Multiplier Input A
SOURCEB	I	Source Selector for Multiplier Input B
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SRIA[17:0]	I	Multiplier shift Input A
SRIB[17:0]	I	Multiplier shift Input B
SROA[17:0]	O	Shift Output A
SROB[17:0]	O	Shift Output B
ROA[17:0]	O	Output A
ROB[17:0]	O	Output B
ROC[17:0]	O	Shift Output C – For right side of the slice only
P[35:0]	O	Product Output
SIGNEDP	O	Signed Bit for the Product Output

**MULT18X18D – Attribute Description**

Table 15-9 describes the attributes for MULT18X18D primitive.

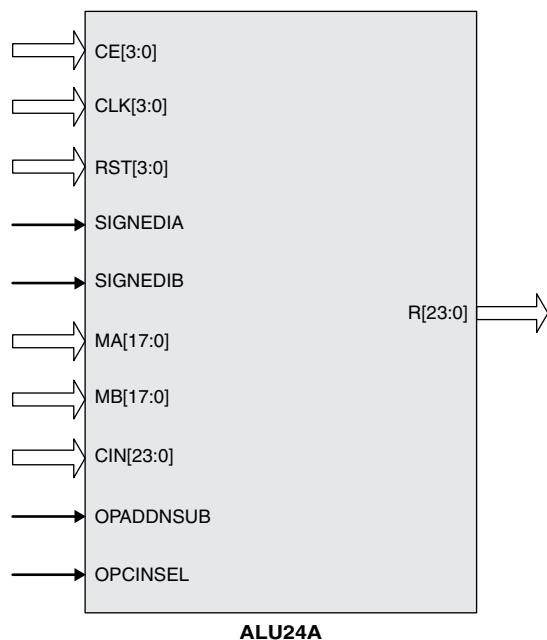
**Table 15-9. Attribute Description for MULT18X18D**

Attribute Name	Values	Default Value	GUI Access
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTC_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTC_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTC_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_PIPELINE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_PIPELINE_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_PIPELINE_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT_RST	RST0, RST1, RST2, RST3	RST0	Y
CLK0_DIV	ENABLED, DISABLED	ENABLED	Y
CLK1_DIV	ENABLED, DISABLED	ENABLED	Y
CLK2_DIV	ENABLED, DISABLED	ENABLED	Y
CLK3_DIV	ENABLED, DISABLED	ENABLED	Y
HIGHSPEED_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
GSR	ENABLED, DISABLED	ENABLED	N
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y
SOURCEB_MODE	B_SHIFT, C_SHIFT, B_C_DYNAMIC, HIGHSPEED	B_SHIFT	Y
MULT_BYPASS	ENABLED, DISABLED	DISABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y

## ALU24A – 24-bit Ternary Adder/ Subtractor

ECP5 devices also allows configuration in an ALU mode. Figure 15-11 shows the ALU24A primitive

**Figure 15-11. ALU24A Primitive**



### ALU24A – I/O Port Description

Table 15-10 describes the port list for ALU24A primitive.

**Table 15-10. ALU24A I/O Port Description**

Port	I/O	Description
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SIGNEDIA	I	Sign Indicator for Input A
SIGNEDIB	I	Sign Indicator for Input B
MA[17:0]	I	Input A
MB[17:0]	I	Input B
CIN[23:0]	I	Carry In Input
OPADDNSUB	I	Add/Sub Selector
OPCINSEL	I	Carry In Selector
R[23:0]	O	Sum Output

**ALU24A – Attribute Description**

Table 15-11 describes the attributes for ALU24A primitive.

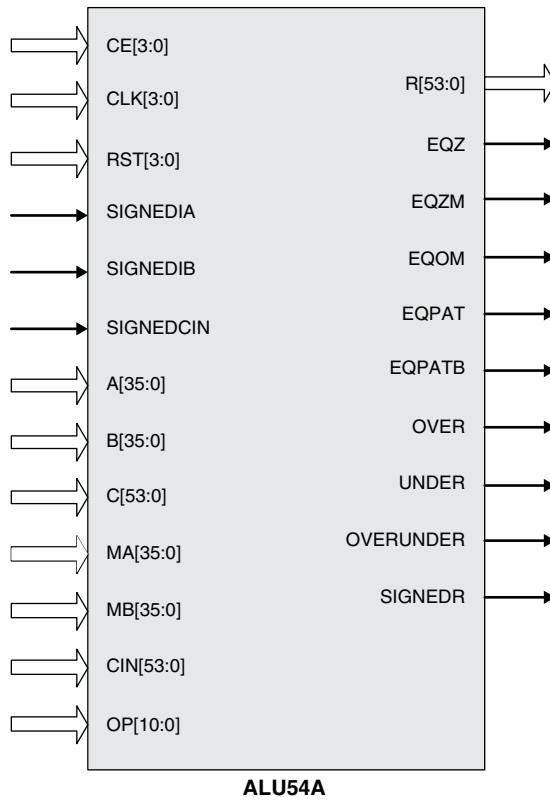
**Table 15-11. Attribute Description for ALU24A**

Attribute Name	Values	Default Value	GUI Access
REG_OUTPUT_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODE_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODE_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODE_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODE_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODE_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODE_1_RST	RST0, RST1, RST2, RST3	RST0	Y
GSR	ENABLED, DISABLED	ENABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y

## **ALU54A – 54-bit Ternary Adder/ Subtractor**

Figure 15-12 shows the ALU54A primitive

**Figure 15-12. ALU54APrimitive**



### **ALU24A – I/O Port Description**

Table 15-12 describes the port list for ALU54A primitive.

**Table 15-12. ALU54A I/O Port Description**

Port	I/O	Description
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SIGNEDIA	I	Sign Bit for Input A
SIGNEDIB	I	Sign Bit for Input B
SIGNEDCIN	I	Sign Bit for Carry In Input
A[35:0]	I	Input A
B[35:0]	I	Input B
C[53:0]	I	Carry In Input
MA[35:0]	I	Input A
MB[35:0]	I	Input B
CIN[53:0]	I	Carry In Input
OP[10:0]	I	Opcode
R[53:0]	O	Sum
EQZ	O	Equal to Zero Flag
EQZM	O	Equal to Zero with Mask Flag
EQOM	O	Equal to One with Mask Flag
EQPAT	O	Equal to Pattern with Mask Flag
EQPATB	O	Equal to Bit Inverted Pattern with Mask Flag
OVER	O	Accumulator Overflow
UNDER	O	Accumulator Underflow
OVERUNDER	O	Either Over or Underflow (may be removed)
SIGNEDR	O	Sign Bit for Sum Output

**ALU54A – Attribute Description**

Table 15-13 describes the attributes for ALU54A primitive.

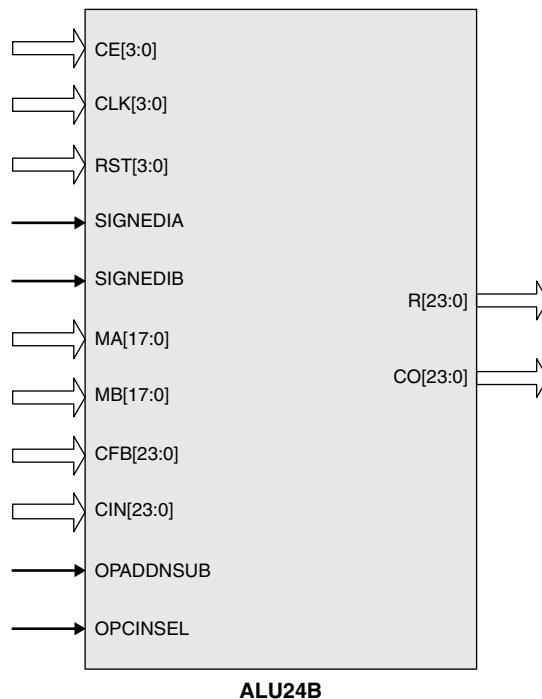
**Table 15-13. Attribute Description for ALU54A**

Attribute Name	Values	Default Value	GUI Access
REG_INPUTC0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTC0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTC0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTC1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTC1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTC1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP0_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEOP0_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP1_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEOP0_1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP1_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEIN_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEIN_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEIN_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEIN_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEIN_1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_FLAG_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_FLAG_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_FLAG_RST	RST0, RST1, RST2, RST3	RST0	Y
MCPAT_SOURCE	STATIC, DYNAMIC	STATIC	Y
MASKPAT_SOURCE	STATIC, DYNAMIC	STATIC	Y
MASK01	0x0000000000000000 to 0xFFFFFFFFFFFFFF	0x0000000000000000	Y
MCPAT	0x0000000000000000 to 0xFFFFFFFFFFFFFF	0x0000000000000000	Y
MASKPAT	0x0000000000000000 to 0xFFFFFFFFFFFFFF	0x0000000000000000	Y
RNDPAT	0x0000000000000000 to 0xFFFFFFFFFFFFFF	0x0000000000000000	Y
GSR	ENABLED, DISABLED	ENABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y
MULT9_MODE	ENABLED, DISABLED	DISABLED	N
LEGACY	ENABLED, DISABLED	DISABLED	Y
FORCE_ZERO_BARREL_SHIFT	ENABLED, DISABLED	DISABLED	N

## ALU24B – 24-bit Ternary Adder/ Subtractor for 9X9 Mode

Figure 15-13 shows the ALU24B primitive.

**Figure 15-13. ALU24B Primitive**



### ALU24B – I/O Port Description

Table 15-14 describes the port list for ALU24B primitive.

**Table 15-14. ALU24B I/O Port Description**

Port	I/O	Description
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SIGNEDIA	I	Sign Bit for Input A
SIGNEDIB	I	Sign Bit for Input B
MA[17:0]	I	Input A
MB[17:0]	I	Input B
CFB[23:0]	I	C Input for Highspeed
CIN[23:0]	I	Carry In Input
OPADDNSUB	I	Add/Sub Selector
OPCINSEL	I	CarryIn Selector
R[23:0]	O	Sum
CO[23:0]	O	Sum – Special Routing output used for Highspeed option

**ALU24B – Attribute Description**

Table 15-15 describes the attributes for ALU24B primitive.

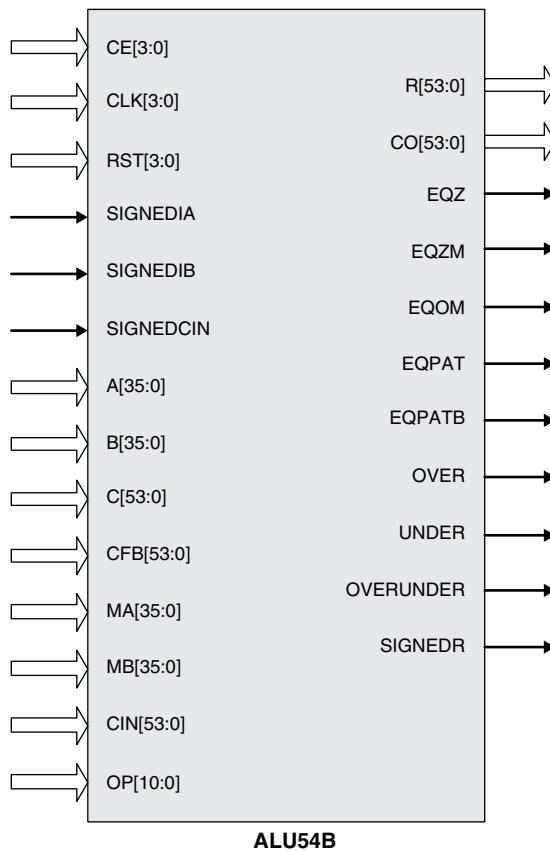
**Table 15-15. Attribute Description for ALU24B**

Attribute Name	Values	Default Value	GUI Access
REG_OUTPUT_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODE_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODE_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODE_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODE_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODE_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODE_1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTCFB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTCFB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTCFB_RST	RST0, RST1, RST2, RST3	RST0	Y
CLK0_DIV	ENABLED, DISABLED	ENABLED	Y
CLK1_DIV	ENABLED, DISABLED	ENABLED	Y
CLK2_DIV	ENABLED, DISABLED	ENABLED	Y
CLK3_DIV	ENABLED, DISABLED	ENABLED	Y
RESETMODE	SYNC, ASYNC	SYNC	Y
GSR	ENABLED, DISABLED	ENABLED	N

## **ALU54B – 54-bit Ternary Adder/ Subtractor for High Speed**

Figure 15-14 shows the ALU54B primitive

**Figure 15-14. ALU54B Primitive**



### **ALU54B – I/O Port Description**

Table 15-16 describes the port list for ALU54B primitive.

**Table 15-16. ALU54B I/O Port Description**

Port	I/O	Description
CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	I	Clock Inputs
RST[3:0]	I	Reset Inputs
SIGNEDIA	I	Sign Bit for Input A
SIGNEDIB	I	Sign Bit for Input B
SIGNEDCIN	I	Sign Bit for Carry In Input
A[35:0]	I	Input A
B[35:0]	I	Input B
C[53:0]	I	Carry In Input /Highspeed Input
CFB[53:0]	I	C Input for Highspeed
MA[35:0]	I	Input A
MB[35:0]	I	Input B
CIN[53:0]	I	Carry In Input
OP[10:0]	I	Opcode
R[53:0]	O	Sum
CO[53:0]	O	Sum – Special Routing output used for Highspeed option
EQZ	O	Equal to Zero Flag
EQZM	O	Equal to Zero with Mask Flag
EQOM	O	Equal to One with Mask Flag
EQPAT	O	Equal to Pattern with Mask Flag
EQPATB	O	Equal to Bit Inverted Pattern with Mask Flag
OVER	O	Accumulator Overflow
UNDER	O	Accumulator Underflow
OVERUNDER	O	Either Over or Underflow (may be removed)
SIGNEDR	O	Sign Bit for Sum Output

**ALU54B – Attribute Description**

Table 15-17 describes the attributes for ALU54B primitive.

**Table 15-17. Attribute Description for ALU54B**

Attribute Name	Values	Default Value	GUI Access
REG_INPUTC0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTC0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTC0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_INPUTC1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTC1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTC1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP0_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEOP0_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP1_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEOP0_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEOP0_1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEOP1_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEIN_0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEIN_0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEIN_0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OPCODEIN_1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OPCODEIN_1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OPCODEIN_1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT0_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT0_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT0_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_OUTPUT1_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_OUTPUT1_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_OUTPUT1_RST	RST0, RST1, RST2, RST3	RST0	Y
REG_FLAG_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_FLAG_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_FLAG_RST	RST0, RST1, RST2, RST3	RST0	Y
MCPAT_SOURCE	STATIC, DYNAMIC	STATIC	Y
MASKPAT_SOURCE	STATIC, DYNAMIC	STATIC	Y
MASK01	0x0000000000000000 to 0xFFFFFFFFFFFFFF	0x0000000000000000	Y
REG_INPUTCFB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y
REG_INPUTCFB_CE	CE0, CE1, CE2, CE3	CE0	Y
REG_INPUTCFB_RST	RST0, RST1, RST2, RST3	RST0	Y
CLK0_DIV	ENABLED, DISABLED	ENABLED	Y
CLK1_DIV	ENABLED, DISABLED	ENABLED	Y
CLK2_DIV	ENABLED, DISABLED	ENABLED	Y
CLK3_DIV	ENABLED, DISABLED	ENABLED	Y
MCPAT	0x0000000000000000 to 0xFFFFFFFFFFFFFF	0x0000000000000000	Y

Attribute Name	Values	Default Value	GUI Access
MASKPAT	0x0000000000000000 to 0xFFFFFFFFFFFFFF	0x0000000000000000	Y
RNDPAT	0x0000000000000000 to 0xFFFFFFFFFFFFFF	0x0000000000000000	Y
GSR	ENABLED, DISABLED	ENABLED	N
RESETMODE	SYNC, ASYNC	SYNC	Y
MULT9_MODE	ENABLED, DISABLED	DISABLED	N
FORCE_ZERO_BARREL_SHIFT	ENABLED, DISABLED	DISABLED	N
LEGACY	ENABLED, DISABLED	DISABLED	Y

In case of ALU54B, it has to be noted that the REG\_INPUT\_C0 corresponds to the lower 27 bits of the C Input, REG\_INPUT\_C1 corresponds to the upper 27 bits of the C Input, and REG\_OUTPUT0\* corresponds to [17:0] of R and REG\_OUTPUT1\_\* corresponds to [53:18] of R.

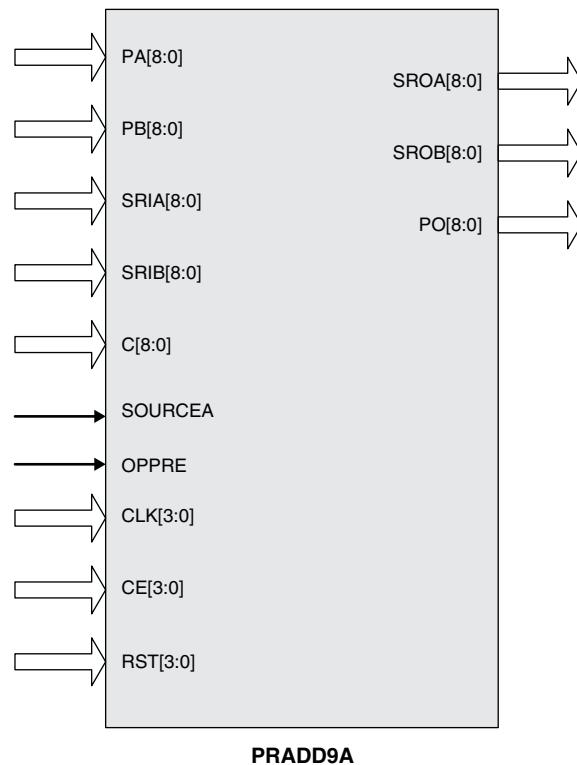
Also, when REG\_INPUTCFB\_CLK = NONE, it means that the CFB ports are not used, and C -> Cr uses the C0/C1\_CLK attributes.

When REG\_INPUTCFB\_CLK != NONE, the CFB ports are being used, CFB -> CO is using these attributes and C -> Cr is unregistered.

## PRADD9A – 9-bit Pre-Adder/Shift

Figure 15-15 shows the PRADD9A primitive.

**Figure 15-15. PRADD9A Primitive**



### PRADD9A – I/O Port Description

Table 15-18 describes the port list for PRADD9A primitive.

**Table 15-18. PRADD9A I/O Port Description**

Port	Tspec Port	I/O	Description
CE[3:0]	CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	CLK[3:0]	I	Clock Inputs
RST[3:0]	RST[3:0]	I	Reset Inputs
SOURCEA	SOURCEA	I	Source Selector for Pre-adder Input A
PA[8:0]	MUA0/A1/A2/A3[8:0]	I	Pre-adder Parallel Input A
PB[8:0]	MUB0/B1/B2/B3[8:0]	I	Pre-adder Parallel Input B
SRIA[8:0]	SRIA[8:0]	I	Pre-adder Shift Input A
SRIB[8:0]	SRI_PRE[8:0]	I	Pre-adder Shift Input B, backward direction
C[8:0]	C[8:0]/C[35:27]	I	Input used for high-speed option
SROA[8:0]	SROA[8:0]	O	Pre-adder Shift Output A
SROB[8:0]	SRO_PRE[8:0]	O	Pre-adder Shift Output B
PO[8:0]	OPA0	O	Pre-adder Addition Output
OPPRE	OP_PRE	I	Opcode for PreAdder

### **PRADD9A – Attribute Description**

Table 15-19 describes the attributes for PRADD9A primitive.

**Table 15-19. Attribute Description for PRADD9A**

Attribute Name	Values	Default Value	GUI Access	Tspec Name
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0	Y	
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0	Y	
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0	Y	
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0	Y	
REG_INPUTC_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	
REG_INPUTC_CE	CE0, CE1, CE2, CE3	CE0	Y	
REG_INPUTC_RST	RST0, RST1, RST2, RST3	RST0	Y	
REG_OPPRE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	
REG_OPPRE_CE	CE0, CE1, CE2, CE3	CE0	Y	
REG_OPPRE_RST	RST0, RST1, RST2, RST3	RST0	Y	
CLK0_DIV	ENABLED, DISABLED	ENABLED	Y	
CLK1_DIV	ENABLED, DISABLED	ENABLED	Y	
CLK2_DIV	ENABLED, DISABLED	ENABLED	Y	
CLK3_DIV	ENABLED, DISABLED	ENABLED	Y	
HIGHSPEED_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	
GSR	ENABLED, DISABLED	ENABLED	N	
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y	
SOURCEA_MODE	A_SHIFT, C_SHIFT, A_C_DYNAMIC, HIGHSPEED	A_SHIFT	Y	
SOURCEB_MODE	SHIFT, PARALLEL, INTERNAL	SHIFT	Y	mc1_pa_b0
FB_MUX	SHIFT, SHIFT_BYPASS, DISABLED	SHIFT	Y	mc1_pa_fb
RESETMODE	SYNC, ASYNC	SYNC	Y	
SYMMETRY_MODE	DIRECT, INTERNAL	DIRECT	Y	MUX_PA0/1/2/3

In the case of PRADD9A, you can also select the source for the input B. The details of SOURCEB\_MODE Attribute for PRADD9A Primitive are given in Table 15-20. The other Source mode attributes and the Feedback Mux information are also included in Table 15-20.

**Table 15-20. SOURCEB\_MODE Attribute for PRADD9A Primitive**

IP Express Operation	SOURCEA_MODE			
Attribute	SOURCEA Port	Mc1_pa_mux3	Mc1_pa_mux4	
Shift	A_SHIFT	1	00	01
A	A_SHIFT	0	00	00
C	C_SHIFT	0	01	00
A/C Dynamic	A_C_DYNAMIC	Live	10	00
HighspeedAC	HIGHSPEED	0	11	00
Dynamic Shift/A	A_SHIFT	Live	00	10
Dynamic Shift/C	C_SHIFT	Live	01	10

**Table 15-21. Details of SOURCEB\_MODE Attribute**

SOURCEB_MODE Attribute	Operation (mc1_pa_b0 mux)
SHIFT	SRIB coming from the adjacent PREADDER on the right
PARALLEL	PB
INTERNAL	Output of Reg. 12

**Table 15-22. Details of FB\_MUX Attribute**

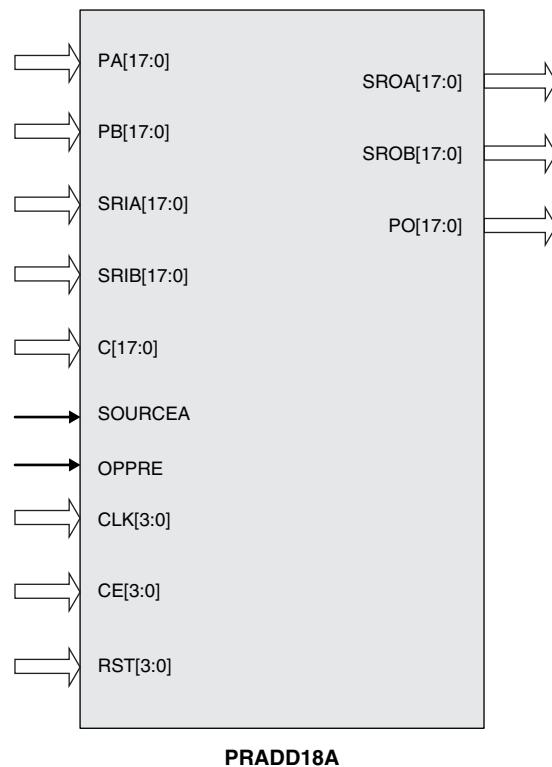
FB_MUX Attribute	Operation (MUX_FB0)
SHIFT	Output of Reg. 16
SHIFT_BYPASS	Output of Reg. 15
DISABLED	For placer only (PreAdder on the left side)

While using the PRADD9A primitive, it should be noted that each of the primitive can only drive PRADD9A in the adjacent column and/or MULT9X9D in the same column.

## PRADD18A – 18-bit Pre-Adder/Shift

Figure 15-16 shows the PRADD18A primitive

**Figure 15-16. PRADD18A Primitive**



## PRADD9A – I/O Port Description

The Table 15-23 describes the port list for PRADD18A primitive.

**Table 15-23. PRADD18A I/O Port Description**

Port	Tspec Port	I/O	Description
CE[3:0]	CE[3:0]	I	Clock Enable Inputs
CLK[3:0]	CLK[3:0]	I	Clock Inputs
RST[3:0]	RST[3:0]	I	Reset Inputs
SOURCEA	SOURCEA_PRE[1:0]	I	Source Selector for Pre-adder Input A
PA[17:0]	MUA0/A1/A2/A3[17:0]	I	Pre-adder Parallel Input A
PB[17:0]	MUB0/B1/B2/B3[17:0]	I	Pre-adder Parallel Input B
SRIA[17:0]	SRIA[17:0]	I	Pre-adder Shift Input A
SRIB[17:0]	SRI_PRE[17:0]	I	Pre-adder Shift Input A, backward direction
C[17:0]	C[17:0]/C[47:27]	I	Input used for high-speed option
SROA[17:0]	SROA[17:0]	O	Pre-adder Shift Output A
SROB[17:0]	SRO_PRE[17:0]	O	Pre-adder Shift Output B
PO[17:0]	OPA0	O	Pre-adder Addition Output
OPPRE	OP_PRE	I	Opcode for PreAdder

**PRADD9A – Attribute Description**

The Table 15-24 describes the attributes for PRADD18A primitive.

**Table 15-24. Attribute Description for PRADD18A**

Attribute Name	Values	Default Value	GUI Access	Tspec Name
REG_INPUTA_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	
REG_INPUTA_CE	CE0, CE1, CE2, CE3	CE0	Y	
REG_INPUTA_RST	RST0, RST1, RST2, RST3	RST0	Y	
REG_INPUTB_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	
REG_INPUTB_CE	CE0, CE1, CE2, CE3	CE0	Y	
REG_INPUTB_RST	RST0, RST1, RST2, RST3	RST0	Y	
REG_INPUTC_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	
REG_INPUTC_CE	CE0, CE1, CE2, CE3	CE0	Y	
REG_INPUTC_RST	RST0, RST1, RST2, RST3	RST0	Y	
REG_OPPRE_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	
REG_OPPRE_CE	CE0, CE1, CE2, CE3	CE0	Y	
REG_OPPRE_RST	RST0, RST1, RST2, RST3	RST0	Y	
CLK0_DIV	ENABLED, DISABLED	ENABLED	Y	
CLK1_DIV	ENABLED, DISABLED	ENABLED	Y	
CLK2_DIV	ENABLED, DISABLED	ENABLED	Y	
CLK3_DIV	ENABLED, DISABLED	ENABLED	Y	
HIGHSPEED_CLK	NONE, CLK0, CLK1, CLK2, CLK3	NONE	Y	
GSR	ENABLED, DISABLED	ENABLED	N	
CAS_MATCH_REG	TRUE, FALSE	FALSE	Y	
SOURCEA_MODE	A_SHIFT, C_SHIFT, A_C_DYNAMIC, HIGHSPEED	A_SHIFT	Y	
SOURCEB_MODE	SHIFT, PARALLEL, INTERNAL	SHIFT	Y	mc1_pa_b_0
FB_MUX	SHIFT, SHIFT_BYPASS, DISABLED	SHIFT	Y	mc1_pa_f_b
RESETMODE	SYNC, ASYNC	SYNC	Y	
PRADD_LOC	0, 1	0	Y	
SYMMETRY_MODE	DIRECT, INTERNAL	DIRECT	Y	MUX_PA0/1/2/3

In case of PRADD18A, you can also select the source for the input B. The details of SOURCEB\_MODE Attribute for PRADD18A Primitive, as given in the Table 15-25. The other Source mode attributes and the Feedback Mux information is also includes the tables that follow.

**Table 15-25. Details of SOURCEA\_MODE Attribute**

Clarity Designer Operation	SOURCEA_MODE Attribute	SOURCEA Port	Mc1_pa_mux3	Mc1_pa_mux4
Shift	A_SHIFT	1	00	01
A	A_SHIFT	0	00	00
C	C_SHIFT	0	01	00
A/C Dynamic	A_C_DYNAMIC	Live	10	00
HighspeedAC	HIGHSPEED	0	11	00
Dynamic Shift/A	A_SHIFT	Live	00	10
Dynamic Shift/C	C_SHIFT	Live	01	10

**Table 15-26. Details of SOURCEB\_MODE Attribute**

SOURCEB_MODE Attribute	Operation (mc1_pa_b0 mux)
SHIFT	SRIB coming from the adjacent PREADDER on the right
PARALLEL	PB
INTERNAL	Output of Reg. 12

**Table 15-27. Details of FB\_MUX Attribute**

FB_MUX Attribute	Operation (MUX_FB0)
SHIFT	Output of Reg. 16
SHIFT_BYPASS	Output of Reg. 15
DISABLED	For placer only (PreAdder on the left side)

While using the PRADD18A primitive, it should be noted that each of the primitive can only drive PRADD18A in the adjacent column and/or MULT18X18D in the same column.

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

Date	Version	Change Summary
March 2014	01.0	Initial release.

## Appendix A: Instantiating DSP Primitives in HDL

This appendix illustrates how to instantiate the ECP5 sysDSP primitives for both Verilog and VHDL.

### Verilog Example Showing Snippet of the MULT18X18D Instantiation

```

defparam dsp_mult_0.CLK3_DIV = "DISABLED" ;
defparam dsp_mult_0.CLK2_DIV = "DISABLED" ;
defparam dsp_mult_0.CLK1_DIV = "DISABLED" ;
defparam dsp_mult_0.CLK0_DIV = "DISABLED" ;
defparam dsp_mult_0.HIGHSPEED_CLK = "CLK0" ;
defparam dsp_mult_0.REG_INPUTC_RST = "RST0" ;
defparam dsp_mult_0.REG_INPUTC_CE = "CE0" ;
defparam dsp_mult_0.REG_INPUTC_CLK = "NONE" ;
defparam dsp_mult_0.SOURCEB_MODE = "B_SHIFT" ;
defparam dsp_mult_0.MULT_BYPASS = "DISABLED" ;
defparam dsp_mult_0.CAS_MATCH_REG = "FALSE" ;
defparam dsp_mult_0.RESETMODE = "SYNC" ;
defparam dsp_mult_0.GSR = "ENABLED" ;
defparam dsp_mult_0.REG_OUTPUT_RST = "RST0" ;
defparam dsp_mult_0.REG_OUTPUT_CE = "CE0" ;
defparam dsp_mult_0.REG_OUTPUT_CLK = "NONE" ;
defparam dsp_mult_0.REG_PIPELINE_RST = "RST0" ;
defparam dsp_mult_0.REG_PIPELINE_CE = "CE0" ;
defparam dsp_mult_0.REG_PIPELINE_CLK = "CLK0" ;
defparam dsp_mult_0.REG_INPUTB_RST = "RST0" ;
defparam dsp_mult_0.REG_INPUTB_CE = "CE0" ;
defparam dsp_mult_0.REG_INPUTB_CLK = "CLK0" ;
defparam dsp_mult_0.REG_INPUTA_RST = "RST0" ;
defparam dsp_mult_0.REG_INPUTA_CE = "CE0" ;
defparam dsp_mult_0.REG_INPUTA_CLK = "CLK0" ;
MULT18X18D dsp_mult_0 (
    .A17(t5M1A_17), .A16(t5M1A_16), .A15(t5M1A_15),
    .A14(t5M1A_14), .A13(t5M1A_13), .A12(t5M1A_12), .A11(t5M1A_11),
    .A10(t5M1A_10), .A9(t5M1A_9), .A8(t5M1A_8), .A7(t5M1A_7), .A6(t5M1A_6),
    .A5(t5M1A_5), .A4(t5M1A_4), .A3(t5M1A_3), .A2(t5M1A_2), .A1(t5M1A_1),
    .A0(t5M1A_0), .B17(scuba_vlo), .B16(scuba_vlo), .B15(scuba_vlo),
    .B14(scuba_vlo), .B13(scuba_vlo), .B12(scuba_vlo), .B11(scuba_vlo),
    .B10(scuba_vlo), .B9(scuba_vlo), .B8(scuba_vlo), .B7(scuba_vlo),
    .B6(scuba_vlo), .B5(scuba_vlo), .B4(scuba_vlo), .B3(scuba_vlo),
    .B2(scuba_vlo), .B1(scuba_vlo), .B0(scuba_vlo), .C17(scuba_vlo),
    .C16(scuba_vlo), .C15(scuba_vlo), .C14(scuba_vlo), .C13(scuba_vlo),
    .C12(scuba_vlo), .C11(scuba_vlo), .C10(scuba_vlo), .C9(scuba_vlo),
    .C8(scuba_vlo), .C7(scuba_vlo), .C6(scuba_vlo), .C5(scuba_vlo),
    .C4(scuba_vlo), .C3(scuba_vlo), .C2(scuba_vlo), .C1(scuba_vlo),
    .C0(scuba_vlo), .SIGNEDA(scuba_vhi), .SIGNEDB(scuba_vhi), .SOURCEA(scuba_vlo),
    .SOURCEB(scuba_vlo), .CE0(ClockEn), .CE1(scuba_vlo), .CE2(scuba_vlo),
    .CE3(scuba_vlo), .CLK0(Clock), .CLK1(Clock_inv), .CLK2(scuba_vlo),
    .CLK3(scuba_vlo), .RST0(Rest), .RST1(scuba_vlo), .RST2(scuba_vlo),
    .RST3(scuba_vlo), .SRIA17(scuba_vlo), .SRIA16(scuba_vlo), .SRIA15(scuba_vlo),
    .SRIA14(scuba_vlo), .SRIA13(scuba_vlo), .SRIA12(scuba_vlo), .SRIA11(scuba_vlo),
    .SRIA10(scuba_vlo), .SRIA9(scuba_vlo), .SRIA8(scuba_vlo), .SRIA7(scuba_vlo),
    .SRIA6(scuba_vlo), .SRIA5(scuba_vlo), .SRIA4(scuba_vlo), .SRIA3(scuba_vlo),
    .SRIA2(scuba_vlo), .SRIA1(scuba_vlo), .SRIA0(scuba_vlo), .SRIB17(scuba_vlo),
    .SRIB16(scuba_vlo), .SRIB15(scuba_vlo), .SRIB14(scuba_vlo), .SRIB13(scuba_vlo),

```

---

```

.SRIB12(scuba_vlo), .SRIB11(scuba_vlo), .SRIB10(scuba_vlo), .SRIB9(scuba_vlo),
.SRIB8(scuba_vlo), .SRIB7(scuba_vlo), .SRIB6(scuba_vlo), .SRIB5(scuba_vlo),
.SRIB4(scuba_vlo), .SRIB3(scuba_vlo), .SRIB2(scuba_vlo), .SRIB1(scuba_vlo),
.SRIB0(scuba_vlo), .SROA17(), .SROA16(), .SROA15(), .SROA14(), .SROA13(),
.SROA12(), .SROA11(), .SROA10(), .SROA9(), .SROA8(), .SROA7(), .SROA6(),
.SROA5(), .SROA4(), .SROA3(), .SROA2(), .SROA1(), .SROA0(), .SROB17(),
.SROB16(), .SROB15(), .SROB14(), .SROB13(), .SROB12(), .SROB11(),
.SROB10(), .SROB9(), .SROB8(), .SROB7(), .SROB6(), .SROB5(), .SROB4(),
.SROB3(), .SROB2(), .SROB1(), .SROB0(), .ROA17(roa1_5_17), .ROA16(roa1_5_16),
.ROA15(roa1_5_15), .ROA14(roa1_5_14), .ROA13(roa1_5_13), .ROA12(roa1_5_12),
.ROA11(roa1_5_11), .ROA10(roa1_5_10), .ROA9(roa1_5_9), .ROA8(roa1_5_8),
.ROA7(roa1_5_7), .ROA6(roa1_5_6), .ROA5(roa1_5_5), .ROA4(roa1_5_4),
.ROA3(roa1_5_3), .ROA2(roa1_5_2), .ROA1(roa1_5_1), .ROA0(roa1_5_0),
.ROB17(rob1_5_17), .ROB16(rob1_5_16), .ROB15(rob1_5_15), .ROB14(rob1_5_14),
.ROB13(rob1_5_13), .ROB12(rob1_5_12), .ROB11(rob1_5_11), .ROB10(rob1_5_10),
.ROB9(rob1_5_9), .ROB8(rob1_5_8), .ROB7(rob1_5_7), .ROB6(rob1_5_6),
.ROB5(rob1_5_5), .ROB4(rob1_5_4), .ROB3(rob1_5_3), .ROB2(rob1_5_2),
.ROB1(rob1_5_1), .ROB0(rob1_5_0), .ROC17(), .ROC16(), .ROC15(),
.ROC14(), .ROC13(), .ROC12(), .ROC11(), .ROC10(), .ROC9(), .ROC8(),
.ROC7(), .ROC6(), .ROC5(), .ROC4(), .ROC3(), .ROC2(), .ROC1(), .ROC0(),
.P35(t5P1_35), .P34(t5P1_34), .P33(t5P1_33), .P32(t5P1_32), .P31(t5P1_31),
.P30(t5P1_30), .P29(t5P1_29), .P28(t5P1_28), .P27(t5P1_27), .P26(t5P1_26),
.P25(t5P1_25), .P24(t5P1_24), .P23(t5P1_23), .P22(t5P1_22), .P21(t5P1_21),
.P20(t5P1_20), .P19(t5P1_19), .P18(t5P1_18), .P17(t5P1_17), .P16(t5P1_16),
.P15(t5P1_15), .P14(t5P1_14), .P13(t5P1_13), .P12(t5P1_12), .P11(t5P1_11),
.P10(t5P1_10), .P9(t5P1_9), .P8(t5P1_8), .P7(t5P1_7), .P6(t5P1_6),
.P5(t5P1_5), .P4(t5P1_4), .P3(t5P1_3), .P2(t5P1_2), .P1(t5P1_1),
.P0(t5P1_0), .SIGNEDP(m5_signedp1)
);

```

## VHDL Example Showing Snippet of the ALU54B Instantiation

```

dsp_alu_0: ALU54B
generic map (
    CLK3_DIV=> "DISABLED", CLK2_DIV=> "DISABLED",
    CLK1_DIV=> "DISABLED", CLK0_DIV=> "DISABLED", REG_INPUTCFB_RST=> "RST0",
    REG_INPUTCFB_CE=> "CE0", REG_INPUTCFB_CLK=> "CLK1",
    REG_OPCODEIN_1_RST=> "RST0", REG_OPCODEIN_1_CE=> "CE0",
    REG_OPCODEIN_1_CLK=> "NONE", REG_OPCODEIN_0_RST=> "RST0",
    REG_OPCODEIN_0_CE=> "CE0", REG_OPCODEIN_0_CLK=> "NONE",
    REG_OPCODEOP1_1_CLK=> "NONE", REG_OPCODEOP1_0_CLK=> "NONE",
    REG_OPCODEOP0_1_RST=> "RST0", REG_OPCODEOP0_1_CE=> "CE0",
    REG_OPCODEOP0_1_CLK=> "NONE", REG_OPCODEOP0_0_RST=> "RST0",
    REG_OPCODEOP0_0_CE=> "CE0", REG_OPCODEOP0_0_CLK=> "NONE",
    REG_INPUTC1_RST=> "RST0", REG_INPUTC1_CE=> "CE0",
    REG_INPUTC1_CLK=> "NONE", REG_INPUTC0_RST=> "RST0",
    REG_INPUTC0_CE=> "CE0", REG_INPUTC0_CLK=> "NONE", LEGACY=> "DISABLED",
    REG_FLAG_RST=> "RST0", REG_FLAG_CE=> "CE0", REG_FLAG_CLK=> "NONE",
    REG_OUTPUT1_RST=> "RST0", REG_OUTPUT1_CE=> "CE0",
    REG_OUTPUT1_CLK=> "CLK0", REG_OUTPUT0_RST=> "RST0",
    REG_OUTPUT0_CE=> "CE0", REG_OUTPUT0_CLK=> "CLK0", MULT9_MODE=> "DISABLED",
    RNDPAT=> "0x0000000000000000", MASKPAT=> "0x0000000000000000", MCPAT=> "0x0000000000000000",
    MASK01=> "0x0000000000000000", MASKPAT_SOURCE=> "STATIC",
    MCPAT_SOURCE=> "STATIC", RESETMODE=> "SYNC", GSR=> "ENABLED"
)
port map (
    A35=>rob0_5_17, A34=>rob0_5_16, A33=>rob0_5_15,
    A32=>rob0_5_14, A31=>rob0_5_13, A30=>rob0_5_12,
    A29=>rob0_5_11, A28=>rob0_5_10, A27=>rob0_5_9, A26=>rob0_5_8,
    A25=>rob0_5_7, A24=>rob0_5_6, A23=>rob0_5_5, A22=>rob0_5_4,
    A21=>rob0_5_3, A20=>rob0_5_2, A19=>rob0_5_1, A18=>rob0_5_0,
    A17=>roa0_5_17, A16=>roa0_5_16, A15=>roa0_5_15,
    A14=>roa0_5_14, A13=>roa0_5_13, A12=>roa0_5_12,
    A11=>roa0_5_11, A10=>roa0_5_10, A9=>roa0_5_9, A8=>roa0_5_8,
    A7=>roa0_5_7, A6=>roa0_5_6, A5=>roa0_5_5, A4=>roa0_5_4,
    A3=>roa0_5_3, A2=>roa0_5_2, A1=>roa0_5_1, A0=>roa0_5_0,
    B35=>rob1_5_17, B34=>rob1_5_16, B33=>rob1_5_15,
    B32=>rob1_5_14, B31=>rob1_5_13, B30=>rob1_5_12,
    B29=>rob1_5_11, B28=>rob1_5_10, B27=>rob1_5_9, B26=>rob1_5_8,
    B25=>rob1_5_7, B24=>rob1_5_6, B23=>rob1_5_5, B22=>rob1_5_4,
    B21=>rob1_5_3, B20=>rob1_5_2, B19=>rob1_5_1, B18=>rob1_5_0,
    B17=>roa1_5_17, B16=>roa1_5_16, B15=>roa1_5_15,
    B14=>roa1_5_14, B13=>roa1_5_13, B12=>roa1_5_12,
    B11=>roa1_5_11, B10=>roa1_5_10, B9=>roa1_5_9, B8=>roa1_5_8,
    B7=>roa1_5_7, B6=>roa1_5_6, B5=>roa1_5_5, B4=>roa1_5_4,
    B3=>roa1_5_3, B2=>roa1_5_2, B1=>roa1_5_1, B0=>roa1_5_0,
    CFB53=>r5_53, CFB52=>r5_52, CFB51=>r5_51, CFB50=>r5_50,
    CFB49=>r5_49, CFB48=>r5_48, CFB47=>r5_47, CFB46=>r5_46,
    CFB45=>r5_45, CFB44=>r5_44, CFB43=>r5_43, CFB42=>r5_42,
    CFB41=>r5_41, CFB40=>r5_40, CFB39=>r5_39, CFB38=>r5_38,
    CFB37=>r5_37, CFB36=>r5_36, CFB35=>r5_35, CFB34=>r5_34,
    CFB33=>r5_33, CFB32=>r5_32, CFB31=>r5_31, CFB30=>r5_30,
    CFB29=>r5_29, CFB28=>r5_28, CFB27=>r5_27, CFB26=>r5_26,
)

```

```

CFB25=>r5_25, CFB24=>r5_24, CFB23=>r5_23, CFB22=>r5_22,
CFB21=>r5_21, CFB20=>r5_20, CFB19=>r5_19, CFB18=>r5_18,
CFB17=>r5_17, CFB16=>r5_16, CFB15=>r5_15, CFB14=>r5_14,
CFB13=>r5_13, CFB12=>r5_12, CFB11=>r5_11, CFB10=>r5_10,
CFB9=>r5_9, CFB8=>r5_8, CFB7=>r5_7, CFB6=>r5_6, CFB5=>r5_5,
CFB4=>r5_4, CFB3=>r5_3, CFB2=>r5_2, CFB1=>r5_1, CFB0=>r5_0,
C53=>scuba_vlo, C52=>scuba_vlo, C51=>scuba_vlo,
C50=>scuba_vlo, C49=>scuba_vlo, C48=>scuba_vlo,
C47=>scuba_vlo, C46=>scuba_vlo, C45=>scuba_vlo,
C44=>scuba_vlo, C43=>scuba_vlo, C42=>scuba_vlo,
C41=>scuba_vlo, C40=>scuba_vlo, C39=>scuba_vlo,
C38=>scuba_vlo, C37=>scuba_vlo, C36=>scuba_vlo,
C35=>scuba_vlo, C34=>scuba_vlo, C33=>scuba_vlo,
C32=>scuba_vlo, C31=>scuba_vlo, C30=>scuba_vlo,
C29=>scuba_vlo, C28=>scuba_vlo, C27=>scuba_vlo,
C26=>scuba_vlo, C25=>scuba_vlo, C24=>scuba_vlo,
C23=>scuba_vlo, C22=>scuba_vlo, C21=>scuba_vlo,
C20=>scuba_vlo, C19=>scuba_vlo, C18=>scuba_vlo,
C17=>scuba_vlo, C16=>scuba_vlo, C15=>scuba_vlo,
C14=>scuba_vlo, C13=>scuba_vlo, C12=>scuba_vlo,
C11=>scuba_vlo, C10=>scuba_vlo, C9=>scuba_vlo, C8=>scuba_vlo,
C7=>scuba_vlo, C6=>scuba_vlo, C5=>scuba_vlo, C4=>scuba_vlo,
C3=>scuba_vlo, C2=>scuba_vlo, C1=>scuba_vlo, C0=>scuba_vlo,
CE0=>ClockEn, CE1=>scuba_vlo, CE2=>scuba_vlo, CE3=>scuba_vlo,
CLK0=>Clock, CLK1=>Clock_inv, CLK2=>scuba_vlo,
CLK3=>scuba_vlo, RST0=>Reset, RST1=>scuba_vlo,
RST2=>scuba_vlo, RST3=>scuba_vlo, SIGNEDIA=>m5_signedp0,
SIGNEDIB=>m5_signedp1, SIGNEDCIN=>signr4, MA35=>t5P0_35,
MA34=>t5P0_34, MA33=>t5P0_33, MA32=>t5P0_32, MA31=>t5P0_31,
MA30=>t5P0_30, MA29=>t5P0_29, MA28=>t5P0_28, MA27=>t5P0_27,
MA26=>t5P0_26, MA25=>t5P0_25, MA24=>t5P0_24, MA23=>t5P0_23,
MA22=>t5P0_22, MA21=>t5P0_21, MA20=>t5P0_20, MA19=>t5P0_19,
MA18=>t5P0_18, MA17=>t5P0_17, MA16=>t5P0_16, MA15=>t5P0_15,
MA14=>t5P0_14, MA13=>t5P0_13, MA12=>t5P0_12, MA11=>t5P0_11,
MA10=>t5P0_10, MA9=>t5P0_9, MA8=>t5P0_8, MA7=>t5P0_7,
MA6=>t5P0_6, MA5=>t5P0_5, MA4=>t5P0_4, MA3=>t5P0_3,
MA2=>t5P0_2, MA1=>t5P0_1, MA0=>t5P0_0, MB35=>t5P1_35,
MB34=>t5P1_34, MB33=>t5P1_33, MB32=>t5P1_32, MB31=>t5P1_31,
MB30=>t5P1_30, MB29=>t5P1_29, MB28=>t5P1_28, MB27=>t5P1_27,
MB26=>t5P1_26, MB25=>t5P1_25, MB24=>t5P1_24, MB23=>t5P1_23,
MB22=>t5P1_22, MB21=>t5P1_21, MB20=>t5P1_20, MB19=>t5P1_19,
MB18=>t5P1_18, MB17=>t5P1_17, MB16=>t5P1_16, MB15=>t5P1_15,
MB14=>t5P1_14, MB13=>t5P1_13, MB12=>t5P1_12, MB11=>t5P1_11,
MB10=>t5P1_10, MB9=>t5P1_9, MB8=>t5P1_8, MB7=>t5P1_7,
MB6=>t5P1_6, MB5=>t5P1_5, MB4=>t5P1_4, MB3=>t5P1_3,
MB2=>t5P1_2, MB1=>t5P1_1, MB0=>t5P1_0, CIN53=>r4_53,
CIN52=>r4_52, CIN51=>r4_51, CIN50=>r4_50, CIN49=>r4_49,
CIN48=>r4_48, CIN47=>r4_47, CIN46=>r4_46, CIN45=>r4_45,
CIN44=>r4_44, CIN43=>r4_43, CIN42=>r4_42, CIN41=>r4_41,
CIN40=>r4_40, CIN39=>r4_39, CIN38=>r4_38, CIN37=>r4_37,
CIN36=>r4_36, CIN35=>r4_35, CIN34=>r4_34, CIN33=>r4_33,
CIN32=>r4_32, CIN31=>r4_31, CIN30=>r4_30, CIN29=>r4_29,
CIN28=>r4_28, CIN27=>r4_27, CIN26=>r4_26, CIN25=>r4_25,
CIN24=>r4_24, CIN23=>r4_23, CIN22=>r4_22, CIN21=>r4_21,

```

```
CIN20=>r4_20, CIN19=>r4_19, CIN18=>r4_18, CIN17=>r4_17,  
CIN16=>r4_16, CIN15=>r4_15, CIN14=>r4_14, CIN13=>r4_13,  
CIN12=>r4_12, CIN11=>r4_11, CIN10=>r4_10, CIN9=>r4_9,  
CIN8=>r4_8, CIN7=>r4_7, CIN6=>r4_6, CIN5=>r4_5, CIN4=>r4_4,  
CIN3=>r4_3, CIN2=>r4_2, CIN1=>r4_1, CIN0=>r4_0,  
OP10=>scuba_vlo, OP9=>scuba_vhi, OP8=>scuba_vlo,  
OP7=>scuba_vlo, OP6=>scuba_vlo, OP5=>scuba_vhi,  
OP4=>scuba_vlo, OP3=>scuba_vhi, OP2=>scuba_vhi,  
OP1=>scuba_vhi, OP0=>scuba_vhi, R53=>r5_53, R52=>r5_52,  
R51=>r5_51, R50=>r5_50, R49=>r5_49, R48=>r5_48, R47=>r5_47,  
R46=>r5_46, R45=>r5_45, R44=>r5_44, R43=>r5_43, R42=>r5_42,  
R41=>r5_41, R40=>r5_40, R39=>r5_39, R38=>r5_38, R37=>r5_37,  
R36=>r5_36, R35=>r5_35, R34=>r5_34, R33=>r5_33, R32=>r5_32,  
R31=>r5_31, R30=>r5_30, R29=>r5_29, R28=>r5_28, R27=>r5_27,  
R26=>r5_26, R25=>r5_25, R24=>r5_24, R23=>r5_23, R22=>r5_22,  
R21=>r5_21, R20=>r5_20, R19=>r5_19, R18=>r5_18, R17=>r5_17,  
R16=>r5_16, R15=>r5_15, R14=>r5_14, R13=>r5_13, R12=>r5_12,  
R11=>r5_11, R10=>r5_10, R9=>r5_9, R8=>r5_8, R7=>r5_7,  
R6=>r5_6, R5=>r5_5, R4=>r5_4, R3=>r5_3, R2=>r5_2, R1=>r5_1,  
R0=>r5_0, CO53=>Result1(53), CO52=>Result1(52),  
CO51=>Result1(51), CO50=>Result1(50), CO49=>Result1(49),  
CO48=>Result1(48), CO47=>Result1(47), CO46=>Result1(46),  
CO45=>Result1(45), CO44=>Result1(44), CO43=>Result1(43),  
CO42=>Result1(42), CO41=>Result1(41), CO40=>Result1(40),  
CO39=>Result1(39), CO38=>Result1(38), CO37=>Result1(37),  
CO36=>Result1(36), CO35=>Result1(35), CO34=>Result1(34),  
CO33=>Result1(33), CO32=>Result1(32), CO31=>Result1(31),  
CO30=>Result1(30), CO29=>Result1(29), CO28=>Result1(28),  
CO27=>Result1(27), CO26=>Result1(26), CO25=>Result1(25),  
CO24=>Result1(24), CO23=>Result1(23), CO22=>Result1(22),  
CO21=>Result1(21), CO20=>Result1(20), CO19=>Result1(19),  
CO18=>Result1(18), CO17=>Result1(17), CO16=>Result1(16),  
CO15=>Result1(15), CO14=>Result1(14), CO13=>Result1(13),  
CO12=>Result1(12), CO11=>Result1(11), CO10=>Result1(10),  
CO9=>Result1(9), CO8=>Result1(8), CO7=>Result1(7),  
CO6=>Result1(6), CO5=>Result1(5), CO4=>Result1(4),  
CO3=>Result1(3), CO2=>Result1(2), CO1=>Result1(1),  
CO0=>Result1(0), EQZ=>open, EQZM=>open, EQOM=>open,  
EQPAT=>open, EQPATB=>open, OVER=>open, UNDER=>open,  
OVERUNDER=>open, SIGNEDR=>signr5  
);
```

## Appendix B: HDL Inference for DSP

Synthesis inference flow enables the design tools to infer sysDSP slices from an HDL design. It is important to note that when using the inference flow, unless the code style matches the sysDSP slice, results will not be optimal. Users can infer the ECP5 sysDSP slice with Synplify Pro® from Synopsys or the Lattice Synthesis Engine (LSE) if certain coding guidelines are followed. The following are VHDL and Verilog examples. This example would not have functional simulation support. This is for example purposes only.

### VHDL Example to Infer Fully Pipelined Multiplier

```

library ieee;
use ieee.std_logic_1164.all;
--use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity mult is
    port (reset, clk : in std_logic;
          dataax, dataay : in std_logic_vector(8 downto 0);
          dataout : out std_logic_vector (17 downto 0));
end;

architecture arch of mult is
    signal dataax_reg, dataay_reg : std_logic_vector (8 downto 0);
    signal dataout_node : std_logic_vector (17 downto 0);
    signal dataout_pipeline : std_logic_vector (17 downto 0);

begin
    process (clk, reset)
begin
    if (reset='1') then
        dataax_reg <= (others => '0');
        dataay_reg <= (others => '0');
    elsif (clk'event and clk='1') then
        dataax_reg <= dataax;
        dataay_reg <= dataay;
    end if;
    end process;

    dataout_node <= dataax_reg * dataay_reg;
    process (clk, reset)
    begin
        if (reset='1') then
            dataout_pipeline <= (others => '0');
        elsif (clk'event and clk='1') then
            dataout_pipeline <= dataout_node;
        end if;
    end process;

    process (clk, reset)
    begin
        if (reset='1') then
            dataout <= (others => '0');
        elsif (clk'event and clk='1') then
            dataout <= dataout_pipeline;
        end if;
    end process;
end arch;.

```

## Verilog Example to Infer Fully Pipelined Multiplier

```
module mult (dataout, dataax, dataay, clk, reset);
output [35:0] dataout;
input [17:0] dataax, dataay;
input clk,reset;

reg [35:0] dataout;
reg [17:0] dataax_reg, dataay_reg;
wire [35:0] dataout_node;
reg [35:0] dataout_reg;

always @(posedge clk or posedge reset)
begin
    if (reset)
begin
    dataax_reg <= 0;
    dataay_reg <= 0;
end
else
begin
    dataax_reg <= dataax;
    dataay_reg <= dataay;
end
end

assign dataout_node = dataax_reg * dataay_reg;
always @(posedge clk or posedge reset)
begin
    if (reset)
    dataout_reg <= 0;
    else
    dataout_reg <= dataout_node;
end

always @(posedge clk or posedge reset)
begin
    if (reset)
    dataout <= 0;
    else
    dataout <= dataout_reg;
end
endmodule
```

---

March 2014

Technical Note TN1269

## Introduction

When designing complex hardware using the ECP5™ FPGA, designers must pay special attention to critical hardware configuration requirements. This technical note steps through these critical hardware implementation items relative to the ECP5 device. The document does not provide detailed step-by-step instructions but gives a high-level summary checklist to assist in the design process.

The device family consists of FPGA LUT densities ranging from 25K to 85K. This technical note assumes that the reader is familiar with the ECP5 device features as described in DS1044, [ECP5 Family Data Sheet](#). The data sheet includes the functional specification for the device. Topics covered in the data sheet include but are not limited to the following:

- High-level functional overview
- Pinouts and packaging information
- Signal descriptions
- Device-specific information about peripherals and registers
- Electrical specifications

Please refer to DS1044, [ECP5 Family Data Sheet](#) for the device-specific details.

The critical hardware areas covered in this technical note are:

- Power supplies as they relate to the ECP5 power supply rails and how to connect them to the PCB and the associated system
- Configuration mode selection for proper power-up behavior
- Device I/O interface and critical signals

*Important:* Users should refer to the following documents for detailed recommendations.

- TN1260, [ECP5 sysCONFIG Usage Guide](#)
- TN1261, [ECP5 SERDES/PCS Usage Guide](#)
- TN1262, [ECP5 sysIO Usage Guide](#)
- TN1263, [ECP5 sysClock PLL/DLL Design and Usage Guide](#)
- TN1264, [ECP5 Memory Usage Guide](#)
- TN1265, [ECP5 High-Speed I/O Interface](#)
- TN1266, [Power Consumption and Management for ECP5 Devices](#)
- TN1267, [ECP5 sysDSP Usage Guide](#)
- TN1114, [Electrical Recommendations for Lattice SERDES](#)
- TN1033, [High-Speed PCB Design Considerations](#)
- TN1068, [Decoupling and Bypass Filtering for Programmable Devices](#)
- TN1084, [LatticeSC SERDES Jitter](#)

- HSPICE SERDES CML simulation package and die models in RLGC format (available under NDA, contact the license administrator at [lic\\_admin@latticesemi.com](mailto:lic_admin@latticesemi.com))
- [ECP5-related pinout information](#) can be found on the Lattice web site.

## Power Supplies

All supplies including  $V_{CC}$ ,  $V_{CCAUX}$  and  $V_{CCIO8}$  power supplies determine the ECP5 internal “power good” condition. These supplies need to be at a valid and stable level before the device can become operational. Several other supplies including  $V_{CCA}$ ,  $V_{CCAUXA}$ ,  $V_{CCHRX}$  and  $V_{CCHTX}$  are used in conjunction with on-board SERDES on ECP5UM devices. Table 1 shows the power supplies and the appropriate voltage levels for each supply.

**Table 1. ECP5 FPGA Power Supplies**

Supply	Voltage (Nominal Value)	Description
VCC	1.1 V	FPGA core power supply
VCCA	1.1 V	Analog power supply for SERDES blocks (For ECP5UM devices). Should be isolated and “clean” from excessive noise.
VCCAUX	2.5 V	Auxiliary power supply
VCCIO[0-4, 8] <sup>1</sup>	1.2 V to 3.3 V	I/O power supply. Seven (eight on LFE5-85 in 756 and 554 caBGA) general purpose I/O banks. Each bank has its own VCCIO supply: $V_{CCIO8}$ is used in conjunction with pins dedicated and shared with device configuration, include JTAG $V_{CCIO0,1,2,3,4,6}$ , and $7$ are optionally used based on per bank usage of I/O.
VCCHRX	1.2 V	Input terminate voltage supply for SERDES inputs (For ECP5UM devices)
VCCHTX	1.2 V	CML output driver/termination voltage supply for SERDES outputs (for ECP5UM devices)
VCCAUXA	2.5 V	Auxiliary power supply for SERDES (for ECP5UM devices)

1. Bank 4 exists only on the LFE5-85 device in 756 caBGA and 554 caBGA. It is not available in any other device/package combinations.

The ECP5 FPGA device has a power-up reset state machine that depends on various power supplies.

These supplies should come up monotonically. A power-up reset counter will begin to count after all of the approximate conditions are met:

- VCC reaches 0.9 V or above
- VCCAUX reaches 2.0 V or above
- VCCIO[8] reaches 0.95 V or above

Initialization of the device will not proceed until the last power supply has reached its minimum operating voltage.

## ECP5 SERDES/PCS Power Supplies

Supplies dedicated to the operation of the SERDES/PCS include VCCA, VCCHRX, VCCHTX. All VCCA supply pins must always be powered to the recommended operating voltage range with the ECP5UM devices. However, if no SERDES is used at all in the ECP5UM device, all power supply pins for the SERDES can be connected to GND (VCCA, VCCAUXA, VCCHRX and VCCHTX).

When SERDES is used in the ECP5 devices, VCCHRX and VCCHTX can be left floating for unused SERDES channels. Unused channel outputs should be left tri-stated.

It is very important that the VCCA supply be low-noise and isolated from heavily loaded switching supplies. Please refer to [TN1114, Electrical Recommendations for Lattice SERDES](#), for recommendations.

## Power Estimation

Once the ECP5 device density, package and logic implementation is decided, power estimation for the system environment should be determined based on the software Power Calculator provided as part of the Lattice Diamond® design tool. When estimating power, the designer should keep two goals in mind:

1. Power supply budgeting should be based on the maximum of the power-up in-rush current, configuration current or maximum DC and AC current for the given system's environmental conditions.
2. The ability for the system environment and ECP5 device packaging to be able to support the specified maximum operating junction temperature. By determining these two criteria, the ECP5 device power requirements are taken into consideration early in the design phase.

## Configuration Considerations

The ECP5 device includes provisions to program the FPGA via a JTAG interface or several modes utilizing the sysCONFIG port. The JTAG port includes a 4-pin interface. The interface requires the following PCB considerations.

**Table 2. JTAG Pin Recommendations**

JTAG Pin	PCB Recommendation
TDI	4.7K Pull-up to VCCJ
TMS	4.7K Pull-up to VCCJ
TDO	4.7K Pull-up to VCCJ
TCK	4.7K Pull-down

Every PCB should have easy access to FPGA JTAG pins. This enables debugging in the final system. For best results, route the TCK, TMS, TDI and TDO signals to a common test header along with VCCJ and ground.

Using other programming modes requires the use of the CFG[2:0] input pins. For JTAG, the MODE pins are not used. The CFG[2:0] pins include weak internal pull-ups. It is recommended that 5-10K external resistors be used when using the sysCONFIG modes. Pull-up resistors should be connected to VCCIO8.

The use of external resistors is always needed if the configuration signals are being used to handshake to other devices. Recommended 4.7K pull-up resistors to VCCIO8 and pull-down to board ground should be used on the following pins.

**Table 3. Pull-up/Pull-down Recommendations for Configuration Pins**

Pin	PCB Connection
PROGRAMN	Pull-up
INITN	Pull-up
CCLK	Pull-down
CFG[0:2]	See Table 4. 1 = 4.7K pull-up, 0 = GND.

**Table 4. Configuration Pins Needed per Programming Mode**

Configuration Mode	Bus Size	Dedicated CFG[2:0]	Clock		Shared Pins	Dedicated Pins
			Pin	I/O		
SSPI	1 Bit	000	CCLK	Input	MISO, MOSI, SI, DOUT, HOLDN	PROGRAMN, INITN, DONE
MSPI	1 Bit	010	MCLK	Output	MISO, MOSI, CSSPIN, DOUT	PROGRAMN, INITN, DONE
	2 Bits				D[1:0], CSSPIN, DOUT	
	4 Bits				D[3:0], CSSPIN, DOUT	
SCM	1 Bit	101	CCLK	Input	DI, DOUT	PROGRAMN, INITN, DONE

**Table 4. Configuration Pins Needed per Programming Mode (Continued)**

Configuration Mode	Bus Size	Dedicated CFG[2:0]	Clock		Shared Pins	Dedicated Pins
			Pin	I/O		
SPCM (Parallel)	8 Bits	111	CCLK	Input	D[7:0], DOUT, CSON, BUSY, WRITEN, CSN, CS1N	PROGRAMN, INITN, DONE
JTAG	1 Bit	xxx	TCK	Input	N/A	TCK, TMS, TDI, TDO

## I/O Pin Assignments

The VCCA provides a “quiet” supply for the internal PLLs and critical SERDES blocks. For the best jitter performance, careful pin assignment will keep “noisy” I/O pins away from “sensitive” pins. The leading causes of PCB related SERDES crosstalk is related to FPGA outputs located in close proximity to the sensitive SERDES power supplies. These supplies require cautious board layout to insure noise immunity to the switching noise generated by FPGA outputs. Guidelines are provided to build quiet filtered supplies for the VCCA, however robust PCB layout is required to insure that noise does not infiltrate into these analog supplies.

Although coupling has been reduced in the device packages of ECP5 devices where little crosstalk is generated, the PCB board can cause significant noise injection from any I/O pin adjacent to SERDES data, reference clock, and power pins as well as other critical I/O pins such as clock signals. TN1114, [Electrical Recommendations for Lattice SERDES](#), provides detailed guidelines for optimizing the hardware to reduce the likelihood of crosstalk to the analog supplies. PCB traces running in parallel for long distances need careful analysis. Simulate any suspicious traces using a PCB crosstalk simulation tool to determine if they will cause problems.

It is common practice for designers to select pinouts for their system very early in the design cycle. For the FPGA designer, this requires a detailed knowledge of the targeted FPGA device. Designers often use a spreadsheet program to initially capture the list of the design I/Os. Lattice provides detailed pinout information that can be downloaded from the Lattice website in .csv format for designers to use as a resource to create pinout information. For example, by navigating to the pinout.csv file, the user can gather the pinout details for all the different package offerings of the device in the family, including I/O banking, differential pairing, Dual Function of the pins, and input and output details.

## Clock Inputs

The ECP5 device does not provide dedicated pins for clock inputs. All clock inputs are shared with the General Purpose I/O pin. When the pin is not used for clocking, the user can use it as a general purpose I/O pin.

However, when these pins are used for clocking purpose, the user needs to pay attention to minimize signal noise on these pins. Please refer to TN1265, [ECP5 High-Speed I/O Interface](#).

The pins can be found under the Dual Function column of the pinlist csv file.

## Pinout Considerations

The ECP5 supports many applications with high-speed interfaces. These include various rule-based pinouts that need to be understood prior to implementation of the PCB design. The pinout selection must be completed with an understanding of the interface building blocks of the FPGA fabric. These include IOLOGIC blocks such as DDR, clock resource connectivity, and PLL and DLL usage. Refer to TN1265, [ECP5 High-Speed I/O Interface](#) for rules pertaining to these interface types.

## LVDS Pin Assignments

True LVDS inputs and outputs are available on I/O pins on the left and right sides of the devices. Top and I/O banks do not support True LVDS standard, but can support emulated LVDS outputs. True LVDS input pairing on left and right banks can be found under the Differential column in the pinlist csv file. True LVDS output pair are available on any A & B pair of the left and right banks.

Emulated LVDS output are available on pairs around all banks, but this will require external termination resistors. This is described in TN1262, [ECP5 sysIO Usage Guide](#).

## HSUL and SSTL Pin Assignments

The HSUL and SSTL interfaces are externally referenced I/O standards require an external reference voltage. The V<sub>REF</sub> pin(s) should get high priority when assigning pins on the PCB. These pins can be found in the Dual Function column with VREF1 label. Each bank includes a separate VREF voltage. VREF1 sets the threshold for the referenced input buffers. Each I/O is individually configurable based on the bank's supply and reference voltages.

## SERDES Pin Considerations

High-speed signaling requires careful PCB design. Maintaining good transmission line characteristics is a requirement. A continuous ground reference should be maintained with high-speed routing. This includes tightly matched differential routing with very few discontinuities. Please refer to TN1033, [High-Speed PCB Design Considerations](#), for suggested methods and guidance.

When operating at 2.5 Gbps or above, use of the following FPGA I/O pins can cause increased jitter. Extra care must be given to these pins when used in combination with the high-speed SERDES interface. High-speed switching output assignments should be minimized or avoided on these pins when the SERDES interface is in use. Only static output or input configuration is recommended.

## ECP5U to ECP5UM Migration

Besides migrating design from one device to another device (i.e. 25K to 45K) on same package (i.e. caBGA554) within its own family in ECP5U and ECP5UM, user can migrate from the non-SERDES (ECP5U) device to SERDES (ECP5UM) device in the same package.

If the user anticipates his design may use SERDES at a later time of his product, he would first design, and making all the connections to, all SERDES circuit on board.

For example, if the user anticipates the need to use the two Dual SERDES on LFE5U-85 product, he has to design his board with LFE5UM-85, which contains the SERDES ports, to the not-yet-populated SERDES circuit on the board. This requires all SERDES powers to be connected to power sources on the board. He can still put in the LFE5U-85 device because the two devices are pin compatible, other than the SERDES pins.

**Table 5. Hardware Checklist**

	Item	OK	NA
<b>1</b>	<b>FPGA Power Supplies</b>		
1.1	VCC core @ 1.1 V +/-5%		
1.1.1	Use a PCB plane for VCC core with proper decoupling		
1.1.2	VCC core sized to meet power requirement calculation from software		
1.2	VCCA @ 1.1 V +/-5%		
1.2.1	VCCA "quiet" and isolated"		
1.2.2	VCCA pins should be ganged together and a solid PCB plane is recommended. This plane should not have adjacent non-SERDES signals passing above or below. It should also be isolated from the VCC core power plane.		
1.3	All VCCIO are between 1.2 V to 3.3 V		
1.3.1	VCCIO8 used with configuration interfaces (i.e. memory devices). Need to match specifications.		
1.3.2	VCCIO[1:7] used based on user design		
1.4	VCCAUX and VCCAUXA @ 2.5 V +/- 5%		
1.6	Power estimation		
<b>2</b>	<b>SERDES Power Supplies</b>		
2.3	VCCHRX and VCCHTX connected for USED SERDES channels		

**Table 5. Hardware Checklist (Continued)**

	Item	OK	NA
2.3.1	VCCHRX and VCCHTX are at 1.2 V +/- 5%		
<b>3</b>	<b>Configuration</b>		
3.1	Pull-ups and pull-downs on configuration specific pins		
3.2	VCCIO8 bank voltage matches sysCONFIG peripheral devices such as SPI Flash		
3.3	Pull-up or pull-down on SPIFASTN pin		
<b>4</b>	<b>SERDES</b>		
4.1	Dedicated reference clock input from clock source meets the DC and AC requirements		
4.1.1	External AC coupling caps may be required for compatibility to common-mode levels		
4.1.2	Ref clock termination resistors may be needed for compatible signaling levels		
4.2	Maintain good high-speed transmission line routing		
4.2.1	Continuous ground reference plane to serial channels		
4.2.2	Tightly length matched differential traces		
4.2.3	Do not pass other signals on the PCB above or below the high-speed SERDES without isolation.		
4.2.4	Keep non-SERDES signal traces from passing above or below the 1.1V VCCA power plane without isolation.		
4.2.5	Avoid the aggressor pins mentioned previously in this document.		
<b>5</b>	<b>Special Pin Assignments</b>		
5.2	VREF assignments followed for single-ended SSTL inputs		
5.2.1	Properly decouple the VREF source		
<b>6</b>	<b>Critical Pinout Selection</b>		
6.1	Pinout has been chosen to address FPGA resource connections to I/O logic and clock resources per TN1265, <a href="#">ECP5 High-Speed I/O Interface</a> .		
6.2	Shared general purpose I/Os are used as inputs for FPGA PLL and Clock inputs.		
<b>7</b>	<b>JTAG</b>		
7.1	Pull-down on TCK. See Table 3.		
7.2	Pull-up on TDI, TMS, TDO. See Table 3.		
<b>8</b>	<b>LPDDR3 and DDR3 Interface Requirements</b>		
8.1	DQ, DM, and DQS signals should be routed in a data group and should have similar routing and matched via counts. Using more than three vias is not recommended in the route between the FPGA controller and memory device.		
8.2	Maintain a maximum of ±50 mil between any DQ/DM and its associated DQS strobe within a DQ group. Use careful serpentine routing to meet this requirement.		
8.3	All data groups must reference a ground plane within the stack-up.		
8.4	DDR trace reference must be solid without slots or breaks. It should be continuous between the FPGA and the memory.		
8.5	Provide a separation of 3 W spacing between a data group and any other unrelated signals to avoid crosstalk issues. Use a minimum of 2 W spacing between all DDR traces excluding differential CK and DQS signals.		
8.6	Assigned FPGA I/O within a data group can be swapped to allow clean layout. Do not swap DQS assignments.		
8.7	Differential pair of DQS to DQS_N trace lengths should be matched at ±10 mil.		
8.8	Resistor terminations (DQ) placed in a fly-by fashion at the FPGA is highly recommended. Stub fashion terminations, if used, should not include a stub longer than 600 mil.		
8.9	LDQS/LDQS_N and UDQS/UDQS_N trace lengths should be matched within ±100 mil.		
8.10	Address/control signals and the associated CK and CK_N differential FPGA clock should be routed with a control trace matching ±100 mil.		
8.11	CK to CK_N trace lengths must be matched to within ±10 mil.		

**Table 5. Hardware Checklist (Continued)**

	<b>Item</b>	<b>OK</b>	<b>NA</b>
8.12	Address and control signals can be referenced to a power plane if a ground plane is not available. Ground reference is preferred.		
8.13	Address and control signals should be kept on a different routing layer from DQ, DQS, and DM to isolate crosstalk between the signals.		
8.14	Differential terminations used by the CLK/CLKN pair must be located as close as possible to the memory.		
8.15	Address and control terminations placed after the memory component using a fly-by technique are highly recommended. Stub fashion terminations, if used, should not include a stub longer than 600 mils.		

## Technical Support Assistance

e-mail: [techsupport@latticesemi.com](mailto:techsupport@latticesemi.com)

Internet: [www.latticesemi.com](http://www.latticesemi.com)

## Revision History

<b>Date</b>	<b>Version</b>	<b>Change Summary</b>
March 2014	01.0	Initial release.



### **Section III. LatticeECP5 Family Handbook Revision History**

---



# LatticeECP5 Family Handbook

## Revision History

---

March 2014

Handbook HB1012

### Revision History

Date	Handbook Revision Number	Change Summary
March 2014	01.1	Initial release.

Note: For detailed revision changes, please refer to the revision history for each document.