LatticeECP/EC Family Handbook

HB1000 Version 03.3, March 2010

LatticeECP/EC Family Handbook
Table of Contents

March 2010

Section I. LatticeECP/EC Family Data Sheet

Introduction

Architecture

Section I. LatticeECP/EC Family Data Sheet

DS1000 Version 02.7, February 2008

Features

- **Extensive Density and Package Options**
  - 1.5K to 32.8K LUT4s
  - 65 to 496 I/Os
  - Density migration supported

- **sysDSP™ Block (LatticeECP™ Versions)**
  - High performance multiply and accumulate
  - 4 to 8 blocks
    - 4 to 8 36x36 multipliers or
    - 16 to 32 18x18 multipliers or
    - 32 to 64 9x9 multipliers

- **Embedded and Distributed Memory**
  - 18 Kbits to 498 Kbits sysMEM™ Embedded Block RAM (EBR)
  - Up to 131 Kbits distributed RAM
  - Flexible memory resources:
    - Distributed and block memory

- **Flexible I/O Buffer**
  - Programmable sysI/O™ buffer supports wide range of interfaces:
    - LVCMOS 3.3/2.5/1.8/1.5/1.2
    - LVTTL
    - SSTL 3/2 Class I, II, SSTL18 Class I
    - HSTL 18 Class I, II, III, HSTL15 Class I, III
    - PCI
    - LVDS, Bus-LVDS, LVPECL, RSDS

- **Dedicated DDR Memory Support**
  - Implements interface up to DDR400 (200MHz)

- **sysCLOCK™ PLLs**
  - Up to four analog PLLs per device
  - Clock multiply, divide and phase shifting

- **System Level Support**
  - IEEE Standard 1149.1 Boundary Scan, plus ispTRACY™ internal logic analyzer capability
  - SPI boot flash interface
  - 1.2V power supply

- **Low Cost FPGA**
  - Features optimized for mainstream applications
  - Low cost TQFP and PQFP packaging

Table 1-1. LatticeECP/EC Family Selection Guide

Device	LFEC1	LFEC3	LFEC6/ LFECP6	LFEC10/ LFECP10	LFEC15/ LFECP15	LFEC20/ LFECP20	LFEC33/ LFECP33
PFU/PFF Rows	12	16	24	32	40	44	64
PFU/PFF Columns	16	24	32	40	48	56	64
PFUs/PFFs	192	384	768	1280	1920	2464	4096
LUTs (K)	1.5	3.1	6.1	10.2	15.4	19.7	32.8
Distributed RAM (Kbits)	6	12	25	41	61	79	131
EBR SRAM (Kbits)	18	55	92	276	350	424	498
EBR SRAM Blocks	2	6	10	30	38	46	54
sysDSP Blocks[1]	—	—	4	5	6	7	8
18x18 Multipliers[1]	—	—	16	20	24	28	32
V_{CC} Voltage (V)	1.2	1.2	1.2	1.2	1.2	1.2	1.2
Number of PLLs	2	2	2	4	4	4	4
Packages and I/O Combinations:							
100-pin TQFP (14 x 14 mm)	67	67					
144-pin TQFP (20 x 20 mm)	97	97	97				
208-pin PQFP (28 x 28 mm)	112	145	147	147			
256-ball fpBGA (17 x 17 mm)		160	195	195	195		
484-ball fpBGA (23 x 23 mm)			224	288	352	360	360
672-ball fpBGA (27 x 27 mm)						400	496

1. LatticeECP devices only.

Introduction

The LatticeECP/EC family of FPGA devices is optimized to deliver mainstream FPGA features at low cost. For maximum performance and value, the LatticeECP™ (<u>EC</u>onomy <u>P</u>lus) FPGA concept combines an efficient FPGA fabric with high-speed dedicated functions. Lattice's first family to implement this approach is the LatticeECP-DSP™ (<u>EC</u>onomy <u>P</u>lus <u>DSP</u>) family, providing dedicated high-performance DSP blocks on-chip. The LatticeEC™ (<u>EC</u>onomy) family supports all the general purpose features of LatticeECP devices without dedicated function blocks to achieve lower cost solutions.

The LatticeECP/EC FPGA fabric, which was designed from the outset with low cost in mind, contains all the critical FPGA elements: LUT-based logic, distributed and embedded memory, PLLs and support for mainstream I/Os. Dedicated DDR memory interface logic is also included to support this memory that is becoming increasingly prevalent in cost-sensitive applications.

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

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

Architecture Overview

The LatticeECP-DSP and LatticeEC architectures contain 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), as shown in Figures 2-1 and 2-2. In addition, LatticeECP-DSP supports an additional row of DSP blocks, as shown in Figure 2-2.

There are two kinds of logic blocks, the Programmable Functional Unit (PFU) and Programmable Functional unit without RAM/ROM (PFF). The PFU contains the building blocks for logic, arithmetic, RAM, ROM and register functions. The PFF block contains building blocks for logic, arithmetic and ROM functions. Both PFU and PFF blocks are optimized for flexibility, allowing complex designs to be implemented quickly and efficiently. Logic Blocks are arranged in a two-dimensional array. Only one type of block is used per row. The PFU blocks are used on the outside rows. The rest of the core consists of rows of PFF blocks interspersed with rows of PFU blocks. For every three rows of PFF blocks there is a row of PFU blocks.

Each PIC block encompasses two PIOs (PIO pairs) with their respective sysI/O interfaces. PIO pairs on the left and right edges of the device can be configured as LVDS transmit/receive pairs. sysMEM EBRs are large dedicated fast memory blocks. They can be configured as RAM or ROM.

The PFU, PFF, PIC and EBR Blocks are arranged in a two-dimensional grid with rows and columns as shown in Figure 2-1. The blocks are connected with many vertical and horizontal routing channel resources. The place and route software tool automatically allocates these routing resources.

At the end of the rows containing the sysMEM Blocks are the sysCLOCK Phase Locked Loop (PLL) Blocks. These PLLs have multiply, divide and phase shifting capability; they are used to manage the phase relationship of the clocks. The LatticeECP/EC architecture provides up to four PLLs per device.

Every device in the family has a JTAG Port with internal Logic Analyzer (ispTRACY) capability. The sysCONFIG™ port which allows for serial or parallel device configuration. The LatticeECP/EC devices use 1.2V as their core voltage.

Figure 2-1. Simplified Block Diagram, LatticeEC Device (Top Level)
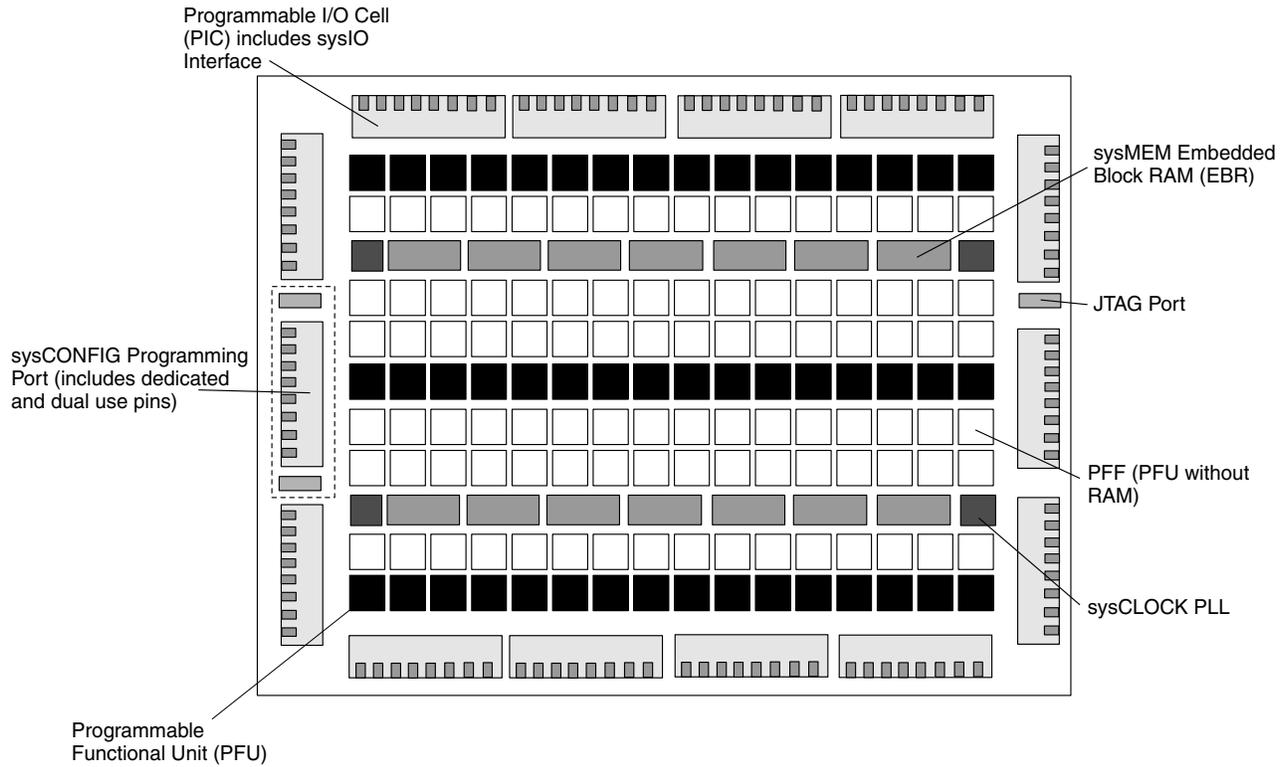
Figure 2-2. Simplified Block Diagram, LatticeECP-DSP Device (Top Level)
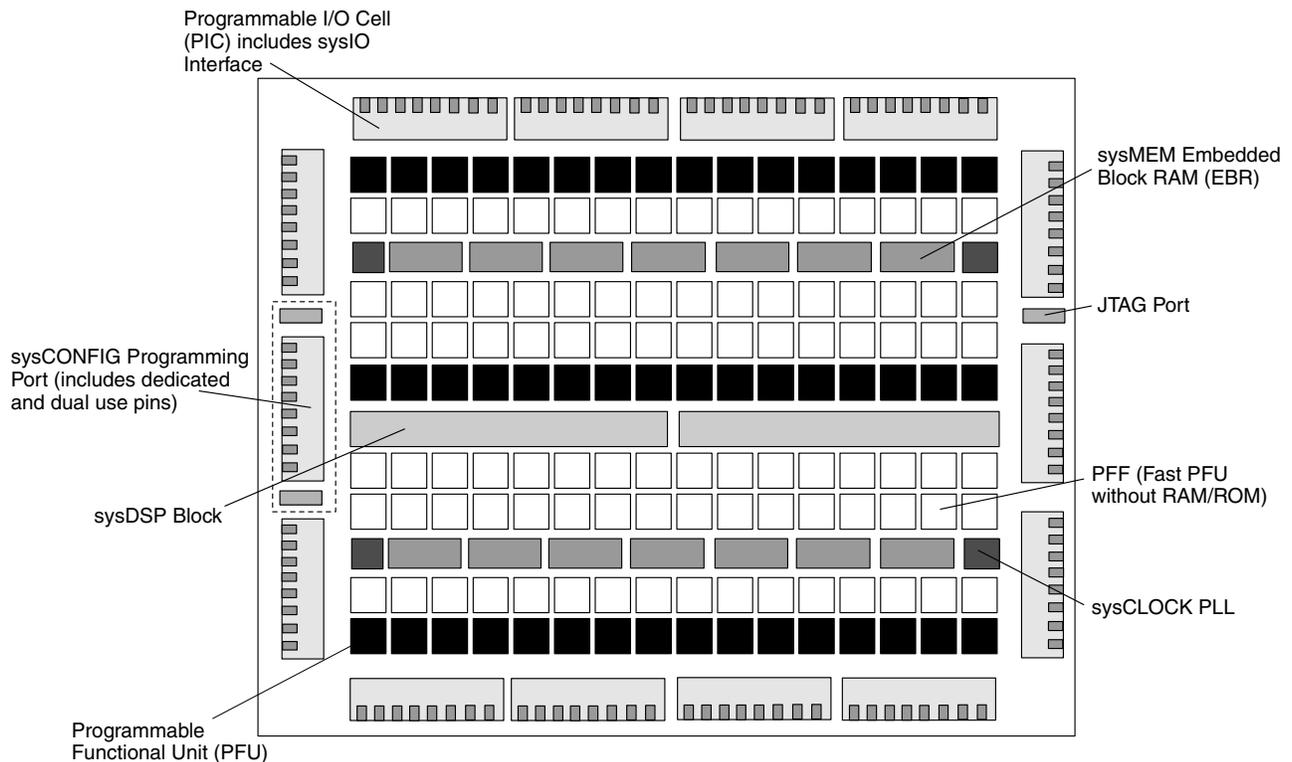
PFU and PFF Blocks

The core of the LatticeECP/EC devices consists of PFU and PFF blocks. The PFUs can be programmed to perform Logic, Arithmetic, Distributed RAM and Distributed ROM functions. PFF blocks can be programmed to perform Logic, Arithmetic and ROM functions. Except where necessary, the remainder of the data sheet will use the term PFU to refer to both PFU and PFF blocks.

Each PFU block consists of four interconnected slices, numbered 0-3 as shown in Figure 2-3. All the interconnections to and from PFU blocks are from routing. There are 53 inputs and 25 outputs associated with each PFU block.

Figure 2-3. PFU Diagram
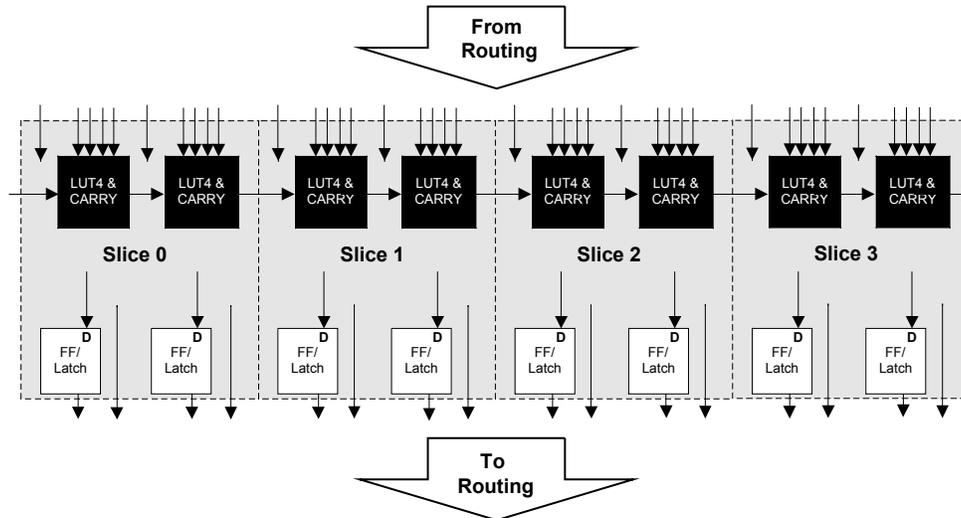
Slice

Each slice contains two LUT4 lookup tables feeding two registers (programmed to be in FF or Latch mode), and some associated logic that allows the LUTs to be combined to perform functions such as LUT5, LUT6, LUT7 and LUT8. There is control logic to perform set/reset functions (programmable as synchronous/asynchronous), clock select, chip-select and wider RAM/ROM functions. Figure 2-4 shows an overview of the internal logic of the slice. The registers in the slice can be configured for positive/negative and edge/level clocks.

There are 14 input signals: 13 signals from routing and one from the carry-chain (from adjacent slice or PFU). There are 7 outputs: 6 to routing and one to carry-chain (to adjacent PFU). Table 2-1 lists the signals associated with each slice.

Figure 2-4. Slice Diagram
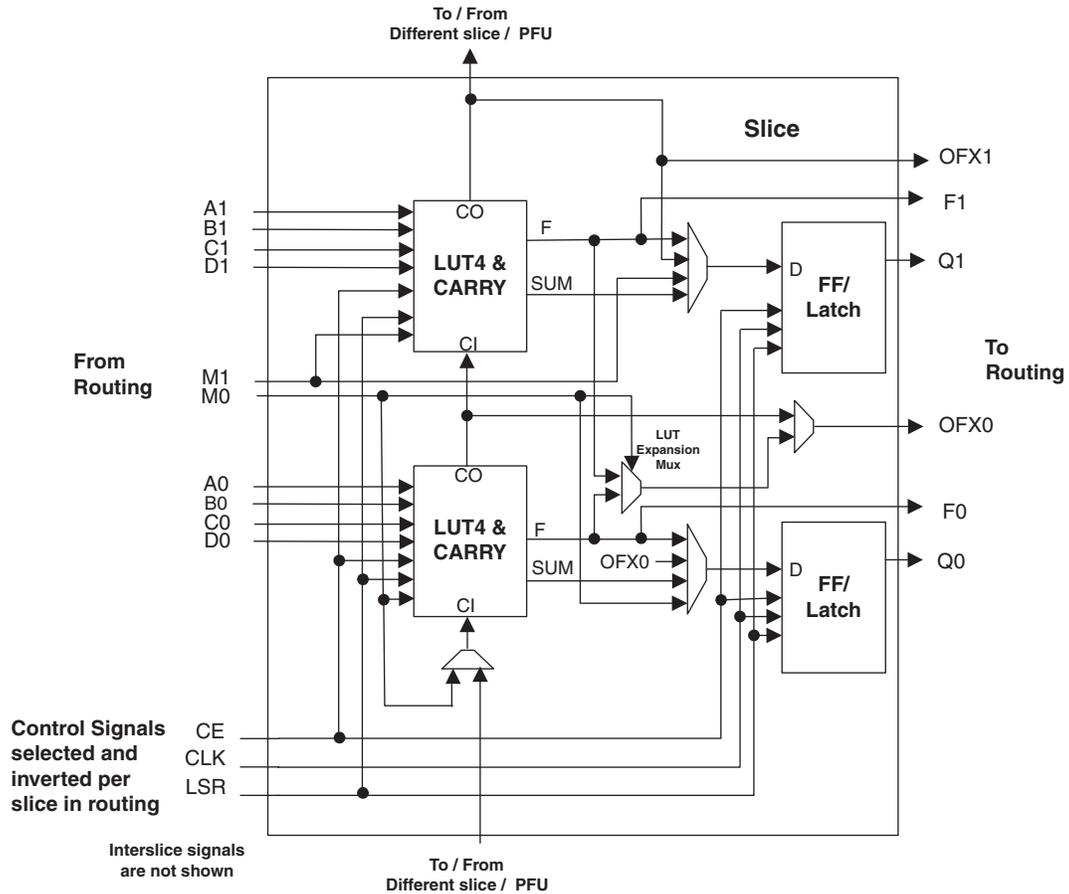
Table 2-1. Slice Signal Descriptions

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	FCIN	Fast Carry In[1]
Output	Data signals	F0, F1	LUT4 output register bypass signals
Output	Data signals	Q0, Q1	Register Outputs
Output	Data signals	OFX0	Output of a LUT5 MUX
Output	Data signals	OFX1	Output of a LUT6, LUT7, LUT8[2] MUX depending on the slice
Output	Inter-PFU signal	FCO	For the right most PFU the fast carry chain output[1]

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

Modes of Operation

Each Slice is capable of four modes of operation: Logic, Ripple, RAM and ROM. The Slice in the PFF is capable of all modes except RAM. Table 2-2 lists the modes and the capability of the Slice blocks.

Table 2-2. Slice Modes

	Logic	**Ripple**	**RAM**	**ROM**
PFU Slice	LUT 4x2 or LUT 5x1	2-bit Arithmetic Unit	SPR16x2	ROM16x1 x 2
PFF Slice	LUT 4x2 or LUT 5x1	2-bit Arithmetic Unit	N/A	ROM16x1 x 2

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 logic function with four inputs can be generated by programming this lookup table. Since there are two LUT4s per Slice, a LUT5 can be constructed within one Slice. Larger lookup tables such as LUT6, LUT7 and LUT8 can be constructed by concatenating other Slices.

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

- Addition 2-bit
- Subtraction 2-bit
- Add/Subtract 2-bit using dynamic control
- Up counter 2-bit
- Down counter 2-bit
- Ripple mode multiplier building block
- 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, distributed RAM can be constructed using each LUT block as a 16x1-bit memory. Through the combination of LUTs and Slices, a variety of different memories can be constructed.

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. Figure 2-5 shows the distributed memory primitive block diagrams. Dual port memories involve the pairing of two Slices, one Slice functions as the read-write port. The other companion Slice supports the read-only port. For more information about using RAM in LatticeECP/EC devices, please see the list of technical documentation at the end of this data sheet.

Table 2-3. Number of Slices Required For Implementing Distributed RAM

	SPR16x2	**DPR16x2**
Number of slices	1	2

Note: SPR = Single Port RAM, DPR = Dual Port RAM

Figure 2-5. Distributed Memory Primitives

SPR16x2

AD0
AD1
AD2
AD3
DI0
DI1
WRE
CK
→ DO0
→ DO1

DPR16x2

WAD0
WAD1
WAD2
WAD3
DI0
DI1
WCK
WRE
← RAD0
← RAD1
← RAD2
← RAD3
→ RDO0
→ RDO1
→ WDO0
→ WDO1

ROM16x1

AD0
AD1
AD2
AD3
→ DO0
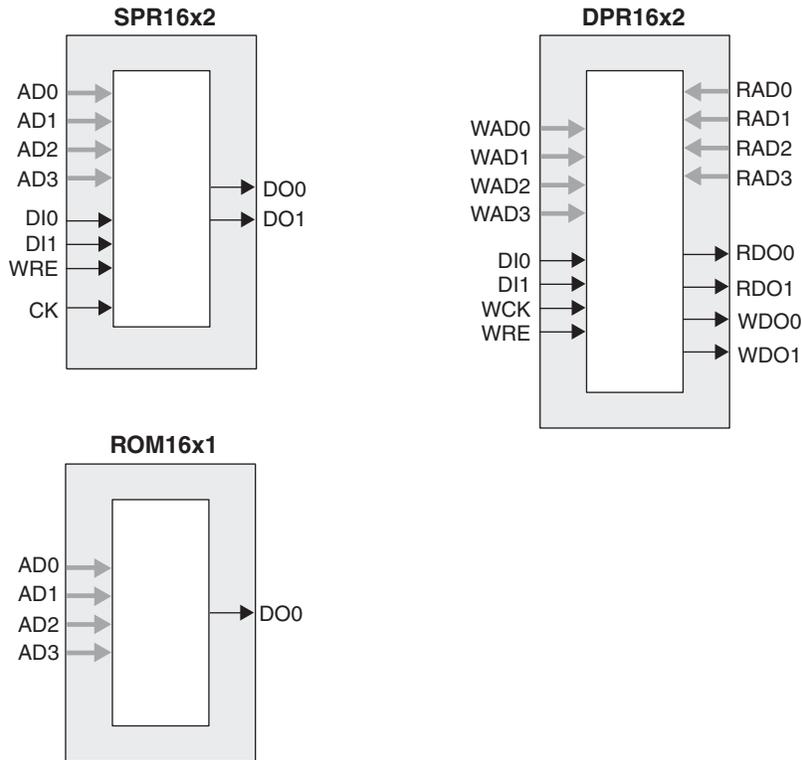
ROM Mode: The ROM mode uses the same principal as the RAM modes, but without the Write port. Pre-loading is accomplished through the programming interface during configuration.

PFU Modes of Operation
Slices can be combined within a PFU to form larger functions. Table 2-4 tabulates these modes and documents the functionality possible at the PFU level.

Table 2-4. PFU Modes of Operation

Logic	Ripple	RAM[1]	ROM
LUT 4x8 or MUX 2x1 x 8	2-bit Add x 4	SPR16x2 x 4 DPR16x2 x 2	ROM16x1 x 8
LUT 5x4 or MUX 4x1 x 4	2-bit Sub x 4	SPR16x4 x 2 DPR16x4 x 1	ROM16x2 x 4
LUT 6x 2 or MUX 8x1 x 2	2-bit Counter x 4	SPR16x8 x 1	ROM16x4 x 2
LUT 7x1 or MUX 16x1 x 1	2-bit Comp x 4		ROM16x8 x 1

1. These modes are not available in PFF blocks

Routing

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

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

The ispLEVER design tool suite takes the output of the synthesis tool and places and routes the design. Generally, the place and route tool is completely automatic, although an interactive routing editor is available to optimize the design.

Clock Distribution Network

The clock inputs are selected from external I/O, the sysCLOCK™ PLLs or routing. These clock inputs are fed through the chip via a clock distribution system.

Primary Clock Sources

LatticeECP/EC devices derive clocks from three primary sources: PLL outputs, dedicated clock inputs and routing. LatticeECP/EC devices have two to four sysCLOCK PLLs, located on the left and right sides of the device. There are four dedicated clock inputs, one on each side of the device. Figure 2-6 shows the 20 primary clock sources.

Figure 2-6. Primary Clock Sources
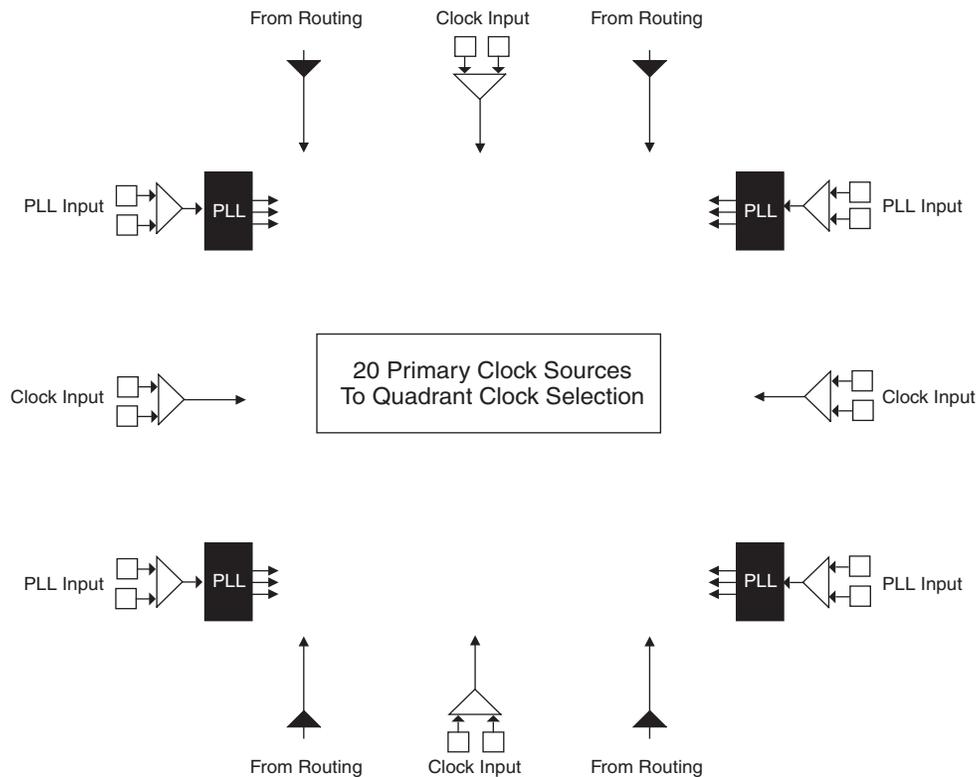
Note: Smaller devices have two PLLs.

Secondary Clock Sources

LatticeECP/EC devices have four secondary clock resources per quadrant. The secondary clock branches are tapped at every PFU. These secondary clock networks can also be used for controls and high fanout data. These secondary clocks are derived from four clock input pads and 16 routing signals as shown in Figure 2-7.

Figure 2-7. Secondary Clock Sources
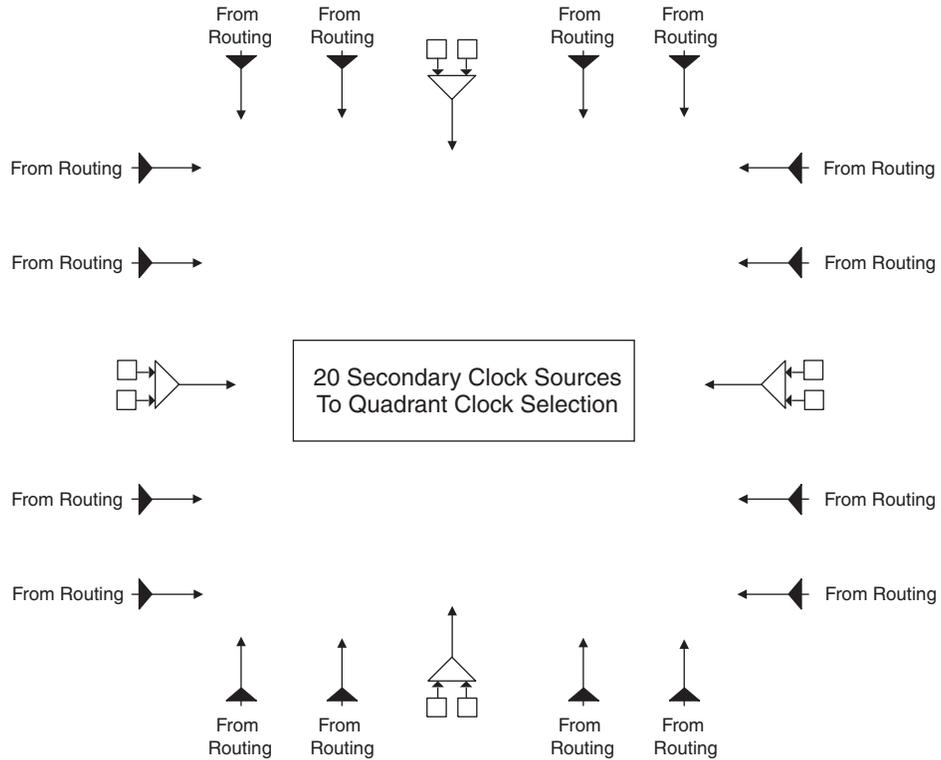
Clock Routing

The clock routing structure in LatticeECP/EC devices consists of four Primary Clock lines and a Secondary Clock network per quadrant. The primary clocks are generated from MUXs located in each quadrant. Figure 2-8 shows this clock routing. The four secondary clocks are generated from MUXs located in each quadrant as shown in Figure 2-9. Each slice derives its clock from the primary clock lines, secondary clock lines and routing as shown in Figure 2-10.

Figure 2-8. Per Quadrant Primary Clock Selection

20 Primary Clock Sources: 12 PLLs + 4 PIOs + 4 Routing[1]

DCS

DCS

4 Primary Clocks (CLK0, CLK1, CLK2, CLK3) per Quadrant

1. Smaller devices have fewer PLL related lines.
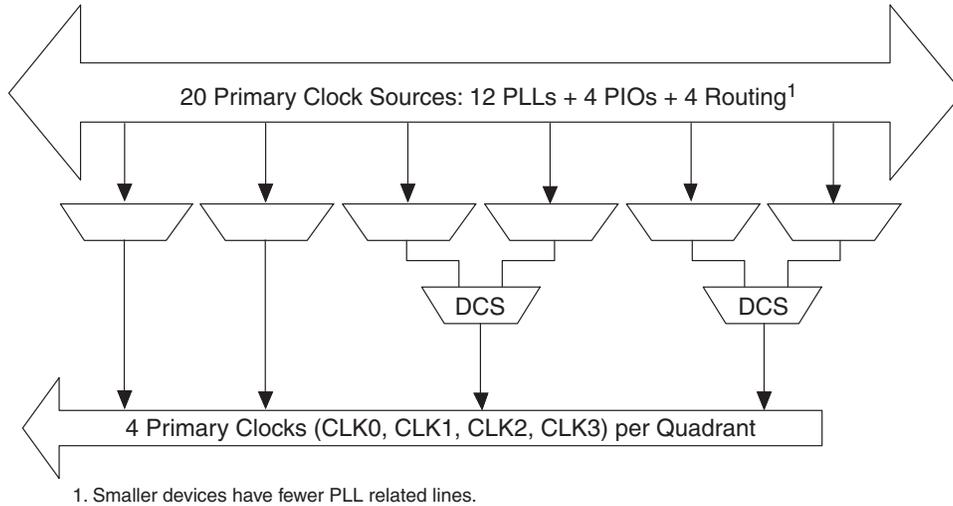
Figure 2-9. Per Quadrant Secondary Clock Selection

20 Secondary Clock Feedlines : 4 Clock Input Pads + 16 Routing Signals

4 Secondary Clocks per Quadrant
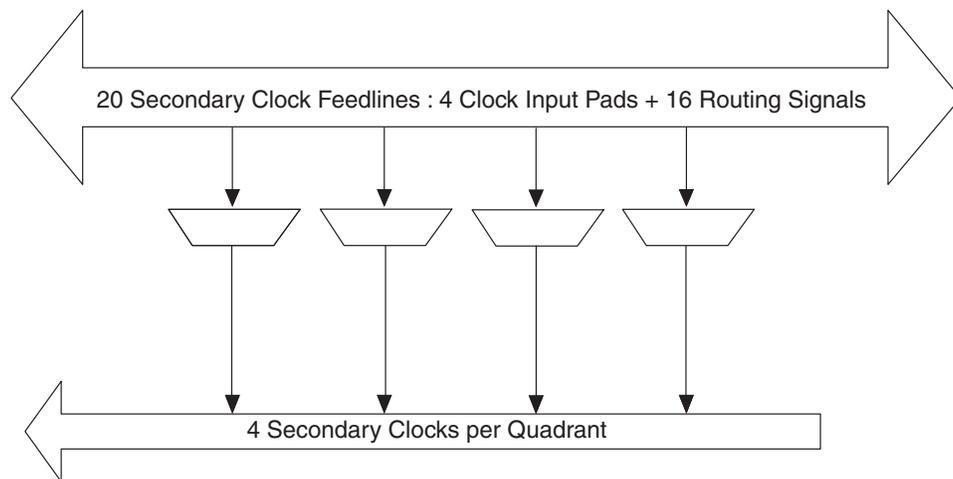
Figure 2-10. Slice Clock Selection

Primary Clock

Secondary Clock

Routing

GND

Clock to
each slice
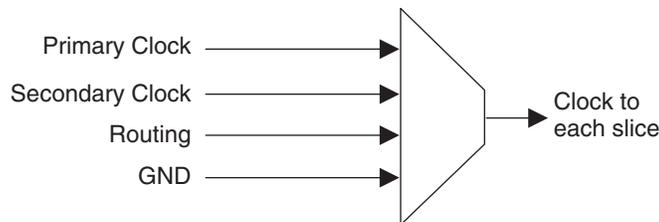
sysCLOCK Phase Locked Loops (PLLs)

The PLL clock input, from pin or routing, feeds into an input clock divider. There are three sources of feedback signal to the feedback divider: from CLKOP (PLL Internal), from clock net (CLKOP) or from a user clock (PIN or logic). There is a PLL_LOCK signal to indicate that VCO has locked on to the input clock signal. Figure 2-11 shows the sysCLOCK PLL diagram.

The setup and hold times of the device can be improved by programming a delay in the feedback or input path of the PLL which will advance or delay the output clock with reference to the input clock. This delay can be either pro-

grammed during configuration or can be adjusted dynamically. In dynamic mode, the PLL may lose lock after adjustment and not relock until the t_{LOCK} parameter has been satisfied. Additionally, the phase and duty cycle block allows the user to adjust the phase and duty cycle of the CLKOS output.

The sysCLOCK PLLs provide the ability to synthesize clock frequencies. Each PLL has four dividers associated with it: input clock divider, feedback divider, post scalar divider and secondary clock divider. The input clock divider is used to divide the input clock signal, while the feedback divider is used to multiply the input clock signal. The post scalar divider allows the VCO to operate at higher frequencies than the clock output, thereby increasing the frequency range. The secondary divider is used to derive lower frequency outputs.

Figure 2-11. PLL Diagram
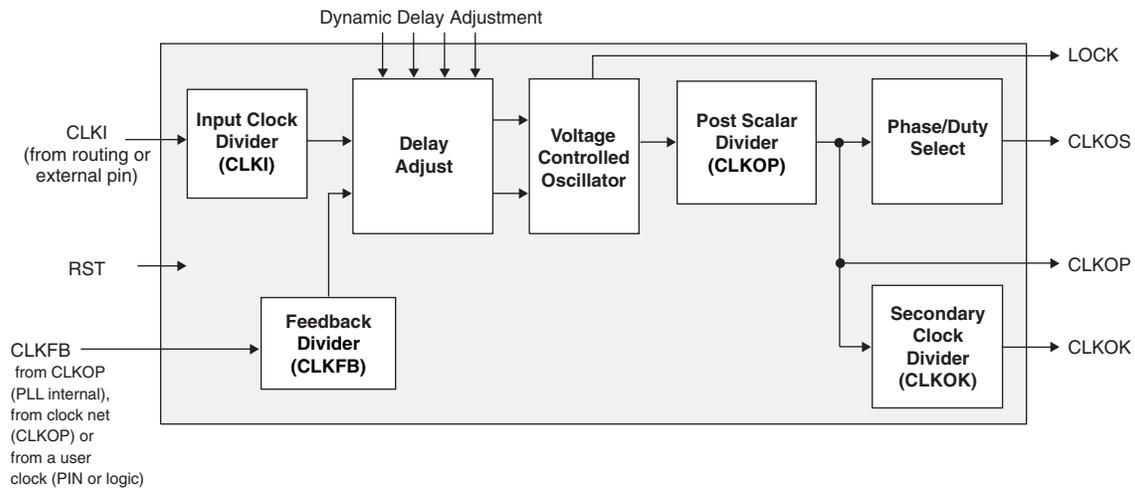
Figure 2-12 shows the available macros for the PLL. Table 2-5 provides signal description of the PLL Block.

Figure 2-12. PLL Primitive
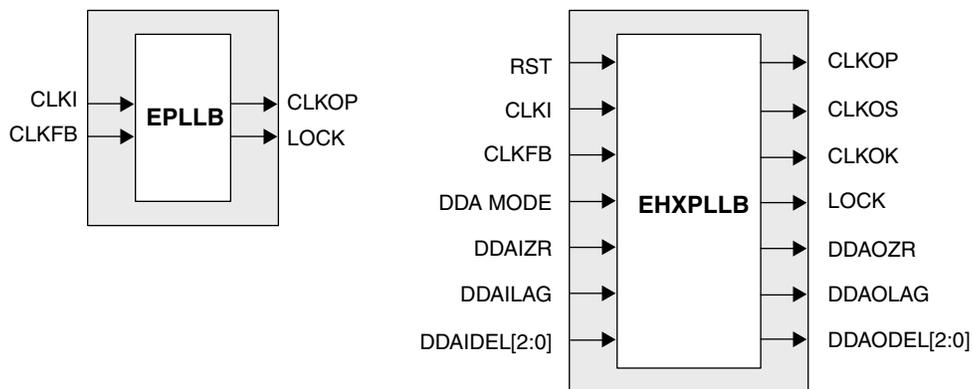
Table 2-5. PLL Signal Descriptions

Signal	I/O	Description
CLKI	I	Clock input from external pin or routing
CLKFB	I	PLL feedback input from CLKOP (PLL internal), from clock net (CLKOP) or from a user clock (PIN or logic)
RST	I	"1" to reset PLL
CLKOS	O	PLL output clock to clock tree (phase shifted/duty cycle changed)
CLKOP	O	PLL output clock to clock tree (No phase shift)
CLKOK	O	PLL output to clock tree through secondary clock divider
LOCK	O	"1" indicates PLL LOCK to CLKI
DDAMODE	I	Dynamic Delay Enable. "1": Pin control (dynamic), "0": Fuse Control (static)
DDAIZR	I	Dynamic Delay Zero. "1": delay = 0, "0": delay = on
DDAILAG	I	Dynamic Delay Lag/Lead. "1": Lead, "0": Lag
DDAIDEL[2:0]	I	Dynamic Delay Input
DDAOZR	O	Dynamic Delay Zero Output
DDAOLAG	O	Dynamic Delay Lag/Lead Output
DDAODEL[2:0]	O	Dynamic Delay Output

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

Dynamic Clock Select (DCS)

The DCS is a global clock buffer with smart multiplexer functions. It takes two independent input clock sources and outputs a clock signal without any glitches or runt pulses. This is achieved regardless of where the select signal is toggled. There are eight DCS blocks per device, located in pairs at the center of each side. Figure 2-13 illustrates the DCS Block Macro.

Figure 2-13. DCS Block Primitive
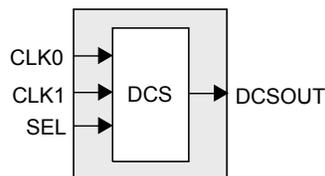
Figure 2-14 shows timing waveforms of the default DCS operating mode. The DCS block can be programmed to other modes. For more information about the DCS, please see the list of technical documentation at the end of this data sheet.

Figure 2-14. DCS Waveforms
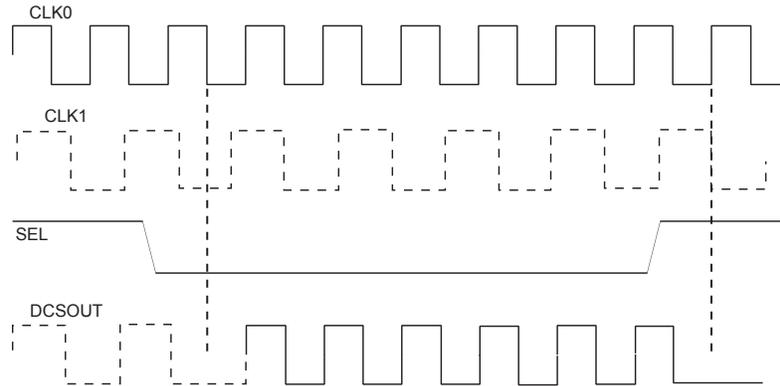
sysMEM Memory

The LatticeECP/EC devices contain a number of sysMEM Embedded Block RAM (EBR). The EBR consists of a 9-Kbit RAM, with dedicated input and output registers.

sysMEM Memory Block

The sysMEM block can implement single port, dual port or pseudo dual port memories. Each block can be used in a variety of depths and widths as shown in Table 2-6.

Table 2-6. sysMEM Block Configurations

Memory Mode	Configurations
Single Port	8,192 x 1 4,096 x 2 2,048 x 4 1,024 x 9 512 x 18 256 x 36
True Dual Port	8,192 x 1 4,096 x 2 2,048 x 4 1,024 x 9 512 x 18
Pseudo Dual Port	8,192 x 1 4,096 x 2 2,048 x 4 1,024 x 9 512 x 18 256 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

Figure 2-15 shows the four basic memory configurations and their input/output names. 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.

Figure 2-15. sysMEM EBR Primitives
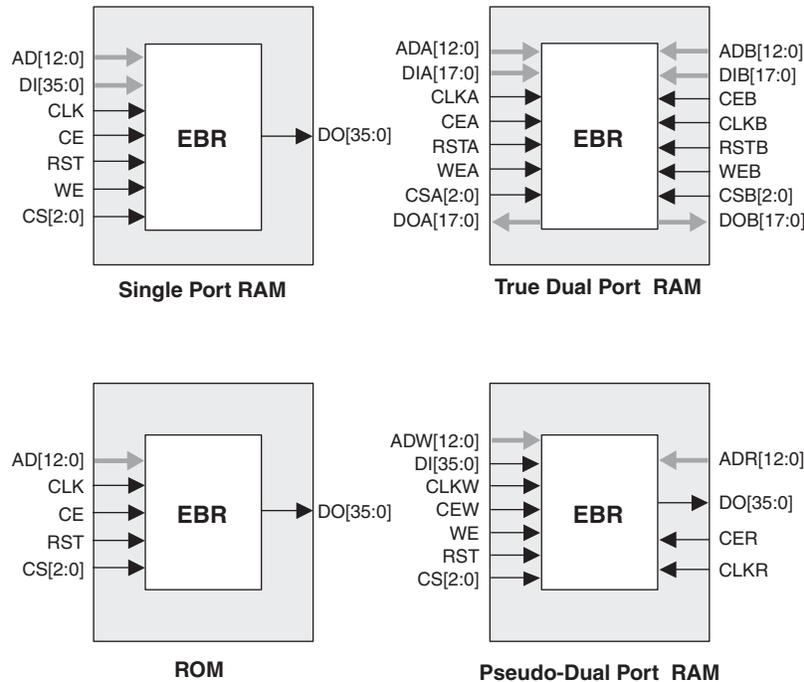
The EBR memory supports three forms of write behavior for single port or dual port operation:

1. **Normal** – data on the output appears only during read cycle. During a write cycle, the data (at the current address) does not appear on the output. This mode is supported for all data widths.

2. **Write Through** – a copy of the input data appears at the output of the same port during a write cycle. This mode is supported for all data widths.

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

Memory Core Reset

The memory array in the EBR utilizes latches at the A and B output ports. These latches can be reset asynchronously or synchronously. RSTA and RSTB are local signals, which reset the output latches associated with Port A and Port B, respectively. The Global Reset (GSRN) signal resets both ports. The output data latches and associated resets for both ports are as shown in Figure 2-16.

Figure 2-16. Memory Core Reset
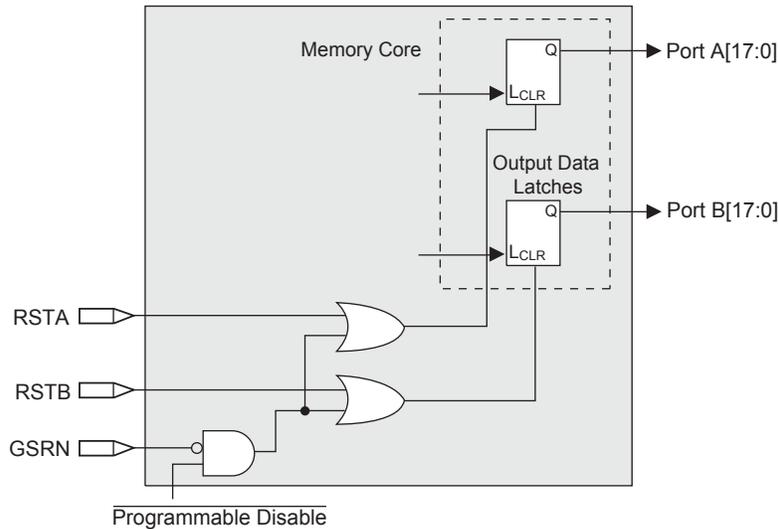
For further information about sysMEM EBR block, please see the the list of technical documentation at the end of this data sheet.

EBR Asynchronous Reset

EBR asynchronous reset or GSR (if used) can only be applied if all clock enables are low for a clock cycle before the reset is applied and released a clock cycle after the reset is released, as shown in Figure 2-17. The GSR input to the EBR is always asynchronous.

Figure 2-17. EBR Asynchronous Reset (Including GSR) Timing Diagram
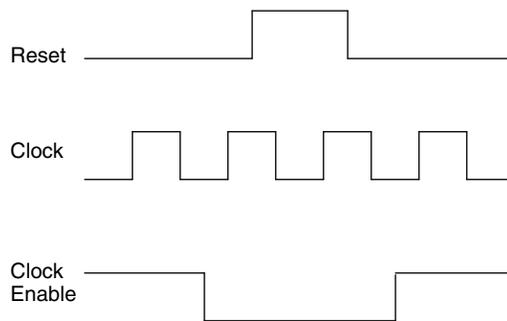
If all clock enables remain enabled, the EBR asynchronous reset or GSR may only be applied and released after the EBR read and write clock inputs are in a steady state condition for a minimum of $1/f_{MAX}$ (EBR clock). The reset release must adhere to the EBR synchronous reset setup time before the next active read or write clock edge.

If an EBR is pre-loaded during configuration, the GSR input must be disabled or the release of the GSR during device Wake Up must occur before the release of the device I/Os becomes active.

These instructions apply to all EBR RAM and ROM implementations.

Note that there are no reset restrictions if the EBR synchronous reset is used and the EBR GSR input is disabled.

sysDSP Block

The LatticeECP-DSP family provides a sysDSP block, making it ideally suited for low cost, high performance Digital Signal Processing (DSP) applications. Typical functions used in these applications are Finite Impulse Response (FIR) filters; Fast Fourier Transforms (FFT) functions, correlators, Reed-Solomon/Turbo/Convolution encoders and

decoders. These complex signal processing functions use similar building blocks such as multiply-adders and multiply-accumulators.

sysDSP Block Approach Compared to General DSP

Conventional general-purpose DSP chips typically contain one to four (Multiply and Accumulate) MAC units with fixed data-width multipliers; this leads to limited parallelism and limited throughput. Their throughput is increased by higher clock speeds. The LatticeECP, on the other hand, has many DSP blocks that support different data-widths. This allows the designer to use highly parallel implementations of DSP functions. The designer can optimize the DSP performance vs. area by choosing an appropriate level of parallelism. Figure 2-18 compares the serial and the parallel implementations.

Figure 2-18. Comparison of General DSP and LatticeECP-DSP Approaches
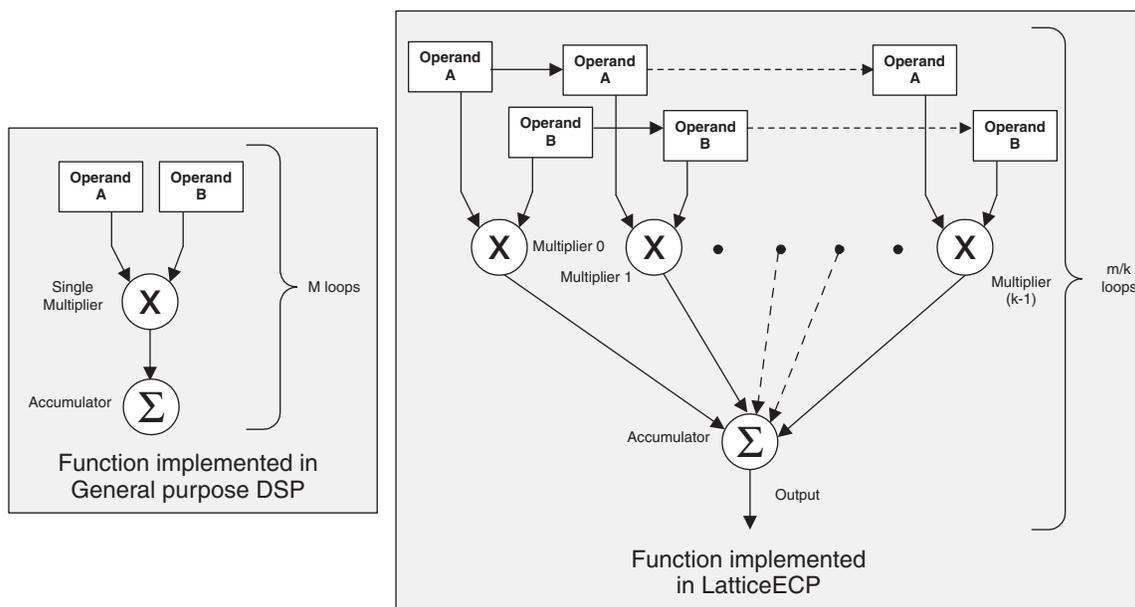
sysDSP Block Capabilities

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

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

- MULT   (Multiply)
- MAC   (Multiply, Accumulate)
- MULTADD   (Multiply, Addition/Subtraction)
- MULTADDSUM (Multiply, Addition/Subtraction, Accumulate)

The number of elements available in each block depends on the width selected from the three available options x9, x18, and x36. A number of these elements are concatenated for highly parallel implementations of DSP functions. Table 2-1 shows the capabilities of the block.

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

Width of Multiply	x9	x18	x36
MULT	8	4	1
MAC	2	2	—
MULTADD	4	2	—
MULTADDSUM	2	1	—

Some options are available in four elements. The input register in all the elements can be directly loaded or can be loaded as shift registers from previous operand registers. In addition by selecting "dynamic operation" in the 'Signed/Unsigned' options the operands can be switched between signed and unsigned on every cycle. Similarly by selecting 'Dynamic operation' in the 'Add/Sub' option the Accumulator can be switched between addition and subtraction on every cycle.

MULT sysDSP Element

This multiplier element implements a multiply with no addition or accumulator nodes. The two operands, A and B, are multiplied and the result is available at the output. The user can enable the input/output and pipeline registers. Figure 2-19 shows the MULT sysDSP element.

Figure 2-19. MULT sysDSP Element
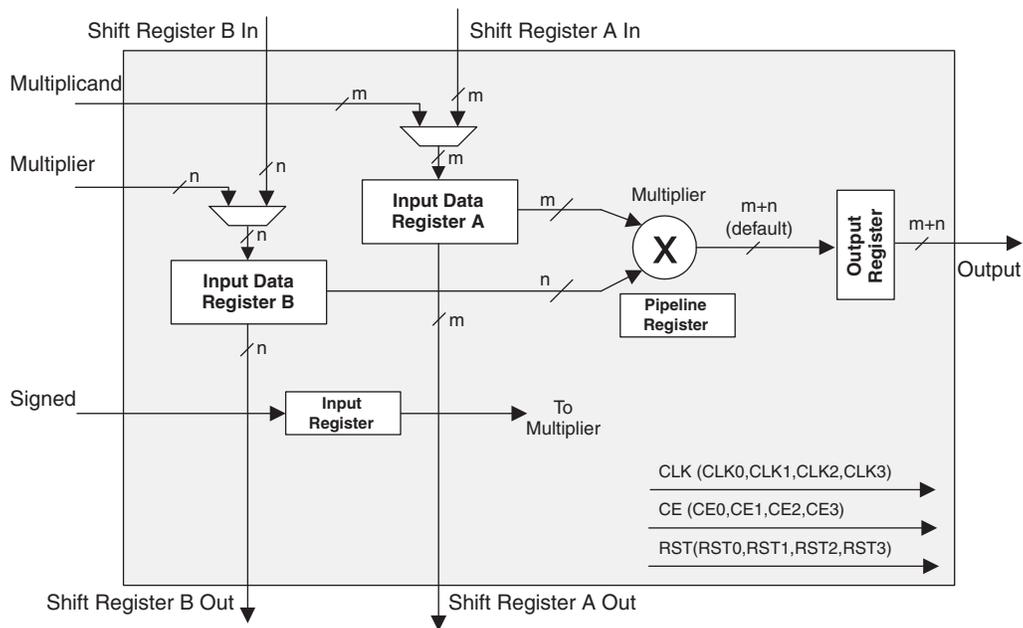
MAC sysDSP Element

In this case the two operands, A and B, are multiplied and the result is added with the previous accumulated value. This accumulated value is available at the output. The user can enable the input and pipeline registers but the output register is always enabled. The output register is used to store the accumulated value. A registered overflow signal is also available. The overflow conditions are provided later in this document. Figure 2-20 shows the MAC sysDSP element.

Figure 2-20. MAC sysDSP Element
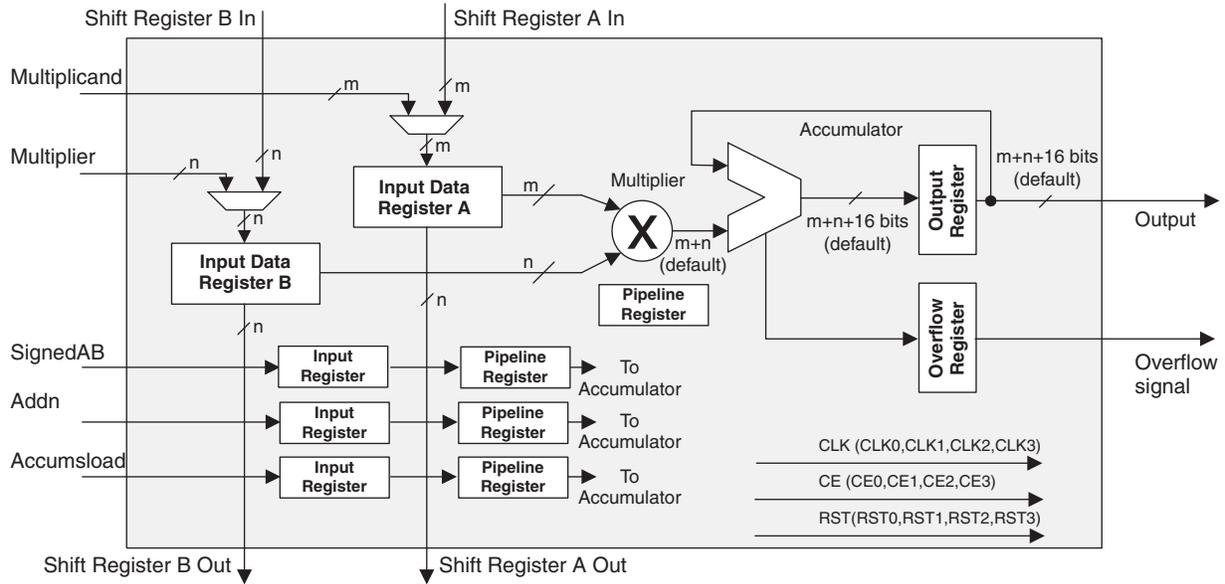
MULTADD sysDSP Element

In this case, the operands A0 and B0 are multiplied and the result is added/subtracted with the result of the multiplier operation of operands A1 and A2. The user can enable the input, output and pipeline registers. Figure 2-21 shows the MULTADD sysDSP element.

Figure 2-21. MULTADD
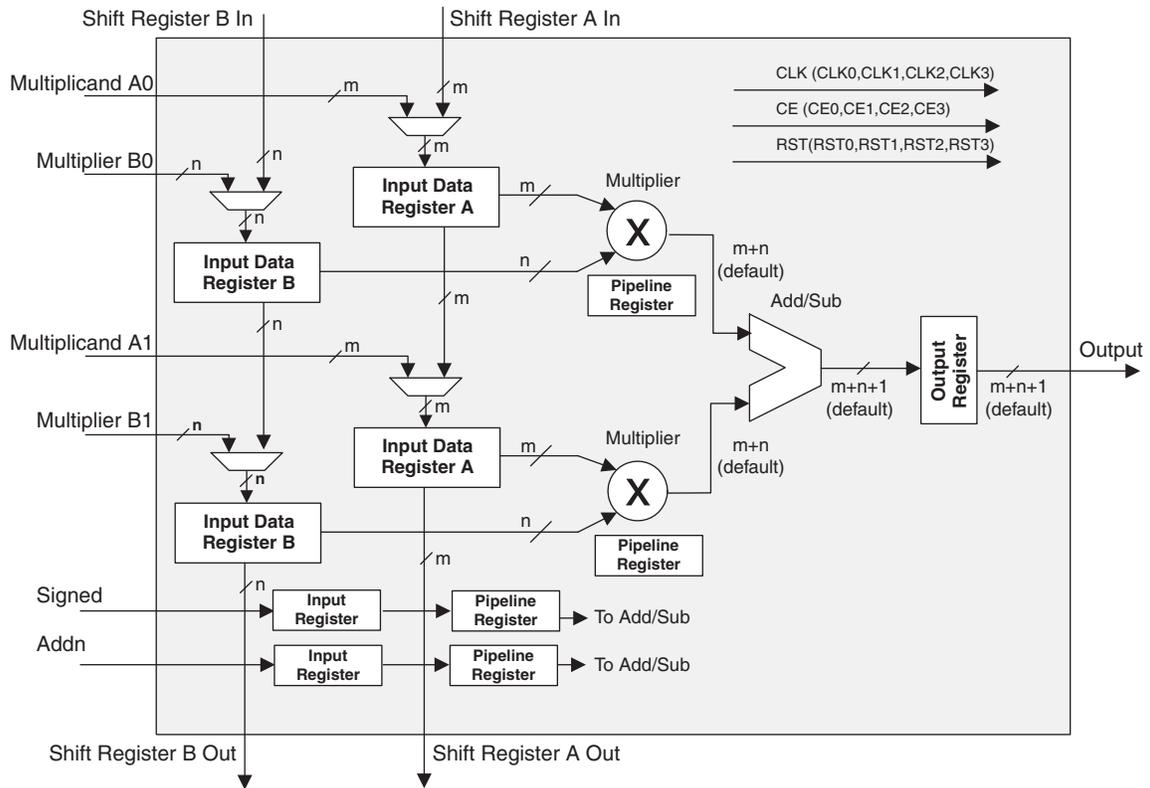
MULTADDSUM sysDSP Element

In this case, the operands A0 and B0 are multiplied and the result is added/subtracted with the result of the multiplier operation of operands A1 and B1. Additionally the operands A2 and B2 are multiplied and the result is added/subtracted with the result of the multiplier operation of operands A3 and B3. The result of both addition/subtraction are added in a summation block. The user can enable the input, output and pipeline registers. Figure 2-22 shows the MULTADDSUM sysDSP element.

Figure 2-22. MULTADDSUM
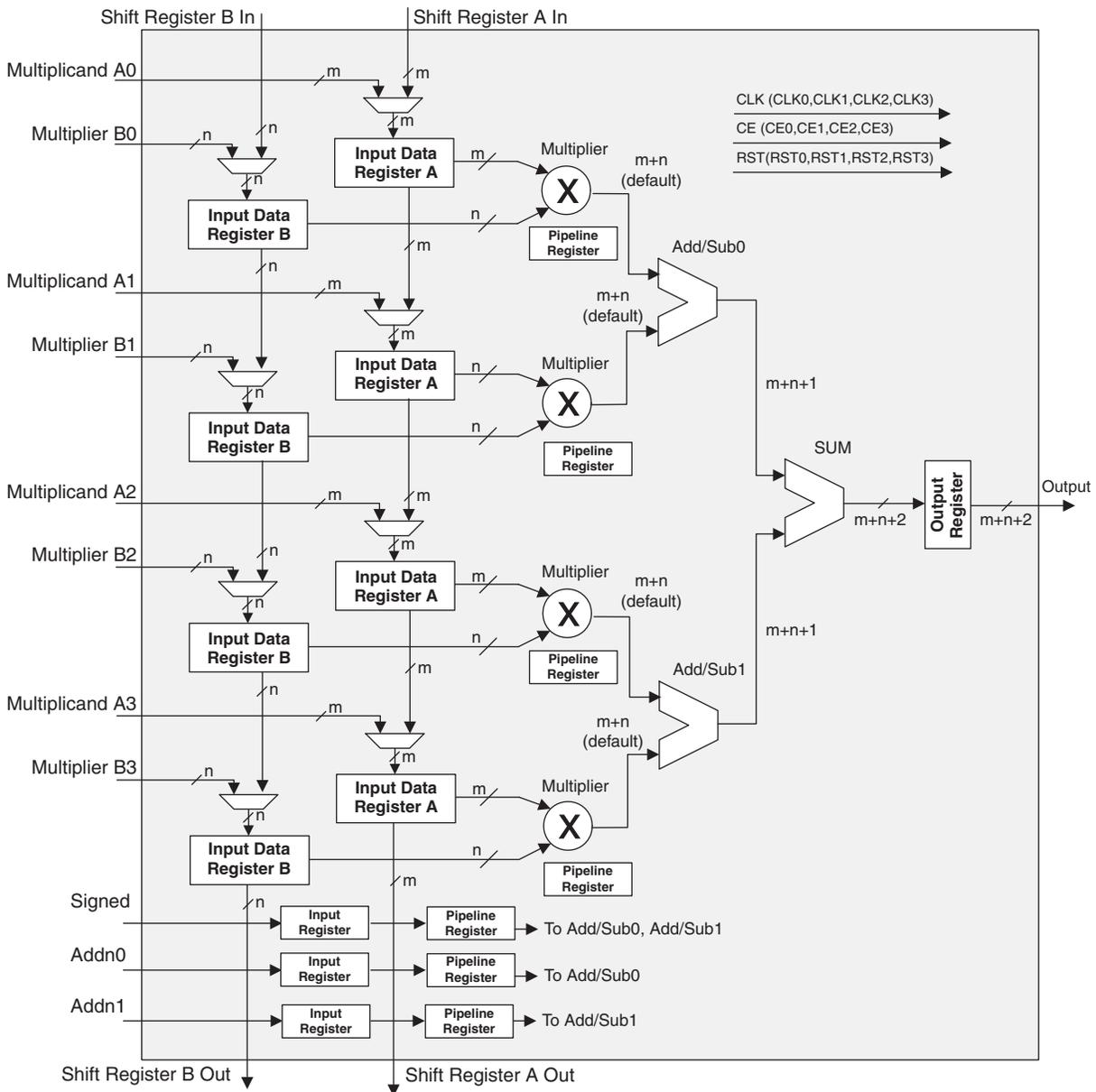
Clock, Clock Enable and Reset Resources

Global Clock, Clock Enable and Reset signals from routing are available to every DSP block. Four Clock, Reset and Clock Enable signals are selected for the sysDSP block. From four clock sources (CLK0, CLK1, CLK2, CLK3) one clock is selected for each input register, pipeline register and output register. Similarly Clock enable (CE) and Reset (RST) are selected from their four respective sources (CE0, CE1, CE2, CE3 and RST0, RST1, RST2, RST3) at each input register, pipeline register and output register.

Signed and Unsigned with Different Widths

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

Table 2-8. An Example of Sign Extension

Number	Unsigned	Unsigned 9-bit	Unsigned 18-bit	Signed	Two's Complement Signed 9-Bits	Two's Complement Signed 18-bits
+5	0101	000000101	00000000000000101	0101	000000101	00000000000000101
-6	0110	000000110	00000000000000110	1010	111111010	11111111111111010

OVERFLOW Flag from MAC

The sysDSP block provides an overflow output to indicate that the accumulator has overflowed. When two unsigned numbers are added and the result is a smaller number then accumulator roll over is said to occur and overflow signal is indicated. When two positive numbers are added with a negative sum and when two negative numbers are added with a positive sum, then the accumulator "roll-over" is said to have occurred and an overflow signal is indicated. Note when overflow occurs the overflow flag is present for only one cycle. By counting these overflow pulses in FPGA logic, larger accumulators can be constructed. The conditions overflow signals for signed and unsigned operands are listed in Figure 2-23.

Figure 2-23. Accumulator Overflow/Underflow Conditions


```
0101111100  252
0101111101  253
0101111110  254
0101111111  255
1010000000  256
1010000001  257
1010000010  258

000000011  3
000000010  2
000000001  1
000000000  0

111111111  511
111111110  510
111111101  509
```

Carry signal is generated for one cycle when this boundary is crossed

Unsigned Operation

```
0101111100  252
0101111101  253
0101111110  254
0101111111  255
1010000000  256
1010000001  255
1010000010  254

000000011  +3
000000010  +2
000000001  +1
000000000   0
111111111  -1
111111110  -2
111111101  -3
```

Overflow signal is generated for one cycle when this boundary is crossed

Signed Operation

IPexpress™

The user can access the sysDSP block via the IPexpress configuration tool, included with the ispLEVER design tool suite. IPexpress has options to configure each DSP module (or group of modules) or through direct HDL instantiation. Additionally Lattice has partnered Mathworks to support instantiation in the Simulink tool, which is a Graphical Simulation Environment. Simulink works with ispLEVER and dramatically shortens the DSP design cycle in Lattice FPGAs.

Optimized DSP Functions

Lattice provides a library of optimized DSP IP functions. Some of the IPs planned for LatticeECP DSP are: Bit Correlators, Fast Fourier Transform, Finite Impulse Response (FIR) Filter, Reed-Solomon Encoder/ Decoder, Turbo Encoder/Decoders and Convolutional Encoder/Decoder. Please contact Lattice to obtain the latest list of available DSP IPs.

Resources Available in the LatticeECP Family

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

Table 2-9. Number of DSP Blocks in LatticeECP Family

Device	DSP Block	9x9 Multiplier	18x18 Multiplier	36x36 Multiplier
LFECP6	4	32	16	4
LFECP10	5	40	20	5
LFECP15	6	48	24	6
LFECP20	7	56	28	7
LFECP33	8	64	32	8

Table 2-10. Embedded SRAM in LatticeECP Family

Device	EBR SRAM Block	Total EBR SRAM (Kbits)
LFECP6	10	92
LFECP10	30	276
LFECP15	38	350
LFECP20	46	424
LFECP33	54	498

DSP Performance of the LatticeECP Family

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

Table 2-11. DSP Block Performance of LatticeECP Family

Device	DSP Block	DSP Performance MMAC
LFECP6	4	3680
LFECP10	5	4600
LFECP15	6	5520
LFECP20	7	6440
LFECP33	8	7360

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

Programmable I/O Cells (PIC)

Each PIC contains two PIOs connected to their respective sysI/O Buffers which are then connected to the PADs as shown in Figure 2-24. The PIO Block supplies the output data (DO) and the Tri-state control signal (TO) to sysI/O buffer, and receives input from the buffer.

Figure 2-24. PIC Diagram
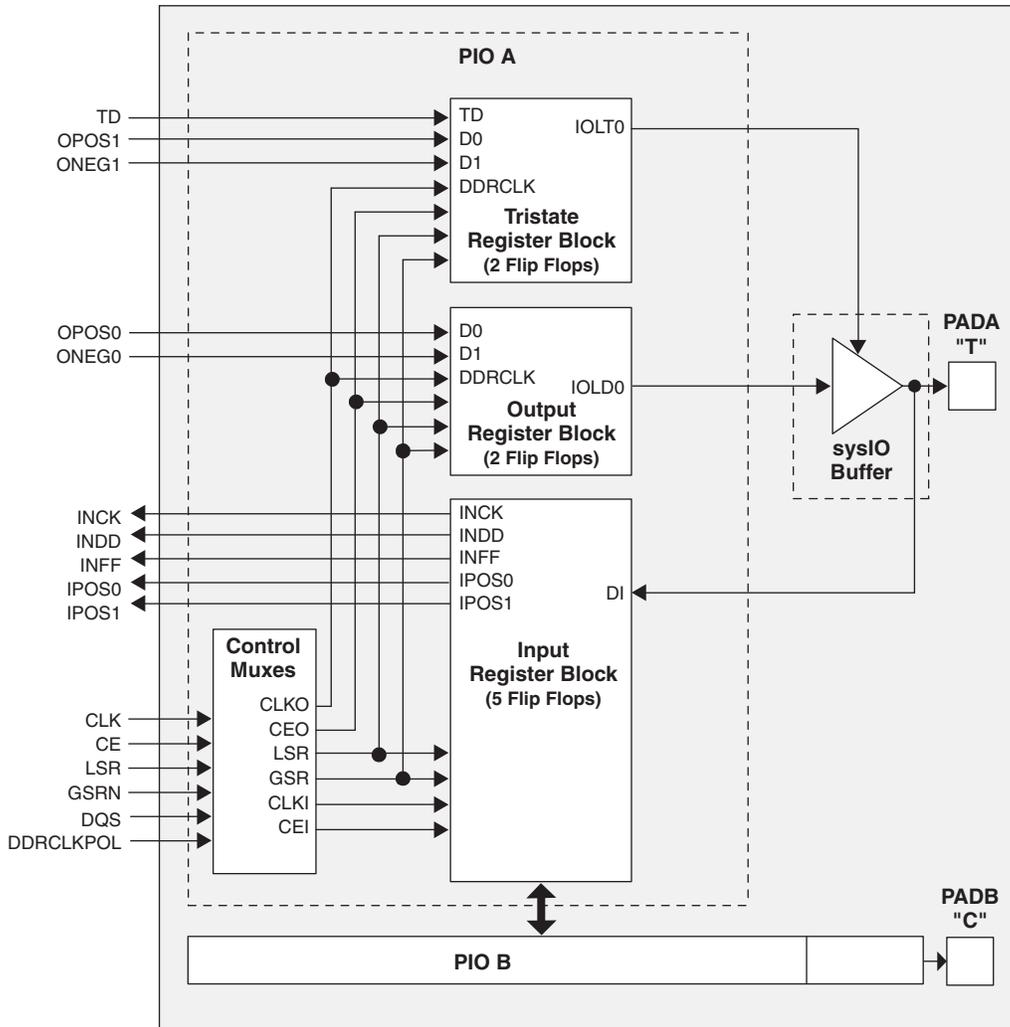
Two adjacent PIOs can be joined to provide a differential I/O pair (labeled as "T" and "C") as shown in Figure 2-25. The PAD Labels "T" and "C" distinguish the two PIOs. Only the PIO pairs on the left and right edges of the device can be configured as LVDS transmit/receive pairs.

One of every 16 PIOs contains a delay element to facilitate the generation of DQS signals. The DQS signal feeds the DQS bus which spans the set of 16 PIOs. Figure 2-25 shows the assignment of DQS pins in each set of 16 PIOs. The exact DQS pins are shown in a dual function in the Logic Signal Connections table at the end of this data sheet. Additional detail is provided in the Signal Descriptions table at the end of this data sheet. The DQS signal from the bus is used to strobe the DDR data from the memory into input register blocks. This interface is designed for memories that support one DQS strobe per eight bits of data.

Table 2-12. PIO Signal List

Name	Type	Description
CE0, CE1	Control from the core	Clock enables for input and output block FFs.
CLK0, CLK1	Control from the core	System clocks for input and output blocks.
LSR	Control from the core	Local Set/Reset.
GSRN	Control from routing	Global Set/Reset (active low).
INCK	Input to the core	Input to Primary Clock Network or PLL reference inputs.
DQS	Input to PIO	DQS signal from logic (routing) to PIO.
INDD	Input to the core	Unregistered data input to core.
INFF	Input to the core	Registered input on positive edge of the clock (CLK0).
IPOS0, IPOS1	Input to the core	DDRX registered inputs to the core.
ONEG0	Control from the core	Output signals from the core for SDR and DDR operation.
OPOS0,	Control from the core	Output signals from the core for DDR operation
OPOS1 ONEG1	Tristate control from the core	Signals to Tristate Register block for DDR operation.
TD	Tristate control from the core	Tristate signal from the core used in SDR operation.
DDRCLKPOL	Control from clock polarity bus	Controls the polarity of the clock (CLK0) that feed the DDR input block.

Figure 2-25. DQS Routing
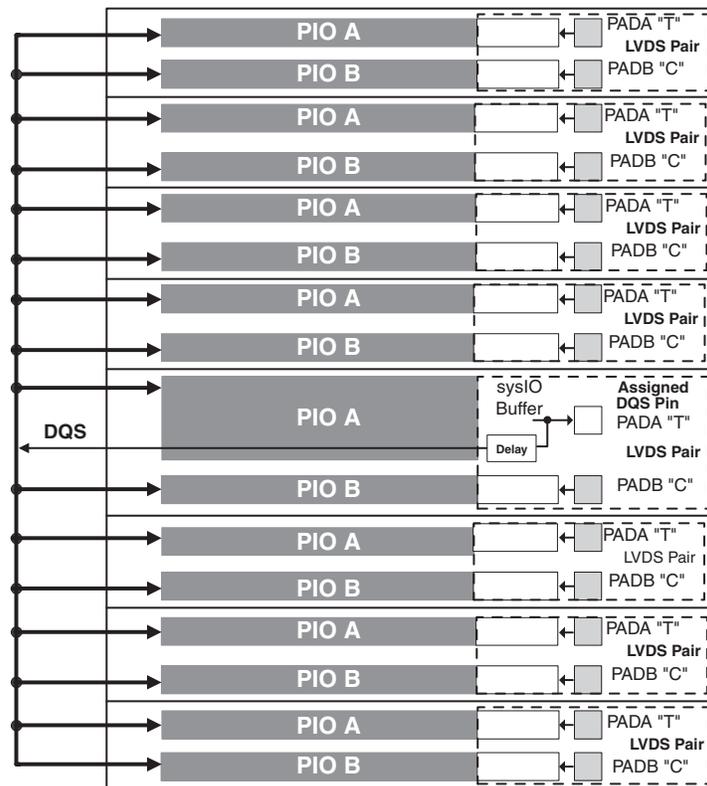
PIO

The PIO contains four blocks: an input register block, output register block, tristate register block and a control logic block. These blocks contain registers for both single data rate (SDR) and double data rate (DDR) operation along with the necessary clock and selection logic. Programmable delay lines used to shift incoming clock and data signals are also included in these blocks.

Input Register Block

The input register block contains delay elements and registers that can be used to condition signals before they are passed to the device core. Figure 2-26 shows the diagram of the input register block.

Input signals are fed from the sysI/O buffer to the input register block (as signal DI). If desired the input signal can bypass the register and delay elements and be used directly as a combinatorial signal (INDD), a clock (INCK) and in selected blocks the input to the DQS delay block. If one of the bypass options is not chosen, the signal first passes through an optional delay block. This delay, if selected, reduces input-register hold-time requirement when using a global clock.

The input block allows two modes of operation. In the single data rate (SDR) the data is registered, by one of the registers in the single data rate sync register block, with the system clock. In the DDR Mode two registers are used to sample the data on the positive and negative edges of the DQS signal creating two data streams, D0 and D2. These two data streams are synchronized with the system clock before entering the core. Further discussion on this topic is in the DDR Memory section of this data sheet.

Figure 2-27 shows the input register waveforms for DDR operation and Figure 2-28 shows the design tool primitives. The SDR/SYNC registers have reset and clock enable available.

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

Figure 2-26. Input Register Diagram
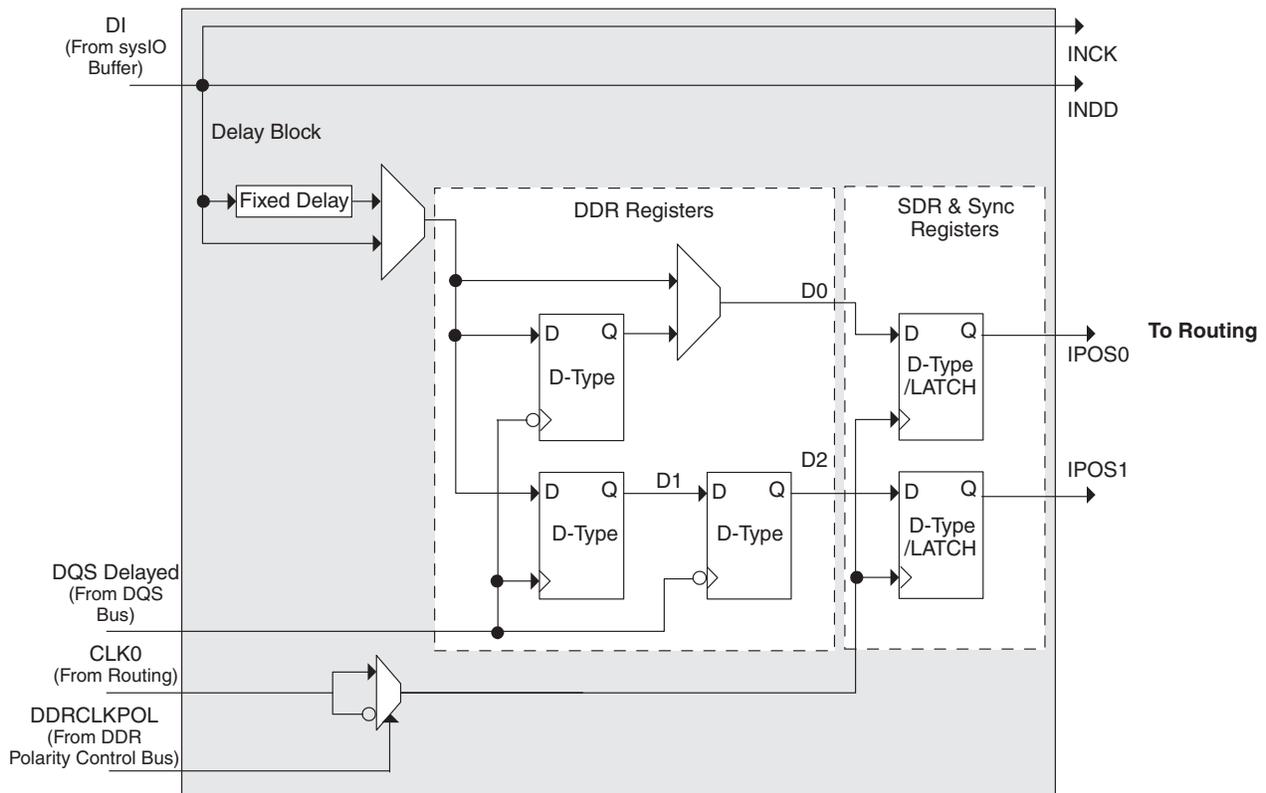
Figure 2-27. Input Register DDR Waveforms
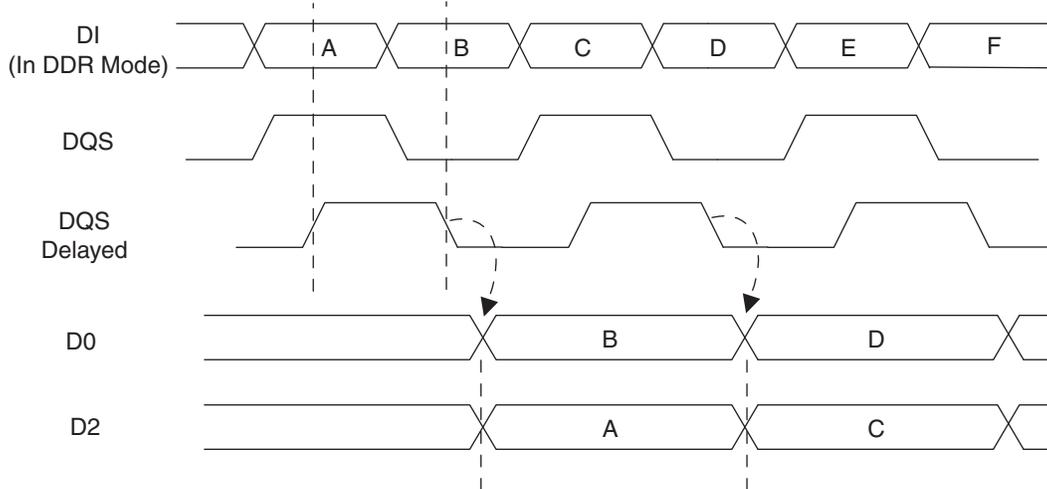
Figure 2-28. INDDRXB Primitive
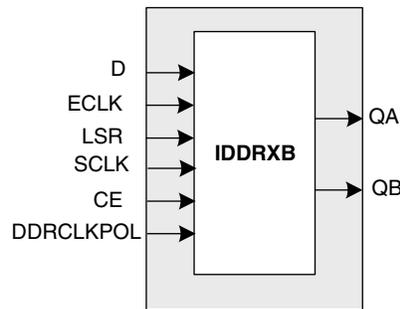
Output Register Block

The output register block provides the ability to register signals from the core of the device before they are passed to the sysI/O buffers. The block contains a register for SDR operation that is combined with an additional latch for DDR operation. Figure 2-29 shows the diagram of the Output Register Block.

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

Figure 2-30 shows the design tool DDR primitives. The SDR output register has reset and clock enable available. The additional register for DDR operation does not have reset or clock enable available.

Figure 2-29. Output Register Block
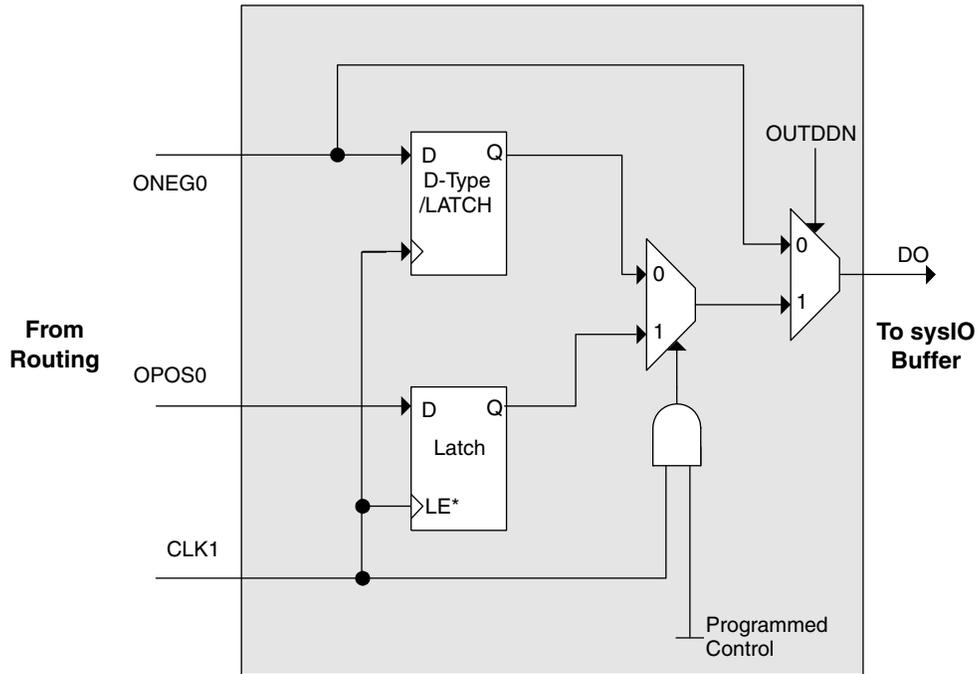
*Latch is transparent when input is low.

Figure 2-30. ODDRXB Primitive
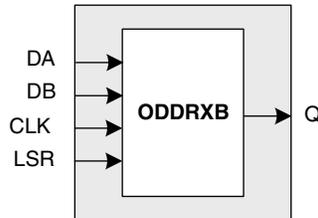
Tristate Register Block

The tristate register block provides the ability to register tri-state control signals from the core of the device before they are passed to the sysI/O buffers. The block contains a register for SDR operation and an additional latch for DDR operation. Figure 2-31 shows the diagram of the Tristate Register Block.

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

Figure 2-31. Tristate Register Block

*Latch is transparent when input is low.
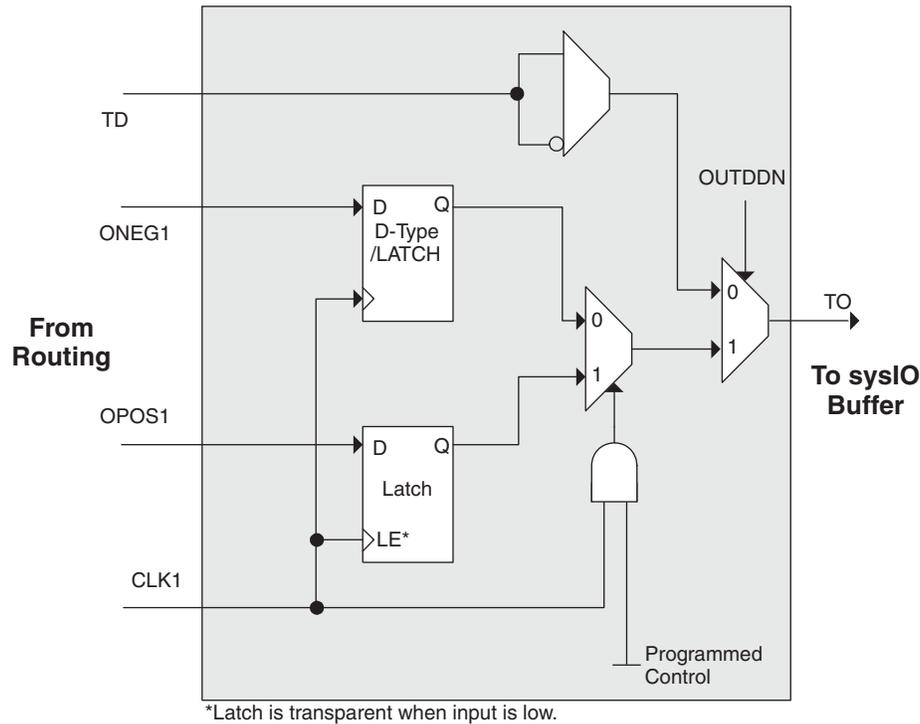
Control Logic Block

The control logic block allows the selection and modification of control signals for use in the PIO block. A clock is selected from one of the clock signals provided from the general purpose routing and a DQS signal provided from the programmable DQS pin. The clock can optionally be inverted.

The clock enable and local reset signals are selected from the routing and optionally inverted. The global tristate signal is passed through this block.

DDR Memory Support

Implementing high performance DDR memory interfaces requires dedicated DDR register structures in the input (for read operations) and in the output (for write operations). As indicated in the PIO Logic section, the LatticeEC devices provide this capability. In addition to these registers, the LatticeEC devices contain two elements to simplify the design of input structures for read operations: the DQS delay block and polarity control logic.

DLL Calibrated DQS Delay Block

Source Synchronous interfaces generally require the input clock to be adjusted in order to correctly capture data at the input register. For most interfaces a PLL is used for this adjustment. However in DDR memories the clock (referred to as DQS) is not free running so this approach cannot be used. The DQS Delay block provides the required clock alignment for DDR memory interfaces.

The DQS signal (selected PIOs only) feeds from the PAD through a DQS delay element to a dedicated DQS routing resource. The DQS signal also feeds polarity control logic, which controls the polarity of the clock to the sync registers in the input register blocks. Figures 2-32 and 2-33 show how the DQS transition signals are routed to the PIOs.

The temperature, voltage and process variations of the DQS delay block are compensated by a set of calibration (6-bit bus) signals from two DLLs on opposite sides of the device. Each DLL compensates DQS Delays in its half of the device as shown in Figure 2-33. The DLL loop is compensated for temperature, voltage and process variations by the system clock and feedback loop.

Figure 2-32. DQS Local Bus.
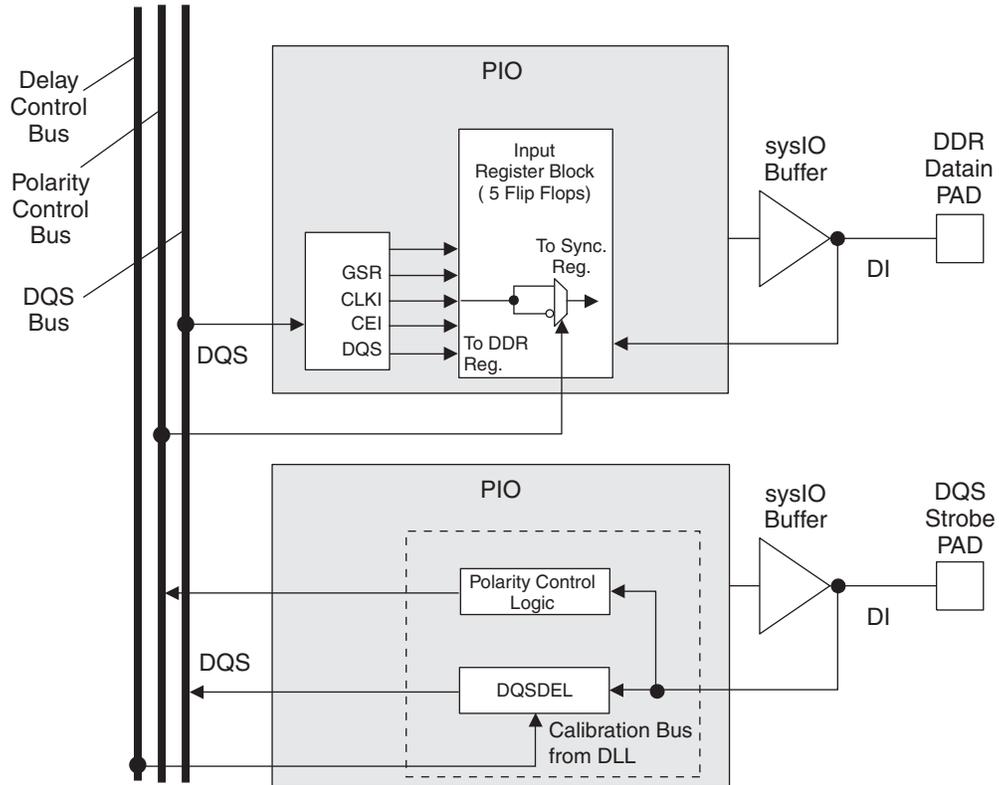
Figure 2-33. DLL Calibration Bus and DQS/DQS Transition Distribution
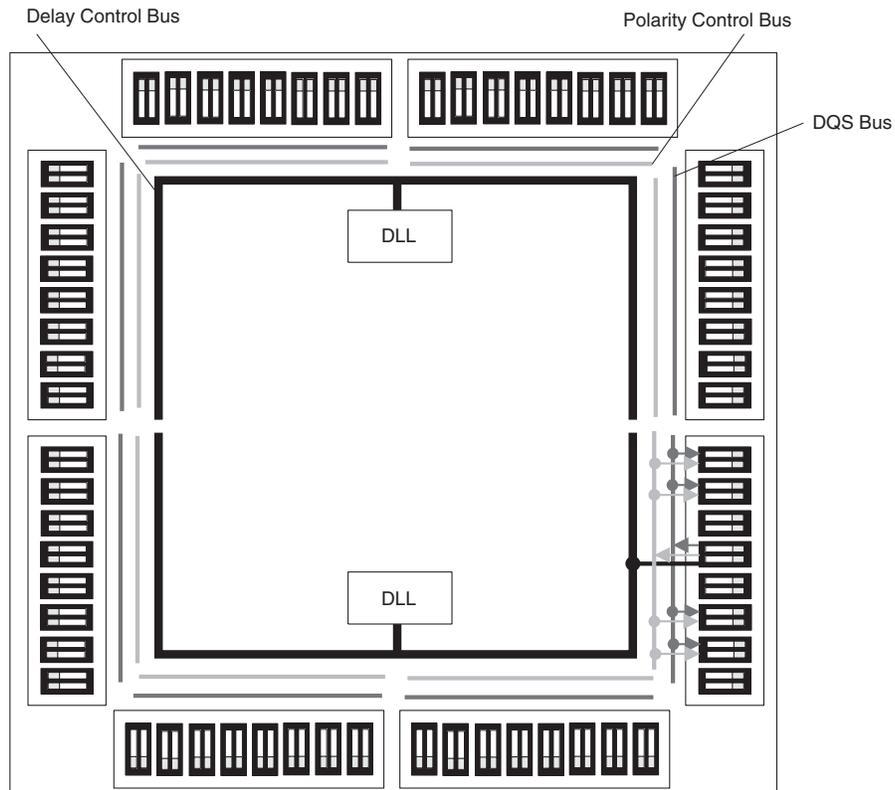
Polarity Control Logic

In a typical DDR Memory interface design, the phase relation between the incoming delayed DQS strobe and the internal system Clock (during the READ cycle) is unknown.

The LatticeECP/EC family contains dedicated circuits to transfer data between these domains. To prevent setup and hold violations at the domain transfer between DQS (delayed) and the system Clock a clock polarity selector is used. This changes the edge on which the data is registered in the synchronizing registers in the input register block. This requires evaluation at the start of each READ cycle for the correct clock polarity.

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

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 eight 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 LVCMOS, SSTL, HSTL, LVDS and LVPECL.

sysI/O Buffer Banks

LatticeECP/EC devices have eight sysI/O buffer banks; each is capable of supporting multiple I/O standards. Each sysI/O bank has its own I/O supply voltage (V_{CCIO}), and two voltage references V_{REF1} and V_{REF2} resources allowing each bank to be completely independent from each other. Figure 2-34 shows the eight banks and their associated supplies.

In the LatticeECP/EC devices, single-ended output buffers and ratioed input buffers (LVTTL, LVCMOS, PCI and PCI-X) are powered using $V_{CCIO.}$ LVTTL, LVCMOS33, LVCMOS25 and LVCMOS12 can also be set as fixed threshold input independent of V_{CCIO}. In addition to the bank V_{CCIO} supplies, the LatticeECP/EC devices have a V_{CC} core logic power supply, and a V_{CCAUX} supply that power all differential and referenced buffers.

Each bank can support up to two separate VREF voltages, VREF1 and VREF2 that set the threshold for the referenced input buffers. In the LatticeECP/EC devices, some dedicated I/O pins in a bank can be configured to be a reference voltage supply pin. Each I/O is individually configurable based on the bank's supply and reference voltages.

Figure 2-34. LatticeECP/EC Banks
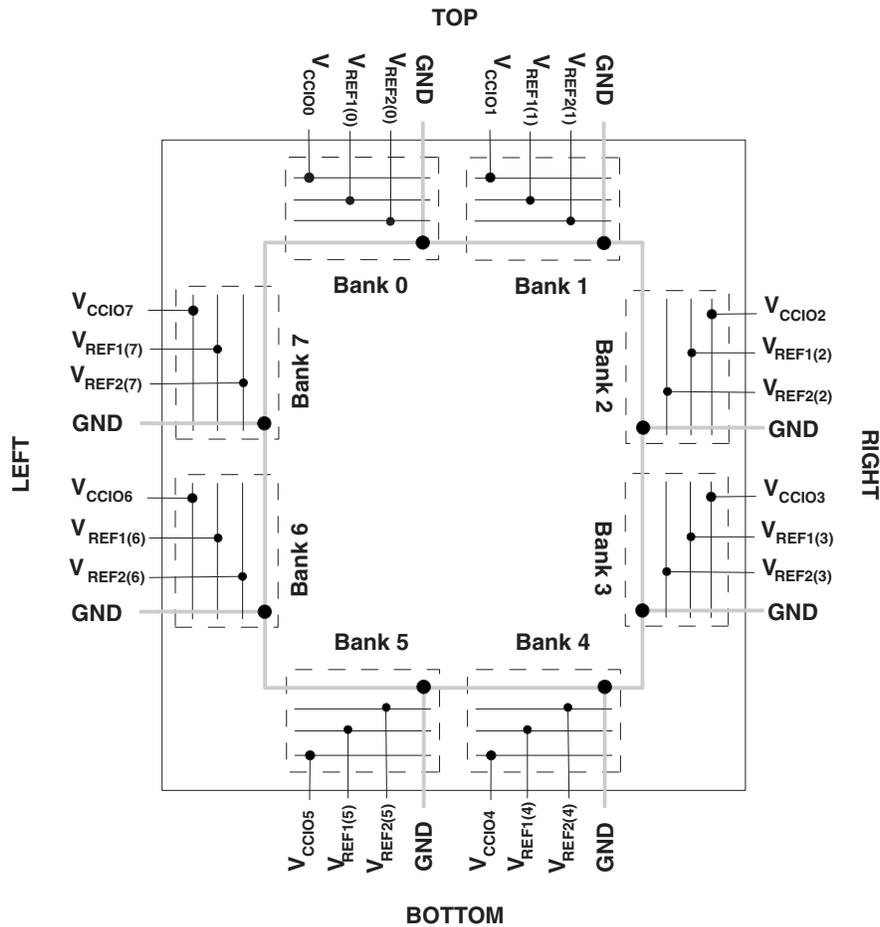
LatticeECP/EC devices contain two types of sysI/O buffer pairs.

1. **Top and Bottom sysI/O Buffer Pairs (Single-Ended Outputs Only)**
   The sysI/O buffer pairs in the top and bottom banks of the device consist of two single-ended output drivers and two sets of single-ended input buffers (both ratioed and referenced). The referenced input buffer can also be configured as a differential input.

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

   Only the I/Os on the top and bottom banks have programmable PCI clamps. These I/O banks also support hot socketing with IDK less than 1mA. Note that the PCI clamp is enabled after V_{CC}, V_{CCAUX} and V_{CCIO} are at valid operating levels and the device has been configured.

2. **Left and Right sysI/O Buffer Pairs (Differential and 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. The referenced input buffer 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.

   Only the left and right banks have LVDS differential output drivers. See the I_{DK} specification for I/O leakage current during power-up.

Typical I/O Behavior During Power-up

The internal power-on-reset (POR) signal is deactivated when V_{CC} 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 LatticeECP/EC 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 Standards

The LatticeECP/EC 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 1.2, 1.5, 1.8, 2.5 and 3.3V standards. In the LVCMOS and LVTTL modes, the buffer has individually configurable options for drive strength, bus maintenance (weak pull-up, weak pull-down, or a bus-keeper latch) and open drain. Other single-ended standards supported include SSTL and HSTL. Differential standards supported include LVDS, BLVDS, LVPECL, RSDS, differential SSTL and differential HSTL. Tables 2-13 and 2-14 show the I/O standards (together with their supply and reference voltages) supported by the LatticeECP/EC devices. For further information about utilizing the sysI/O buffer to support a variety of standards please see the the list of technical information at the end of this data sheet.

Table 2-13. Supported Input Standards

Input Standard	V_{REF} (Nom.)	V_{CCIO}[1] (Nom.)
Single Ended Interfaces		
LVTTL	—	—
LVCMOS33[2]	—	—
LVCMOS25[2]	—	—
LVCMOS18	—	1.8
LVCMOS15	—	1.5
LVCMOS12[2]	—	—
PCI	—	3.3
HSTL18 Class I, II	0.9	—
HSTL18 Class III	1.08	—
HSTL15 Class I	0.75	—
HSTL15 Class III	0.9	—
SSTL3 Class I, II	1.5	—
SSTL2 Class I, II	1.25	—
SSTL18 Class I	0.9	—
Differential Interfaces		
Differential SSTL18 Class I	—	—
Differential SSTL2 Class I, II	—	—
Differential SSTL3 Class I, II	—	—
Differential HSTL15 Class I, III	—	—
Differential HSTL18 Class I, II, III	—	—
LVDS, LVPECL, BLVDS, RSDS	—	—

1. When not specified V_{CCIO} can be set anywhere in the valid operating range.
2. JTAG inputs do not have a fixed threshold option and always follow V_{CCJ}.

Table 2-14. Supported Output Standards

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

1. Emulated with external resistors.

Hot Socketing

The LatticeECP/EC devices have been carefully designed to ensure predictable behavior during power-up and power-down. Power supplies can be sequenced in any order. During power up and power-down sequences, the I/Os remain in 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, this allows for easy integration with the rest of the system. These capabilities make the LatticeECP/EC ideal for many multiple power supply and hot-swap applications.

Configuration and Testing

The following section describes the configuration and testing features of the LatticeECP/EC devices.

IEEE 1149.1-Compliant Boundary Scan Testability

All LatticeECP/EC devices have boundary scan cells that are accessed through an IEEE 1149.1 compliant test access port (TAP). This allows functional testing of the circuit board, on which the device is mounted, through a serial scan path that can access all critical logic nodes. Internal registers are linked internally, allowing test data to

be shifted in and loaded directly onto test nodes, or test data to be captured and shifted out for verification. The test access port consists of dedicated I/Os: TDI, TDO, TCK and TMS. The test access port has its own supply voltage V_{CCJ} and can operate with LVCMOS3.3, 2.5, 1.8, 1.5 and 1.2 standards.

For more details on boundary scan test, please see information regarding additional technical documentation at the end of this data sheet.

Device Configuration

All LatticeECP/EC devices contain two possible ports that can be used for device configuration. The test access port (TAP), which supports bit-wide configuration, and the sysCONFIG port that supports both byte-wide and serial configuration.

The TAP supports both the IEEE Std. 1149.1 Boundary Scan specification and the IEEE Std. 1532 In-System Configuration specification. The sysCONFIG port is a 20-pin interface with six of the I/Os used as dedicated pins and the rest being dual-use pins (please refer to TN1053 for more information about using the dual-use pins as general purpose I/O). There are four configuration options for LatticeECP/EC devices:

1. Industry standard SPI memories.

2. Industry standard byte wide flash and ispMACH 4000 for control/addressing.

3. Configuration from system microprocessor via the configuration bus or TAP.

4. Industry standard FPGA board memory.

On power-up, the FPGA SRAM is ready to be configured with the sysCONFIG port active. The IEEE 1149.1 serial mode can be activated any time after power-up by sending the appropriate command through the TAP port. Once a configuration port is selected, that port is locked and another configuration port cannot be activated until the next power-up sequence.

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

Internal Logic Analyzer Capability (ispTRACY)

All LatticeECP/EC devices support an internal logic analyzer diagnostic feature. The diagnostic features provide capabilities similar to an external logic analyzer, such as programmable event and trigger condition and deep trace memory. This feature is enabled by Lattice's ispTRACY. The ispTRACY utility is added into the user design at compile time.

For more information about ispTRACY, please see information regarding additional technical documentation at the end of this data sheet.

External Resistor

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

Oscillator

Every LatticeECP/EC device has an internal CMOS oscillator which is used to derive a master clock for configuration. The oscillator and the master clock run continuously. The default value of the master clock is 2.5MHz. Table 2-15 lists all the available Master Clock frequencies. When a different Master Clock is selected during the design process, the following sequence takes place:

1. User selects a different Master Clock frequency.

2. During configuration the device starts with the default (2.5MHz) Master Clock frequency.

3. The clock configuration settings are contained in the early configuration bit stream.

4. The Master Clock frequency changes to the selected frequency once the clock configuration bits are received.

For further information about the use of this oscillator for configuration, please see the list of technical documentation at the end of this data sheet.

Table 2-15. Selectable Master Clock (CCLK) Frequencies During Configuration

CCLK (MHz)	CCLK (MHz)	CCLK (MHz)
2.5*	13	45
4.3	15	51
5.4	20	55
6.9	26	60
8.1	30	130
9.2	34	—
10.0	41	—

Density Shifting

The LatticeECP/EC family has been designed to ensure that different density devices in the same package have the same pin-out. Furthermore, the architecture ensures a high success rate when performing design migration from lower density parts to higher density parts. In many cases, it is also possible to shift a lower utilization design targeted for a high-density device to a lower density device. However, the exact details of the final resource utilization will impact the likely success in each case.

Absolute Maximum Ratings[1, 2, 3]

Supply Voltage V_{CC} . . . . -0.5 to 1.32V

Supply Voltage V_{CCAUX} . . . . -0.5 to 3.75V

Supply Voltage V_{CCJ} . . . . -0.5 to 3.75V

Output Supply Voltage V_{CCIO} . . . . -0.5 to 3.75V

Dedicated Input Voltage Applied[4] . . . . -0.5 to 4.25V

I/O Tristate Voltage Applied [4] . . . . -0.5 to 3.75V

Storage Temperature (Ambient) . . . . -65 to 150°C

Junction Temp. (Tj) . . . . +125°C

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

Recommended Operating Conditions

Symbol	Parameter	Min.	Max.	Units
V_{CC}	Core Supply Voltage	1.14	1.26	V
V_{CCAUX}^{3}	Auxiliary Supply Voltage	3.135	3.465	V
V_{CCPLL}	PLL Supply Voltage for ECP/EC33	1.14	1.26	V
$V_{CCIO}^{1, 2}$	I/O Driver Supply Voltage	1.140	3.465	V
V_{CCJ}^{1}	Supply Voltage for IEEE 1149.1 Test Access Port	1.140	3.465	V
t_{JCOM}	Junction Commercial Operation	0	85	°C
t_{JIND}	Junction Industrial Operation	-40	100	°C

1. If V_{CCIO} or V_{CCJ} is set to 1.2V, they must be connected to the same power supply as V_{CC}. If V_{CCIO} or V_{CCJ} is set to 3.3V, they must be connected to the same power supply as V_{CCAUX}.
2. See recommended voltages by I/O standard in subsequent table.
3. V_{CCAUX} ramp rate must not exceed 3mV/μs for commercial and 0.6 mV/μs for industrial device operations during power up when transitioning between 0.8V and 1.8V.

Hot Socketing Specifications[1, 2, 3, 4]

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
Top and Bottom General Purpose sysI/Os (Banks 0, 1, 4 and 5), JTAG and Dedicated sysCONFIG Pins						
I_{DK_TB}	Input or I/O Leakage Current	0 ð V_{IN} ð V_{IH} (MAX.)	—	—	+/-1000	μA
Left and Right General Purpose sysI/Os (Banks 2, 3, 6 and 7)						
I_{DK_LR}	Input or I/O Leakage Current	V_{IN} ð V_{CCIO}	—	—	+/-1000	μA
		V_{IN} > V_{CCIO}	—	35	—	mA

1. Insensitive to sequence of V_{CC}, V_{CCAUX} and V_{CCIO}. However, assumes monotonic rise/fall rates for V_{CC}, V_{CCAUX} and V_{CCIO}.
2. 0 ð V_{CC} ð V_{CC} (MAX), 0 ð V_{CCIO} ð V_{CCIO} (MAX) or 0 ð V_{CCAUX} ð V_{CCAUX} (MAX).
3. I_{DK} is additive to I_{PU}, I_{PW} or I_{BH}.
4. LVCMOS and LVTTL only.

DC Electrical Characteristics

Over Recommended Operating Conditions

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
I_{IL}, I_{IH}[1]	Input or I/O Leakage	$0 \leq V_{IN} \leq (V_{CCIO} - 0.2V)$	—	—	10	µA
I_{IH}[1, 3]	Input or I/O High Leakage	$(V_{CCIO} - 0.2V) \leq V_{IH} \leq 3.6V$	—	—	40	µA
I_{PU}	I/O Active Pull-up Current	$0 \leq V_{IN} \leq 0.7\ V_{CCIO}$	-30	—	-150	µA
I_{PD}	I/O Active Pull-down Current	V_{IL} (MAX) $\leq V_{IN} \leq V_{IH}$ (MAX)	30	—	150	µA
I_{BHLS}	Bus Hold Low sustaining current	$V_{IN} = V_{IL}$ (MAX)	30	—	—	µA
I_{BHHS}	Bus Hold High sustaining current	$V_{IN} = 0.7V_{CCIO}$	-30	—	—	µA
I_{BHLO}	Bus Hold Low Overdrive current	$0 \leq V_{IN} \leq V_{IH}$ (MAX)	—	—	150	µA
I_{BHLH}	Bus Hold High Overdrive current	$0 \leq V_{IN} \leq V_{IH}$ (MAX)	—	—	-150	µA
V_{BHT}	Bus Hold trip Points	$0 \leq V_{IN} \leq V_{IH}$ (MAX)	V_{IL} (MAX)	—	V_{IH} (MIN)	V
C1	I/O Capacitance[2]	V_{CCIO} = 3.3V, 2.5V, 1.8V, 1.5V, 1.2V, V_{CC} = 1.2V, V_{IO} = 0 to V_{IH} (MAX)	—	8	—	pf
C2	Dedicated Input Capacitance[2]	V_{CCIO} = 3.3V, 2.5V, 1.8V, 1.5V, 1.2V, V_{CC} = 1.2V, V_{IO} = 0 to V_{IH} (MAX)	—	6	—	pf

1. Input or I/O leakage current is measured with the pin configured as an input or as an I/O with the output driver tri-stated. It is not measured with the output driver active. Bus maintenance circuits are disabled.
2. T_A 25°C, f = 1.0MHz
3. For top and bottom general purpose I/O pins, when V_{IH} is higher than V_{CCIO}, a transient current typically of 30ns in duration or less with a peak current of 6mA can occur on the high-to-low transition. For left and right I/O banks, V_{IH} must be less than or equal to V_{CCIO}.

Supply Current (Standby)[1, 2, 3, 4]

Over Recommended Operating Conditions

Symbol	Parameter	Device	Typ.[5]	Units
I_{CC}	Core Power Supply Current	LFEC1	6	mA
		LFEC3	10	mA
		LFECP6/LFEC6	15	mA
		LFECP10/LFEC10	25	mA
		LFECP15/LFEC15	35	mA
		LFECP20/LFEC20	60	mA
		LFECP33/LFEC33	85	mA
I_{CCAUX}	Auxiliary Power Supply Current		15	mA
I_{CCPLL}	PLL Power Supply Current		5	mA
I_{CCIO}	Bank Power Supply Current[6]		2	mA
I_{CCJ}	V_{CCJ} Power Supply Current		5	mA

1. For further information about 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 0MHz.
4. Pattern represents a "blank" configuration data file.
5. T_J=25°C, power supplies at nominal voltage.
6. Per bank.

Initialization Supply Current[1, 2, 3, 4, 5, 6]

Over Recommended Operating Conditions

Symbol	Parameter	Devices	Typ.[6]	Units
I_{CC}	Core Power Supply Current	LFEC1	25	mA
		LFEC3	40	mA
		LFECP6/LFEC6	50	mA
		LFECP10/LFEC10	60	mA
		LFECP15/LFEC15	70	mA
		LFECP20/LFEC20	150	mA
		LFECP33/LFEC33	220	mA
I_{CCAUX}	Auxiliary Power Supply Current	LFEC1	30	mA
		LFEC3	30	mA
		LFECP6/LFEC6	30	mA
		LFECP10/LFEC10	35	mA
		LFECP15/LFEC15	35	mA
		LFECP20/LFEC20	40	mA
		LFECP33/LFEC33	40	mA
I_{CCPLL}	PLL Power Supply Current		12	mA
I_{CCIO}	Bank Power Supply Current[7]	LFEC1	4	mA
		LFEC3	5	mA
		LFECP6/LFEC6	6	mA
		LFECP10/LFEC10	6	mA
		LFECP15/LFEC15	7	mA
		LFECP20/LFEC20	8	mA
		LFECP33/LFEC33	8	mA
I_{CCJ}	V_{CCJ} Power Supply Current		20	mA

1. Until DONE signal is active.
2. For further information about supply current, please see the list of technical documentation at the end of this data sheet.
3. Assumes all outputs are tristated, all inputs are configured as LVCMOS and held at the V_{CCIO} or GND.
4. Frequency 0MHz.
5. Pattern represents typical design with 65% logic, 55% EBR, 10% routing utilization.
6. T_J=25ºC, power supplies at nominal voltage.
7. Per bank.

sysI/O Recommended Operating Conditions

Standard	V_CCIO			V_REF (V)		
	Min.	Typ.	Max.	Min.	Typ.	Max.
LVCMOS 3.3	3.135	3.3	3.465	—	—	—
LVCMOS 2.5	2.375	2.5	2.625	—	—	—
LVCMOS 1.8	1.71	1.8	1.89	—	—	—
LVCMOS 1.5	1.425	1.5	1.575	—	—	—
LVCMOS 1.2	1.14	1.2	1.26	—	—	—
LVTTL	3.135	3.3	3.465	—	—	—
PCI	3.135	3.3	3.465	—	—	—
SSTL18 Class I	1.71	1.8	1.89	0.833	0.90	0.969
SSTL2 Class I, II	2.375	2.5	2.625	1.15	1.25	1.35
SSTL3 Class I, II	3.135	3.3	3.465	1.3	1.5	1.7
HSTL15 Class I	1.425	1.5	1.575	0.68	0.75	0.9
HSTL15 Class III	1.425	1.5	1.575	—	0.9	—
HSTL 18 Class I, II	1.71	1.8	1.89	—	0.9	—
HSTL 18 Class III	1.71	1.8	1.89	—	1.08	—
LVDS	2.375	2.5	2.625	—	—	—
LVPECL[1]	3.135	3.3	3.465	—	—	—
BLVDS[1]	2.375	2.5	2.625	—	—	—
RSDS[1]	2.375	2.5	2.625	—	—	—

1. Outputs are implemented with the addition of external resistors. V_{CCIO} applies to outputs only.

sysI/O Single-Ended DC Electrical Characteristics

Input/Output Standard	V_{IL}		V_{IH}		V_{OL} Max. (V)	V_{OH} Min. (V)	I_{OL}[1] (mA)	I_{OH}[1] (mA)
	Min. (V)	Max. (V)	Min. (V)	Max. (V)				
LVCMOS 3.3	-0.3	0.8	2.0	3.6	0.4	V_{CCIO} - 0.4	20, 16, 12, 8, 4	-20, -16, -12, -8, -4
					0.2	V_{CCIO} - 0.2	0.1	-0.1
LVTTL	-0.3	0.8	2.0	3.6	0.4	V_{CCIO} - 0.4	20, 16, 12, 8, 4	-20, -16, -12, -8, -4
					0.2	V_{CCIO} - 0.2	0.1	-0.1
LVCMOS 2.5	-0.3	0.7	1.7	3.6	0.4	V_{CCIO} - 0.4	20, 16, 12, 8, 4	-20, -16, -12, -8, -4
					0.2	V_{CCIO} - 0.2	0.1	-0.1
LVCMOS 1.8	-0.3	$0.35V_{CCIO}$	$0.65V_{CCIO}$	3.6	0.4	V_{CCIO} - 0.4	16, 12, 8, 4	-16, -12, -8, -4
					0.2	V_{CCIO} - 0.2	0.1	-0.1
LVCMOS 1.5	-0.3	$0.35V_{CCIO}$	$0.65V_{CCIO}$	3.6	0.4	V_{CCIO} - 0.4	8, 4	-8, -4
					0.2	V_{CCIO} - 0.2	0.1	-0.1
LVCMOS 1.2	-0.3	$0.35V_{CC}$	$0.65V_{CC}$	3.6	0.4	V_{CCIO} - 0.4	6, 2	-6, -2
					0.2	V_{CCIO} - 0.2	0.1	-0.1
PCI	-0.3	$0.3V_{CCIO}$	$0.5V_{CCIO}$	3.6	$0.1V_{CCIO}$	$0.9V_{CCIO}$	1.5	-0.5
SSTL3 class I	-0.3	V_{REF} - 0.2	V_{REF} + 0.2	3.6	0.7	V_{CCIO} - 1.1	8	-8
SSTL3 class II	-0.3	V_{REF} - 0.2	V_{REF} + 0.2	3.6	0.5	V_{CCIO} - 0.9	16	-16
SSTL2 class I	-0.3	V_{REF} - 0.18	V_{REF} + 0.18	3.6	0.54	V_{CCIO} - 0.62	7.6	-7.6
SSTL2 class II	-0.3	V_{REF} - 0.18	V_{REF} + 0.18	3.6	0.35	V_{CCIO} - 0.43	15.2	-15.2
SSTL18 class I	-0.3	V_{REF} - 0.125	V_{REF} + 0.125	3.6	0.4	V_{CCIO} - 0.4	6.7	-6.7
HSTL15 class I	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	3.6	0.4	V_{CCIO} - 0.4	8	-8
HSTL15 class III	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	3.6	0.4	V_{CCIO} - 0.4	24	-8
HSTL18 class I	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	3.6	0.4	V_{CCIO} - 0.4	9.6	-9.6
HSTL18 class II	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	3.6	0.4	V_{CCIO} - 0.4	16	-16
HSTL18 class III	-0.3	V_{REF} - 0.1	V_{REF} + 0.1	3.6	0.4	V_{CCIO} - 0.4	24	-8

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

sysI/O Differential Electrical Characteristics

LVDS

Over Recommended Operating Conditions

Parameter Symbol	Parameter Description	Test Conditions	Min.	Typ.	Max.	Units
V_{INP}, V_{INM}	Input voltage		0	—	2.4	V
V_{THD}	Differential input threshold		+/-100	—	—	mV
V_{CM}	Input common mode voltage	100mV ð V_{THD}	V_{THD}/2	1.2	1.8	V
		200mV ð V_{THD}	V_{THD}/2	1.2	1.9	V
		350mV ð V_{THD}	V_{THD}/2	1.2	2.0	V
I_{IN}	Input current	Power on or power off	—	—	+/-10	μA
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
ΔV_{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	1.25	1.375	V
ΔV_{OS}	Change in V_{OS} between H and L		—	—	50	mV
I_{OSD}	Output short circuit current	V_{OD} = 0V Driver outputs shorted	—	—	6	mA

Differential HSTL and SSTL

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

LVDS25E

The top and bottom side of LatticeECP/EC 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
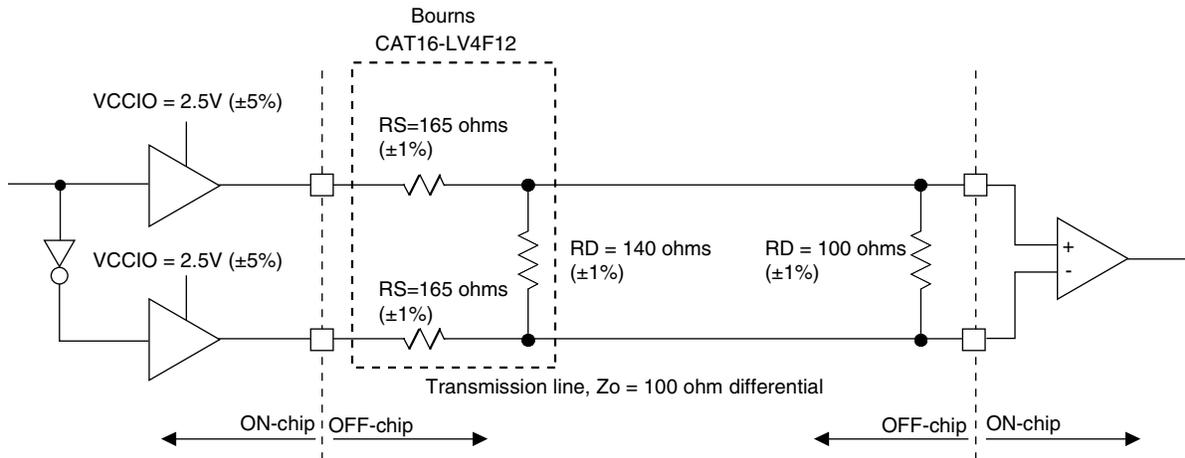
Table 3-1. LVDS25E DC Conditions

Parameter	Description	Typical	Units
V_{OH}	Output high voltage	1.42	V
V_{OL}	Output low voltage	1.08	V
V_{OD}	Output differential voltage	0.35	V
V_{CM}	Output common mode voltage	1.25	V
Z_{BACK}	Back impedance	100	¾

BLVDS

The LatticeECP/EC devices support 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. BLVDS Multi-point Output Example
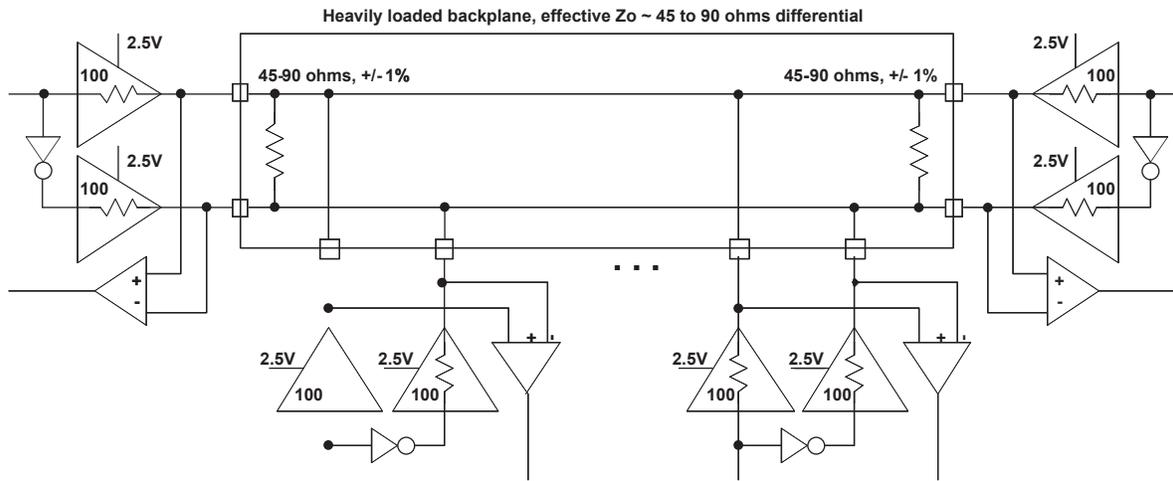
Table 3-2. BLVDS DC Conditions[1]

Over Recommended Operating Conditions

Parameter	Description	Typical		Units
		Zo = 45	Zo = 90	
Z_{OUT}	Output impedance	100	100	ohm
R_{TLEFT}	Left end termination	45	90	ohm
R_{TRIGHT}	Right end termination	45	90	ohm
V_{OH}	Output high voltage	1.375	1.48	V
V_{OL}	Output low voltage	1.125	1.02	V
V_{OD}	Output differential voltage	0.25	0.46	V
V_{CM}	Output common mode voltage	1.25	1.25	V
I_{DC}	DC output current	11.2	10.2	mA

1. For input buffer, see LVDS table.

LVPECL

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

Figure 3-3. Differential LVPECL
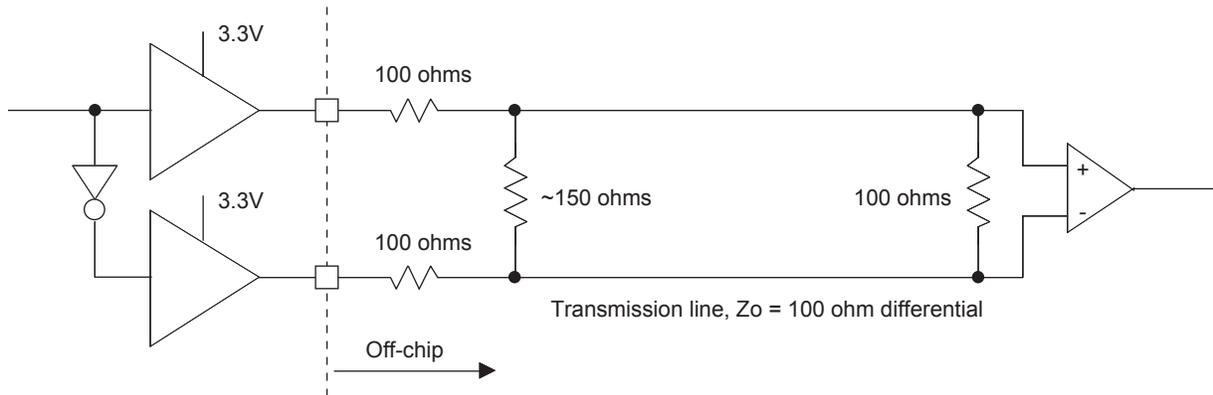
Table 3-3. LVPECL DC Conditions[1]

Over Recommended Operating Conditions

Parameter	Description	Typical	Units
Z_{OUT}	Output impedance	100	ohm
R_P	Driver parallel resistor	150	ohm
R_T	Receiver termination	100	ohm
V_{OH}	Output high voltage	2.03	V
V_{OL}	Output low voltage	1.27	V
V_{OD}	Output differential voltage	0.76	V
V_{CM}	Output common mode voltage	1.65	V
Z_{BACK}	Back impedance	85.7	ohm
I_{DC}	DC output current	12.7	mA

1. For input buffer, see LVDS table.

For further information about LVPECL, BLVDS and other differential interfaces please see the list of technical information at the end of this data sheet.

RSDS

The LatticeECP/EC devices support differential RSDS standard. This standard is emulated using complementary LVCMOS outputs in conjunction with a parallel resistor across the driver outputs. The RSDS input standard is supported by the LVDS differential input buffer. The scheme shown in Figure 3-4 is one possible solution for RSDS standard implementation. Use LVDS25E mode with suggested resistors for RSDS operation. Resistor values in Figure 3-4 are industry standard values for 1% resistors.

Figure 3-4. RSDS (Reduced Swing Differential Standard)
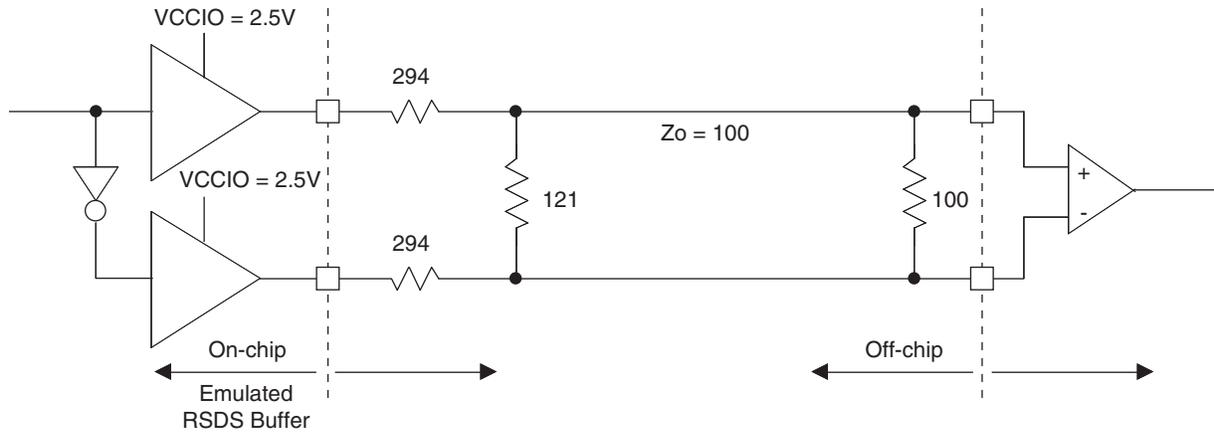
Table 3-4. RSDS DC Conditions

Parameter	Description	Typical	Units
Z_{OUT}	Output impedance	20	ohm
R_S	Driver series resistor	294	ohm
R_P	Driver parallel resistor	121	ohm
R_T	Receiver termination	100	ohm
V_{OH}	Output high voltage	1.35	V
V_{OL}	Output low voltage	1.15	V
V_{OD}	Output differential voltage	0.20	V
V_{CM}	Output common mode voltage	1.25	V
Z_{BACK}	Back impedance	101.5	ohm
I_{DC}	DC output current	3.66	mA

Typical Building Block Function Performance

Pin-to-Pin Performance (LVCMOS25 12mA Drive)

Function	-5 Timing	Units
Basic Functions		
16-bit decoder	5.5	ns
32-bit decoder	6.9	ns
64-bit decoder	7.1	ns
4:1 MUX	4.3	ns
8:1 MUX	4.7	ns
16:1 MUX	5.0	ns
32:1 MUX	5.5	ns

Register-to-Register Performance[1]

Function	-5 Timing	Units
Basic Functions		
16 bit decoder	410	MHz
32 bit decoder	283	MHz
64 bit decoder	272	MHz
4:1 MUX	613	MHz
8:1 MUX	565	MHz
16:1 MUX	526	MHz
32:1 MUX	442	MHz
8-bit adder	363	MHz
16-bit adder	353	MHz
64-bit adder	196	MHz
16-bit counter	414	MHz
32-bit counter	317	MHz
64-bit counter	216	MHz
64-bit accumulator	178	MHz
Embedded Memory Functions		
256x36 Single Port RAM	280	MHz
512x18 True-Dual Port RAM	280	MHz
Distributed Memory Functions		
16x2 Single Port RAM	460	MHz
64x2 Single Port RAM	375	MHz
128x4 Single Port RAM	294	MHz
32x2 Pseudo-Dual Port RAM	392	MHz
64x4 Pseudo-Dual Port RAM	332	MHz
DSP Function[2]		
9x9 Pipelined Multiply/Accumulate	242	MHz
18x18 Pipelined Multiply/Accumulate	238	MHz
36x36 Pipelined Multiply	235	MHz

1. These timing numbers were generated using the ispLEVER design tool. Exact performance may vary with design and tool version. The tool uses internal parameters that have been characterized but are not tested on every device.
2. Applies to LatticeECP devices only.

Timing v.G 0.30

Derating Timing Tables

Logic Timing provided in the following sections of the data sheet and the ispLEVER design tools are worst-case numbers in the operating range. Actual delays at nominal temperature and voltage for best-case process, can be much better than the values given in the tables. To calculate logic timing numbers at a particular temperature and voltage multiply the noted numbers with the derating factors provided below.

The junction temperature for the FPGA depends on the power dissipation by the device, the package thermal characteristics (Θ_{JA}), and the ambient temperature, as calculated with the following equation:

$$T_{JMAX} = T_{AMAX} + (Power * \Theta_{JA})$$

The user must determine this temperature and then use it to determine the derating factor based on the following derating tables: T_J °C.

Table 3-5. Delay Derating Table for Internal Blocks

T_J °C Commercial	T_J °C Industrial	Power Supply Voltage		
		1.14V	1.2V	1.26V
—	-40	0.82	0.77	0.71
—	-25	0.82	0.76	0.71
0	20	0.89	0.83	0.78
25	45	0.93	0.87	0.81
85	105	1.00	0.94	0.89

LatticeECP/EC External Switching Characteristics

Over Recommended Operating Conditions

Parameter	Description	Device	-5 Min.	-5 Max.	-4 Min.	-4 Max.	-3 Min.	-3 Max.	Units
General I/O Pin Parameters (Using Primary Clock without PLL)[1]									
t_{CO}[7]	Clock to Output - PIO Output Register	LFEC1	—	5.09	—	6.11	—	7.13	ns
		LFEC3	—	5.71	—	6.85	—	7.99	ns
		LFEC6	—	5.60	—	6.72	—	7.84	ns
		LFEC10	—	5.47	—	6.57	—	7.66	ns
		LFEC15	—	5.67	—	6.81	—	7.94	ns
		LFEC20	—	5.89	—	7.07	—	8.25	ns
		LFEC33	—	6.19	—	7.42	—	8.66	ns
t_{SU}[7]	Clock to Data Setup - PIO Input Register	LFEC1	-0.08	—	-0.10	—	-0.12	—	ns
		LFEC3	-0.70	—	-0.84	—	-0.98	—	ns
		LFEC6	-0.63	—	-0.76	—	-0.89	—	ns
		LFEC10	-0.43	—	-0.52	—	-0.61	—	ns
		LFEC15	-0.70	—	-0.84	—	-0.98	—	ns
		LFEC20	-0.88	—	-1.06	—	-1.24	—	ns
		LFEC33	-1.12	—	-1.34	—	-1.56	—	ns
t_H[7]	Clock to Data Hold - PIO Input Register	LFEC1	2.19	—	2.62	—	3.06	—	ns
		LFEC3	2.80	—	3.36	—	3.92	—	ns
		LFEC6	2.69	—	3.23	—	3.77	—	ns
		LFEC10	2.56	—	3.08	—	3.59	—	ns
		LFEC15	2.76	—	3.32	—	3.87	—	ns
		LFEC20	2.99	—	3.58	—	4.18	—	ns
		LFEC33	3.28	—	3.93	—	4.59	—	ns
t_{SU_DEL}[7]	Clock to Data Setup - PIO Input Register with Data Input Delay	LFEC1	3.36	—	4.03	—	4.70	—	ns
		LFEC3	2.74	—	3.29	—	3.84	—	ns
		LFEC6	2.81	—	3.37	—	3.93	—	ns
		LFEC10	3.01	—	3.61	—	4.21	—	ns
		LFEC15	2.74	—	3.29	—	3.83	—	ns
		LFEC20	2.56	—	3.07	—	3.58	—	ns
		LFEC33	2.32	—	2.79	—	3.25	—	ns
t_{H_DEL}[7]	Clock to Data Hold - PIO Input Register with Input Data Delay	LFEC1	-1.31	—	-1.57	—	-1.83	—	ns
		LFEC3	-0.70	—	-0.83	—	-0.97	—	ns
		LFEC6	-0.80	—	-0.96	—	-1.12	—	ns
		LFEC10	-0.93	—	-1.12	—	-1.30	—	ns
		LFEC15	-0.73	—	-0.88	—	-1.02	—	ns
		LFEC20	-0.51	—	-0.61	—	-0.71	—	ns
		LFEC33	-0.22	—	-0.26	—	-0.30	—	ns
f_{MAX_IO}[2]	Clock Frequency of I/O and PFU Register	All	—	420	—	378	—	340	Mhz
DDR I/O Pin Parameters[3, 4, 5]									
t_{DVADQ}	Data Valid After DQS (DDR Read)	All	—	0.19	—	0.19	—	0.19	UI
t_{DVEDQ}	Data Hold After DQS (DDR Read)	All	0.67	—	0.67	—	0.67	—	UI

LatticeECP/EC External Switching Characteristics (Continued)

Over Recommended Operating Conditions

Parameter	Description	Device	-5 Min.	-5 Max.	-4 Min.	-4 Max.	-3 Min.	-3 Max.	Units
t$_{DQVBS}$	Data Valid Before DQS	All	0.20	—	0.20	—	0.20	—	UI
t$_{DQVAS}$	Data Valid After DQS	All	0.20	—	0.20	—	0.20	—	UI
f$_{MAX\_DDR}$	DDR Clock Frequency	All	95	200	95	166	95	133	MHz
Primary and Secondary Clock[6]									
f$_{MAX\_PRI}$[2]	Frequency for Primary Clock Tree	All	—	420	—	378	—	340	MHz
t$_{W\_PRI}$	Clock Pulse Width for Primary Clock	All	1.19	—	1.19	—	1.19	—	ns
t$_{SKEW\_PRI}$	Primary Clock Skew within an I/O Bank	All	—	250	—	300	—	350	ps

1. General timing numbers based on LVCMOS2.5V, 12 mA. Loading of 0 pF.
2. Using LVDS I/O standard.
3. DDR timing numbers based on SSTL I/O.
4. DDR specifications are characterized but not tested.
5. UI is average bit period.
6. Based on a single primary clock.
7. These timing numbers were generated using ispLEVER design tool. Exact performance may vary with design and tool version. The tool uses internal parameters that have been characterized but are not tested on every device.

Timing v.G 0.30

Figure 3-5. DDR Timings
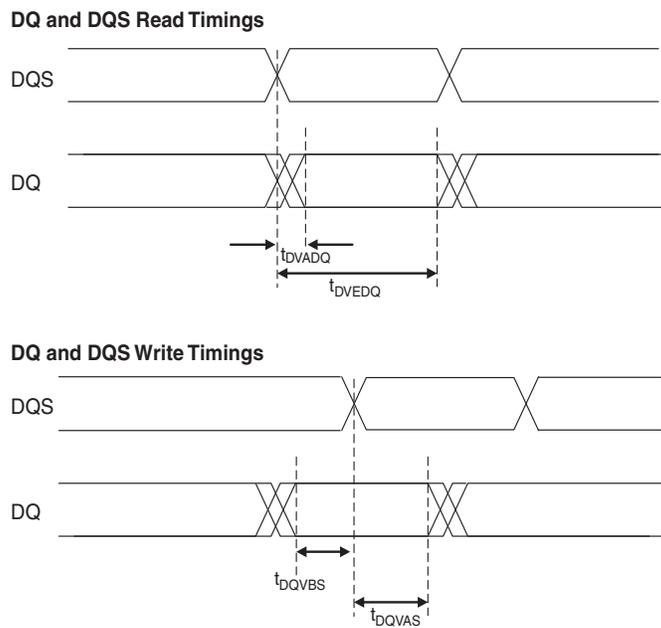
DQ and DQS Read Timings

DQ and DQS Write Timings

LatticeECP/EC Internal Switching Characteristics

Over Recommended Operating Conditions

Parameter	Description	-5 Min.	-5 Max.	-4 Min.	-4 Max.	-3 Min.	-3 Max.	Units
PFU/PFF Logic Mode Timing								
t_{LUT4_PFU}	LUT4 Delay (A to D Inputs to F Output)	—	0.25	—	0.31	—	0.36	ns
t_{LUT6_PFU}	LUT6 Delay (A to D Inputs to OFX Output)	—	0.40	—	0.48	—	0.56	ns
t_{LSR_PFU}	Set/Reset to Output of PFU	—	0.81	—	0.98	—	1.14	ns
t_{SUM_PFU}	Clock to Mux (M0,M1) Input Setup Time	0.12	—	0.14	—	0.16	—	ns
t_{HM_PFU}	Clock to Mux (M0,M1) Input Hold Time	-0.05	—	-0.06	—	-0.06	—	ns
t_{SUD_PFU}	Clock to D Input Setup Time	0.12	—	0.14	—	0.16	—	ns
t_{HD_PFU}	Clock to D Input Hold time	-0.03	—	-0.03	—	-0.04	—	ns
t_{CK2Q_PFU}	Clock to Q Delay, D-type Register Configuration	—	0.36	—	0.44	—	0.51	ns
t_{LE2Q_PFU}	Clock to Q Delay Latch Configuration	—	0.48	—	0.58	—	0.68	ns
t_{LD2Q_PFU}	D to Q Throughput Delay when Latch is Enabled	—	0.50	—	0.60	—	0.69	ns
PFU Dual Port Memory Mode Timing								
t_{CORAM_PFU}	Clock to Output	—	0.36	—	0.44	—	0.51	ns
t_{SUDATA_PFU}	Data Setup Time	-0.20	—	-0.24	—	-0.28	—	ns
t_{HDATA_PFU}	Data Hold Time	0.26	—	0.31	—	0.36	—	ns
t_{SUADDR_PFU}	Address Setup Time	-0.51	—	-0.62	—	-0.72	—	ns
t_{HADDR_PFU}	Address Hold Time	0.64	—	0.77	—	0.90	—	ns
t_{SUWREN_PFU}	Write/Read Enable Setup Time	-0.24	—	-0.29	—	-0.34	—	ns
t_{HWREN_PFU}	Write/Read Enable Hold Time	0.30	—	0.36	—	0.42	—	ns
PIC Timing								
PIO Input/Output Buffer Timing								
t_{IN_PIO}	Input Buffer Delay	—	0.56	—	0.67	—	0.78	ns
t_{OUT_PIO}	Output Buffer Delay	—	1.92	—	2.31	—	2.69	ns
IOLOGIC Input/Output Timing								
t_{SUI_PIO}	Input Register Setup Time (Data Before Clock)	0.90	—	1.08	—	1.26	—	ns
t_{HI_PIO}	Input Register Hold Time (Data after Clock)	0.62	—	0.74	—	0.87	—	ns
t_{COO_PIO}	Output Register Clock to Output Delay	—	0.33	—	0.40	—	0.46	ns
t_{SUCE_PIO}	Input Register Clock Enable Setup Time	-0.10	—	-0.12	—	-0.14	—	ns
t_{HCE_PIO}	Input Register Clock Enable Hold Time	0.12	—	0.14	—	0.17	—	ns
t_{SULSR_PIO}	Set/Reset Setup Time	0.18	—	0.21	—	0.25	—	ns
t_{HLSR_PIO}	Set/Reset Hold Time	-0.15	—	-0.18	—	-0.21	—	ns
EBR Timing								
t_{CO_EBR}	Clock to Output from Address or Data	—	3.64	—	4.37	—	5.10	ns
t_{COO_EBR}	Clock to Output from EBR output Register	—	0.74	—	0.88	—	1.03	ns
t_{SUDATA_EBR}	Setup Data to EBR Memory	-0.29	—	-0.35	—	-0.41	—	ns
t_{HDATA_EBR}	Hold Data to EBR Memory	0.37	—	0.44	—	0.52	—	ns
t_{SUADDR_EBR}	Setup Address to EBR Memory	-0.29	—	-0.35	—	-0.41	—	ns
t_{HADDR_EBR}	Hold Address to EBR Memory	0.37	—	0.45	—	0.52	—	ns
t_{SUWREN_EBR}	Setup Write/Read Enable to EBR Memory	-0.18	—	-0.22	—	-0.26	—	ns
t_{HWREN_EBR}	Hold Write/Read Enable to EBR Memory	0.23	—	0.28	—	0.33	—	ns

LatticeECP/EC Internal Switching Characteristics (Continued)

Over Recommended Operating Conditions

Parameter	Description	-5 Min.	-5 Max.	-4 Min.	-4 Max.	-3 Min.	-3 Max.	Units
t_{SUCE_EBR}	Clock Enable Setup Time to EBR Output Register	0.18	—	0.21	—	0.25	—	ns
t_{HCE_EBR}	Clock Enable Hold Time to EBR Output Register	-0.14	—	-0.17	—	-0.20	—	ns
t_{RSTO_EBR}	Reset To Output Delay Time from EBR Output Register	—	1.47	—	1.76	—	2.05	ns
PLL Parameters								
t_{RSTREC}	Reset Recovery to Rising Clock	1.00	—	1.00	—	1.00	—	ns
t_{RSTSU}	Reset Signal Setup Time	1.00	—	1.00	—	1.00	—	ns
DSP Block Timing[2, 3]								
t_{SUI_DSP}	Input Register Setup Time	-0.38	—	-0.30	—	-0.23	—	ns
t_{HI_DSP}	Input Register Hold Time	0.71	—	0.86	—	1.00	—	ns
t_{SUP_DSP}	Pipeline Register Setup Time	3.31	—	3.98	—	4.64	—	ns
t_{HP_DSP}	Pipeline Register Hold Time	0.71	—	0.86	—	1.00	—	ns
t_{SUO_DSP}[4]	Output Register Setup Time	5.54	—	6.64	—	7.75	—	ns
t_{HO_DSP}[4]	Output Register Hold Time	0.71	—	0.86	—	1.00	—	ns
t_{COI_DSP}[4]	Input Register Clock to Output Time	—	7.50	—	9.00	—	10.50	ns
t_{COP_DSP}[4]	Pipeline Register Clock to Output Time	—	4.66	—	5.60	—	6.53	ns
t_{COO_DSP}	Output Register Clock to Output Time	—	1.47	—	1.77	—	2.06	ns
$t_{SUADSUB}$	AdSub Input Register Setup Time	-0.38	—	-0.30	—	-0.23	—	ns
t_{HADSUB}	AdSub Input Register Hold Time	0.71	—	0.86	—	1.00	—	ns

1. Internal parameters are characterized but not tested on every device.
2. These parameters apply to LatticeECP devices only.
3. DSP Block is configured in Multiply Add/Sub 18 x 18 Mode.
4. These parameters include the Adder Subtractor block in the path.
Timing v.G 0.30

Timing Diagrams

PFU Timing Diagrams

Figure 3-6. Slice Single/Dual Port Write Cycle Timing
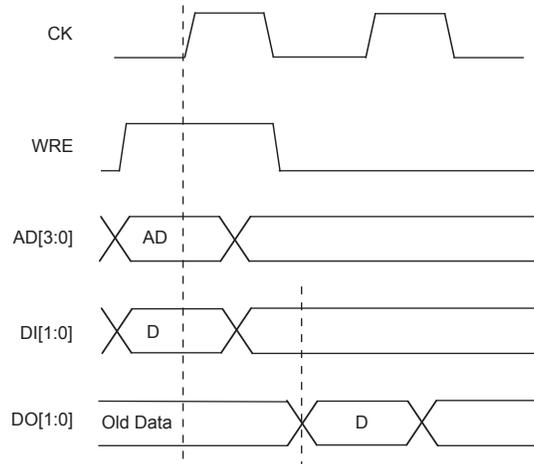
Figure 3-7. Slice Single /Dual Port Read Cycle Timing
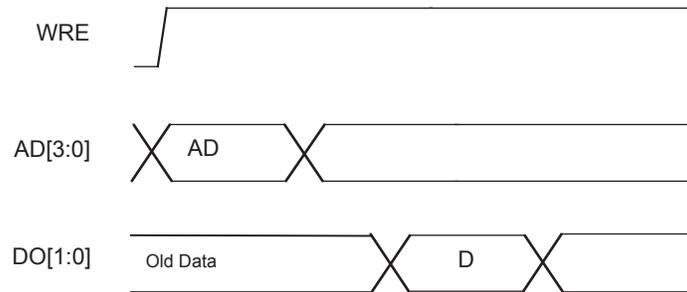
EBR Memory Timing Diagrams

Figure 3-8. Read/Write Mode (Normal)
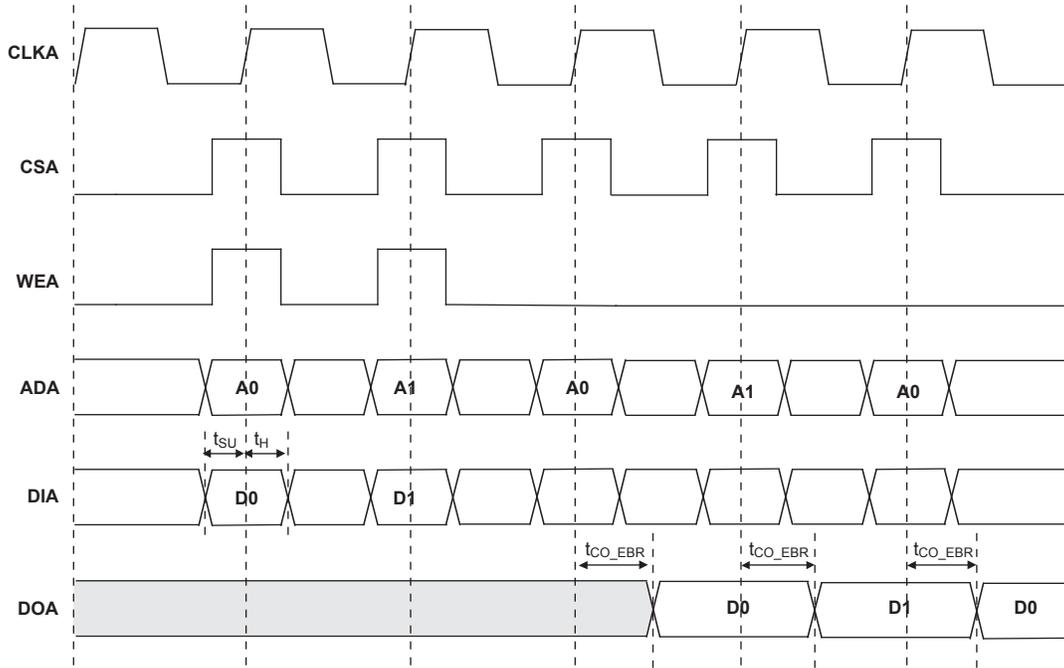
Note: Input data and address are registered at the positive edge of the clock and output data appears after the positive edge of the clock.

Figure 3-9. Read/Write Mode with Input and Output Registers
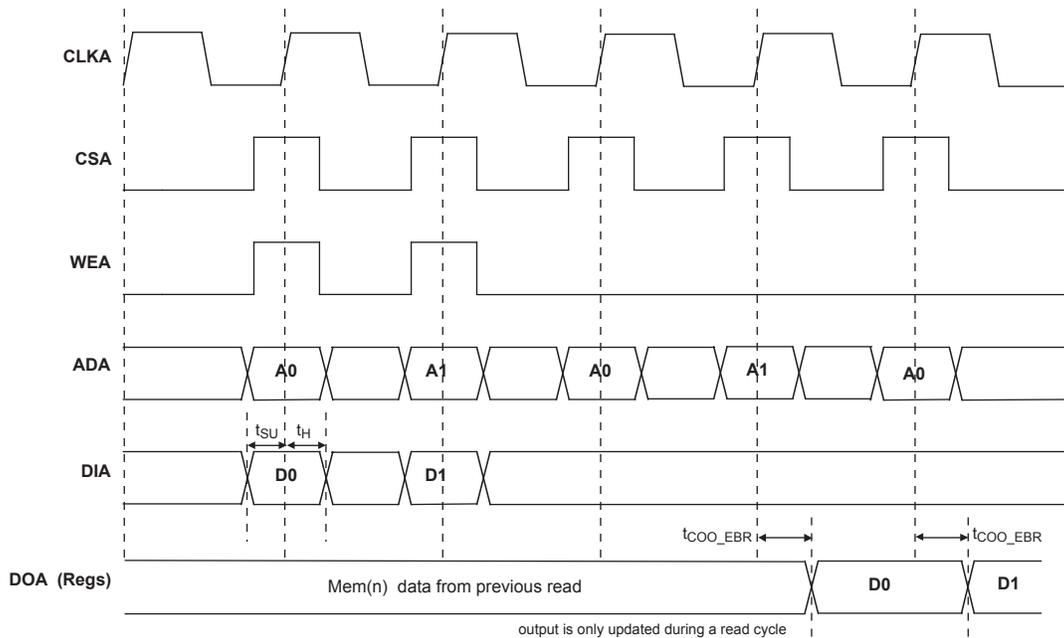
Figure 3-10. Read Before Write (SP Read/Write on Port A, Input Registers Only)
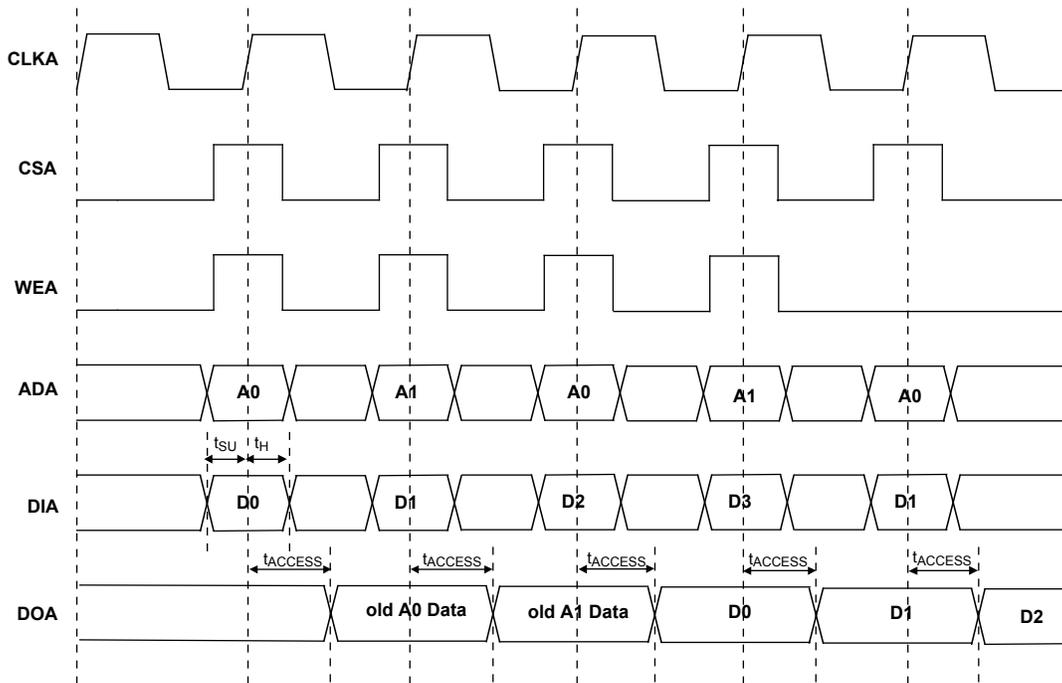
Note: Input data and address are registered at the positive edge of the clock and output data appears after the positive edge of the clock.

Figure 3-11. Write Through (SP Read/Write On Port A, Input Registers Only)
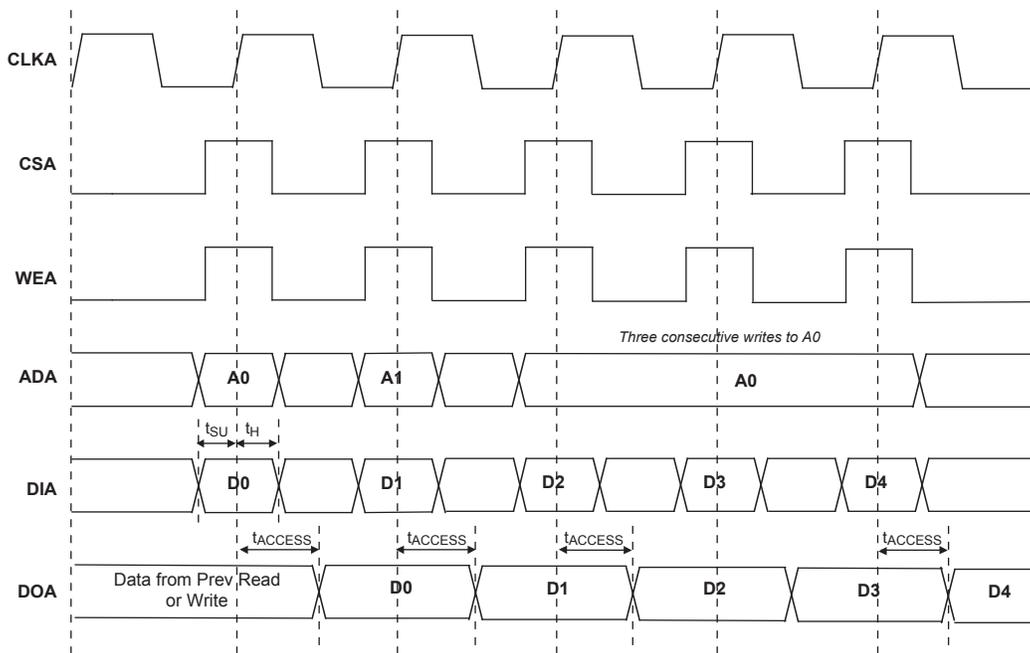
Note: Input data and address are registered at the positive edge of the clock and output data appears after the positive edge of the clock.

LatticeECP/EC Family Timing Adders[1, 2, 3]

Over Recommended Operating Conditions

Buffer Type	Description	-5	-4	-3	Units
Input Adjusters					
LVDS25	LVDS	0.41	0.50	0.58	ns
BLVDS25	BLVDS	0.41	0.50	0.58	ns
LVPECL33	LVPECL	0.50	0.60	0.70	ns
HSTL18_I	HSTL_18 class I	0.41	0.49	0.57	ns
HSTL18_II	HSTL_18 class II	0.41	0.49	0.57	ns
HSTL18_III	HSTL_18 class III	0.41	0.49	0.57	ns
HSTL18D_I	Differential HSTL 18 class I	0.37	0.44	0.52	ns
HSTL18D_II	Differential HSTL 18 class II	0.37	0.44	0.52	ns
HSTL18D_III	Differential HSTL 18 class III	0.37	0.44	0.52	ns
HSTL15_I	HSTL_15 class I	0.40	0.48	0.56	ns
HSTL15_III	HSTL_15 class III	0.40	0.48	0.56	ns
HSTL15D_I	Differential HSTL 15 class I	0.37	0.44	0.51	ns
HSTL15D_III	Differential HSTL 15 class III	0.37	0.44	0.51	ns
SSTL33_I	SSTL_3 class I	0.46	0.55	0.64	ns
SSTL33_II	SSTL_3 class II	0.46	0.55	0.64	ns
SSTL33D_I	Differential SSTL_3 class I	0.39	0.47	0.55	ns
SSTL33D_II	Differential SSTL_3 class II	0.39	0.47	0.55	ns
SSTL25_I	SSTL_2 class I	0.43	0.51	0.60	ns
SSTL25_II	SSTL_2 class II	0.43	0.51	0.60	ns
SSTL25D_I	Differential SSTL_2 class I	0.38	0.45	0.53	ns
SSTL25D_II	Differential SSTL_2 class II	0.38	0.45	0.53	ns
SSTL18_I	SSTL_18 class I	0.40	0.48	0.56	ns
SSTL18D_I	Differential SSTL_18 class I	0.37	0.44	0.51	ns
LVTTL33	LVTTL	0.07	0.09	0.10	ns
LVCMOS33	LVCMOS 3.3	0.07	0.09	0.10	ns
LVCMOS25	LVCMOS 2.5	0.00	0.00	0.00	ns
LVCMOS18	LVCMOS 1.8	0.07	0.09	0.10	ns
LVCMOS15	LVCMOS 1.5	0.24	0.29	0.33	ns
LVCMOS12	LVCMOS 1.2	1.27	1.52	1.77	ns
PCI33	PCI	0.07	0.09	0.10	ns
Output Adjusters					
LVDS25E	LVDS 2.5 E	0.12	0.14	0.17	ns
LVDS25	LVDS 2.5	-0.44	-0.53	-0.62	ns
BLVDS25	BLVDS 2.5	0.33	0.40	0.46	ns
LVPECL33	LVPECL 3.3	0.20	0.24	0.28	ns
HSTL18_I	HSTL_18 class I	-0.10	-0.12	-0.14	ns
HSTL18_II	HSTL_18 class II	0.06	0.07	0.08	ns
HSTL18_III	HSTL_18 class III	0.15	0.19	0.22	ns
HSTL18D_I	Differential HSTL 18 class I	-0.10	-0.12	-0.14	ns
HSTL18D_II	Differential HSTL 18 class II	0.06	0.07	0.08	ns
HSTL18D_III	Differential HSTL 18 class III	0.15	0.19	0.22	ns
HSTL15_I	HSTL_15 class I	0.08	0.10	0.11	ns

LatticeECP/EC Family Timing Adders[1, 2, 3] (Continued)

Over Recommended Operating Conditions

Buffer Type	Description	-5	-4	-3	Units
HSTL15_II	HSTL_15 class II	0.10	0.12	0.14	ns
HSTL15_III	HSTL_15 class III	0.10	0.12	0.14	ns
HSTL15D_I	Differential HSTL 15 class I	0.08	0.10	0.11	ns
HSTL15D_III	Differential HSTL 15 class III	0.10	0.12	0.14	ns
SSTL33_I	SSTL_3 class I	-0.05	-0.06	-0.07	ns
SSTL33_II	SSTL_3 class II	0.40	0.48	0.56	ns
SSTL33D_I	Differential SSTL_3 class I	-0.05	-0.06	-0.07	ns
SSTL33D_II	Differential SSTL_3 class II	0.40	0.48	0.56	ns
SSTL25_I	SSTL_2 class I	0.05	0.07	0.08	ns
SSTL25_II	SSTL_2 class II	0.25	0.30	0.35	ns
SSTL25D_I	Differential SSTL_2 class I	0.05	0.07	0.08	ns
SSTL25D_II	Differential SSTL_2 class II	0.25	0.30	0.35	ns
SSTL18_I	SSTL_1.8 class I	0.01	0.01	0.01	ns
SSTL18D_I	Differential SSTL_1.8 class I	0.01	0.01	0.01	ns
LVTTL33_4mA	LVTTL 4mA drive	0.09	0.11	0.13	ns
LVTTL33_8mA	LVTTL 8mA drive	0.07	0.08	0.09	ns
LVTTL33_12mA	LVTTL 12mA drive	-0.03	-0.04	-0.05	ns
LVTTL33_16mA	LVTTL 16mA drive	0.36	0.43	0.51	ns
LVTTL33_20mA	LVTTL 20mA drive	0.28	0.33	0.39	ns
LVCMOS33_4mA	LVCMOS 3.3 4mA drive	0.09	0.11	0.13	ns
LVCMOS33_8mA	LVCMOS 3.3 8mA drive	0.07	0.08	0.09	ns
LVCMOS33_12mA	LVCMOS 3.3 12mA drive	-0.03	-0.04	-0.05	ns
LVCMOS33_16mA	LVCMOS 3.3 16mA drive	0.36	0.43	0.51	ns
LVCMOS33_20mA	LVCMOS 3.3 20mA drive	0.28	0.33	0.39	ns
LVCMOS25_4mA	LVCMOS 2.5 4mA drive	0.18	0.21	0.25	ns
LVCMOS25_8mA	LVCMOS 2.5 8mA drive	0.10	0.12	0.14	ns
LVCMOS25_12mA	LVCMOS 2.5 12mA drive	0.00	0.00	0.00	ns
LVCMOS25_16mA	LVCMOS 2.5 16mA drive	0.22	0.26	0.31	ns
LVCMOS25_20mA	LVCMOS 2.5 20mA drive	0.14	0.16	0.19	ns
LVCMOS18_4mA	LVCMOS 1.8 4mA drive	0.15	0.18	0.21	ns
LVCMOS18_8mA	LVCMOS 1.8 8mA drive	0.06	0.08	0.09	ns
LVCMOS18_12mA	LVCMOS 1.8 12mA drive	0.01	0.01	0.01	ns
LVCMOS18_16mA	LVCMOS 1.8 16mA drive	0.16	0.19	0.22	ns
LVCMOS15_4mA	LVCMOS 1.5 4mA drive	0.26	0.31	0.36	ns
LVCMOS15_8mA	LVCMOS 1.5 8mA drive	0.04	0.04	0.05	ns
LVCMOS12_2mA	LVCMOS 1.2 2mA drive	0.36	0.43	0.50	ns
LVCMOS12_6mA	LVCMOS 1.2 6mA drive	0.08	0.10	0.11	ns
LVCMOS12_4mA	LVCMOS 1.2 4mA drive	0.36	0.43	0.50	ns
PCI33	PCI33	1.05	1.26	1.46	ns

1. Timing adders are characterized but not tested on every device.
2. LVCMOS timing measured with the load specified in Switching Test Conditions table of this document.
3. All other standards according to the appropriate specification.
Timing v.G 0.30

sysCLOCK PLL Timing

Over Recommended Operating Conditions

Parameter	Description	Conditions	Min.	Typ.	Max.	Units
f_{IN}	Input Clock Frequency (CLKI, CLKFB)		25	—	420	MHz
f_{OUT}	Output Clock Frequency (CLKOP, CLKOS)		25	—	420	MHz
f_{OUT2}	K-Divider Output Frequency (CLKOK)		0.195	—	210	MHz
f_{VCO}	PLL VCO Frequency		420	—	840	MHz
f_{PFD}	Phase Detector Input Frequency		25	—	—	MHz
AC Characteristics						
t_{DT}	Output Clock Duty Cycle	Default Duty Cycle Elected[3]	45	50	55	%
t_{PH}[4]	Output Phase Accuracy		—	—	0.05	UI
t_{OPJIT}[1]	Output Clock Period Jitter	f_{OUT} >= 100MHz	—	—	+/- 125	ps
		f_{OUT} < 100MHz	—	—	0.02	UIPP
t_{SK}	Input Clock to Output Clock Skew	Divider ratio = integer	—	—	+/- 200	ps
t_W	Output Clock Pulse Width	At 90% or 10%[3]	1	—	—	ns
t_{LOCK}[2]	PLL Lock-in Time		—	—	150	µs
t_{PA}	Programmable Delay Unit		100	250	450	ps
t_{IPJIT}	Input Clock Period Jitter		—	—	+/- 200	ps
t_{FBKDLY}	External Feedback Delay		—	—	10	ns
t_{HI}	Input Clock High Time	90% to 90%	0.5	—	—	ns
t_{LO}	Input Clock Low Time	10% to 10%	0.5	—	—	ns
t_{RST}	RST Pulse Width		10	—	—	ns

1. Jitter sample is taken over 10,000 samples of the primary PLL output with clean reference clock.
2. Output clock is valid after t_{LOCK} for PLL reset and dynamic delay adjustment.
3. Using LVDS output buffers.
4. Relative to CLKOP.
Timing v.G 0.30

LatticeECP/EC sysCONFIG Port Timing Specifications

Over Recommended Operating Conditions

Parameter	Description	Min.	Typ.	Max.	Units
sysCONFIG Byte Data Flow					
t$_{SUCBDI}$	Byte D[0:7] Setup Time to CCLK	7		—	ns
t$_{HCBDI}$	Byte D[0:7] Hold Time to CCLK	1		—	ns
t$_{CODO}$	Clock to Dout in Flowthrough Mode	—		12	ns
t$_{SUCS}$	CS[0:1] Setup Time to CCLK	7		—	ns
t$_{HCS}$	CS[0:1] Hold Time to CCLK	1		—	ns
t$_{SUWD}$	Write Signal Setup Time to CCLK	7		—	ns
t$_{HWD}$	Write Signal Hold Time to CCLK	1		—	ns
t$_{DCB}$	CCLK to BUSY Delay Time	—		12	ns
t$_{CORD}$	Clock to Out for Read Data	—		12	ns
sysCONFIG Byte Slave Clocking					
t$_{BSCH}$	Byte Slave Clock Minimum High Pulse	6		—	ns
t$_{BSCL}$	Byte Slave Clock Minimum Low Pulse	9		—	ns
t$_{BSCYC}$	Byte Slave Clock Cycle Time	15		—	ns
t$_{SUSCDI}$	Din Setup time to CCLK Slave Mode	7		—	ns
t$_{HSCDI}$	Din Hold Time to CCLK Slave Mode	1		—	ns
t$_{CODO}$	Clock to Dout in Flowthrough Mode	—		12	ns
sysCONFIG Serial (Bit) Data Flow					
t$_{SUMCDI}$	Din Setup time to CCLK Master Mode	7		—	ns
t$_{HMCDI}$	Din Hold Time to CCLK Master Mode	1		—	ns
sysCONFIG Serial Slave Clocking					
t$_{SSCH}$	Serial Slave Clock Minimum High Pulse	6		—	ns
t$_{SSCL}$	Serial Slave Clock Minimum Low Pulse	6		—	ns
sysCONFIG POR, Initialization and Wake Up					
t$_{ICFG}$	Minimum Vcc to INIT High	—		50	ms
t$_{VMC}$	Time from tICFG to Valid Master Clock	—		2	us
t$_{PRGMRJ}$	Program Pin Pulse Rejection	—		8	ns
t$_{PRGM}$	PROGRAMN Low Time to Start Configuration	25		—	ns
t$_{DINIT}$	INIT Low Time	—		1	ms
t$_{DPPINIT}$	Delay Time from PROGRAMN Low to INIT Low	—		37	ns
t$_{DINITD}$	Delay Time from PROGRAMN Low to DONE Low	—		37	ns
t$_{IODISS}$	User I/O Disable from PROGRAMN Low	—		35	ns
t$_{IOENSS}$	User I/O Enabled Time from CCLK Edge During Wake Up Sequence	—		25	ns
t$_{MWC}$	Additional Wake Master Clock Signals after Done Pin High	120		—	cycles
t$_{SUCFG}$	CFG to INITN Setup Time	100		—	ns
t$_{HCFG}$	CFG to INITN Hold Time	100		—	ns
sysCONFIG SPI Port					
t$_{CFGX}$	Init High to CCLK Low	—		80	ns
t$_{CSSPI}$	Init High to CSSPIN Low	—		2	us
t$_{CSCCLK}$	CCLK Low Before CSSPIN Low	0		-	ns
t$_{SOCDO}$	CCLK Low to Output Valid	—		15	ns

LatticeECP/EC sysCONFIG Port Timing Specifications (Continued)

Over Recommended Operating Conditions

Parameter	Description	Min.	Typ.	Max.	Units
t$_{SOE}$	CSSPIN Active Setup Time	300		—	ns
t$_{CSPID}$	CSSPIN Low to First Clock Edge Setup Time	300+3cyc		600+6cyc	ns
f$_{MAXSPI}$	Max Frequency for SPI	—		25	MHz
t$_{SUSPI}$	SOSPI Data Setup Time Before CCLK	7		—	ns
t$_{HSPI}$	SOSPI Data Hold Time After CCLK	1		—	ns

Timing v.G 0.30

Master Clock

Clock Mode	Min.	Typ.	Max.	Units
2.5MHz	1.75	2.5	3.25	MHz
5 MHz	3.78	5.4	7.02	MHz
10 MHz	7	10	13	MHz
15 MHz	10.5	15	19.5	MHz
20 MHz	14	20	26	MHz
25 MHz	18.2	26	33.8	MHz
30 MHz	21	30	39	MHz
35 MHz	23.8	34	44.2	MHz
40 MHz	28.7	41	53.3	MHz
45 MHz	31.5	45	58.5	MHz
50 MHz	35.7	51	66.3	MHz
55 MHz	38.5	55	71.5	MHz
60 MHz	42	60	78	MHz
Duty Cycle	40	—	60	%

Timing v.G 0.30

Figure 3-12. sysCONFIG Parallel Port Read Cycle
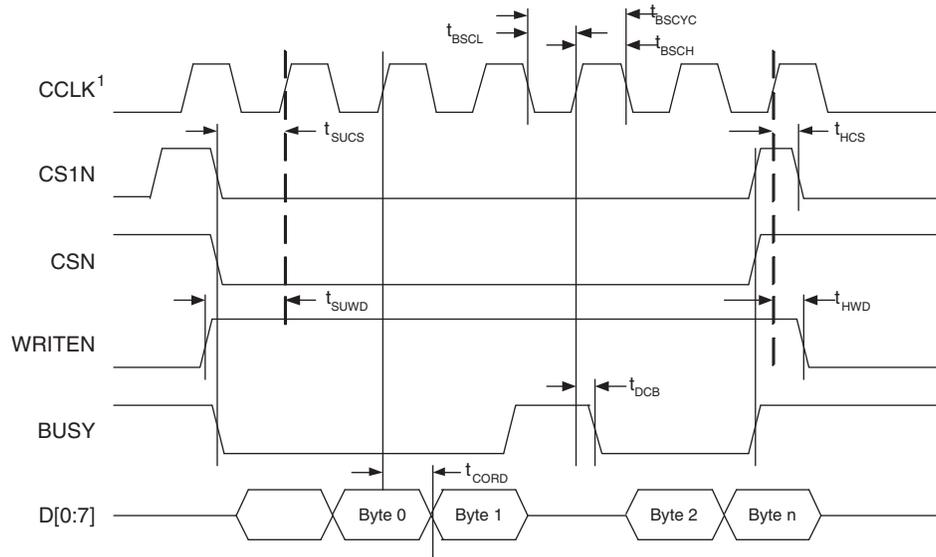
1. In Master Parallel Mode the FPGA provides CCLK. In Slave Parallel Mode the external device provides CCLK.

Figure 3-13. sysCONFIG Parallel Port Write Cycle
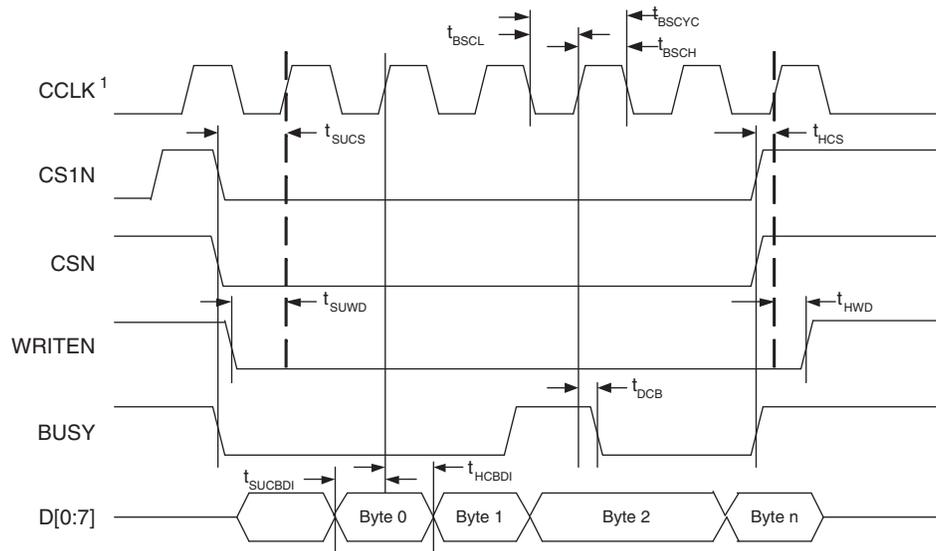
1. In Master Parallel Mode the FPGA provides CCLK. In Slave Parallel Mode the external device provides CCLK.

Figure 3-14. sysCONFIG Master Serial Port Timing
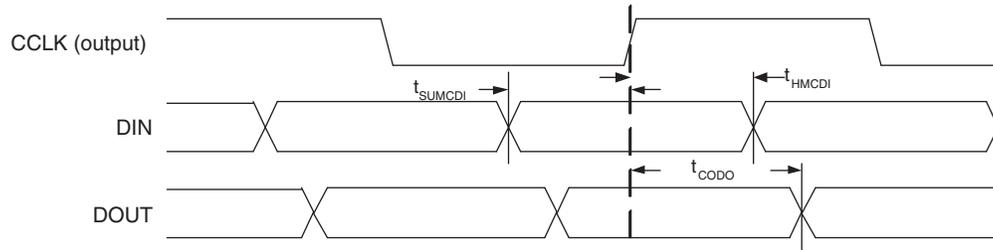
Figure 3-15. sysCONFIG Slave Serial Port Timing
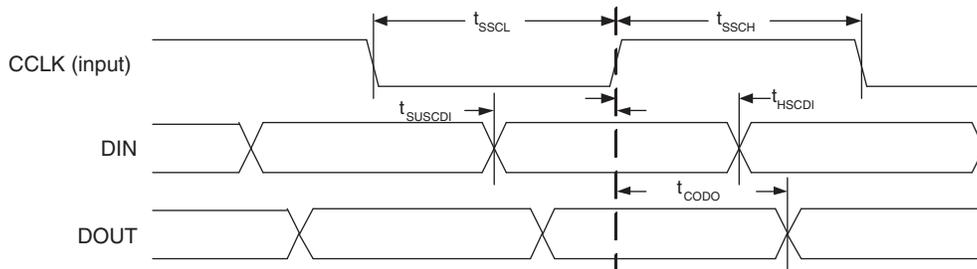
Figure 3-16. Power-On-Reset (POR) Timing
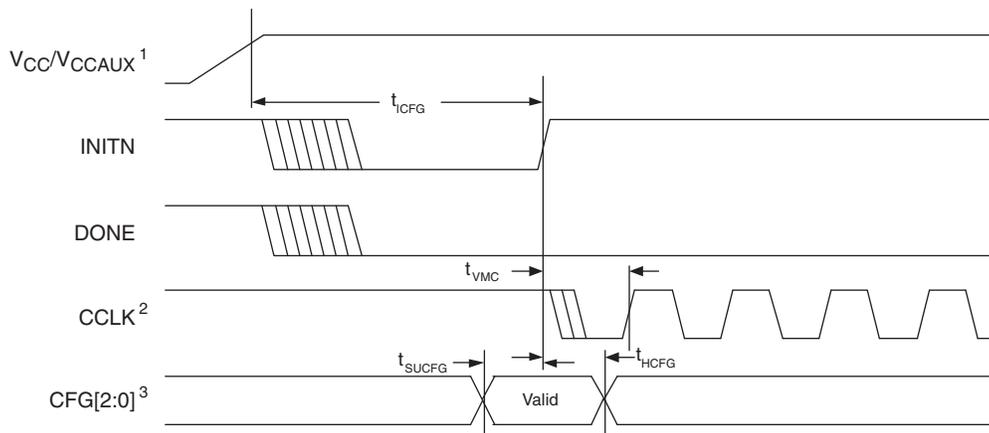
1. Time taken from V_{CC} or V_{CCAUX}, whichever is the last to reach its V_{MIN}.
2. Device is in a Master Mode.
3. The CFG pins are normally static (hard wired).

Figure 3-17. Configuration from PROGRAMN Timing
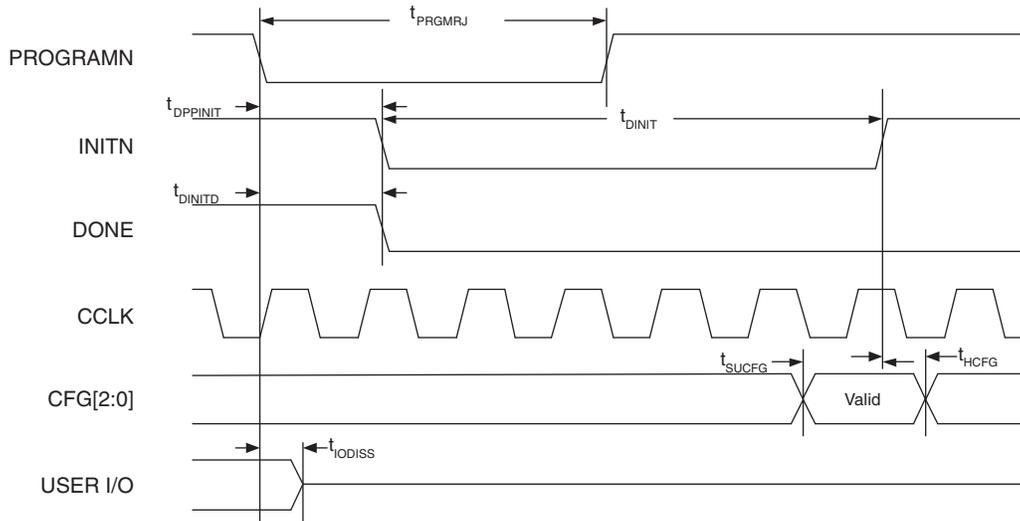
1. The CFG pins are normally static (hard wired)

Figure 3-18. Wake-Up Timing
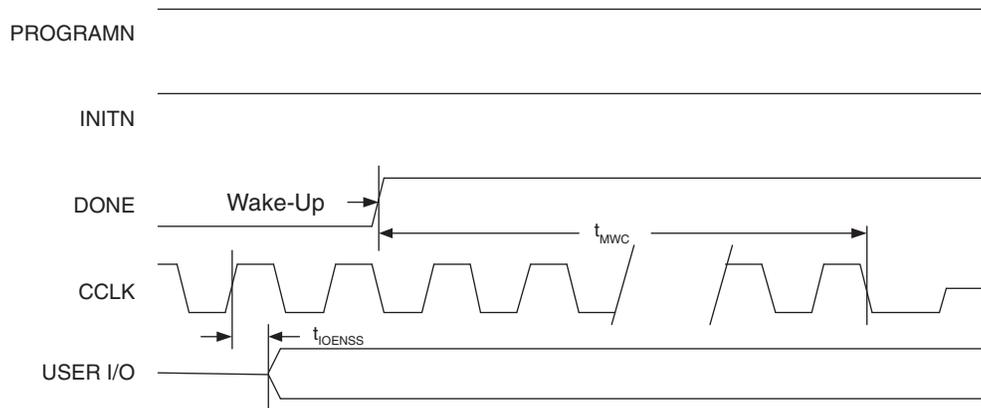
Figure 3-19. sysCONFIG SPI Port Sequence
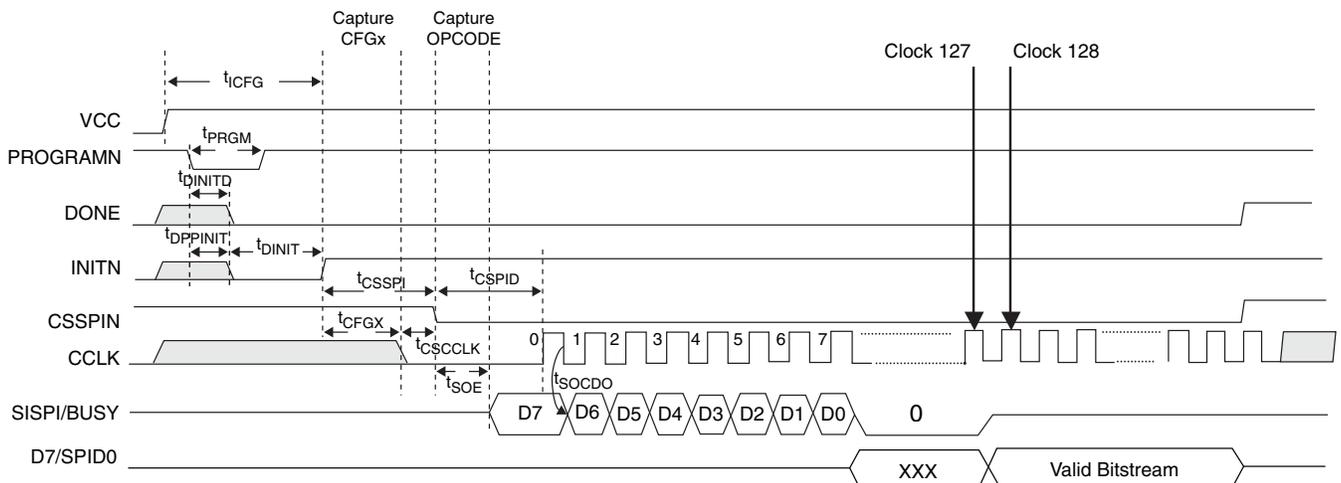
JTAG Port Timing Specifications

Over Recommended Operating Conditions

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

Timing v.G 0.30

Figure 3-20. JTAG Port Timing Waveforms
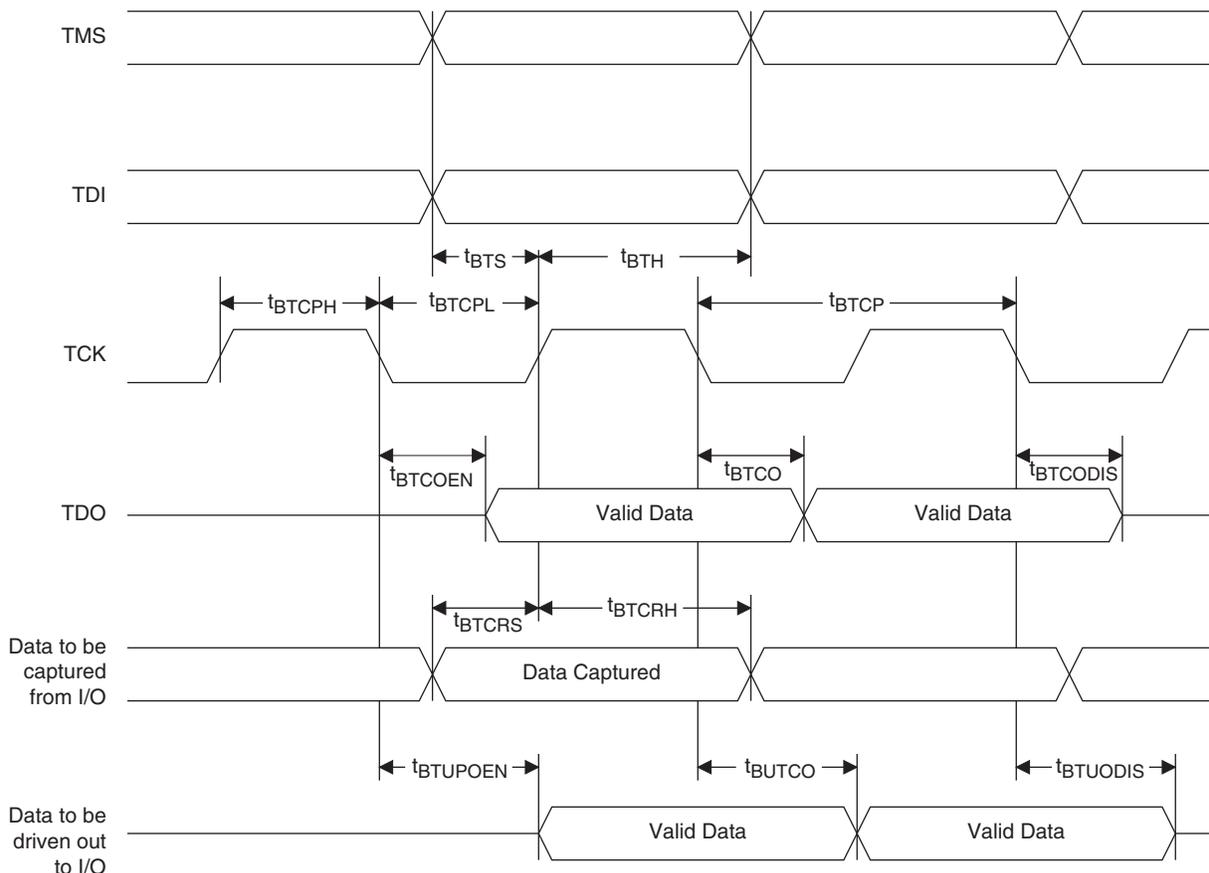
Switching Test Conditions

Figure 3-21 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 3-6.

Figure 3-21. Output Test Load, LVTTL and LVCMOS Standards
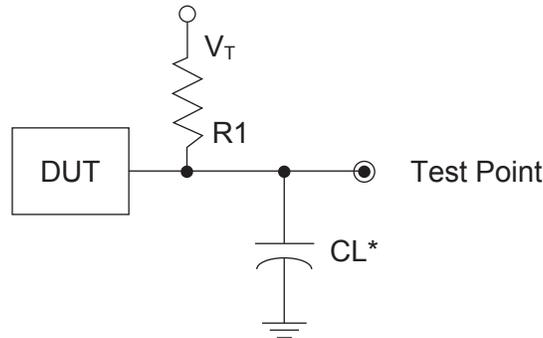
*CL Includes Test Fixture and Probe Capacitance

Table 3-6. Test Fixture Required Components, Non-Terminated Interfaces

Test Condition	R_1	C_L	Timing Ref.	V_T
LVTTL and other LVCMOS settings (L -> H, H -> L)	∞	0pF	LVCMOS 3.3 = 1.5V	—
			LVCMOS 2.5 = V_{CCIO}/2	—
			LVCMOS 1.8 = V_{CCIO}/2	—
			LVCMOS 1.5 = V_{CCIO}/2	—
			LVCMOS 1.2 = V_{CCIO}/2	—
LVCMOS 2.5 I/O (Z -> H)	188¾	0pF	V_{CCIO}/2	V_{OL}
LVCMOS 2.5 I/O (Z -> L)			V_{CCIO}/2	V_{OH}
LVCMOS 2.5 I/O (H -> Z)			V_{OH} - 0.15	V_{OL}
LVCMOS 2.5 I/O (L -> Z)			V_{OL} + 0.15	V_{OH}

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

Signal Descriptions

Signal Name	I/O	Description
General Purpose		
P[Edge] [Row/Column Number*]_[A/B]	I/O	[Edge] indicates the edge of the device on which the pad is located. Valid edge designations are L (Left), B (Bottom), R (Right), T (Top). [Row/Column Number] indicates the PFU row or the column of the device on which the PIC exists. When Edge is T (Top) or (Bottom), only need to specify Row Number. When Edge is L (Left) or R (Right), only need to specify Column Number. [A/B] indicates the PIO within the PIC to which the pad is connected. Some of these user-programmable pins are shared with special function pins. These pin when not used as special purpose pins can be programmed as I/Os for user logic. During configuration the user-programmable I/Os are tri-stated with an internal pull-up resistor enabled. If any pin is not used (or not bonded to a package pin), it is also tri-stated with an internal pull-up resistor enabled after configuration.
GSRN	I	Global RESET signal (active low). Any I/O pin can be GSRN.
NC	—	No connect.
GND	—	Ground. Dedicated pins.
V_{CC}	—	Power supply pins for core logic. Dedicated pins.
V_{CCAUX}	—	Auxiliary power supply pin. It powers all the differential and referenced input buffers. Dedicated pins.
V_{CCIOx}	—	Power supply pins for I/O bank x. Dedicated pins.
V_{REF1_x}, V_{REF2_x}	—	Reference supply pins for I/O bank x. Pre-determined pins in each bank are assigned as V_{REF} inputs. When not used, they may be used as I/O pins.
XRES	—	10K ohm +/-1% resistor must be connected between this pad and ground.
V_{CCPLL}	—	Power supply pin for PLL.þApplicable to ECP/EC33 device.
PLL and Clock Functions (Used as user programmable I/O pins when not in use for PLL or clock pins)		
[LOC][num]_PLL[T, C]_IN_A	I	Reference clock (PLL) input pads: ULM, LLM, URM, LRM, num = row from center, T = true and C = complement, index A,B,C...at each side.
[LOC][num]_PLL[T, C]_FB_A	I	Optional feedback (PLL) input pads: ULM, LLM, URM, LRM, num = row from center, T = true and C = complement, index A,B,C...at each side.
PCLK[T, C]_[n:0]_[3:0]	I	Primary Clock pads, T = true and C = complement, n per side, indexed by bank and 0,1,2,3 within bank.
[LOC]DQS[num]	I	DQS input pads: T (Top), R (Right), B (Bottom), L (Left), DQS, num = ball function number. Any pad can be configured to be output.
Test and Programming (Dedicated pins)		
TMS	I	Test Mode Select input, used to control the 1149.1 state machine. Pull-up is enabled during configuration.
TCK	I	Test Clock input pin, used to clock the 1149.1 state machine. No pull-up enabled.

Signal Descriptions (Cont.)

Signal Name	I/O	Description
TDI	I	Test Data in pin. Used to load data into device using 1149.1 state machine. After power-up, this TAP port can be activated for configuration by sending appropriate command. (Note: once a configuration port is selected it is locked. Another configuration port cannot be selected until the power-up sequence). Pull-up is enabled during configuration.
TDO	O	Output pin. Test Data out pin used to shift data out of device using 1149.1.
V_{CCJ}	—	V_{CCJ} - The power supply pin for JTAG Test Access Port.
Configuration Pads (used during sysCONFIG)		
CFG[2:0]	I	Mode pins used to specify configuration modes values latched on rising edge of INITN. During configuration, a pull-up is enabled. These are dedicated pins.
INITN	I/O	Open Drain pin. Indicates the FPGA is ready to be configured. During configuration, a pull-up is enabled. It is a dedicated pin.
PROGRAMN	I	Initiates configuration sequence when asserted low. This pin always has an active pull-up. This is a dedicated pin.
DONE	I/O	Open Drain pin. Indicates that the configuration sequence is complete, and the startup sequence is in progress. This is a dedicated pin.
CCLK	I/O	Configuration Clock for configuring an FPGA in sysCONFIG mode.
BUSY/SISPI	I/O	Read control command in SPI3 or SPIX mode.
CSN	I	sysCONFIG chip select (Active low). During configuration, a pull-up is enabled.
CS1N	I	sysCONFIG chip select (Active low). During configuration, a pull-up is enabled.
WRITEN	I	Write Data on Parallel port (Active low).
D[7:0]/SPID[0:7]	I/O	sysCONFIG Port Data I/O.
DOUT/CSON	O	Output for serial configuration data (rising edge of CCLK) when using sysCONFIG port.
DI/CSSPIN	I/O	Input for serial configuration data (clocked with CCLK) when using sysCONFIG port. During configuration, a pull-up is enabled. Output when used in SPI/SPIX modes.

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
P[Edge] [n-4]	A	DQ
	B	DQ
P[Edge] [n-3]	A	DQ
	B	DQ
P[Edge] [n-2]	A	DQ
	B	DQ
P[Edge] [n-1]	A	DQ
	B	DQ
P[Edge] [n]	A	[Edge]DQSn
	B	DQ
P[Edge] [n+1]	A	DQ
	B	DQ
P[Edge] [n+2]	A	DQ
	B	DQ
P[Edge] [n+3]	A	DQ
	B	DQ

Notes:
1. "n" is a Row/Column PIC number
2. The DDR interface is designed for memories that support one DQS strobe per eight bits of data. In some packages, all the potential DDR data (DQ) pins may not be available.
3. PIC numbering definitions are provided in the "Signal Names" column of the Signal Descriptions table.

Pin Information Summary

Pin Type		LFEC1			LFEC3				LFECP6/EC6				LFECP/EC10		
		100-TQFP	144-TQFP	208-PQFP	100-TQFP	144-TQFP	208-PQFP	256-fpBGA	144-TQFP	208-PQFP	256-fpBGA	484-fpBGA	208-PQFP	256-fpBGA	484-fpBGA
Single Ended User I/O		67	97	112	67	97	145	160	97	147	195	224	147	195	288
Differential Pair User I/O		29	46	56	29	46	72	80	46	72	97	112	72	97	144
Configuration	Dedicated	13	13	13	13	13	13	13	13	13	13	13	13	13	13
	Muxed	48	48	48	48	48	48	48	48	48	48	48	56	56	56
TAP		5	5	5	5	5	5	5	5	5	5	5	5	5	5
Dedicated (total without supplies)		80	110	160	80	110	160	208	110	160	208	373	160	208	373
V_{CC}		2	3	3	2	3	3	10	4	4	10	20	6	10	20
V_{CCAUX}		2	2	2	4	4	4	4	2	4	2	12	4	2	12
V_{CCPLL}		0	0	0	0	0	0	0	0	0	0	0	0	0	0
V_{CCIO}	Bank0	1	2	2	1	2	3	2	2	3	2	4	3	2	4
	Bank1	1	2	2	1	2	2	2	2	2	2	4	2	2	4
	Bank2	1	1	1	2	2	2	2	1	2	2	4	2	2	4
	Bank3	1	2	2	1	2	2	2	2	2	2	4	2	2	4
	Bank4	1	2	2	1	2	2	2	2	2	2	4	2	2	4
	Bank5	1	2	2	1	2	2	2	2	3	2	4	3	2	4
	Bank6	1	2	2	1	2	2	2	2	2	2	4	2	2	4
	Bank7	1	1	1	2	2	2	2	1	2	2	4	2	2	4
GND, GND0-GND7		8	13	13	8	13	16	20	14	18	20	44	20	20	44
NC		0	2	51	0	2	9	35	0	4	0	139	0	0	75
Single Ended/ Differential I/O Pair per Bank	Bank 0	11/5	14/7	16/8	11/5	14/7	26/13	32/16	14/7	26/13	32/16	32/16	26/13	32/16	48/24
	Bank 1	11/5	13/6	16/8	11/5	13/6	16/8	16/8	13/6	17/8	18/9	32/16	17/8	18/9	32/16
	Bank 2	3/1	8/4	8/4	3/1	8/4	14/7	16/8	8/4	14/7	16/8	16/8	14/7	16/8	32/16
	Bank 3	8/4	13/6	16/8	8/4	13/6	16/8	16/8	13/6	16/8	32/16	32/16	16/8	32/16	32/16
	Bank 4	12/4	14/6	16/8	12/4	14/6	16/8	16/8	14/6	17/8	17/8	32/16	17/8	17/8	32/16
	Bank 5	9/4	13/6	16/8	9/4	13/6	26/13	32/16	13/6	26/13	32/16	32/16	26/13	32/16	48/24
	Bank 6	5/2	14/7	16/8	5/2	14/7	16/8	16/8	14/7	16/8	32/16	32/16	16/8	32/16	32/16
	Bank 7	8/4	8/4	8/4	8/4	8/4	15/7	16/8	8/4	15/7	16/8	16/8	15/7	16/8	32/16
V_{CCJ}		1	1	1	1	1	1	1	1	1	1	1	1	1	1

Note: During configuration the user-programmable I/Os are tri-stated with an internal pull-up resistor enabled. If any pin is not used (or not bonded to a package pin), it is also tri-stated with an internal pull-up resistor enabled after configuration.

Pin Information Summary (Cont.)

Pin Type		LFECP/EC15		LFECP20/EC20		LFECP/EC33	
		256-fpBGA	484-fpBGA	484-fpBGA	672-fpBGA	484-fpBGA	672-fpBGA
Single Ended User I/O		195	352	360	400	360	496
Differential Pair User I/O		97	176	180	200	180	248
Configuration	Dedicated	13	13	13	13	13	13
	Muxed	56	56	56	56	56	56
TAP		5	5	5	5	5	5
Dedicated (total without supplies)		208	373	373	509	373	509
V_{CC}		10	20	20	32	16	28
V_{CCAUX}		2	12	12	20	12	20
V_{CCPLL}		0	0	0	0	4	4
V_{CCIO}	Bank0	2	4	4	6	4	6
	Bank1	2	4	4	6	4	6
	Bank2	2	4	4	6	4	6
	Bank3	2	4	4	6	4	6
	Bank4	2	4	4	6	4	6
	Bank5	2	4	4	6	4	6
	Bank6	2	4	4	6	4	6
	Bank7	2	4	4	6	4	6
GND, GND0-GND7		20	44	44	63	44	63
NC		0	11	3	96	3	0
Single Ended/ Differential I/O Pair per Bank	Bank0	32/16	48/24	48/24	64/32	48/24	64/32
	Bank1	18/9	48/24	48/24	48/24	48/24	64/32
	Bank2	16/8	40/20	40/20	40/20	40/20	56/28
	Bank3	32/16	40/20	44/22	48/24	44/22	64/32
	Bank4	17/8	48/24	48/24	48/24	48/24	64/32
	Bank5	32/16	48/24	48/24	64/32	48/24	64/32
	Bank6	32/16	40/20	44/22	48/24	44/22	64/32
	Bank7	16/8	40/20	40/20	40/20	40/20	56/28
V_{CCJ}		1	1	1	1	1	1

Note: During configuration the user-programmable I/Os are tri-stated with an internal pull-up resistor enabled. If any pin is not used (or not bonded to a package pin), it is also tri-stated with an internal pull-up resistor enabled after configuration.

Power Supply and NC Connections

Signals	100 TQFP	144 TQFP	208 PQFP	256 fpBGA
VCC	12, 64	**EC1, EC3:** 13, 92, 99 **ECP/EC6:** 11, 13, 92, 99	**EC1, EC3:** 26, 128, 135 **ECP/EC6:** 24, 26, 128, 135 **ECP/EC10:** 5, 24, 26, 128, 135, 152	E12, E5, E8, M12, M5, M9, F6, F11, L11, L6
VCCIO0	100	136, 143	**EC1:** 187, 208 **EC3, ECP/EC6, ECP/EC10:** 187, 197, 208	F7, F8
VCCIO1	86	110, 125	157, 176	F9, F10
VCCIO2	73	108	**EC1:** 155 **EC3, ECP/EC6, ECP/EC10:** 145, 155	G11, H11
VCCIO3	56	73, 84	106, 120	J11, K11
VCCIO4	38	55, 71	85, 104	L9, L10
VCCIO5	26	38, 44	**EC1:** 53, 74 **EC2, ECP/EC6, ECP/EC10:** 53, 64, 74	L7, L8
VCCIO6	24	24, 36	37, 51	J6, K6
VCCIO7	2	1	**EC1:** 2 **EC3, ECP/EC6, ECP/EC10:** 2, 13	G6, H6
VCCJ	18	19	32	L4
VCCAUX	37, 87	54, 126	**EC1:** 84, 177 **EC3, ECP/EC6, ECP/EC10:** 22, 84, 136, 177	B15, R2
VCCPLL	—	—	—	—
GND, GND0-GND7	1, 14, 25, 35, 51, 68, 74, 89	**EC1, EC3:** 15, 28, 37, 52, 63, 72, 80, 96, 98, 109, 117, 128, 144 **ECP/EC6:** 12, 15, 28, 37, 52, 63, 72, 80, 96, 98, 109, 117, 128, 144	**EC1:** 1, 28, 41, 52, 82, 93, 105, 116, 132, 134, 156, 168, 179 **EC3:** 1, 28, 41, 52, 72, 82, 93, 105, 116, 132, 134, 138, 156, 168, 179, 189 **ECP/EC6:** 1, 18, 25, 28, 41, 52, 72, 82, 93, 105, 116, 132, 134, 138, 156, 168, 179, 189 ECP/EC10: 1, 6, 18, 25, 28, 41, 52, 72, 82, 93, 105, 116, 132, 134, 138, 151, 156, 168, 179, 189	A1, A16, G10, G7, G8, G9, H10, H7, H8, H9, J10, J7, J8, J9, K10, K7, K8, K9, T1, T16
NC	—	**EC1, EC3:** 11, 12 **ECP6/EC6:** None	**EC1:** 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 18, 22, 24, 25, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 72, 103, 136, 138, 144, 145, 146, 147, 148, 149, 150, 151, 152, 158, 189, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207 **EC3:** 5, 6, 18, 24, 25, 103, 151, 152, 158 **ECP/EC6:** 5, 6, 151, 152 **ECP/EC10:** None	**EC3:** G5, H5, F2, F1, H4, H3, G2, G1, J4, J3, J5, K5, H2, H1, J2, J1, R12, H16, H15, G16, G15, K12, J12, J14, J15, F16, F15, J13, H13, H14, G14, E16, E15, B13, C13 **ECP/EC10:** None **ECP/EC15:** None

Power Supply and NC Connections (Cont.)

Signals	484 fpBGA	672 fpBGA
VCC	J16, J7, K16, K17, K6, K7, L17, L6, M17, M6, N16, N17, N6, N7, P16, P7, J6, J17, P6, P17	H10, H11, H16, H17, H18, H19, H8, H9, J18, J9, K8, L19, M19, N7, R20, R7, T19, V18, V8, V9, W10, W11, W16, W17, W18, W19, W8, W9, K19, L8, U19, U8
VCCIO0	G11, H10, H11, H9	H12, H13, J10, J11, J12, J13
VCCIO1	G12, H12, H13, H14	H14, H15, J14, J15, J16, J17
VCCIO2	J15, K15, L15, L16	K17, K18, L18, M18, N18, N19
VCCIO3	M15, M16, N15, P15	P18, P19, R18, R19, T18, U18
VCCIO4	R12, R13, R14, T12	V14, V15, V16, V17, W14, W15
VCCIO5	R10, R11, R9, T11	V10, V11, V12, V13, W12, W13
VCCIO6	M7, M8, N8, P8	P8, P9, R8, R9, T9, U9
VCCIO7	J8, K8, L7, L8	K9, L9, M8, M9, N8, N9
VCCJ	U2	U6
VCCAUX	G15, G16, G7, G8, H16, H7, R16, R7, T15, T16, T7, T8	G13, H20, H7, J19, J8, K7, L20, M20, M7, N20, P20, P7, T20, T7, T8, V19, V7, W20, Y13, Y7
VCCPLL	**ECP/EC20**: None **ECP/EC33**: J6, J17, P6, P17	**ECP/EC20**: None **ECP/EC33**: K19, L8, U19, U8
GND, GND0-GND7	A1, A22, AB1, AB22, H15, H8, J10, J11, J12, J13, J14, J9, K10, K11, K12, K13, K14, K9, L10, L11, L12, L13, L14, L9, M10, M11, M12, M13, M14, M9, N10, N11, N12, N13, N14, N9, P10, P11, P12, P13, P14, P9, R15, R8	K10, K11, K12, K13, K14, K15, K16, L10, L11, L12, L13, L14, L15, L16, L17, M10, M11, M12, M13, M14, M15, M16, M17, N10, N11, N12, N13, N14, N15, N16, N17, P10, P11, P12, P13, P14, P15, P16, P17, R10, R11, R12, R13, R14, R15, R16, R17, T10, T11, T12, T13, T14, T15, T16, T17, U10, U11, U12, U13, U14, U15, U16, U17
NC	**ECP/EC6**: C3, B2, E5, F5, D3, C2, F4, G4, E3, D2, B1, C1, F3, E2, G5, H6, G3, H4, J5, H5, F2, F1, E1, D1, R6, P5, P3, P4, R1, R2, R5, R4, T1, T2, R3, T3, V7, T6, V8, U7, W5, U6, AA3, AB3, Y6, V6, AA5, W6, Y5, Y4, AA4, AB4, W16, U15, V16, U16, Y17, V17, AB20, AA19, Y16, W17, AA20, Y19, Y18, W18, T17, U17, T18, R17, R19, R18, U22, T22, R21, R22, P20, N20, P19, P18, E21, D22, G21, G20, J18, H19, J19, H20, H17, H18, D21, C22, G19, G18, F20, F19, E20, D20, C21, C20, F18, E18, B22, B21, G17, F17, D18, C18, C19, B20, D17, C16, B19, A20, E17, C17, F16, E16, F15, D16, A4, B4, C4, C5, D6, B5, E6, C6, A3, B3, F6, D5, F7, E8, G6, E7, A2, AB2, A21 **ECP/EC10**: G5, H6, G3, H4, J5, H5, F2, F1, R6, P5, P3, P4, R2, R1, R5, R4, T1, T2, R3, T3, W16, U15, V16, U16, Y17, V17, AB20, AA19, Y16, W17, AA20, Y19, Y18, W18, T17, U17, T18, R17, R19, R18, U22, T22, R21, R22, P20, N20, P19, P18, G21, G20, J18, H19, J19, H20, H17, H18, G17, F17, D18, C18, C19, B20, D17, C16, B19, A20, E17, C17, F16, E16, F15, D16, A2, AB2, A21 **ECP/EC15:** T1, T2, R3, T3, T18, R17, R19, R18, A2, AB2, A21 **ECP/EC20:** A2, AB2, A21 **ECP/EC33:** A2, AB2, A21	**ECP/EC20:** E5, D5, F4, F5, C3, D3, C2, B2, H6, J7, G5, H5, H3, J3, H2, J2, AA2, AA3, W5, Y5, Y6, W7, AA4, AB3, AC2, AC3, AA5, AB5, AD3, AD2, AE1, AD1, AD19, AD20, AC19, AB19, AD21, AC20, AF25, AE25, AB21, AB20, AE24, AD23, AD22, AC21, AC22, AB22, AD24, AD25, AE26, AD26, Y20, Y19, AA23, AA22, AB23, AB24, Y21, AA21, Y23, Y22, AA24, Y24, J21, J22, J23, H22, G26, F26, E26, E25, F24, F23, E24, D24, E22, F22, E21, D22, G20, F20, D21, C21, C23, C22, B23, C24, D20, E19, B25, B24, B26, A25, C20, C19 **ECP/EC33:** None

LFEC1, LFEC3 Logic Signal Connections: 100 TQFP

Pin Number	LFEC1 Pin Function	Bank	LVDS	Dual Function	LFEC3 Pin Function	Bank	LVDS	Dual Function
1*	GND0 GND7	-			GND0 GND7	-		
2	VCCIO7	7			VCCIO7	7		
3	PL2A	7	T	VREF2_7	PL2A	7	T	VREF2_7
4	PL2B	7	C	VREF1_7	PL2B	7	C	VREF1_7
5	PL3A	7	T		PL7A	7	T	
6	PL3B	7	C		PL7B	7	C	
7	PL4A	7	T		PL8A	7	T	
8	PL4B	7	C		PL8B	7	C	
9	PL5A	7	T	PCLKT7_0	PL9A	7	T	PCLKT7_0
10	PL5B	7	C	PCLKC7_0	PL9B	7	C	PCLKC7_0
11	XRES	6			XRES	6		
12	VCC	-			VCC	-		
13	TCK	6			TCK	6		
14	GND	-			GND	-		
15	TDI	6			TDI	6		
16	TMS	6			TMS	6		
17	TDO	6			TDO	6		
18	VCCJ	6			VCCJ	6		
19	PL7A	6	T	LLM0_PLLT_IN_A	PL11A	6	T	LUM0_PLLT_IN_A
20	PL7B	6	C	LLM0_PLLC_IN_A	PL11B	6	C	LUM0_PLLC_IN_A
21	PL8A	6	T	LLM0_PLLT_FB_A	PL12A	6	T	LUM0_PLLT_FB_A
22	PL8B	6	C	LLM0_PLLC_FB_A	PL12B	6	C	LUM0_PLLC_FB_A
23	PL14A	6		VREF1_6	PL18A	6		VREF1_6
24	VCCIO6	6			VCCIO6	6		
25*	GND5 GND6	-			GND5 GND6	-		
26	VCCIO5	5			VCCIO5	5		
27	PB2A	5	T		PB10A	5	T	
28	PB2B	5	C		PB10B	5	C	
29	PB3A	5	T		PB11A	5	T	
30	PB3B	5	C		PB11B	5	C	
31	PB6A	5		BDQS6	PB14A	5		BDQS14
32	PB8A	5	T	VREF2_5	PB16A	5	T	VREF2_5
33	PB8B	5	C	VREF1_5	PB16B	5	C	VREF1_5
34	PB9A	5	T	PCLKT5_0	PB17A	5	T	PCLKT5_0
35	GND5	5			GND5	5		
36	PB9B	5	C	PCLKC5_0	PB17B	5	C	PCLKC5_0
37	VCCAUX	-			VCCAUX	-		
38	VCCIO4	4			VCCIO4	4		
39	PB10A	4	T	WRITEN	PB18A	4	T	WRITEN
40	PB10B	4	C	CS1N	PB18B	4	C	CS1N

LFEC1, LFEC3 Logic Signal Connections: 100 TQFP (Cont.)

Pin Number	LFEC1				LFEC3			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
41	PB11A	4	T	VREF1_4	PB19A	4	T	VREF1_4
42	PB11B	4	C	CSN	PB19B	4	C	CSN
43	PB12B	4		D0/SPID7	PB20B	4		D0/SPID7
44	PB13A	4	T	D2/SPID5	PB21A	4	T	D2/SPID5
45	PB13B	4	C	D1/SPID6	PB21B	4	C	D1/SPID6
46	PB14A	4	T	BDQS14	PB22A	4	T	BDQS22
47	PB14B	4	C	D3/SPID4	PB22B	4	C	D3/SPID4
48	PB15B	4		D4/SPID3	PB23B	4		D4/SPID3
49	PB16B	4		D5/SPID2	PB24B	4		D5/SPID2
50	PB17B	4		D6/SPID1	PB25B	4		D6/SPID1
51*	GND3 GND4	-			GND3 GND4	-		
52	PR10B	3	C	RLM0_PLLC_FB_A	PR14B	3	C	RLM0_PLLC_FB_A
53	PR10A	3	T	RLM0_PLLT_FB_A	PR14A	3	T	RLM0_PLLT_FB_A
54	PR9B	3	C	RLM0_PLLC_IN_A	PR13B	3	C	RLM0_PLLC_IN_A
55	PR9A	3	T	RLM0_PLLT_IN_A	PR13A	3	T	RLM0_PLLT_IN_A
56	VCCIO3	3			VCCIO3	3		
57	PR8B	3	C	DI/CSSPIN	PR12B	3	C	DI/CSSPIN
58	PR8A	3	T	DOUT/CSON	PR12A	3	T	DOUT/CSON
59	PR7B	3	C	BUSY/SISPI	PR11B	3	C	BUSY/SISPI
60	PR7A	3	T	D7/SPID0	PR11A	3	T	D7/SPID0
61	CFG2	3			CFG2	3		
62	CFG1	3			CFG1	3		
63	CFG0	3			CFG0	3		
64	VCC	-			VCC	-		
65	PROGRAMN	3			PROGRAMN	3		
66	CCLK	3			CCLK	3		
67	INITN	3			INITN	3		
68	GND	-			GND	-		
69	DONE	3			DONE	3		
70	PR5B	2	C	PCLKC2_0	PR9B	2	C	PCLKC2_0
71	PR5A	2	T	PCLKT2_0	PR9A	2	T	PCLKT2_0
72	PR2B	2		VREF1_2	PR2B	2		VREF1_2
73	VCCIO2	2			VCCIO2	2		
74	GND2	2			GND2	2		
75	PT17B	1	C		PT25B	1	C	
76	PT17A	1	T		PT25A	1	T	
77	PT14B	1	C		PT22B	1	C	
78	PT14A	1	T	TDQS14	PT22A	1	T	TDQS22
79	PT13A	1			PT21A	1		
80	PT12B	1	C		PT20B	1	C	
81	PT12A	1	T		PT20A	1	T	

LFEC1, LFEC3 Logic Signal Connections: 100 TQFP (Cont.)

Pin Number	LFEC1				LFEC3			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
82	PT11B	1	C	VREF2_1	PT19B	1	C	VREF2_1
83	PT11A	1	T	VREF1_1	PT19A	1	T	VREF1_1
84	PT10B	1	C		PT18B	1	C	
85	PT10A	1	T		PT18A	1	T	
86	VCCIO1	1			VCCIO1	1		
87	VCCAUX	-			VCCAUX	-		
88	PT9B	0	C	PCLKC0_0	PT17B	0	C	PCLKC0_0
89	GND0	0			GND0	0		
90	PT9A	0	T	PCLKT0_0	PT17A	0	T	PCLKT0_0
91	PT8B	0	C	VREF1_0	PT16B	0	C	VREF1_0
92	PT8A	0	T	VREF2_0	PT16A	0	T	VREF2_0
93	PT7B	0			PT15B	0		
94	PT6B	0	C		PT14B	0	C	
95	PT6A	0	T	TDQS6	PT14A	0	T	TDQS14
96	PT4B	0	C		PT12B	0	C	
97	PT4A	0	T		PT12A	0	T	
98	PT2B	0	C		PT10B	0	C	
99	PT2A	0	T		PT10A	0	T	
100	VCCIO0	0			VCCIO0	0		

*Double bonded to the pin.

LFEC1, LFEC3, LFECP/EC6 Logic Signal Connections: 144 TQFP

Pin Number	LFEC1				LFEC3				LFECP6/EC6			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
1	VCCIO7	7			VCCIO7	7			VCCIO7	7		
2	PL2A	7	T	VREF2_7	PL2A	7	T	VREF2_7	PL2A	7	T	VREF2_7
3	PL2B	7	C	VREF1_7	PL2B	7	C	VREF1_7	PL2B	7	C	VREF1_7
4	PL3A	7	T		PL7A	7	T		PL7A	7	T	
5	PL3B	7	C		PL7B	7	C		PL7B	7	C	
6	PL4A	7	T		PL8A	7	T		PL8A	7	T	
7	PL4B	7	C		PL8B	7	C		PL8B	7	C	
8	PL5A	7	T	PCLKT7_0	PL9A	7	T	PCLKT7_0	PL9A	7	T	PCLKT7_0
9	PL5B	7	C	PCLKC7_0	PL9B	7	C	PCLKC7_0	PL9B	7	C	PCLKC7_0
10	XRES	6			XRES	6			XRES	6		
11	NC	-			NC	-			VCC	-		
12	NC	-			NC	-			GND	-		
13	VCC	-			VCC	-			VCC	-		
14	TCK	6			TCK	6			TCK	6		
15	GND	-			GND	-			GND	-		
16	TDI	6			TDI	6			TDI	6		
17	TMS	6			TMS	6			TMS	6		
18	TDO	6			TDO	6			TDO	6		
19	VCCJ	6			VCCJ	6			VCCJ	6		
20	PL7A	6	T	LLM0_PLLT_IN_A	PL11A	6	T	LLM0_PLLT_IN_A	PL20A	6	T	LLM0_PLLT_IN_A
21	PL7B	6	C	LLM0_PLLC_IN_A	PL11B	6	C	LLM0_PLLC_IN_A	PL20B	6	C	LLM0_PLLC_IN_A
22	PL8A	6	T	LLM0_PLLT_FB_A	PL12A	6	T	LLM0_PLLT_FB_A	PL21A	6	T	LLM0_PLLT_FB_A
23	PL8B	6	C	LLM0_PLLC_FB_A	PL12B	6	C	LLM0_PLLC_FB_A	PL21B	6	C	LLM0_PLLC_FB_A
24	VCCIO6	6			VCCIO6	6			VCCIO6	6		
25	PL9A	6	T		PL13A	6	T		PL22A	6	T	
26	PL9B	6	C		PL13B	6	C		PL22B	6	C	
27	PL10A	6	T		PL14A	6	T		PL23A	6	T	
28	GND6	6			GND6	6			GND6	6		
29	PL10B	6	C		PL14B	6	C		PL23B	6	C	
30	PL11A	6	T	LDQS11	PL15A	6	T	LDQS15	PL24A	6	T	LDQS24
31	PL11B	6	C		PL15B	6	C		PL24B	6	C	
32	PL12A	6	T		PL16A	6	T		PL25A	6	T	
33	PL12B	6	C		PL16B	6	C		PL25B	6	C	
34	PL14A	6	T	VREF1_6	PL18A	6	T	VREF1_6	PL27A	6	T	VREF1_6
35	PL14B	6	C	VREF2_6	PL18B	6	C	VREF2_6	PL27B	6	C	VREF2_6
36	VCCIO6	6			VCCIO6	6			VCCIO6	6		
37*	GND5 GND6	-			GND5 GND6	-			GND5 GND6	-		
38	VCCIO5	5			VCCIO5	5			VCCIO5	5		
39	PB2A	5	T		PB10A	5	T		PB10A	5	T	
40	PB2B	5	C		PB10B	5	C		PB10B	5	C	
41	PB3A	5	T		PB11A	5	T		PB11A	5	T	
42	PB3B	5	C		PB11B	5	C		PB11B	5	C	
43	PB5B	5			PB13B	5			PB13B	5		
44	VCCIO5	5			VCCIO5	5			VCCIO5	5		
45	PB6A	5	T	BDQS6	PB14A	5	T	BDQS14	PB14A	5	T	BDQS14
46	PB6B	5	C		PB14B	5	C		PB14B	5	C	
47	PB7A	5	T		PB15A	5	T		PB15A	5	T	
48	PB7B	5	C		PB15B	5	C		PB15B	5	C	
49	PB8A	5	T	VREF2_5	PB16A	5	T	VREF2_5	PB16A	5	T	VREF2_5

LFEC1, LFEC3, LFECP/EC6 Logic Signal Connections: 144 TQFP (Cont.)

Pin Number	LFEC1				LFEC3				LFECP6/EC6			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
50	PB8B	5	C	VREF1_5	PB16B	5	C	VREF1_5	PB16B	5	C	VREF1_5
51	PB9A	5	T	PCLKT5_0	PB17A	5	T	PCLKT5_0	PB17A	5	T	PCLKT5_0
52	GND5	5			GND5	5			GND5	5		
53	PB9B	5	C	PCLKC5_0	PB17B	5	C	PCLKC5_0	PB17B	5	C	PCLKC5_0
54	VCCAUX	-			VCCAUX	-			VCCAUX	-		
55	VCCIO4	4			VCCIO4	4			VCCIO4	4		
56	PB10A	4	T	WRITEN	PB18A	4	T	WRITEN	PB18A	4	T	WRITEN
57	PB10B	4	C	CS1N	PB18B	4	C	CS1N	PB18B	4	C	CS1N
58	PB11A	4	T	VREF1_4	PB19A	4	T	VREF1_4	PB19A	4	T	VREF1_4
59	PB11B	4	C	CSN	PB19B	4	C	CSN	PB19B	4	C	CSN
60	PB12A	4	T	VREF2_4	PB20A	4	T	VREF2_4	PB20A	4	T	VREF2_4
61	PB12B	4	C	D0/SPID7	PB20B	4	C	D0/SPID7	PB20B	4	C	D0/SPID7
62	PB13A	4	T	D2/SPID5	PB21A	4	T	D2/SPID5	PB21A	4	T	D2/SPID5
63	GND4	4			GND4	4			GND4	4		
64	PB13B	4	C	D1/SPID6	PB21B	4	C	D1/SPID6	PB21B	4	C	D1/SPID6
65	PB14A	4	T	BDQS14	PB22A	4	T	BDQS22	PB22A	4	T	BDQS22
66	PB14B	4	C	D3/SPID4	PB22B	4	C	D3/SPID4	PB22B	4	C	D3/SPID4
67	PB15A	4	T		PB23A	4	T		PB23A	4	T	
68	PB15B	4	C	D4/SPID3	PB23B	4	C	D4/SPID3	PB23B	4	C	D4/SPID3
69	PB16B	4		D5/SPID2	PB24B	4		D5/SPID2	PB24B	4		D5/SPID2
70	PB17B	4		D6/SPID1	PB25B	4		D6/SPID1	PB25B	4		D6/SPID1
71	VCCIO4	4			VCCIO4	4			VCCIO4	4		
72*	GND3 GND4	-			GND3 GND4	-			GND3 GND4	-		
73	VCCIO3	3			VCCIO3	3			VCCIO3	3		
74	PR14A	3		VREF1_3	PR18A	3		VREF1_3	PR27A	3		VREF1_3
75	PR12B	3	C		PR16B	3	C		PR25B	3	C	
76	PR12A	3	T		PR16A	3	T		PR25A	3	T	
77	PR11B	3	C		PR15B	3	C		PR24B	3	C	
78	PR11A	3	T	RDQS11	PR15A	3	T	RDQS15	PR24A	3	T	RDQS24
79	PR10B	3	C	RLM0_PLLC_FB_A	PR14B	3	C	RLM0_PLLC_FB_A	PR23B	3	C	RLM0_PLLC_FB_A
80	GND3	3			GND3	3			GND3	3		
81	PR10A	3	T	RLM0_PLLT_FB_A	PR14A	3	T	RLM0_PLLT_FB_A	PR23A	3	T	RLM0_PLLT_FB_A
82	PR9B	3	C	RLM0_PLLC_IN_A	PR13B	3	C	RLM0_PLLC_IN_A	PR22B	3	C	RLM0_PLLC_IN_A
83	PR9A	3	T	RLM0_PLLT_IN_A	PR13A	3	T	RLM0_PLLT_IN_A	PR22A	3	T	RLM0_PLLT_IN_A
84	VCCIO3	3			VCCIO3	3			VCCIO3	3		
85	PR8B	3	C	DI/CSSPIN	PR12B	3	C	DI/CSSPIN	PR21B	3	C	DI/CSSPIN
86	PR8A	3	T	DOUT/CSON	PR12A	3	T	DOUT/CSON	PR21A	3	T	DOUT/CSON
87	PR7B	3	C	BUSY/SISPI	PR11B	3	C	BUSY/SISPI	PR20B	3	C	BUSY/SISPI
88	PR7A	3	T	D7/SPID0	PR11A	3	T	D7/SPID0	PR20A	3	T	D7/SPID0
89	CFG2	3			CFG2	3			CFG2	3		
90	CFG1	3			CFG1	3			CFG1	3		
91	CFG0	3			CFG0	3			CFG0	3		
92	VCC	-			VCC	-			VCC	-		
93	PROGRAMN	3			PROGRAMN	3			PROGRAMN	3		
94	CCLK	3			CCLK	3			CCLK	3		
95	INITN	3			INITN	3			INITN	3		
96	GND	-			GND	-			GND	-		
97	DONE	3			DONE	3			DONE	3		
98	GND	-			GND	-			GND	-		

LFEC1, LFEC3, LFECP/EC6 Logic Signal Connections: 144 TQFP (Cont.)

Pin Number	LFEC1				LFEC3				LFECP6/EC6			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
99	VCC	-			VCC	-			VCC	-		
100	PR5B	2	C	PCLKC2_0	PR9B	2	C	PCLKC2_0	PR9B	2	C	PCLKC2_0
101	PR5A	2	T	PCLKT2_0	PR9A	2	T	PCLKT2_0	PR9A	2	T	PCLKT2_0
102	PR4B	2	C		PR8B	2	C		PR8B	2	C	
103	PR4A	2	T		PR8A	2	T		PR8A	2	T	
104	PR3B	2	C		PR7B	2	C		PR7B	2	C	
105	PR3A	2	T		PR7A	2	T		PR7A	2	T	
106	PR2B	2	C	VREF1_2	PR2B	2	C	VREF1_2	PR2B	2	C	VREF1_2
107	PR2A	2	T	VREF2_2	PR2A	2	T	VREF2_2	PR2A	2	T	VREF2_2
108	VCCIO2	2			VCCIO2	2			VCCIO2	2		
109*	GND1 GND2	-			GND1 GND2	-			GND1 GND2	-		
110	VCCIO1	1			VCCIO1	1			VCCIO1	1		
111	PT17B	1	C		PT25B	1	C		PT25B	1	C	
112	PT17A	1	T		PT25A	1	T		PT25A	1	T	
113	PT15A	1			PT23A	1			PT23A	1		
114	PT14B	1	C		PT22B	1	C		PT22B	1	C	
115	PT14A	1	T	TDQS14	PT22A	1	T	TDQS22	PT22A	1	T	TDQS22
116	PT13B	1	C		PT21B	1	C		PT21B	1	C	
117	GND1	1			GND1	1			GND1	1		
118	PT13A	1	T		PT21A	1	T		PT21A	1	T	
119	PT12B	1	C		PT20B	1	C		PT20B	1	C	
120	PT12A	1	T		PT20A	1	T		PT20A	1	T	
121	PT11B	1	C	VREF2_1	PT19B	1	C	VREF2_1	PT19B	1	C	VREF2_1
122	PT11A	1	T	VREF1_1	PT19A	1	T	VREF1_1	PT19A	1	T	VREF1_1
123	PT10B	1	C		PT18B	1	C		PT18B	1	C	
124	PT10A	1	T		PT18A	1	T		PT18A	1	T	
125	VCCIO1	1			VCCIO1	1			VCCIO1	1		
126	VCCAUX	-			VCCAUX	-			VCCAUX	-		
127	PT9B	0	C	PCLKC0_0	PT17B	0	C	PCLKC0_0	PT17B	0	C	PCLKC0_0
128	GND0	0			GND0	0			GND0	0		
129	PT9A	0	T	PCLKT0_0	PT17A	0	T	PCLKT0_0	PT17A	0	T	PCLKT0_0
130	PT8B	0	C	VREF1_0	PT16B	0	C	VREF1_0	PT16B	0	C	VREF1_0
131	PT8A	0	T	VREF2_0	PT16A	0	T	VREF2_0	PT16A	0	T	VREF2_0
132	PT7B	0	C		PT15B	0	C		PT15B	0	C	
133	PT7A	0	T		PT15A	0	T		PT15A	0	T	
134	PT6B	0	C		PT14B	0	C		PT14B	0	C	
135	PT6A	0	T	TDQS6	PT14A	0	T	TDQS14	PT14A	0	T	TDQS14
136	VCCIO0	0			VCCIO0	0			VCCIO0	0		
137	PT5B	0	C		PT13B	0	C		PT13B	0	C	
138	PT5A	0	T		PT13A	0	T		PT13A	0	T	
139	PT4B	0	C		PT12B	0	C		PT12B	0	C	
140	PT4A	0	T		PT12A	0	T		PT12A	0	T	
141	PT2B	0	C		PT10B	0	C		PT10B	0	C	
142	PT2A	0	T		PT10A	0	T		PT10A	0	T	
143	VCCIO0	0			VCCIO0	0			VCCIO0	0		
144*	GND0 GND7	-			GND0 GND7	-			GND0 GND7	-		

*Double bonded to the pin.

LFEC1, LFEC3 Logic Signal Connections: 208 PQFP

Pin Number	LFEC1				LFEC3			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
1*	GND0 GND7	-			GND0 GND7	-		
2	VCCIO7	7			VCCIO7	7		
3	PL2A	7	T	VREF2_7	PL2A	7	T	VREF2_7
4	PL2B	7	C	VREF1_7	PL2B	7	C	VREF1_7
5	NC	-			NC	-		
6	NC	-			NC	-		
7	NC	-			PL3B	7		
8	NC	-			PL4A	7	T	
9	NC	-			PL4B	7	C	
10	NC	-			PL5A	7	T	
11	NC	-			PL5B	7	C	
12	NC	-			PL6A	7	T	LDQS6
13	NC	-			VCCIO7	7		
14	NC	-			PL6B	7	C	
15	PL3A	7	T		PL7A	7	T	
16	PL3B	7	C		PL7B	7	C	
17	PL4A	7	T		PL8A	7	T	
18	NC	-			NC	-		
19	PL4B	7	C		PL8B	7	C	
20	PL5A	7	T	PCLKT7_0	PL9A	7	T	PCLKT7_0
21	PL5B	7	C	PCLKC7_0	PL9B	7	C	PCLKC7_0
22	NC	-			VCCAUX	-		
23	XRES	6			XRES	6		
24	NC	-			NC	-		
25	NC	-			NC	-		
26	VCC	-			VCC	-		
27	TCK	6			TCK	6		
28	GND	-			GND	-		
29	TDI	6			TDI	6		
30	TMS	6			TMS	6		
31	TDO	6			TDO	6		
32	VCCJ	6			VCCJ	6		
33	PL7A	6	T	LLM0_PLLT_IN_A	PL11A	6	T	LLM0_PLLT_IN_A
34	PL7B	6	C	LLM0_PLLC_IN_A	PL11B	6	C	LLM0_PLLC_IN_A
35	PL8A	6	T	LLM0_PLLT_FB_A	PL12A	6	T	LLM0_PLLT_FB_A
36	PL8B	6	C	LLM0_PLLC_FB_A	PL12B	6	C	LLM0_PLLC_FB_A
37	VCCIO6	6			VCCIO6	6		
38	PL9A	6	T		PL13A	6	T	
39	PL9B	6	C		PL13B	6	C	
40	PL10A	6	T		PL14A	6	T	
41	GND6	6			GND6	6		
42	PL10B	6	C		PL14B	6	C	

LFEC1, LFEC3 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFEC1				LFEC3			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
43	PL11A	6	T	LDQS11	PL15A	6	T	LDQS15
44	PL11B	6	C		PL15B	6	C	
45	PL12A	6	T		PL16A	6	T	
46	PL12B	6	C		PL16B	6	C	
47	PL13A	6	T		PL17A	6	T	
48	PL13B	6	C		PL17B	6	C	
49	PL14A	6	T	VREF1_6	PL18A	6	T	VREF1_6
50	PL14B	6	C	VREF2_6	PL18B	6	C	VREF2_6
51	VCCIO6	6			VCCIO6	6		
52*	GND5 GND6	-			GND5 GND6	-		
53	VCCIO5	5			VCCIO5	5		
54	NC	-			PB2A	5	T	
55	NC	-			PB2B	5	C	
56	NC	-			PB3A	5	T	
57	NC	-			PB3B	5	C	
58	NC	-			PB4A	5	T	
59	NC	-			PB4B	5	C	
60	NC	-			PB5A	5	T	
61	NC	-			PB5B	5	C	
62	NC	-			PB6A	5	T	BDQS6
63	NC	-			PB6B	5	C	
64	NC	-			VCCIO5	5		
65	PB2A	5	T		PB10A	5	T	
66	PB2B	5	C		PB10B	5	C	
67	PB3A	5	T		PB11A	5	T	
68	PB3B	5	C		PB11B	5	C	
69	PB4A	5	T		PB12A	5	T	
70	PB4B	5	C		PB12B	5	C	
71	PB5A	5	T		PB13A	5	T	
72	NC	-			GND5	5		
73	PB5B	5	C		PB13B	5	C	
74	VCCIO5	5			VCCIO5	5		
75	PB6A	5	T	BDQS6	PB14A	5	T	BDQS14
76	PB6B	5	C		PB14B	5	C	
77	PB7A	5	T		PB15A	5	T	
78	PB7B	5	C		PB15B	5	C	
79	PB8A	5	T	VREF2_5	PB16A	5	T	VREF2_5
80	PB8B	5	C	VREF1_5	PB16B	5	C	VREF1_5
81	PB9A	5	T	PCLKT5_0	PB17A	5	T	PCLKT5_0
82	GND5	5			GND5	5		
83	PB9B	5	C	PCLKC5_0	PB17B	5	C	PCLKC5_0
84	VCCAUX	-			VCCAUX	-		

LFEC1, LFEC3 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFEC1				LFEC3			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
85	VCCIO4	4			VCCIO4	4		
86	PB10A	4	T	WRITEN	PB18A	4	T	WRITEN
87	PB10B	4	C	CS1N	PB18B	4	C	CS1N
88	PB11A	4	T	VREF1_4	PB19A	4	T	VREF1_4
89	PB11B	4	C	CSN	PB19B	4	C	CSN
90	PB12A	4	T	VREF2_4	PB20A	4	T	VREF2_4
91	PB12B	4	C	D0/SPID7	PB20B	4	C	D0/SPID7
92	PB13A	4	T	D2/SPID5	PB21A	4	T	D2/SPID5
93	GND4	4			GND4	4		
94	PB13B	4	C	D1/SPID6	PB21B	4	C	D1/SPID6
95	PB14A	4	T	BDQS14	PB22A	4	T	BDQS22
96	PB14B	4	C	D3/SPID4	PB22B	4	C	D3/SPID4
97	PB15A	4	T		PB23A	4	T	
98	PB15B	4	C	D4/SPID3	PB23B	4	C	D4/SPID3
99	PB16A	4	T		PB24A	4	T	
100	PB16B	4	C	D5/SPID2	PB24B	4	C	D5/SPID2
101	PB17A	4	T		PB25A	4	T	
102	PB17B	4	C	D6/SPID1	PB25B	4	C	D6/SPID1
103	NC	-			NC	-		
104	VCCIO4	4			VCCIO4	4		
105*	GND3 GND4	-			GND3 GND4	-		
106	VCCIO3	3			VCCIO3	3		
107	PR14B	3	C	VREF2_3	PR18B	3	C	VREF2_3
108	PR14A	3	T	VREF1_3	PR18A	3	T	VREF1_3
109	PR13B	3	C		PR17B	3	C	
110	PR13A	3	T		PR17A	3	T	
111	PR12B	3	C		PR16B	3	C	
112	PR12A	3	T		PR16A	3	T	
113	PR11B	3	C		PR15B	3	C	
114	PR11A	3	T	RDQS11	PR15A	3	T	RDQS15
115	PR10B	3	C	RLM0_PLLC_FB_A	PR14B	3	C	RLM0_PLLC_FB_A
116	GND3	3			GND3	3		
117	PR10A	3	T	RLM0_PLLT_FB_A	PR14A	3	T	RLM0_PLLT_FB_A
118	PR9B	3	C	RLM0_PLLC_IN_A	PR13B	3	C	RLM0_PLLC_IN_A
119	PR9A	3	T	RLM0_PLLT_IN_A	PR13A	3	T	RLM0_PLLT_IN_A
120	VCCIO3	3			VCCIO3	3		
121	PR8B	3	C	DI/CSSPIN	PR12B	3	C	DI/CSSPIN
122	PR8A	3	T	DOUT/CSON	PR12A	3	T	DOUT/CSON
123	PR7B	3	C	BUSY/SISPI	PR11B	3	C	BUSY/SISPI
124	PR7A	3	T	D7/SPID0	PR11A	3	T	D7/SPID0
125	CFG2	3			CFG2	3		
126	CFG1	3			CFG1	3		

LFEC1, LFEC3 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFEC1				LFEC3			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
127	CFG0	3			CFG0	3		
128	VCC	-			VCC	-		
129	PROGRAMN	3			PROGRAMN	3		
130	CCLK	3			CCLK	3		
131	INITN	3			INITN	3		
132	GND	-			GND	-		
133	DONE	3			DONE	3		
134	GND	-			GND	-		
135	VCC	-			VCC	-		
136	NC	-			VCCAUX	-		
137	PR5B	2	C	PCLKC2_0	PR9B	2	C	PCLKC2_0
138	NC	-			GND2	2		
139	PR5A	2	T	PCLKT2_0	PR9A	2	T	PCLKT2_0
140	PR4B	2	C		PR8B	2	C	
141	PR4A	2	T		PR8A	2	T	
142	PR3B	2	C		PR7B	2	C	
143	PR3A	2	T		PR7A	2	T	
144	NC	-			PR6B	2	C	
145	NC	-			VCCIO2	2		
146	NC	-			PR6A	2	T	RDQS6
147	NC	-			PR5B	2	C	
148	NC	-			PR5A	2	T	
149	NC	-			PR4B	2	C	
150	NC	-			PR4A	2	T	
151	NC	-			NC	-		
152	NC	-			NC	-		
153	PR2B	2	C	VREF1_2	PR2B	2	C	VREF1_2
154	PR2A	2	T	VREF2_2	PR2A	2	T	VREF2_2
155	VCCIO2	2			VCCIO2	2		
156*	GND1 GND2	-			GND1 GND2	-		
157	VCCIO1	1			VCCIO1	1		
158	NC	-			NC	-		
159	PT17B	1	C		PT25B	1	C	
160	PT17A	1	T		PT25A	1	T	
161	PT16B	1	C		PT24B	1	C	
162	PT16A	1	T		PT24A	1	T	
163	PT15B	1	C		PT23B	1	C	
164	PT15A	1	T		PT23A	1	T	
165	PT14B	1	C		PT22B	1	C	
166	PT14A	1	T	TDQS14	PT22A	1	T	TDQS22
167	PT13B	1	C		PT21B	1	C	
168	GND1	1			GND1	1		

LFEC1, LFEC3 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFEC1				LFEC3			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
169	PT13A	1	T		PT21A	1	T	
170	PT12B	1	C		PT20B	1	C	
171	PT12A	1	T		PT20A	1	T	
172	PT11B	1	C	VREF2_1	PT19B	1	C	VREF2_1
173	PT11A	1	T	VREF1_1	PT19A	1	T	VREF1_1
174	PT10B	1	C		PT18B	1	C	
175	PT10A	1	T		PT18A	1	T	
176	VCCIO1	1			VCCIO1	1		
177	VCCAUX	-			VCCAUX	-		
178	PT9B	0	C	PCLKC0_0	PT17B	0	C	PCLKC0_0
179	GND0	0			GND0	0		
180	PT9A	0	T	PCLKT0_0	PT17A	0	T	PCLKT0_0
181	PT8B	0	C	VREF1_0	PT16B	0	C	VREF1_0
182	PT8A	0	T	VREF2_0	PT16A	0	T	VREF2_0
183	PT7B	0	C		PT15B	0	C	
184	PT7A	0	T		PT15A	0	T	
185	PT6B	0	C		PT14B	0	C	
186	PT6A	0	T	TDQS6	PT14A	0	T	TDQS14
187	VCCIO0	0			VCCIO0	0		
188	PT5B	0	C		PT13B	0	C	
189	NC	-			GND0	0		
190	PT5A	0	T		PT13A	0	T	
191	PT4B	0	C		PT12B	0	C	
192	PT4A	0	T		PT12A	0	T	
193	PT3B	0	C		PT11B	0	C	
194	PT3A	0	T		PT11A	0	T	
195	PT2B	0	C		PT10B	0	C	
196	PT2A	0	T		PT10A	0	T	
197	NC	-			VCCIO0	0		
198	NC	-			PT6B	0	C	
199	NC	-			PT6A	0	T	TDQS6
200	NC	-			PT5B	0	C	
201	NC	-			PT5A	0	T	
202	NC	-			PT4B	0	C	
203	NC	-			PT4A	0	T	
204	NC	-			PT3B	0	C	
205	NC	-			PT3A	0	T	
206	NC	-			PT2B	0	C	
207	NC	-			PT2A	0	T	
208	VCCIO0	0			VCCIO0	0		

* Double bonded to the pin.

LFECP/EC6, LFECP/EC10 Logic Signal Connections: 208 PQFP

Pin Number	LFECP6/LFEC6				LFECP10/LFEC10			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
1*	GND0 GND7	-			GND0 GND7	-		
2	VCCIO7	7			VCCIO7	7		
3	PL2A	7	T	VREF2_7	PL2A	7	T	VREF2_7
4	PL2B	7	C	VREF1_7	PL2B	7	C	VREF1_7
5	NC	-			VCC	-		
6	NC	-			GND	-		
7	PL3B	7			PL12B	7		
8	PL4A	7	T		PL13A	7	T	
9	PL4B	7	C		PL13B	7	C	
10	PL5A	7	T		PL14A	7	T	
11	PL5B	7	C		PL14B	7	C	
12	PL6A	7	T	LDQS6	PL15A	7	T	LDQS15
13	VCCIO7	7			VCCIO7	7		
14	PL6B	7	C		PL15B	7	C	
15	PL7A	7	T		PL16A	7	T	
16	PL7B	7	C		PL16B	7	C	
17	PL8A	7	T		PL17A	7	T	
18	GND7	7			GND7	7		
19	PL8B	7	C		PL17B	7	C	
20	PL9A	7	T	PCLKT7_0	PL18A	7	T	PCLKT7_0
21	PL9B	7	C	PCLKC7_0	PL18B	7	C	PCLKC7_0
22	VCCAUX	-			VCCAUX	-		
23	XRES	6			XRES	6		
24	VCC	-			VCC	-		
25	GND	-			GND	-		
26	VCC	-			VCC	-		
27	TCK	6			TCK	6		
28	GND	-			GND	-		
29	TDI	6			TDI	6		
30	TMS	6			TMS	6		
31	TDO	6			TDO	6		
32	VCCJ	6			VCCJ	6		
33	PL20A	6	T	LLM0_PLLT_IN_A	PL29A	6	T	LLM0_PLLT_IN_A
34	PL20B	6	C	LLM0_PLLC_IN_A	PL29B	6	C	LLM0_PLLC_IN_A
35	PL21A	6	T	LLM0_PLLT_FB_A	PL30A	6	T	LLM0_PLLT_FB_A
36	PL21B	6	C	LLM0_PLLC_FB_A	PL30B	6	C	LLM0_PLLC_FB_A
37	VCCIO6	6			VCCIO6	6		
38	PL22A	6	T		PL31A	6	T	
39	PL22B	6	C		PL31B	6	C	
40	PL23A	6	T		PL32A	6	T	
41	GND6	6			GND6	6		
42	PL23B	6	C		PL32B	6	C	

LFECP/EC6, LFECP/EC10 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFECP6/LFEC6				LFECP10/LFEC10			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
43	PL24A	6	T	LDQS24	PL33A	6	T	LDQS33
44	PL24B	6	C		PL33B	6	C	
45	PL25A	6	T		PL34A	6	T	
46	PL25B	6	C		PL34B	6	C	
47	PL26A	6	T		PL35A	6	T	
48	PL26B	6	C		PL35B	6	C	
49	PL27A	6	T	VREF1_6	PL36A	6	T	VREF1_6
50	PL27B	6	C	VREF2_6	PL36B	6	C	VREF2_6
51	VCCIO6	6			VCCIO6	6		
52*	GND5 GND6	-			GND5 GND6	-		
53	VCCIO5	5			VCCIO5	5		
54	PB2A	5	T		PB2A	5	T	
55	PB2B	5	C		PB2B	5	C	
56	PB3A	5	T		PB3A	5	T	
57	PB3B	5	C		PB3B	5	C	
58	PB4A	5	T		PB4A	5	T	
59	PB4B	5	C		PB4B	5	C	
60	PB5A	5	T		PB5A	5	T	
61	PB5B	5	C		PB5B	5	C	
62	PB6A	5	T	BDQS6	PB6A	5	T	BDQS6
63	PB6B	5	C		PB6B	5	C	
64	VCCIO5	5			VCCIO5	5		
65	PB10A	5	T		PB18A	5	T	
66	PB10B	5	C		PB18B	5	C	
67	PB11A	5	T		PB19A	5	T	
68	PB11B	5	C		PB19B	5	C	
69	PB12A	5	T		PB20A	5	T	
70	PB12B	5	C		PB20B	5	C	
71	PB13A	5	T		PB21A	5	T	
72	GND5	5			GND5	5		
73	PB13B	5	C		PB21B	5	C	
74	VCCIO5	5			VCCIO5	5		
75	PB14A	5	T	BDQS14	PB22A	5	T	BDQS22
76	PB14B	5	C		PB22B	5	C	
77	PB15A	5	T		PB23A	5	T	
78	PB15B	5	C		PB23B	5	C	
79	PB16A	5	T	VREF2_5	PB24A	5	T	VREF2_5
80	PB16B	5	C	VREF1_5	PB24B	5	C	VREF1_5
81	PB17A	5	T	PCLKT5_0	PB25A	5	T	PCLKT5_0
82	GND5	5			GND5	5		
83	PB17B	5	C	PCLKC5_0	PB25B	5	C	PCLKC5_0
84	VCCAUX	-			VCCAUX	-		

LFECP/EC6, LFECP/EC10 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFECP6/LFEC6				LFECP10/LFEC10			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
85	VCCIO4	4			VCCIO4	4		
86	PB18A	4	T	WRITEN	PB26A	4	T	WRITEN
87	PB18B	4	C	CS1N	PB26B	4	C	CS1N
88	PB19A	4	T	VREF1_4	PB27A	4	T	VREF1_4
89	PB19B	4	C	CSN	PB27B	4	C	CSN
90	PB20A	4	T	VREF2_4	PB28A	4	T	VREF2_4
91	PB20B	4	C	D0/SPID7	PB28B	4	C	D0/SPID7
92	PB21A	4	T	D2/SPID5	PB29A	4	T	D2/SPID5
93	GND4	4			GND4	4		
94	PB21B	4	C	D1/SPID6	PB29B	4	C	D1/SPID6
95	PB22A	4	T	BDQS22	PB30A	4	T	BDQS30
96	PB22B	4	C	D3/SPID4	PB30B	4	C	D3/SPID4
97	PB23A	4	T		PB31A	4	T	
98	PB23B	4	C	D4/SPID3	PB31B	4	C	D4/SPID3
99	PB24A	4	T		PB32A	4	T	
100	PB24B	4	C	D5/SPID2	PB32B	4	C	D5/SPID2
101	PB25A	4	T		PB33A	4	T	
102	PB25B	4	C	D6/SPID1	PB33B	4	C	D6/SPID1
103	PB33A	4			PB41A	4		
104	VCCIO4	4			VCCIO4	4		
105*	GND3 GND4	-			GND3 GND4	-		
106	VCCIO3	3			VCCIO3	3		
107	PR27B	3	C	VREF2_3	PR36B	3	C	VREF2_3
108	PR27A	3	T	VREF1_3	PR36A	3	T	VREF1_3
109	PR26B	3	C		PR35B	3	C	
110	PR26A	3	T		PR35A	3	T	
111	PR25B	3	C		PR34B	3	C	
112	PR25A	3	T		PR34A	3	T	
113	PR24B	3	C		PR33B	3	C	
114	PR24A	3	T	RDQS24	PR33A	3	T	RDQS33
115	PR23B	3	C	RLM0_PLLC_FB_A	PR32B	3	C	RLM0_PLLC_FB_A
116	GND3	3			GND3	3		
117	PR23A	3	T	RLM0_PLLT_FB_A	PR32A	3	T	RLM0_PLLT_FB_A
118	PR22B	3	C	RLM0_PLLC_IN_A	PR31B	3	C	RLM0_PLLC_IN_A
119	PR22A	3	T	RLM0_PLLT_IN_A	PR31A	3	T	RLM0_PLLT_IN_A
120	VCCIO3	3			VCCIO3	3		
121	PR21B	3	C	DI/CSSPIN	PR30B	3	C	DI/CSSPIN
122	PR21A	3	T	DOUT/CSON	PR30A	3	T	DOUT/CSON
123	PR20B	3	C	BUSY/SISPI	PR29B	3	C	BUSY/SISPI
124	PR20A	3	T	D7/SPID0	PR29A	3	T	D7/SPID0
125	CFG2	3			CFG2	3		
126	CFG1	3			CFG1	3		

LFECP/EC6, LFECP/EC10 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFECP6/LFEC6				LFECP10/LFEC10			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
127	CFG0	3			CFG0	3		
128	VCC	-			VCC	-		
129	PROGRAMN	3			PROGRAMN	3		
130	CCLK	3			CCLK	3		
131	INITN	3			INITN	3		
132	GND	-			GND	-		
133	DONE	3			DONE	3		
134	GND	-			GND	-		
135	VCC	-			VCC	-		
136	VCCAUX	-			VCCAUX	-		
137	PR9B	2	C	PCLKC2_0	PR18B	2	C	PCLKC2_0
138	GND2	2			GND2	2		
139	PR9A	2	T	PCLKT2_0	PR18A	2	T	PCLKT2_0
140	PR8B	2	C		PR17B	2	C	
141	PR8A	2	T		PR17A	2	T	
142	PR7B	2	C		PR16B	2	C	
143	PR7A	2	T		PR16A	2	T	
144	PR6B	2	C		PR15B	2	C	
145	VCCIO2	2			VCCIO2	2		
146	PR6A	2	T	RDQS6	PR15A	2	T	RDQS15
147	PR5B	2	C		PR14B	2	C	
148	PR5A	2	T		PR14A	2	T	
149	PR4B	2	C		PR13B	2	C	
150	PR4A	2	T		PR13A	2	T	
151	NC	-			GND	-		
152	NC	-			VCC	-		
153	PR2B	2	C	VREF1_2	PR2B	2	C	VREF1_2
154	PR2A	2	T	VREF2_2	PR2A	2	T	VREF2_2
155	VCCIO2	2			VCCIO2	2		
156*	GND1 GND2	-			GND1 GND2	-		
157	VCCIO1	1			VCCIO1	1		
158	PT33A	1			PT41A	1		
159	PT25B	1	C		PT33B	1	C	
160	PT25A	1	T		PT33A	1	T	
161	PT24B	1	C		PT32B	1	C	
162	PT24A	1	T		PT32A	1	T	
163	PT23B	1	C		PT31B	1	C	
164	PT23A	1	T		PT31A	1	T	
165	PT22B	1	C		PT30B	1	C	
166	PT22A	1	T	TDQS22	PT30A	1	T	TDQS30
167	PT21B	1	C		PT29B	1	C	
168	GND1	1			GND1	1		

LFECP/EC6, LFECP/EC10 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFECP6/LFEC6				LFECP10/LFEC10			
	Pin Function	Bank	LVDS	Dual Function	Pin Function	Bank	LVDS	Dual Function
169	PT21A	1	T		PT29A	1	T	
170	PT20B	1	C		PT28B	1	C	
171	PT20A	1	T		PT28A	1	T	
172	PT19B	1	C	VREF2_1	PT27B	1	C	VREF2_1
173	PT19A	1	T	VREF1_1	PT27A	1	T	VREF1_1
174	PT18B	1	C		PT26B	1	C	
175	PT18A	1	T		PT26A	1	T	
176	VCCIO1	1			VCCIO1	1		
177	VCCAUX	-			VCCAUX	-		
178	PT17B	0	C	PCLKC0_0	PT25B	0	C	PCLKC0_0
179	GND0	0			GND0	0		
180	PT17A	0	T	PCLKT0_0	PT25A	0	T	PCLKT0_0
181	PT16B	0	C	VREF1_0	PT24B	0	C	VREF1_0
182	PT16A	0	T	VREF2_0	PT24A	0	T	VREF2_0
183	PT15B	0	C		PT23B	0	C	
184	PT15A	0	T		PT23A	0	T	
185	PT14B	0	C		PT22B	0	C	
186	PT14A	0	T	TDQS14	PT22A	0	T	TDQS22
187	VCCIO0	0			VCCIO0	0		
188	PT13B	0	C		PT21B	0	C	
189	GND0	0			GND0	0		
190	PT13A	0	T		PT21A	0	T	
191	PT12B	0	C		PT20B	0	C	
192	PT12A	0	T		PT20A	0	T	
193	PT11B	0	C		PT19B	0	C	
194	PT11A	0	T		PT19A	0	T	
195	PT10B	0	C		PT18B	0	C	
196	PT10A	0	T		PT18A	0	T	
197	VCCIO0	0			VCCIO0	0		
198	PT6B	0	C		PT6B	0	C	
199	PT6A	0	T	TDQS6	PT6A	0	T	TDQS6
200	PT5B	0	C		PT5B	0	C	
201	PT5A	0	T		PT5A	0	T	
202	PT4B	0	C		PT4B	0	C	
203	PT4A	0	T		PT4A	0	T	
204	PT3B	0	C		PT3B	0	C	
205	PT3A	0	T		PT3A	0	T	
206	PT2B	0	C		PT2B	0	C	
207	PT2A	0	T		PT2A	0	T	
208	VCCIO0	0			VCCIO0	0		

*Double bonded to the pin.

LFEC3 and LFECP/EC6 Logic Signal Connections: 256 fpBGA

Ball Number	LFEC3				LFECP6/LFEC6			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
GND	GND7	7			GND7	7		
D4	PL2A	7	T	VREF2_7	PL2A	7	T	VREF2_7
D3	PL2B	7	C	VREF1_7	PL2B	7	C	VREF1_7
C3	PL3A	7	T		PL3A	7	T	
C2	PL3B	7	C		PL3B	7	C	
B1	PL4A	7	T		PL4A	7	T	
C1	PL4B	7	C		PL4B	7	C	
E3	PL5A	7	T		PL5A	7	T	
E4	PL5B	7	C		PL5B	7	C	
F4	PL6A	7	T	LDQS6	PL6A	7	T	LDQS6
F5	PL6B	7	C		PL6B	7	C	
G4	PL7A	7	T		PL7A	7	T	
G3	PL7B	7	C		PL7B	7	C	
D2	PL8A	7	T		PL8A	7	T	
D1	PL8B	7	C		PL8B	7	C	
E1	PL9A	7	T	PCLKT7_0	PL9A	7	T	PCLKT7_0
GND	GND7	7			GND7	7		
E2	PL9B	7	C	PCLKC7_0	PL9B	7	C	PCLKC7_0
F3	XRES	6			XRES	6		
G5	NC	-			PL11A	6	T	
H5	NC	-			PL11B	6	C	
F2	NC	-			PL12A	6	T	
F1	NC	-			PL12B	6	C	
H4	NC	-			PL13A	6	T	
H3	NC	-			PL13B	6	C	
G2	NC	-			PL14A	6	T	
-	-	-			GND6	6		
G1	NC	-			PL14B	6	C	
J4	NC	-			PL15A	6	T	LDQS15
J3	NC	-			PL15B	6	C	
J5	NC	-			PL16A	6	T	
K5	NC	-			PL16B	6	C	
H2	NC	-			PL17A	6	T	
H1	NC	-			PL17B	6	C	
J2	NC	-			PL18A	6	T	
-	-	-			GND6	6		
J1	NC	-			PL18B	6	C	
K4	TCK	6			TCK	6		
K3	TDI	6			TDI	6		
L3	TMS	6			TMS	6		
L5	TDO	6			TDO	6		
L4	VCCJ	6			VCCJ	6		

LFEC3 and LFECP/EC6 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFEC3				LFECP6/LFEC6			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
K2	PL11A	6	T	LLM0_PLLT_IN_A	PL20A	6	T	LLM0_PLLT_IN_A
K1	PL11B	6	C	LLM0_PLLC_IN_A	PL20B	6	C	LLM0_PLLC_IN_A
L2	PL12A	6	T	LLM0_PLLT_FB_A	PL21A	6	T	LLM0_PLLT_FB_A
L1	PL12B	6	C	LLM0_PLLC_FB_A	PL21B	6	C	LLM0_PLLC_FB_A
M2	PL13A	6	T		PL22A	6	T	
M1	PL13B	6	C		PL22B	6	C	
N1	PL14A	6	T		PL23A	6	T	
GND	GND6	6			GND6	6		
N2	PL14B	6	C		PL23B	6	C	
M4	PL15A	6	T	LDQS15	PL24A	6	T	LDQS24
M3	PL15B	6	C		PL24B	6	C	
P1	PL16A	6	T		PL25A	6	T	
R1	PL16B	6	C		PL25B	6	C	
P2	PL17A	6	T		PL26A	6	T	
P3	PL17B	6	C		PL26B	6	C	
N3	PL18A	6	T	VREF1_6	PL27A	6	T	VREF1_6
N4	PL18B	6	C	VREF2_6	PL27B	6	C	VREF2_6
GND	GND6	6			GND6	6		
GND	GND5	5			GND5	5		
P4	PB2A	5	T		PB2A	5	T	
N5	PB2B	5	C		PB2B	5	C	
P5	PB3A	5	T		PB3A	5	T	
P6	PB3B	5	C		PB3B	5	C	
R4	PB4A	5	T		PB4A	5	T	
R3	PB4B	5	C		PB4B	5	C	
T2	PB5A	5	T		PB5A	5	T	
T3	PB5B	5	C		PB5B	5	C	
R5	PB6A	5	T	BDQS6	PB6A	5	T	BDQS6
R6	PB6B	5	C		PB6B	5	C	
T4	PB7A	5	T		PB7A	5	T	
T5	PB7B	5	C		PB7B	5	C	
N6	PB8A	5	T		PB8A	5	T	
M6	PB8B	5	C		PB8B	5	C	
T6	PB9A	5	T		PB9A	5	T	
GND	GND5	5			GND5	5		
T7	PB9B	5	C		PB9B	5	C	
P7	PB10A	5	T		PB10A	5	T	
N7	PB10B	5	C		PB10B	5	C	
R7	PB11A	5	T		PB11A	5	T	
R8	PB11B	5	C		PB11B	5	C	
M7	PB12A	5	T		PB12A	5	T	
M8	PB12B	5	C		PB12B	5	C	
T8	PB13A	5	T		PB13A	5	T	

LFEC3 and LFECP/EC6 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFEC3				LFECP6/LFEC6			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
GND	GND5	5			GND5	5		
T9	PB13B	5	C		PB13B	5	C	
P8	PB14A	5	T	BDQS14	PB14A	5	T	BDQS14
N8	PB14B	5	C		PB14B	5	C	
R9	PB15A	5	T		PB15A	5	T	
R10	PB15B	5	C		PB15B	5	C	
P9	PB16A	5	T	VREF2_5	PB16A	5	T	VREF2_5
N9	PB16B	5	C	VREF1_5	PB16B	5	C	VREF1_5
T10	PB17A	5	T	PCLKT5_0	PB17A	5	T	PCLKT5_0
GND	GND5	5			GND5	5		
T11	PB17B	5	C	PCLKC5_0	PB17B	5	C	PCLKC5_0
T12	PB18A	4	T	WRITEN	PB18A	4	T	WRITEN
T13	PB18B	4	C	CS1N	PB18B	4	C	CS1N
P10	PB19A	4	T	VREF1_4	PB19A	4	T	VREF1_4
N10	PB19B	4	C	CSN	PB19B	4	C	CSN
T14	PB20A	4	T	VREF2_4	PB20A	4	T	VREF2_4
T15	PB20B	4	C	D0/SPID7	PB20B	4	C	D0/SPID7
M10	PB21A	4	T	D2/SPID5	PB21A	4	T	D2/SPID5
GND	GND4	4			GND4	4		
M11	PB21B	4	C	D1/SPID6	PB21B	4	C	D1/SPID6
R11	PB22A	4	T	BDQS22	PB22A	4	T	BDQS22
P11	PB22B	4	C	D3/SPID4	PB22B	4	C	D3/SPID4
R13	PB23A	4	T		PB23A	4	T	
R14	PB23B	4	C	D4/SPID3	PB23B	4	C	D4/SPID3
P12	PB24A	4	T		PB24A	4	T	
P13	PB24B	4	C	D5/SPID2	PB24B	4	C	D5/SPID2
N11	PB25A	4	T		PB25A	4	T	
-	-	-			GND4	4		
N12	PB25B	4	C	D6/SPID1	PB25B	4	C	D6/SPID1
R12	NC	-			PB26A	4		
GND	GND4	4			GND4	4		
-	-	-			GND4	4		
GND	GND3	3			GND3	3		
N13	PR18B	3	C	VREF2_3	PR27B	3	C	VREF2_3
N14	PR18A	3	T	VREF1_3	PR27A	3	T	VREF1_3
P14	PR17B	3	C		PR26B	3	C	
P15	PR17A	3	T		PR26A	3	T	
R15	PR16B	3	C		PR25B	3	C	
R16	PR16A	3	T		PR25A	3	T	
M13	PR15B	3	C		PR24B	3	C	
M14	PR15A	3	T	RDQS15	PR24A	3	T	RDQS24
P16	PR14B	3	C	RLM0_PLLC_FB_A	PR23B	3	C	RLM0_PLLC_FB_A
GND	GND3	3			GND3	3		

LFEC3 and LFECP/EC6 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFEC3				LFECP6/LFEC6			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
N16	PR14A	3	T	RLM0_PLLT_FB_A	PR23A	3	T	RLM0_PLLT_FB_A
N15	PR13B	3	C	RLM0_PLLC_IN_A	PR22B	3	C	RLM0_PLLC_IN_A
M15	PR13A	3	T	RLM0_PLLT_IN_A	PR22A	3	T	RLM0_PLLT_IN_A
M16	PR12B	3	C	DI/CSSPIN	PR21B	3	C	DI/CSSPIN
L16	PR12A	3	T	DOUT/CSON	PR21A	3	T	DOUT/CSON
K16	PR11B	3	C	BUSY/SISPI	PR20B	3	C	BUSY/SISPI
J16	PR11A	3	T	D7/SPID0	PR20A	3	T	D7/SPID0
L12	CFG2	3			CFG2	3		
L14	CFG1	3			CFG1	3		
L13	CFG0	3			CFG0	3		
K13	PROGRAMN	3			PROGRAMN	3		
L15	CCLK	3			CCLK	3		
K15	INITN	3			INITN	3		
K14	DONE	3			DONE	3		
	-	-			GND3	3		
H16	NC	-			PR18B	3	C	
H15	NC	-			PR18A	3	T	
G16	NC	-			PR17B	3	C	
G15	NC	-			PR17A	3	T	
K12	NC	-			PR16B	3	C	
J12	NC	-			PR16A	3	T	
J14	NC	-			PR15B	3	C	
J15	NC	-			PR15A	3	T	RDQS15
F16	NC	-			PR14B	3	C	
-	-	-			GND3	3		
F15	NC	-			PR14A	3	T	
J13	NC	-			PR13B	3	C	
H13	NC	-			PR13A	3	T	
H14	NC	-			PR12B	3	C	
G14	NC	-			PR12A	3	T	
E16	NC	-			PR11B	3	C	
E15	NC	-			PR11A	3	T	
H12	PR9B	2	C	PCLKC2_0	PR9B	2	C	PCLKC2_0
GND	GND2	2			GND2			
G12	PR9A	2	T	PCLKT2_0	PR9A	2	T	PCLKT2_0
G13	PR8B	2	C		PR8B	2	C	
F13	PR8A	2	T		PR8A	2	T	
F12	PR7B	2	C		PR7B	2	C	
E13	PR7A	2	T		PR7A	2	T	
D16	PR6B	2	C		PR6B	2	C	
D15	PR6A	2	T	RDQS6	PR6A	2	T	RDQS6
F14	PR5B	2	C		PR5B	2	C	
E14	PR5A	2	T		PR5A	2	T	

LFEC3 and LFECP/EC6 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFEC3				LFECP6/LFEC6			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
C16	PR4B	2	C		PR4B	2	C	
B16	PR4A	2	T		PR4A	2	T	
C15	PR3B	2	C		PR3B	2	C	
C14	PR3A	2	T		PR3A	2	T	
D14	PR2B	2	C	VREF1_2	PR2B	2	C	VREF1_2
D13	PR2A	2	T	VREF2_2	PR2A	2	T	VREF2_2
GND	GND2	2			GND2	2		
GND	GND1	1			GND1	1		
-	-	-			GND1	1		
B13	NC	-			PT26B	1	C	
C13	NC	-			PT26A	1	T	
C12	PT25B	1	C		PT25B	1	C	
-	-	-			GND1	1		
D12	PT25A	1	T		PT25A	1	T	
A15	PT24B	1	C		PT24B	1	C	
B14	PT24A	1	T		PT24A	1	T	
D11	PT23B	1	C		PT23B	1	C	
C11	PT23A	1	T		PT23A	1	T	
E10	PT22B	1	C		PT22B	1	C	
E11	PT22A	1	T	TDQS22	PT22A	1	T	TDQS22
A14	PT21B	1	C		PT21B	1	C	
GND	GND1	1			GND1	1		
A13	PT21A	1	T		PT21A	1	T	
D10	PT20B	1	C		PT20B	1	C	
C10	PT20A	1	T		PT20A	1	T	
A12	PT19B	1	C	VREF2_1	PT19B	1	C	VREF2_1
B12	PT19A	1	T	VREF1_1	PT19A	1	T	VREF1_1
A11	PT18B	1	C		PT18B	1	C	
B11	PT18A	1	T		PT18A	1	T	
A10	PT17B	0	C	PCLKC0_0	PT17B	0	C	PCLKC0_0
GND	GND0	0			GND0	0		
B10	PT17A	0	T	PCLKT0_0	PT17A	0	T	PCLKT0_0
C9	PT16B	0	C	VREF1_0	PT16B	0	C	VREF1_0
B9	PT16A	0	T	VREF2_0	PT16A	0	T	VREF2_0
E9	PT15B	0	C		PT15B	0	C	
D9	PT15A	0	T		PT15A	0	T	
D8	PT14B	0	C		PT14B	0	C	
C8	PT14A	0	T	TDQS14	PT14A	0	T	TDQS14
A9	PT13B	0	C		PT13B	0	C	
GND	GND0	0			GND0	0		
A8	PT13A	0	T		PT13A	0	T	
B8	PT12B	0	C		PT12B	0	C	
B7	PT12A	0	T		PT12A	0	T	

LFEC3 and LFECP/EC6 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFEC3				LFECP6/LFEC6			
	Ball Function	**Bank**	**LVDS**	**Dual Function**	**Ball Function**	**Bank**	**LVDS**	**Dual Function**
D7	PT11B	0	C		PT11B	0	C	
C7	PT11A	0	T		PT11A	0	T	
A7	PT10B	0	C		PT10B	0	C	
A6	PT10A	0	T		PT10A	0	T	
E7	PT9B	0	C		PT9B	0	C	
GND	GND0	0			GND0	0		
E6	PT9A	0	T		PT9A	0	T	
D6	PT8B	0	C		PT8B	0	C	
C6	PT8A	0	T		PT8A	0	T	
B6	PT7B	0	C		PT7B	0	C	
B5	PT7A	0	T		PT7A	0	T	
A5	PT6B	0	C		PT6B	0	C	
A4	PT6A	0	T	TDQS6	PT6A	0	T	TDQS6
A3	PT5B	0	C		PT5B	0	C	
A2	PT5A	0	T		PT5A	0	T	
B2	PT4B	0	C		PT4B	0	C	
B3	PT4A	0	T		PT4A	0	T	
D5	PT3B	0	C		PT3B	0	C	
C5	PT3A	0	T		PT3A	0	T	
C4	PT2B	0	C		PT2B	0	C	
B4	PT2A	0	T		PT2A	0	T	
GND	GND0	0			GND0	0		
A1	GND	-			GND	-		
A16	GND	-			GND	-		
G10	GND	-			GND	-		
G7	GND	-			GND	-		
G8	GND	-			GND	-		
G9	GND	-			GND	-		
H10	GND	-			GND	-		
H7	GND	-			GND	-		
H8	GND	-			GND	-		
H9	GND	-			GND	-		
J10	GND	-			GND	-		
J7	GND	-			GND	-		
J8	GND	-			GND	-		
J9	GND	-			GND	-		
K10	GND	-			GND	-		
K7	GND	-			GND	-		
K8	GND	-			GND	-		
K9	GND	-			GND	-		
T1	GND	-			GND	-		
T16	GND	-			GND	-		
E12	VCC	-			VCC	-		

LFEC3 and LFECP/EC6 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFEC3				LFECP6/LFEC6			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
E5	VCC	-			VCC	-		
E8	VCC	-			VCC	-		
M12	VCC	-			VCC	-		
M5	VCC	-			VCC	-		
M9	VCC	-			VCC	-		
B15	VCCAUX	-			VCCAUX	-		
R2	VCCAUX	-			VCCAUX	-		
F7	VCCIO0	0			VCCIO0	0		
F8	VCCIO0	0			VCCIO0	0		
F10	VCCIO1	1			VCCIO1	1		
F9	VCCIO1	1			VCCIO1	1		
G11	VCCIO2	2			VCCIO2	2		
H11	VCCIO2	2			VCCIO2	2		
J11	VCCIO3	3			VCCIO3	3		
K11	VCCIO3	3			VCCIO3	3		
L10	VCCIO4	4			VCCIO4	4		
L9	VCCIO4	4			VCCIO4	4		
L7	VCCIO5	5			VCCIO5	5		
L8	VCCIO5	5			VCCIO5	5		
J6	VCCIO6	6			VCCIO6	6		
K6	VCCIO6	6			VCCIO6	6		
G6	VCCIO7	7			VCCIO7	7		
H6	VCCIO7	7			VCCIO7	7		
F6	VCC	-			VCC	-		
F11	VCC	-			VCC	-		
L11	VCC	-			VCC	-		
L6	VCC	-			VCC	-		

LFECP/EC10 and LFECP/EC15 Logic Signal Connections: 256 fpBGA

Ball Number	LFECP10/LFEC10				LFECP15/LFEC15			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
GND	GND7	7			GND7	7		
D4	PL2A	7	T	VREF2_7	PL2A	7	T	VREF2_7
D3	PL2B	7	C	VREF1_7	PL2B	7	C	VREF1_7
GND	GND7	7			GND7	7		
C3	PL12A	7	T		PL16A	7	T	
C2	PL12B	7	C		PL16B	7	C	
B1	PL13A	7	T		PL17A	7	T	
C1	PL13B	7	C		PL17B	7	C	
E3	PL14A	7	T		PL18A	7	T	
GND	GND7	7			GND7	7		
-	-	-			GND7	7		
E4	PL14B	7	C		PL18B	7	C	
F4	PL15A	7	T	LDQS15	PL19A	7	T	LDQS19
F5	PL15B	7	C		PL19B	7	C	
G4	PL16A	7	T		PL20A	7	T	
G3	PL16B	7	C		PL20B	7	C	
D2	PL17A	7	T		PL21A	7	T	
D1	PL17B	7	C		PL21B	7	C	
E1	PL18A	7	T	PCLKT7_0	PL22A	7	T	PCLKT7_0
GND	GND7	7			GND7	7		
E2	PL18B	7	C	PCLKC7_0	PL22B	7	C	PCLKC7_0
F3	XRES	6			XRES	6		
G5	PL20A	6	T		PL24A	6	T	
H5	PL20B	6	C		PL24B	6	C	
F2	PL21A	6	T		PL25A	6	T	
F1	PL21B	6	C		PL25B	6	C	
H4	PL22A	6	T		PL26A	6	T	
H3	PL22B	6	C		PL26B	6	C	
G2	PL23A	6	T		PL27A	6	T	
GND	GND6	6			GND6	6		
G1	PL23B	6	C		PL27B	6	C	
J4	PL24A	6	T	LDQS24	PL28A	6	T	LDQS28
J3	PL24B	6	C		PL28B	6	C	
J5	PL25A	6	T		PL29A	6	T	
K5	PL25B	6	C		PL29B	6	C	
H2	PL26A	6	T		PL30A	6	T	
H1	PL26B	6	C		PL30B	6	C	
J2	PL27A	6	T		PL31A	6	T	
GND	GND6	6			GND6	6		
J1	PL27B	6	C		PL31B	6	C	
K4	TCK	6			TCK	6		
K3	TDI	6			TDI	6		

LFECP/EC10 and LFECP/EC15 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFECP10/LFEC10				LFECP15/LFEC15			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
L3	TMS	6			TMS	6		
L5	TDO	6			TDO	6		
L4	VCCJ	6			VCCJ	6		
K2	PL29A	6	T	LLM0_PLLT_IN_A	PL37A	6	T	LLM0_PLLT_IN_A
K1	PL29B	6	C	LLM0_PLLC_IN_A	PL37B	6	C	LLM0_PLLC_IN_A
L2	PL30A	6	T	LLM0_PLLT_FB_A	PL38A	6	T	LLM0_PLLT_FB_A
L1	PL30B	6	C	LLM0_PLLC_FB_A	PL38B	6	C	LLM0_PLLC_FB_A
M2	PL31A	6	T		PL39A	6	T	
M1	PL31B	6	C		PL39B	6	C	
N1	PL32A	6	T		PL40A	6	T	
GND	GND6	6			GND6	6		
-	-	-			GND6	6		
N2	PL32B	6	C		PL40B	6	C	
M4	PL33A	6	T	LDQS33	PL41A	6	T	LDQS41
M3	PL33B	6	C		PL41B	6	C	
P1	PL34A	6	T		PL42A	6	T	
R1	PL34B	6	C		PL42B	6	C	
P2	PL35A	6	T		PL43A	6	T	
P3	PL35B	6	C		PL43B	6	C	
N3	PL36A	6	T	VREF1_6	PL44A	6	T	VREF1_6
N4	PL36B	6	C	VREF2_6	PL44B	6	C	VREF2_6
GND	GND6	6			GND6	6		
GND	GND5	5			GND5	5		
GND	GND5	5			GND5	5		
P4	PB10A	5	T		PB10A	5	T	
N5	PB10B	5	C		PB10B	5	C	
P5	PB11A	5	T		PB11A	5	T	
P6	PB11B	5	C		PB11B	5	C	
R4	PB12A	5	T		PB12A	5	T	
R3	PB12B	5	C		PB12B	5	C	
T2	PB13A	5	T		PB13A	5	T	
GND	GND5	5			GND5	5		
T3	PB13B	5	C		PB13B	5	C	
R5	PB14A	5	T	BDQS14	PB14A	5	T	BDQS14
R6	PB14B	5	C		PB14B	5	C	
T4	PB15A	5	T		PB15A	5	T	
T5	PB15B	5	C		PB15B	5	C	
N6	PB16A	5	T		PB16A	5	T	
M6	PB16B	5	C		PB16B	5	C	
T6	PB17A	5	T		PB17A	5	T	
GND	GND5	5			GND5	5		
T7	PB17B	5	C		PB17B	5	C	
P7	PB18A	5	T		PB18A	5	T	

LFECP/EC10 and LFECP/EC15 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFECP10/LFEC10				LFECP15/LFEC15			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
N7	PB18B	5	C		PB18B	5	C	
R7	PB19A	5	T		PB19A	5	T	
R8	PB19B	5	C		PB19B	5	C	
M7	PB20A	5	T		PB20A	5	T	
M8	PB20B	5	C		PB20B	5	C	
T8	PB21A	5	T		PB21A	5	T	
GND	GND5	5			GND5	5		
T9	PB21B	5	C		PB21B	5	C	
P8	PB22A	5	T	BDQS22	PB22A	5	T	BDQS22
N8	PB22B	5	C		PB22B	5	C	
R9	PB23A	5	T		PB23A	5	T	
R10	PB23B	5	C		PB23B	5	C	
P9	PB24A	5	T	VREF2_5	PB24A	5	T	VREF2_5
N9	PB24B	5	C	VREF1_5	PB24B	5	C	VREF1_5
T10	PB25A	5	T	PCLKT5_0	PB25A	5	T	PCLKT5_0
GND	GND5	5			GND5	5		
T11	PB25B	5	C	PCLKC5_0	PB25B	5	C	PCLKC5_0
T12	PB26A	4	T	WRITEN	PB26A	4	T	WRITEN
T13	PB26B	4	C	CS1N	PB26B	4	C	CS1N
P10	PB27A	4	T	VREF1_4	PB27A	4	T	VREF1_4
N10	PB27B	4	C	CSN	PB27B	4	C	CSN
T14	PB28A	4	T	VREF2_4	PB28A	4	T	VREF2_4
T15	PB28B	4	C	D0/SPID7	PB28B	4	C	D0/SPID7
M10	PB29A	4	T	D2/SPID5	PB29A	4	T	D2/SPID5
GND	GND4	4			GND4	4		
M11	PB29B	4	C	D1/SPID6	PB29B	4	C	D1/SPID6
R11	PB30A	4	T	BDQS30	PB30A	4	T	BDQS30
P11	PB30B	4	C	D3/SPID4	PB30B	4	C	D3/SPID4
R13	PB31A	4	T		PB31A	4	T	
R14	PB31B	4	C	D4/SPID3	PB31B	4	C	D4/SPID3
P12	PB32A	4	T		PB32A	4	T	
P13	PB32B	4	C	D5/SPID2	PB32B	4	C	D5/SPID2
N11	PB33A	4	T		PB33A	4	T	
GND	GND4	4			GND4	4		
N12	PB33B	4	C	D6/SPID1	PB33B	4	C	D6/SPID1
R12	PB34A	4			PB34A	4		
GND	GND4	4			GND4	4		
GND	GND4	4			GND4	4		
-	-	-			GND4	4		
-	-	-			GND4	4		
GND	GND3	3			GND3	3		
N13	PR36B	3	C	VREF2_3	PR44B	3	C	VREF2_3
N14	PR36A	3	T	VREF1_3	PR44A	3	T	VREF1_3

LFECP/EC10 and LFECP/EC15 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFECP10/LFEC10				LFECP15/LFEC15			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
P14	PR35B	3	C		PR43B	3	C	
P15	PR35A	3	T		PR43A	3	T	
R15	PR34B	3	C		PR42B	3	C	
R16	PR34A	3	T		PR42A	3	T	
M13	PR33B	3	C		PR41B	3	C	
M14	PR33A	3	T	RDQS33	PR41A	3	T	RDQS41
P16	PR32B	3	C	RLM0_PLLC_FB_A	PR40B	3	C	RLM0_PLLC_FB_A
GND	GND3	3			GND3	3		
N16	PR32A	3	T	RLM0_PLLT_FB_A	PR40A	3	T	RLM0_PLLT_FB_A
N15	PR31B	3	C	RLM0_PLLC_IN_A	PR39B	3	C	RLM0_PLLC_IN_A
M15	PR31A	3	T	RLM0_PLLT_IN_A	PR39A	3	T	RLM0_PLLT_IN_A
M16	PR30B	3	C	DI/CSSPIN	PR38B	3	C	DI/CSSPIN
L16	PR30A	3	T	DOUT/CSON	PR38A	3	T	DOUT/CSON
K16	PR29B	3	C	BUSY/SISPI	PR37B	3	C	BUSY/SISPI
J16	PR29A	3	T	D7/SPID0	PR37A	3	T	D7/SPID0
L12	CFG2	3			CFG2	3		
L14	CFG1	3			CFG1	3		
L13	CFG0	3			CFG0	3		
K13	PROGRAMN	3			PROGRAMN	3		
L15	CCLK	3			CCLK	3		
K15	INITN	3			INITN	3		
K14	DONE	3			DONE	3		
GND	GND3	3			GND3	3		
H16	PR27B	3	C		PR31B	3	C	
-	-	-			GND3	3		
H15	PR27A	3	T		PR31A	3	T	
G16	PR26B	3	C		PR30B	3	C	
G15	PR26A	3	T		PR30A	3	T	
K12	PR25B	3	C		PR29B	3	C	
J12	PR25A	3	T		PR29A	3	T	
J14	PR24B	3	C		PR28B	3	C	
J15	PR24A	3	T	RDQS24	PR28A	3	T	RDQS28
F16	PR23B	3	C		PR27B	3	C	
GND	GND3	3			GND3	3		
F15	PR23A	3	T		PR27A	3	T	
J13	PR22B	3	C		PR26B	3	C	
H13	PR22A	3	T		PR26A	3	T	
H14	PR21B	3	C		PR25B	3	C	
G14	PR21A	3	T		PR25A	3	T	
E16	PR20B	3	C		PR24B	3	C	
E15	PR20A	3	T		PR24A	3	T	
H12	PR18B	2	C	PCLKC2_0	PR22B	2	C	PCLKC2_0
GND	GND2	2			GND2	2		

LFECP/EC10 and LFECP/EC15 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFECP10/LFEC10				LFECP15/LFEC15			
	Ball Function	**Bank**	**LVDS**	**Dual Function**	**Ball Function**	**Bank**	**LVDS**	**Dual Function**
G12	PR18A	2	T	PCLKT2_0	PR22A	2	T	PCLKT2_0
G13	PR17B	2	C		PR21B	2	C	
F13	PR17A	2	T		PR21A	2	T	
F12	PR16B	2	C		PR20B	2	C	
E13	PR16A	2	T		PR20A	2	T	
D16	PR15B	2	C		PR19B	2	C	
D15	PR15A	2	T		PR19A	2	T	RDQS19
F14	PR14B	2	C		PR18B	2	C	
GND	GND2	2			GND2	2		
E14	PR14A	2	T		PR18A	2	T	
C16	PR13B	2	C		PR17B	2	C	
B16	PR13A	2	T		PR17A	2	T	
C15	PR12B	2	C		PR16B	2	C	
C14	PR12A	2	T		PR16A	2	T	
GND	GND2	2			GND2	2		
-	-	-			GND2	2		
D14	PR2B	2	C	VREF1_2	PR2B	2	C	VREF1_2
D13	PR2A	2	T	VREF2_2	PR2A	2	T	VREF2_2
GND	GND2	2			GND2	2		
GND	GND1	1			GND1	1		
GND	GND1	1			GND1	1		
-	-	-			GND1	1		
-	-	-			GND1	1		
B13	PT34B	1	C		PT34B	1	C	
C13	PT34A	1	T		PT34A	1	T	
C12	PT33B	1	C		PT33B	1	C	
GND	GND1	1			GND1	1		
D12	PT33A	1	T		PT33A	1	T	
A15	PT32B	1	C		PT32B	1	C	
B14	PT32A	1	T		PT32A	1	T	
D11	PT31B	1	C		PT31B	1	C	
C11	PT31A	1	T		PT31A	1	T	
E10	PT30B	1	C		PT30B	1	C	
E11	PT30A	1	T	TDQS30	PT30A	1	T	TDQS30
A14	PT29B	1	C		PT29B	1	C	
GND	GND1	1			GND1	1		
A13	PT29A	1	T		PT29A	1	T	
D10	PT28B	1	C		PT28B	1	C	
C10	PT28A	1	T		PT28A	1	T	
A12	PT27B	1	C	VREF2_1	PT27B	1	C	VREF2_1
B12	PT27A	1	T	VREF1_1	PT27A	1	T	VREF1_1
A11	PT26B	1	C		PT26B	1	C	
B11	PT26A	1	T		PT26A	1	T	

LFECP/EC10 and LFECP/EC15 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFECP10/LFEC10				LFECP15/LFEC15			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
A10	PT25B	0	C	PCLKC0_0	PT25B	0	C	PCLKC0_0
GND	GND0	0			GND0	0		
B10	PT25A	0	T	PCLKT0_0	PT25A	0	T	PCLKT0_0
C9	PT24B	0	C	VREF1_0	PT24B	0	C	VREF1_0
B9	PT24A	0	T	VREF2_0	PT24A	0	T	VREF2_0
E9	PT23B	0	C		PT23B	0	C	
D9	PT23A	0	T		PT23A	0	T	
D8	PT22B	0	C		PT22B	0	C	
C8	PT22A	0	T	TDQS22	PT22A	0	T	TDQS22
A9	PT21B	0	C		PT21B	0	C	
GND	GND0	0			GND0	0		
A8	PT21A	0	T		PT21A	0	T	
B8	PT20B	0	C		PT20B	0	C	
B7	PT20A	0	T		PT20A	0	T	
D7	PT19B	0	C		PT19B	0	C	
C7	PT19A	0	T		PT19A	0	T	
A7	PT18B	0	C		PT18B	0	C	
A6	PT18A	0	T		PT18A	0	T	
E7	PT17B	0	C		PT17B	0	C	
GND	GND0	0			GND0	0		
E6	PT17A	0	T		PT17A	0	T	
D6	PT16B	0	C		PT16B	0	C	
C6	PT16A	0	T		PT16A	0	T	
B6	PT15B	0	C		PT15B	0	C	
B5	PT15A	0	T		PT15A	0	T	
A5	PT14B	0	C		PT14B	0	C	
A4	PT14A	0	T	TDQS14	PT14A	0	T	TDQS14
A3	PT13B	0	C		PT13B	0	C	
-	GND0	0			GND0	0		
A2	PT13A	0	T		PT13A	0	T	
B2	PT12B	0	C		PT12B	0	C	
B3	PT12A	0	T		PT12A	0	T	
D5	PT11B	0	C		PT11B	0	C	
C5	PT11A	0	T		PT11A	0	T	
C4	PT10B	0	C		PT10B	0	C	
B4	PT10A	0	T		PT10A	0	T	
GND	GND0	0			GND0	0		
GND	GND0	0			GND0	0		
A1	GND	-			GND	-		
A16	GND	-			GND	-		
G10	GND	-			GND	-		
G7	GND	-			GND	-		
G8	GND	-			GND	-		

LFECP/EC10 and LFECP/EC15 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFECP10/LFEC10				LFECP15/LFEC15			
	Ball Function	Bank	LVDS	Dual Function	Ball Function	Bank	LVDS	Dual Function
G9	GND	-			GND	-		
H10	GND	-			GND	-		
H7	GND	-			GND	-		
H8	GND	-			GND	-		
H9	GND	-			GND	-		
J10	GND	-			GND	-		
J7	GND	-			GND	-		
J8	GND	-			GND	-		
J9	GND	-			GND	-		
K10	GND	-			GND	-		
K7	GND	-			GND	-		
K8	GND	-			GND	-		
K9	GND	-			GND	-		
T1	GND	-			GND	-		
T16	GND	-			GND	-		
E12	VCC	-			VCC	-		
E5	VCC	-			VCC	-		
E8	VCC	-			VCC	-		
M12	VCC	-			VCC	-		
M5	VCC	-			VCC	-		
M9	VCC	-			VCC	-		
B15	VCCAUX	-			VCCAUX	-		
R2	VCCAUX	-			VCCAUX	-		
F7	VCCIO0	0			VCCIO0	0		
F8	VCCIO0	0			VCCIO0	0		
F10	VCCIO1	1			VCCIO1	1		
F9	VCCIO1	1			VCCIO1	1		
G11	VCCIO2	2			VCCIO2	2		
H11	VCCIO2	2			VCCIO2	2		
J11	VCCIO3	3			VCCIO3	3		
K11	VCCIO3	3			VCCIO3	3		
L10	VCCIO4	4			VCCIO4	4		
L9	VCCIO4	4			VCCIO4	4		
L7	VCCIO5	5			VCCIO5	5		
L8	VCCIO5	5			VCCIO5	5		
J6	VCCIO6	6			VCCIO6	6		
K6	VCCIO6	6			VCCIO6	6		
G6	VCCIO7	7			VCCIO7	7		
H6	VCCIO7	7			VCCIO7	7		
F6	VCC	-			VCC	-		
F11	VCC	-			VCC	-		
L11	VCC	-			VCC	-		
L6	VCC	-			VCC	-		

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
	LFECP6/LFEC6					**LFECP10/LFEC10**					**LFECP/LFEC15**			
GND	GND7	7			GND	GND7	7			GND	GND7	7		
D4	PL2A	7	T	VREF2_7	D4	PL2A	7	T	VREF2_7	D4	PL2A	7	T	VREF2_7
E4	PL2B	7	C	VREF1_7	E4	PL2B	7	C	VREF1_7	E4	PL2B	7	C	VREF1_7
C3	NC	-			C3	PL3A	7	T		C3	PL3A	7	T	
B2	NC	-			B2	PL3B	7	C		B2	PL3B	7	C	
E5	NC	-			E5	PL4A	7	T		E5	PL4A	7	T	
F5	NC	-			F5	PL4B	7	C		F5	PL4B	7	C	
D3	NC	-			D3	PL5A	7	T		D3	PL5A	7	T	
C2	NC	-			C2	PL5B	7	C		C2	PL5B	7	C	
F4	NC	-			F4	PL6A	7	T	LDQS6	F4	PL6A	7	T	LDQS6
G4	NC	-			G4	PL6B	7	C		G4	PL6B	7	C	
E3	NC	-			E3	PL7A	7	T		E3	PL7A	7	T	
D2	NC	-			D2	PL7B	7	C		D2	PL7B	7	C	
B1	NC	-			B1	PL8A	7	T	LUM0_PLLT_IN_A	B1	PL8A	7	T	LUM0_PLLT_IN_A
C1	NC	-			C1	PL8B	7	C	LUM0_PLLC_IN_A	C1	PL8B	7	C	LUM0_PLLC_IN_A
F3	NC	-			F3	PL9A	7	T	LUM0_PLLT_FB_A	F3	PL9A	7	T	LUM0_PLLT_FB_A
GND	-	-			GND	GND7	7			GND	GND7	7		
E2	NC	-			E2	PL9B	7	C	LUM0_PLLC_FB_A	E2	PL9B	7	C	LUM0_PLLC_FB_A
G5	NC	-			G5	NC	-			G5	PL11A	7	T	
H6	NC	-			H6	NC	-			H6	PL11B	7	C	
G3	NC	-			G3	NC	-			G3	PL12A	7	T	
H4	NC	-			H4	NC	-			H4	PL12B	7	C	
J5	NC	-			J5	NC	-			J5	PL13A	7	T	
H5	NC	-			H5	NC	-			H5	PL13B	7	C	
F2	NC	-			F2	NC	-			F2	PL14A	7	T	
GND	-	-			GND	-	-			GND	GND7	7		
F1	NC	-			F1	NC	-			F1	PL14B	7	C	
E1	NC	-			E1	PL11A	7	T		E1	PL15A	7	T	
D1	NC	-			D1	PL11B	7	C		D1	PL15B	7	C	
H3	PL3A	7	T		H3	PL12A	7	T		H3	PL16A	7	T	
G2	PL3B	7	C		G2	PL12B	7	C		G2	PL16B	7	C	
H2	PL4A	7	T		H2	PL13A	7	T		H2	PL17A	7	T	
G1	PL4B	7	C		G1	PL13B	7	C		G1	PL17B	7	C	
J4	PL5A	7	T		J4	PL14A	7	T		J4	PL18A	7	T	
GND	-	-			GND	GND7	7			GND	GND7	7		
J3	PL5B	7	C		J3	PL14B	7	C		J3	PL18B	7	C	
J2	PL6A	7	T	LDQS6	J2	PL15A	7	T	LDQS15	J2	PL19A	7	T	LDQS19
H1	PL6B	7	C		H1	PL15B	7	C		H1	PL19B	7	C	
K4	PL7A	7	T		K4	PL16A	7	T		K4	PL20A	7	T	
K5	PL7B	7	C		K5	PL16B	7	C		K5	PL20B	7	C	
K3	PL8A	7	T		K3	PL17A	7	T		K3	PL21A	7	T	
K2	PL8B	7	C		K2	PL17B	7	C		K2	PL21B	7	C	
J1	PL9A	7	T	PCLKT7_0	J1	PL18A	7	T	PCLKT7_0	J1	PL22A	7	T	PCLKT7_0
GND	GND7	7			GND	GND7	7			GND	GND7	7		
K1	PL9B	7	C	PCLKC7_0	K1	PL18B	7	C	PCLKC7_0	K1	PL22B	7	C	PCLKC7_0
L3	XRES	6			L3	XRES	6			L3	XRES	6		
L4	PL11A	6	T		L4	PL20A	6	T		L4	PL24A	6	T	
L5	PL11B	6	C		L5	PL20B	6	C		L5	PL24B	6	C	
L2	PL12A	6	T		L2	PL21A	6	T		L2	PL25A	6	T	
L1	PL12B	6	C		L1	PL21B	6	C		L1	PL25B	6	C	

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA (Cont.)

LFECP6/LFEC6					LFECP10/LFEC10					LFECP/LFEC15				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
M4	PL13A	6	T		M4	PL22A	6	T		M4	PL26A	6	T	
M5	PL13B	6	C		M5	PL22B	6	C		M5	PL26B	6	C	
M1	PL14A	6	T		M1	PL23A	6	T		M1	PL27A	6	T	
GND	GND6	6			GND	GND6	6			GND	GND6	6		
M2	PL14B	6	C		M2	PL23B	6	C		M2	PL27B	6	C	
N3	PL15A	6	T	LDQS15	N3	PL24A	6	T	LDQS24	N3	PL28A	6	T	LDQS28
M3	PL15B	6	C		M3	PL24B	6	C		M3	PL28B	6	C	
N5	PL16A	6	T		N5	PL25A	6	T		N5	PL29A	6	T	
N4	PL16B	6	C		N4	PL25B	6	C		N4	PL29B	6	C	
N1	PL17A	6	T		N1	PL26A	6	T		N1	PL30A	6	T	
N2	PL17B	6	C		N2	PL26B	6	C		N2	PL30B	6	C	
P1	PL18A	6	T		P1	PL27A	6	T		P1	PL31A	6	T	
GND	GND6	6			GND	GND6	6			GND	GND6	6		
P2	PL18B	6	C		P2	PL27B	6	C		P2	PL31B	6	C	
R6	NC	-			R6	NC	-			R6	PL32A	6	T	
P5	NC	-			P5	NC	-			P5	PL32B	6	C	
P3	NC	-			P3	NC	-			P3	PL33A	6	T	
P4	NC	-			P4	NC	-			P4	PL33B	6	C	
R1	NC	-			R1	NC	-			R1	PL34A	6	T	
R2	NC	-			R2	NC	-			R2	PL34B	6	C	
R5	NC	-			R5	NC	-			R5	PL35A	6	T	
GND	-	-			-	-	-			GND	GND6	6		
R4	NC	-			R4	NC	-			R4	PL35B	6	C	
T1	NC	-			T1	NC	-			T1	NC	-		
T2	NC	-			T2	NC	-			T2	NC	-		
R3	NC	-			R3	NC	-			R3	NC	-		
T3	NC	-			T3	NC	-			T3	NC	-		
T5	TCK	6			T5	TCK	6			T5	TCK	6		
U5	TDI	6			U5	TDI	6			U5	TDI	6		
T4	TMS	6			T4	TMS	6			T4	TMS	6		
U1	TDO	6			U1	TDO	6			U1	TDO	6		
U2	VCCJ	6			U2	VCCJ	6			U2	VCCJ	6		
V1	PL20A	6	T	LLM0_PLLT_IN_A	V1	PL29A	6	T	LLM0_PLLT_IN_A	V1	PL37A	6	T	LLM0_PLLT_IN_A
V2	PL20B	6	C	LLM0_PLLC_IN_A	V2	PL29B	6	C	LLM0_PLLC_IN_A	V2	PL37B	6	C	LLM0_PLLC_IN_A
U3	PL21A	6	T	LLM0_PLLT_FB_A	U3	PL30A	6	T	LLM0_PLLT_FB_A	U3	PL38A	6	T	LLM0_PLLT_FB_A
V3	PL21B	6	C	LLM0_PLLC_FB_A	V3	PL30B	6	C	LLM0_PLLC_FB_A	V3	PL38B	6	C	LLM0_PLLC_FB_A
U4	PL22A	6	T		U4	PL31A	6	T		U4	PL39A	6	T	
V5	PL22B	6	C		V5	PL31B	6	C		V5	PL39B	6	C	
W1	PL23A	6	T		W1	PL32A	6	T		W1	PL40A	6	T	
GND	GND6	6			GND	GND6	6			GND	GND6	6		
W2	PL23B	6	C		W2	PL32B	6	C		W2	PL40B	6	C	
Y1	PL24A	6	T	LDQS24	Y1	PL33A	6	T	LDQS33	Y1	PL41A	6	T	LDQS41
Y2	PL24B	6	C		Y2	PL33B	6	C		Y2	PL41B	6	C	
AA1	PL25A	6	T		AA1	PL34A	6	T		AA1	PL42A	6	T	
AA2	PL25B	6	C		AA2	PL34B	6	C		AA2	PL42B	6	C	
W4	PL26A	6	T		W4	PL35A	6	T		W4	PL43A	6	T	
V4	PL26B	6	C		V4	PL35B	6	C		V4	PL43B	6	C	
W3	PL27A	6	T	VREF1_6	W3	PL36A	6	T	VREF1_6	W3	PL44A	6	T	VREF1_6
Y3	PL27B	6	C	VREF2_6	Y3	PL36B	6	C	VREF2_6	Y3	PL44B	6	C	VREF2_6
GND	GND6	6			GND	GND6	6			GND	GND6	6		

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA (Cont.)

LFECP6/LFEC6					LFECP10/LFEC10					LFECP/LFEC15				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
GND	GND5	5			GND	GND5	5			GND	GND5	5		
V7	NC	-			V7	PB2A	5	T		V7	PB2A	5	T	
T6	NC	-			T6	PB2B	5	C		T6	PB2B	5	C	
V8	NC	-			V8	PB3A	5	T		V8	PB3A	5	T	
U7	NC	-			U7	PB3B	5	C		U7	PB3B	5	C	
W5	NC	-			W5	PB4A	5	T		W5	PB4A	5	T	
U6	NC	-			U6	PB4B	5	C		U6	PB4B	5	C	
AA3	NC	-			AA3	PB5A	5	T		AA3	PB5A	5	T	
AB3	NC	-			AB3	PB5B	5	C		AB3	PB5B	5	C	
Y6	NC	-			Y6	PB6A	5	T	BDQS6	Y6	PB6A	5	T	BDQS6
V6	NC	-			V6	PB6B	5	C		V6	PB6B	5	C	
AA5	NC	-			AA5	PB7A	5	T		AA5	PB7A	5	T	
W6	NC	-			W6	PB7B	5	C		W6	PB7B	5	C	
Y5	NC	-			Y5	PB8A	5	T		Y5	PB8A	5	T	
Y4	NC	-			Y4	PB8B	5	C		Y4	PB8B	5	C	
AA4	NC	-			AA4	PB9A	5	T		AA4	PB9A	5	T	
GND	-	-			GND	GND5	5			GND	GND5	5		
AB4	NC	-			AB4	PB9B	5	C		AB4	PB9B	5	C	
Y7	PB2A	5	T		Y7	PB10A	5	T		Y7	PB10A	5	T	
W8	PB2B	5	C		W8	PB10B	5	C		W8	PB10B	5	C	
W7	PB3A	5	T		W7	PB11A	5	T		W7	PB11A	5	T	
U8	PB3B	5	C		U8	PB11B	5	C		U8	PB11B	5	C	
W9	PB4A	5	T		W9	PB12A	5	T		W9	PB12A	5	T	
U9	PB4B	5	C		U9	PB12B	5	C		U9	PB12B	5	C	
Y8	PB5A	5	T		Y8	PB13A	5	T		Y8	PB13A	5	T	
GND	-	-			GND	GND5	5			GND	GND5	5		
Y9	PB5B	5	C		Y9	PB13B	5	C		Y9	PB13B	5	C	
V9	PB6A	5	T	BDQS6	V9	PB14A	5	T	BDQS14	V9	PB14A	5	T	BDQS14
T9	PB6B	5	C		T9	PB14B	5	C		T9	PB14B	5	C	
W10	PB7A	5	T		W10	PB15A	5	T		W10	PB15A	5	T	
U10	PB7B	5	C		U10	PB15B	5	C		U10	PB15B	5	C	
V10	PB8A	5	T		V10	PB16A	5	T		V10	PB16A	5	T	
T10	PB8B	5	C		T10	PB16B	5	C		T10	PB16B	5	C	
AA6	PB9A	5	T		AA6	PB17A	5	T		AA6	PB17A	5	T	
GND	GND5	5			GND	GND5	5			GND	GND5	5		
AB5	PB9B	5	C		AB5	PB17B	5	C		AB5	PB17B	5	C	
AA8	PB10A	5	T		AA8	PB18A	5	T		AA8	PB18A	5	T	
AA7	PB10B	5	C		AA7	PB18B	5	C		AA7	PB18B	5	C	
AB6	PB11A	5	T		AB6	PB19A	5	T		AB6	PB19A	5	T	
AB7	PB11B	5	C		AB7	PB19B	5	C		AB7	PB19B	5	C	
Y10	PB12A	5	T		Y10	PB20A	5	T		Y10	PB20A	5	T	
W11	PB12B	5	C		W11	PB20B	5	C		W11	PB20B	5	C	
AB8	PB13A	5	T		AB8	PB21A	5	T		AB8	PB21A	5	T	
GND	GND5	5			GND	GND5	5			GND	GND5	5		
AB9	PB13B	5	C		AB9	PB21B	5	C		AB9	PB21B	5	C	
AA10	PB14A	5	T	BDQS14	AA10	PB22A	5	T	BDQS22	AA10	PB22A	5	T	BDQS22
AA9	PB14B	5	C		AA9	PB22B	5	C		AA9	PB22B	5	C	
Y11	PB15A	5	T		Y11	PB23A	5	T		Y11	PB23A	5	T	
AA11	PB15B	5	C		AA11	PB23B	5	C		AA11	PB23B	5	C	
V11	PB16A	5	T	VREF2_5	V11	PB24A	5	T	VREF2_5	V11	PB24A	5	T	VREF2_5

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA (Cont.)

LFECP6/LFEC6					LFECP10/LFEC10					LFECP/LFEC15				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
V12	PB16B	5	C	VREF1_5	V12	PB24B	5	C	VREF1_5	V12	PB24B	5	C	VREF1_5
AB10	PB17A	5	T	PCLKT5_0	AB10	PB25A	5	T	PCLKT5_0	AB10	PB25A	5	T	PCLKT5_0
GND	GND5	5			GND	GND5	5			GND	GND5	5		
AB11	PB17B	5	C	PCLKC5_0	AB11	PB25B	5	C	PCLKC5_0	AB11	PB25B	5	C	PCLKC5_0
Y12	PB18A	4	T	WRITEN	Y12	PB26A	4	T	WRITEN	Y12	PB26A	4	T	WRITEN
U11	PB18B	4	C	CS1N	U11	PB26B	4	C	CS1N	U11	PB26B	4	C	CS1N
W12	PB19A	4	T	VREF1_4	W12	PB27A	4	T	VREF1_4	W12	PB27A	4	T	VREF1_4
U12	PB19B	4	C	CSN	U12	PB27B	4	C	CSN	U12	PB27B	4	C	CSN
W13	PB20A	4	T	VREF2_4	W13	PB28A	4	T	VREF2_4	W13	PB28A	4	T	VREF2_4
U13	PB20B	4	C	D0/SPID7	U13	PB28B	4	C	D0/SPID7	U13	PB28B	4	C	D0/SPID7
AA12	PB21A	4	T	D2/SPID5	AA12	PB29A	4	T	D2/SPID5	AA12	PB29A	4	T	D2/SPID5
GND	GND4	4			GND	GND4	4			GND	GND4	4		
AB12	PB21B	4	C	D1/SPID6	AB12	PB29B	4	C	D1/SPID6	AB12	PB29B	4	C	D1/SPID6
T13	PB22A	4	T	BDQS22	T13	PB30A	4	T	BDQS30	T13	PB30A	4	T	BDQS30
V13	PB22B	4	C	D3/SPID4	V13	PB30B	4	C	D3/SPID4	V13	PB30B	4	C	D3/SPID4
W14	PB23A	4	T		W14	PB31A	4	T		W14	PB31A	4	T	
U14	PB23B	4	C	D4/SPID3	U14	PB31B	4	C	D4/SPID3	U14	PB31B	4	C	D4/SPID3
Y13	PB24A	4	T		Y13	PB32A	4	T		Y13	PB32A	4	T	
V14	PB24B	4	C	D5/SPID2	V14	PB32B	4	C	D5/SPID2	V14	PB32B	4	C	D5/SPID2
AA13	PB25A	4	T		AA13	PB33A	4	T		AA13	PB33A	4	T	
GND	GND4	4			GND	GND4	4			GND	GND4	4		
AB13	PB25B	4	C	D6/SPID1	AB13	PB33B	4	C	D6/SPID1	AB13	PB33B	4	C	D6/SPID1
AA14	PB26A	4	T		AA14	PB34A	4	T		AA14	PB34A	4	T	
Y14	PB26B	4	C		Y14	PB34B	4	C		Y14	PB34B	4	C	
Y15	PB27A	4	T		Y15	PB35A	4	T		Y15	PB35A	4	T	
W15	PB27B	4	C		W15	PB35B	4	C		W15	PB35B	4	C	
V15	PB28A	4	T		V15	PB36A	4	T		V15	PB36A	4	T	
T14	PB28B	4	C		T14	PB36B	4	C		T14	PB36B	4	C	
AB14	PB29A	4	T		AB14	PB37A	4	T		AB14	PB37A	4	T	
GND	GND4	4			GND	GND4	4			GND	GND4	4		
AB15	PB29B	4	C		AB15	PB37B	4	C		AB15	PB37B	4	C	
AB16	PB30A	4	T	BDQS30	AB16	PB38A	4	T	BDQS38	AB16	PB38A	4	T	BDQS38
AA15	PB30B	4	C		AA15	PB38B	4	C		AA15	PB38B	4	C	
AB17	PB31A	4	T		AB17	PB39A	4	T		AB17	PB39A	4	T	
AA16	PB31B	4	C		AA16	PB39B	4	C		AA16	PB39B	4	C	
AB18	PB32A	4	T		AB18	PB40A	4	T		AB18	PB40A	4	T	
AA17	PB32B	4	C		AA17	PB40B	4	C		AA17	PB40B	4	C	
AB19	PB33A	4	T		AB19	PB41A	4	T		AB19	PB41A	4	T	
GND	-	-			GND	-	-			GND	GND4	4		
AA18	PB33B	4	C		AA18	PB41B	4	C		AA18	PB41B	4	C	
W16	NC	-			W16	NC	-			W16	PB42A	4	T	
U15	NC	-			U15	NC	-			U15	PB42B	4	C	
V16	NC	-			V16	NC	-			V16	PB43A	4	T	
U16	NC	-			U16	NC	-			U16	PB43B	4	C	
Y17	NC	-			Y17	NC	-			Y17	PB44A	4	T	
V17	NC	-			V17	NC	-			V17	PB44B	4	C	
AB20	NC	-			AB20	NC	-			AB20	PB45A	4	T	
GND	-	-			GND	-	-			GND	GND4	4		
AA19	NC	-			AA19	NC	-			AA19	PB45B	4	C	
Y16	NC	-			Y16	NC	-			Y16	PB46A	4	T	BDQS46

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA (Cont.)

LFECP6/LFEC6					LFECP10/LFEC10					LFECP/LFEC15				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
W17	NC	-			W17	NC	-			W17	PB46B	4	C	
AA20	NC	-			AA20	NC	-			AA20	PB47A	4	T	
Y19	NC	-			Y19	NC	-			Y19	PB47B	4	C	
Y18	NC	-			Y18	NC	-			Y18	PB48A	4	T	
W18	NC	-			W18	NC	-			W18	PB48B	4	C	
T17	NC	-			T17	NC	-			T17	PB49A	4	T	
U17	NC	-			U17	NC	-			U17	PB49B	4	C	
GND	GND4	4			GND	GND4	4			GND	GND4	4		
GND	GND3	3			GND	GND3	3			GND	GND3	3		
W20	PR27B	3	C	VREF2_3	W20	PR36B	3	C	VREF2_3	W20	PR44B	3	C	VREF2_3
Y20	PR27A	3	T	VREF1_3	Y20	PR36A	3	T	VREF1_3	Y20	PR44A	3	T	VREF1_3
AA21	PR26B	3	C		AA21	PR35B	3	C		AA21	PR43B	3	C	
AB21	PR26A	3	T		AB21	PR35A	3	T		AB21	PR43A	3	T	
W19	PR25B	3	C		W19	PR34B	3	C		W19	PR42B	3	C	
V19	PR25A	3	T		V19	PR34A	3	T		V19	PR42A	3	T	
Y21	PR24B	3	C		Y21	PR33B	3	C		Y21	PR41B	3	C	
AA22	PR24A	3	T	RDQS24	AA22	PR33A	3	T	RDQS33	AA22	PR41A	3	T	RDQS41
V20	PR23B	3	C	RLM0_PLLC_FB_A	V20	PR32B	3	C	RLM0_PLLC_FB_A	V20	PR40B	3	C	RLM0_PLLC_FB_A
GND	GND3	3			GND	GND3	3			GND	GND3	3		
U20	PR23A	3	T	RLM0_PLLT_FB_A	U20	PR32A	3	T	RLM0_PLLT_FB_A	U20	PR40A	3	T	RLM0_PLLT_FB_A
W21	PR22B	3	C	RLM0_PLLC_IN_A	W21	PR31B	3	C	RLM0_PLLC_IN_A	W21	PR39B	3	C	RLM0_PLLC_IN_A
Y22	PR22A	3	T	RLM0_PLLT_IN_A	Y22	PR31A	3	T	RLM0_PLLT_IN_A	Y22	PR39A	3	T	RLM0_PLLT_IN_A
V21	PR21B	3	C	DI/CSSPIN	V21	PR30B	3	C	DI/CSSPIN	V21	PR38B	3	C	DI/CSSPIN
W22	PR21A	3	T	DOUT/CSON	W22	PR30A	3	T	DOUT/CSON	W22	PR38A	3	T	DOUT/CSON
U21	PR20B	3	C	BUSY/SISPI	U21	PR29B	3	C	BUSY/SISPI	U21	PR37B	3	C	BUSY/SISPI
V22	PR20A	3	T	D7/SPID0	V22	PR29A	3	T	D7/SPID0	V22	PR37A	3	T	D7/SPID0
T19	CFG2	3			T19	CFG2	3			T19	CFG2	3		
U19	CFG1	3			U19	CFG1	3			U19	CFG1	3		
U18	CFG0	3			U18	CFG0	3			U18	CFG0	3		
V18	PROGRAMN	3			V18	PROGRAMN	3			V18	PROGRAMN	3		
T20	CCLK	3			T20	CCLK	3			T20	CCLK	3		
T21	INITN	3			T21	INITN	3			T21	INITN	3		
R20	DONE	3			R20	DONE	3			R20	DONE	3		
T18	NC	-			T18	NC	-			T18	NC	-		
R17	NC	-			R17	NC	-			R17	NC	-		
R19	NC	-			R19	NC	-			R19	NC	-		
R18	NC	-			R18	NC	-			R18	NC	-		
U22	NC	-			U22	NC	-			U22	PR35B	3	C	
GND	-	-			GND	-	-			GND	GND3	3		
T22	NC	-			T22	NC	-			T22	PR35A	3	T	
R21	NC	-			R21	NC	-			R21	PR34B	3	C	
R22	NC	-			R22	NC	-			R22	PR34A	3	T	
P20	NC	-			P20	NC	-			P20	PR33B	3	C	
N20	NC	-			N20	NC	-			N20	PR33A	3	T	
P19	NC	-			P19	NC	-			P19	PR32B	3	C	
P18	NC	-			P18	NC	-			P18	PR32A	3	T	
P21	PR18B	3	C		P21	PR27B	3	C		P21	PR31B	3	C	
GND	GND3	3			GND	GND3	3			GND	GND3	3		
P22	PR18A	3	T		P22	PR27A	3	T		P22	PR31A	3	T	
N21	PR17B	3	C		N21	PR26B	3	C		N21	PR30B	3	C	

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA (Cont.)

LFECP6/LFEC6					LFECP10/LFEC10					LFECP/LFEC15				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
N22	PR17A	3	T		N22	PR26A	3	T		N22	PR30A	3	T	
N19	PR16B	3	C		N19	PR25B	3	C		N19	PR29B	3	C	
N18	PR16A	3	T		N18	PR25A	3	T		N18	PR29A	3	T	
M21	PR15B	3	C		M21	PR24B	3	C		M21	PR28B	3	C	
L20	PR15A	3	T	RDQS15	L20	PR24A	3	T	RDQS24	L20	PR28A	3	T	RDQS28
L21	PR14B	3	C		L21	PR23B	3	C		L21	PR27B	3	C	
GND	GND3	3			GND	GND3	3			GND	GND3	3		
M20	PR14A	3	T		M20	PR23A	3	T		M20	PR27A	3	T	
M18	PR13B	3	C		M18	PR22B	3	C		M18	PR26B	3	C	
M19	PR13A	3	T		M19	PR22A	3	T		M19	PR26A	3	T	
M22	PR12B	3	C		M22	PR21B	3	C		M22	PR25B	3	C	
L22	PR12A	3	T		L22	PR21A	3	T		L22	PR25A	3	T	
K22	PR11B	3	C		K22	PR20B	3	C		K22	PR24B	3	C	
K21	PR11A	3	T		K21	PR20A	3	T		K21	PR24A	3	T	
J22	PR9B	2	C	PCLKC2_0	J22	PR18B	2	C	PCLKC2_0	J22	PR22B	2	C	PCLKC2_0
GND	GND2	2			GND	GND2	2			GND	GND2	2		
J21	PR9A	2	T	PCLKT2_0	J21	PR18A	2	T	PCLKT2_0	J21	PR22A	2	T	PCLKT2_0
H22	PR8B	2	C		H22	PR17B	2	C		H22	PR21B	2	C	
H21	PR8A	2	T		H21	PR17A	2	T		H21	PR21A	2	T	
L19	PR7B	2	C		L19	PR16B	2	C		L19	PR20B	2	C	
L18	PR7A	2	T		L18	PR16A	2	T		L18	PR20A	2	T	
K20	PR6B	2	C		K20	PR15B	2	C		K20	PR19B	2	C	
J20	PR6A	2	T	RDQS6	J20	PR15A	2	T	RDQS15	J20	PR19A	2	T	RDQS19
K19	PR5B	2	C		K19	PR14B	2	C		K19	PR18B	2	C	
GND	-	-			GND	GND2	2			GND	GND2	2		
K18	PR5A	2	T		K18	PR14A	2	T		K18	PR18A	2	T	
G22	PR4B	2	C		G22	PR13B	2	C		G22	PR17B	2	C	
F22	PR4A	2	T		F22	PR13A	2	T		F22	PR17A	2	T	
F21	PR3B	2	C		F21	PR12B	2	C		F21	PR16B	2	C	
E22	PR3A	2	T		E22	PR12A	2	T		E22	PR16A	2	T	
E21	NC	-			E21	PR11B	2	C		E21	PR15B	2	C	
D22	NC	-			D22	PR11A	2	T		D22	PR15A	2	T	
G21	NC	-			G21	NC	-			G21	PR14B	2	C	
G20	NC	-			G20	NC	-			GND	GND2	2		
GND	-	-			-	-	-			G20	PR14A	2	T	
J18	NC	-			J18	NC	-			J18	PR13B	2	C	
H19	NC	-			H19	NC	-			H19	PR13A	2	T	
J19	NC	-			J19	NC	-			J19	PR12B	2	C	
H20	NC	-			H20	NC	-			H20	PR12A	2	T	
H17	NC	-			H17	NC	-			H17	PR11B	2	C	
H18	NC	-			H18	NC	-			H18	PR11A	2	T	
D21	NC	-			D21	PR9B	2	C	RUM0_PLLC_FB_A	D21	PR9B	2	C	RUM0_PLLC_FB_A
GND	-	-			GND	GND2	2			GND	GND2	2		
C22	NC	-			C22	PR9A	2	T	RUM0_PLLT_FB_A	C22	PR9A	2	T	RUM0_PLLT_FB_A
G19	NC	-			G19	PR8B	2	C	RUM0_PLLC_IN_A	G19	PR8B	2	C	RUM0_PLLC_IN_A
G18	NC	-			G18	PR8A	2	T	RUM0_PLLT_IN_A	G18	PR8A	2	T	RUM0_PLLT_IN_A
F20	NC	-			F20	PR7B	2	C		F20	PR7B	2	C	
F19	NC	-			F19	PR7A	2	T		F19	PR7A	2	T	
E20	NC	-			E20	PR6B	2	C		E20	PR6B	2	C	
D20	NC	-			D20	PR6A	2	T	RDQS6	D20	PR6A	2	T	RDQS6

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA (Cont.)

LFECP6/LFEC6					LFECP10/LFEC10					LFECP/LFEC15				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
C21	NC	-			C21	PR5B	2	C		C21	PR5B	2	C	
C20	NC	-			C20	PR5A	2	T		C20	PR5A	2	T	
F18	NC	-			F18	PR4B	2	C		F18	PR4B	2	C	
E18	NC	-			E18	PR4A	2	T		E18	PR4A	2	T	
B22	NC	-			B22	PR3B	2	C		B22	PR3B	2	C	
B21	NC	-			B21	PR3A	2	T		B21	PR3A	2	T	
E19	PR2B	2	C	VREF1_2	E19	PR2B	2	C	VREF1_2	E19	PR2B	2	C	VREF1_2
D19	PR2A	2	T	VREF2_2	D19	PR2A	2	T	VREF2_2	D19	PR2A	2	T	VREF2_2
GND	GND2	2			GND	GND2	2			GND	GND2	2		
GND	GND1	1			GND	GND1	1			GND	GND1	1		
G17	NC	-			G17	NC	-			G17	PT49B	1	C	
F17	NC	-			F17	NC	-			F17	PT49A	1	T	
D18	NC	-			D18	NC	-			D18	PT48B	1	C	
C18	NC	-			C18	NC	-			C18	PT48A	1	T	
C19	NC	-			C19	NC	-			C19	PT47B	1	C	
B20	NC	-			B20	NC	-			B20	PT47A	1	T	
D17	NC	-			D17	NC	-			D17	PT46B	1	C	
C16	NC	-			C16	NC	-			C16	PT46A	1	T	TDQS46
B19	NC	-			B19	NC	-			B19	PT45B	1	C	
GND	-	-			GND	-	-			GND	GND1	1		
A20	NC	-			A20	NC	-			A20	PT45A	1	T	
E17	NC	-			E17	NC	-			E17	PT44B	1	C	
C17	NC	-			C17	NC	-			C17	PT44A	1	T	
F16	NC	-			F16	NC	-			F16	PT43B	1	C	
E16	NC	-			E16	NC	-			E16	PT43A	1	T	
F15	NC	-			F15	NC	-			F15	PT42B	1	C	
D16	NC	-			D16	NC	-			D16	PT42A	1	T	
B18	PT33B	1	C		B18	PT41B	1	C		B18	PT41B	1	C	
GND	-	-			GND	-	-			GND	GND1	1		
A19	PT33A	1	T		A19	PT41A	1	T		A19	PT41A	1	T	
B17	PT32B	1	C		B17	PT40B	1	C		B17	PT40B	1	C	
A18	PT32A	1	T		A18	PT40A	1	T		A18	PT40A	1	T	
B16	PT31B	1	C		B16	PT39B	1	C		B16	PT39B	1	C	
A17	PT31A	1	T		A17	PT39A	1	T		A17	PT39A	1	T	
B15	PT30B	1	C		B15	PT38B	1	C		B15	PT38B	1	C	
A16	PT30A	1	T	TDQS30	A16	PT38A	1	T	TDQS38	A16	PT38A	1	T	TDQS38
A15	PT29B	1	C		A15	PT37B	1	C		A15	PT37B	1	C	
GND	GND1	1			GND	GND1	1			GND	GND1	1		
A14	PT29A	1	T		A14	PT37A	1	T		A14	PT37A	1	T	
G14	PT28B	1	C		G14	PT36B	1	C		G14	PT36B	1	C	
E15	PT28A	1	T		E15	PT36A	1	T		E15	PT36A	1	T	
D15	PT27B	1	C		D15	PT35B	1	C		D15	PT35B	1	C	
C15	PT27A	1	T		C15	PT35A	1	T		C15	PT35A	1	T	
C14	PT26B	1	C		C14	PT34B	1	C		C14	PT34B	1	C	
B14	PT26A	1	T		B14	PT34A	1	T		B14	PT34A	1	T	
A13	PT25B	1	C		A13	PT33B	1	C		A13	PT33B	1	C	
GND	GND1	1			GND	GND1	1			GND	GND1	1		
B13	PT25A	1	T		B13	PT33A	1	T		B13	PT33A	1	T	
E14	PT24B	1	C		E14	PT32B	1	C		E14	PT32B	1	C	
C13	PT24A	1	T		C13	PT32A	1	T		C13	PT32A	1	T	

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA (Cont.)

LFECP6/LFEC6					LFECP10/LFEC10					LFECP/LFEC15				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
F14	PT23B	1	C		F14	PT31B	1	C		F14	PT31B	1	C	
D14	PT23A	1	T		D14	PT31A	1	T		D14	PT31A	1	T	
E13	PT22B	1	C		E13	PT30B	1	C		E13	PT30B	1	C	
G13	PT22A	1	T	TDQS22	G13	PT30A	1	T	TDQS30	G13	PT30A	1	T	TDQS30
A12	PT21B	1	C		A12	PT29B	1	C		A12	PT29B	1	C	
GND	GND1	1			GND	GND1	1			GND	GND1	1		
B12	PT21A	1	T		B12	PT29A	1	T		B12	PT29A	1	T	
F13	PT20B	1	C		F13	PT28B	1	C		F13	PT28B	1	C	
D13	PT20A	1	T		D13	PT28A	1	T		D13	PT28A	1	T	
F12	PT19B	1	C	VREF2_1	F12	PT27B	1	C	VREF2_1	F12	PT27B	1	C	VREF2_1
D12	PT19A	1	T	VREF1_1	D12	PT27A	1	T	VREF1_1	D12	PT27A	1	T	VREF1_1
F11	PT18B	1	C		F11	PT26B	1	C		F11	PT26B	1	C	
C12	PT18A	1	T		C12	PT26A	1	T		C12	PT26A	1	T	
A11	PT17B	0	C	PCLKC0_0	A11	PT25B	0	C	PCLKC0_0	A11	PT25B	0	C	PCLKC0_0
GND	GND0	0			GND	GND0	0			GND	GND0	0		
A10	PT17A	0	T	PCLKT0_0	A10	PT25A	0	T	PCLKT0_0	A10	PT25A	0	T	PCLKT0_0
E12	PT16B	0	C	VREF1_0	E12	PT24B	0	C	VREF1_0	E12	PT24B	0	C	VREF1_0
E11	PT16A	0	T	VREF2_0	E11	PT24A	0	T	VREF2_0	E11	PT24A	0	T	VREF2_0
B11	PT15B	0	C		B11	PT23B	0	C		B11	PT23B	0	C	
C11	PT15A	0	T		C11	PT23A	0	T		C11	PT23A	0	T	
B9	PT14B	0	C		B9	PT22B	0	C		B9	PT22B	0	C	
B10	PT14A	0	T	TDQS14	B10	PT22A	0	T	TDQS22	B10	PT22A	0	T	TDQS22
A9	PT13B	0	C		A9	PT21B	0	C		A9	PT21B	0	C	
GND	GND0	0			GND	GND0	0			GND	GND0	0		
A8	PT13A	0	T		A8	PT21A	0	T		A8	PT21A	0	T	
D11	PT12B	0	C		D11	PT20B	0	C		D11	PT20B	0	C	
C10	PT12A	0	T		C10	PT20A	0	T		C10	PT20A	0	T	
A7	PT11B	0	C		A7	PT19B	0	C		A7	PT19B	0	C	
A6	PT11A	0	T		A6	PT19A	0	T		A6	PT19A	0	T	
B7	PT10B	0	C		B7	PT18B	0	C		B7	PT18B	0	C	
B8	PT10A	0	T		B8	PT18A	0	T		B8	PT18A	0	T	
A5	PT9B	0	C		A5	PT17B	0	C		A5	PT17B	0	C	
GND	GND0	0			GND	GND0	0			GND	GND0	0		
B6	PT9A	0	T		B6	PT17A	0	T		B6	PT17A	0	T	
G10	PT8B	0	C		G10	PT16B	0	C		G10	PT16B	0	C	
E10	PT8A	0	T		E10	PT16A	0	T		E10	PT16A	0	T	
F10	PT7B	0	C		F10	PT15B	0	C		F10	PT15B	0	C	
D10	PT7A	0	T		D10	PT15A	0	T		D10	PT15A	0	T	
G9	PT6B	0	C		G9	PT14B	0	C		G9	PT14B	0	C	
E9	PT6A	0	T	TDQS6	E9	PT14A	0	T	TDQS14	E9	PT14A	0	T	TDQS14
C9	PT5B	0	C		C9	PT13B	0	C		C9	PT13B	0	C	
GND	-	-			GND	GND0	0			GND	GND0	0		
C8	PT5A	0	T		C8	PT13A	0	T		C8	PT13A	0	T	
F9	PT4B	0	C		F9	PT12B	0	C		F9	PT12B	0	C	
D9	PT4A	0	T		D9	PT12A	0	T		D9	PT12A	0	T	
F8	PT3B	0	C		F8	PT11B	0	C		F8	PT11B	0	C	
D7	PT3A	0	T		D7	PT11A	0	T		D7	PT11A	0	T	
D8	PT2B	0	C		D8	PT10B	0	C		D8	PT10B	0	C	
C7	PT2A	0	T		C7	PT10A	0	T		C7	PT10A	0	T	
GND	GND0	0			GND	GND0	0			GND	GND0	0		

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
	LFECP6/LFEC6					LFECP10/LFEC10					LFECP/LFEC15			
A4	NC	-			A4	PT9B	0	C		A4	PT9B	0	C	
B4	NC	-			B4	PT9A	0	T		B4	PT9A	0	T	
C4	NC	-			C4	PT8B	0	C		C4	PT8B	0	C	
C5	NC	-			C5	PT8A	0	T		C5	PT8A	0	T	
D6	NC	-			D6	PT7B	0	C		D6	PT7B	0	C	
B5	NC	-			B5	PT7A	0	T		B5	PT7A	0	T	
E6	NC	-			E6	PT6B	0	C		E6	PT6B	0	C	
C6	NC	-			C6	PT6A	0	T	TDQS6	C6	PT6A	0	T	TDQS6
A3	NC	-			A3	PT5B	0	C		A3	PT5B	0	C	
B3	NC	-			B3	PT5A	0	T		B3	PT5A	0	T	
F6	NC	-			F6	PT4B	0	C		F6	PT4B	0	C	
D5	NC	-			D5	PT4A	0	T		D5	PT4A	0	T	
F7	NC	-			F7	PT3B	0	C		F7	PT3B	0	C	
E8	NC	-			E8	PT3A	0	T		E8	PT3A	0	T	
G6	NC	-			G6	PT2B	0	C		G6	PT2B	0	C	
E7	NC	-			E7	PT2A	0	T		E7	PT2A	0	T	
GND	-	-			GND	GND0	0			GND	GND0	0		
A1	GND	-			A1	GND	-			A1	GND	-		
A22	GND	-			A22	GND	-			A22	GND	-		
AB1	GND	-			AB1	GND	-			AB1	GND	-		
AB22	GND	-			AB22	GND	-			AB22	GND	-		
H15	GND	-			H15	GND	-			H15	GND	-		
H8	GND	-			H8	GND	-			H8	GND	-		
J10	GND	-			J10	GND	-			J10	GND	-		
J11	GND	-			J11	GND	-			J11	GND	-		
J12	GND	-			J12	GND	-			J12	GND	-		
J13	GND	-			J13	GND	-			J13	GND	-		
J14	GND	-			J14	GND	-			J14	GND	-		
J9	GND	-			J9	GND	-			J9	GND	-		
K10	GND	-			K10	GND	-			K10	GND	-		
K11	GND	-			K11	GND	-			K11	GND	-		
K12	GND	-			K12	GND	-			K12	GND	-		
K13	GND	-			K13	GND	-			K13	GND	-		
K14	GND	-			K14	GND	-			K14	GND	-		
K9	GND	-			K9	GND	-			K9	GND	-		
L10	GND	-			L10	GND	-			L10	GND	-		
L11	GND	-			L11	GND	-			L11	GND	-		
L12	GND	-			L12	GND	-			L12	GND	-		
L13	GND	-			L13	GND	-			L13	GND	-		
L14	GND	-			L14	GND	-			L14	GND	-		
L9	GND	-			L9	GND	-			L9	GND	-		
M10	GND	-			M10	GND	-			M10	GND	-		
M11	GND	-			M11	GND	-			M11	GND	-		
M12	GND	-			M12	GND	-			M12	GND	-		
M13	GND	-			M13	GND	-			M13	GND	-		
M14	GND	-			M14	GND	-			M14	GND	-		
M9	GND	-			M9	GND	-			M9	GND	-		
N10	GND	-			N10	GND	-			N10	GND	-		
N11	GND	-			N11	GND	-			N11	GND	-		
N12	GND	-			N12	GND	-			N12	GND	-		

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
	LFECP6/LFEC6					LFECP10/LFEC10					LFECP/LFEC15			
N13	GND	-			N13	GND	-			N13	GND	-		
N14	GND	-			N14	GND	-			N14	GND	-		
N9	GND	-			N9	GND	-			N9	GND	-		
P10	GND	-			P10	GND	-			P10	GND	-		
P11	GND	-			P11	GND	-			P11	GND	-		
P12	GND	-			P12	GND	-			P12	GND	-		
P13	GND	-			P13	GND	-			P13	GND	-		
P14	GND	-			P14	GND	-			P14	GND	-		
P9	GND	-			P9	GND	-			P9	GND	-		
R15	GND	-			R15	GND	-			R15	GND	-		
R8	GND	-			R8	GND	-			R8	GND	-		
J16	VCC	-			J16	VCC	-			J16	VCC	-		
J7	VCC	-			J7	VCC	-			J7	VCC	-		
K16	VCC	-			K16	VCC	-			K16	VCC	-		
K17	VCC	-			K17	VCC	-			K17	VCC	-		
K6	VCC	-			K6	VCC	-			K6	VCC	-		
K7	VCC	-			K7	VCC	-			K7	VCC	-		
L17	VCC	-			L17	VCC	-			L17	VCC	-		
L6	VCC	-			L6	VCC	-			L6	VCC	-		
M17	VCC	-			M17	VCC	-			M17	VCC	-		
M6	VCC	-			M6	VCC	-			M6	VCC	-		
N16	VCC	-			N16	VCC	-			N16	VCC	-		
N17	VCC	-			N17	VCC	-			N17	VCC	-		
N6	VCC	-			N6	VCC	-			N6	VCC	-		
N7	VCC	-			N7	VCC	-			N7	VCC	-		
P16	VCC	-			P16	VCC	-			P16	VCC	-		
P7	VCC	-			P7	VCC	-			P7	VCC	-		
G11	VCCIO0	0			G11	VCCIO0	0			G11	VCCIO0	0		
H10	VCCIO0	0			H10	VCCIO0	0			H10	VCCIO0	0		
H11	VCCIO0	0			H11	VCCIO0	0			H11	VCCIO0	0		
H9	VCCIO0	0			H9	VCCIO0	0			H9	VCCIO0	0		
G12	VCCIO1	1			G12	VCCIO1	1			G12	VCCIO1	1		
H12	VCCIO1	1			H12	VCCIO1	1			H12	VCCIO1	1		
H13	VCCIO1	1			H13	VCCIO1	1			H13	VCCIO1	1		
H14	VCCIO1	1			H14	VCCIO1	1			H14	VCCIO1	1		
J15	VCCIO2	2			J15	VCCIO2	2			J15	VCCIO2	2		
K15	VCCIO2	2			K15	VCCIO2	2			K15	VCCIO2	2		
L15	VCCIO2	2			L15	VCCIO2	2			L15	VCCIO2	2		
L16	VCCIO2	2			L16	VCCIO2	2			L16	VCCIO2	2		
M15	VCCIO3	3			M15	VCCIO3	3			M15	VCCIO3	3		
M16	VCCIO3	3			M16	VCCIO3	3			M16	VCCIO3	3		
N15	VCCIO3	3			N15	VCCIO3	3			N15	VCCIO3	3		
P15	VCCIO3	3			P15	VCCIO3	3			P15	VCCIO3	3		
R12	VCCIO4	4			R12	VCCIO4	4			R12	VCCIO4	4		
R13	VCCIO4	4			R13	VCCIO4	4			R13	VCCIO4	4		
R14	VCCIO4	4			R14	VCCIO4	4			R14	VCCIO4	4		
T12	VCCIO4	4			T12	VCCIO4	4			T12	VCCIO4	4		
R10	VCCIO5	5			R10	VCCIO5	5			R10	VCCIO5	5		
R11	VCCIO5	5			R11	VCCIO5	5			R11	VCCIO5	5		
R9	VCCIO5	5			R9	VCCIO5	5			R9	VCCIO5	5		

LFECP/EC6, LFECP/EC10, LFECP/EC15 Logic Signal Connections: 484 fpBGA (Cont.)

LFECP6/LFEC6					LFECP10/LFEC10					LFECP/LFEC15				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
T11	VCCIO5	5			T11	VCCIO5	5			T11	VCCIO5	5		
M7	VCCIO6	6			M7	VCCIO6	6			M7	VCCIO6	6		
M8	VCCIO6	6			M8	VCCIO6	6			M8	VCCIO6	6		
N8	VCCIO6	6			N8	VCCIO6	6			N8	VCCIO6	6		
P8	VCCIO6	6			P8	VCCIO6	6			P8	VCCIO6	6		
J8	VCCIO7	7			J8	VCCIO7	7			J8	VCCIO7	7		
K8	VCCIO7	7			K8	VCCIO7	7			K8	VCCIO7	7		
L7	VCCIO7	7			L7	VCCIO7	7			L7	VCCIO7	7		
L8	VCCIO7	7			L8	VCCIO7	7			L8	VCCIO7	7		
G15	VCCAUX	-			G15	VCCAUX	-			G15	VCCAUX	-		
G16	VCCAUX	-			G16	VCCAUX	-			G16	VCCAUX	-		
G7	VCCAUX	-			G7	VCCAUX	-			G7	VCCAUX	-		
G8	VCCAUX	-			G8	VCCAUX	-			G8	VCCAUX	-		
H16	VCCAUX	-			H16	VCCAUX	-			H16	VCCAUX	-		
H7	VCCAUX	-			H7	VCCAUX	-			H7	VCCAUX	-		
R16	VCCAUX	-			R16	VCCAUX	-			R16	VCCAUX	-		
R7	VCCAUX	-			R7	VCCAUX	-			R7	VCCAUX	-		
T15	VCCAUX	-			T15	VCCAUX	-			T15	VCCAUX	-		
T16	VCCAUX	-			T16	VCCAUX	-			T16	VCCAUX	-		
T7	VCCAUX	-			T7	VCCAUX	-			T7	VCCAUX	-		
T8	VCCAUX	-			T8	VCCAUX	-			T8	VCCAUX	-		
J6	VCC	-			J6	VCC	-			J6	VCC	-		
J17	VCC	-			J17	VCC	-			J17	VCC	-		
P6	VCC	-			P6	VCC	-			P6	VCC	-		
P17	VCC	-			P17	VCC	-			P17	VCC	-		
A2	NC	-			A2	NC	-			A2	NC	-		
AB2	NC	-			AB2	NC	-			AB2	NC	-		
A21	NC	-			A21	NC	-			A21	NC	-		

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

LFECP20/LFEC20					LFECP/LFEC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
GND	GND7	7			GND	GND7	7		
D4	PL2A	7	T	VREF2_7	D4	PL2A	7	T	VREF2_7
E4	PL2B	7	C	VREF1_7	E4	PL2B	7	C	VREF1_7
GND	-	-			GND	GND7	7		
C3	PL3A	7	T		C3	PL10A	7	T	
B2	PL3B	7	C		B2	PL10B	7	C	
E5	PL4A	7	T		E5	PL11A	7	T	
F5	PL4B	7	C		F5	PL11B	7	C	
D3	PL5A	7	T		D3	PL12A	7	T	
C2	PL5B	7	C		C2	PL12B	7	C	
GND	-	-			GND	GND7	7		
F4	PL6A	7	T	LDQS6	F4	PL14A	7	T	LDQS14
G4	PL6B	7	C		G4	PL14B	7	C	
E3	PL7A	7	T		E3	PL15A	7	T	
D2	PL7B	7	C		D2	PL15B	7	C	
B1	PL8A	7	T	LUM0_PLLT_IN_A	B1	PL16A	7	T	LUM0_PLLT_IN_A
C1	PL8B	7	C	LUM0_PLLC_IN_A	C1	PL16B	7	C	LUM0_PLLC_IN_A
F3	PL9A	7	T	LUM0_PLLT_FB_A	F3	PL17A	7	T	LUM0_PLLT_FB_A
GND	GND7	7			GND	GND7	7		
E2	PL9B	7	C	LUM0_PLLC_FB_A	E2	PL17B	7	C	LUM0_PLLC_FB_A
GND	-	-			GND	GND7	7		
G5	PL11A	7	T		G5	PL23A	7	T	LDQS23
H6	PL11B	7	C		H6	PL23B	7	C	
G3	PL12A	7	T		G3	PL24A	7	T	
H4	PL12B	7	C		H4	PL24B	7	C	
J5	PL13A	7	T		J5	PL25A	7	T	
H5	PL13B	7	C		H5	PL25B	7	C	
F2	PL14A	7	T		F2	PL26A	7	T	
GND	GND7	7			GND	GND7	7		
F1	PL14B	7	C		F1	PL26B	7	C	
E1	PL15A	7	T		E1	PL27A	7	T	
D1	PL15B	7	C		D1	PL27B	7	C	
H3	PL16A	7	T		H3	PL28A	7	T	
G2	PL16B	7	C		G2	PL28B	7	C	
H2	PL17A	7	T		H2	PL29A	7	T	
G1	PL17B	7	C		G1	PL29B	7	C	
J4	PL18A	7	T		J4	PL30A	7	T	
GND	GND7	7			GND	GND7	7		
J3	PL18B	7	C		J3	PL30B	7	C	
J2	PL19A	7	T	LDQS19	J2	PL31A	7	T	LDQS31
H1	PL19B	7	C		H1	PL31B	7	C	
K4	PL20A	7	T		K4	PL32A	7	T	
K5	PL20B	7	C		K5	PL32B	7	C	
K3	PL21A	7	T		K3	PL33A	7	T	

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

LFECP20/LFEC20					LFECP/LFEC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
K2	PL21B	7	C		K2	PL33B	7	C	
J1	PL22A	7	T	PCLKT7_0	J1	PL34A	7	T	PCLKT7_0
GND	GND7	7			GND	GND7	7		
K1	PL22B	7	C	PCLKC7_0	K1	PL34B	7	C	PCLKC7_0
L3	XRES	6			L3	XRES	6		
L4	PL24A	6	T		L4	PL36A	6	T	
L5	PL24B	6	C		L5	PL36B	6	C	
L2	PL25A	6	T		L2	PL37A	6	T	
L1	PL25B	6	C		L1	PL37B	6	C	
M4	PL26A	6	T		M4	PL38A	6	T	
M5	PL26B	6	C		M5	PL38B	6	C	
M1	PL27A	6	T		M1	PL39A	6	T	
GND	GND6	6			GND	GND6	6		
M2	PL27B	6	C		M2	PL39B	6	C	
N3	PL28A	6	T	LDQS28	N3	PL40A	6	T	LDQS40
M3	PL28B	6	C		M3	PL40B	6	C	
N5	PL29A	6	T		N5	PL41A	6	T	
N4	PL29B	6	C		N4	PL41B	6	C	
N1	PL30A	6	T		N1	PL42A	6	T	
N2	PL30B	6	C		N2	PL42B	6	C	
P1	PL31A	6	T		P1	PL43A	6	T	
GND	GND6	6			GND	GND6	6		
P2	PL31B	6	C		P2	PL43B	6	C	
R6	PL32A	6	T		R6	PL44A	6	T	
P5	PL32B	6	C		P5	PL44B	6	C	
P3	PL33A	6	T		P3	PL45A	6	T	
P4	PL33B	6	C		P4	PL45B	6	C	
R1	PL34A	6	T		R1	PL46A	6	T	
R2	PL34B	6	C		R2	PL46B	6	C	
R5	PL35A	6	T		R5	PL47A	6	T	
GND	GND6	6			GND	GND6	6		
R4	PL35B	6	C		R4	PL47B	6	C	
T1	PL36A	6	T	LDQS36	T1	PL48A	6	T	LDQS48
T2	PL36B	6	C		T2	PL48B	6	C	
R3	PL37A	6	T		R3	PL49A	6	T	
T3	PL37B	6	C		T3	PL49B	6	C	
GND	GND6	6			GND	GND6	6		
T5	TCK	6			T5	TCK	6		
U5	TDI	6			U5	TDI	6		
T4	TMS	6			T4	TMS	6		
U1	TDO	6			U1	TDO	6		
U2	VCCJ	6			U2	VCCJ	6		
V1	PL41A	6	T	LLM0_PLLT_IN_A	V1	PL53A	6	T	LLM0_PLLT_IN_A
V2	PL41B	6	C	LLM0_PLLC_IN_A	V2	PL53B	6	C	LLM0_PLLC_IN_A

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

	LFECP20/LFEC20					LFECP/LFEC33			
Ball Number	**Ball Function**	**Bank**	**LVDS**	**Dual Function**	**Ball Number**	**Ball Function**	**Bank**	**LVDS**	**Dual Function**
U3	PL42A	6	T	LLM0_PLLT_FB_A	U3	PL54A	6	T	LLM0_PLLT_FB_A
V3	PL42B	6	C	LLM0_PLLC_FB_A	V3	PL54B	6	C	LLM0_PLLC_FB_A
U4	PL43A	6	T		U4	PL55A	6	T	
V5	PL43B	6	C		V5	PL55B	6	C	
W1	PL44A	6	T		W1	PL56A	6	T	
GND	GND6	6			GND	GND6	6		
W2	PL44B	6	C		W2	PL56B	6	C	
Y1	PL45A	6	T	LDQS45	Y1	PL57A	6	T	LDQS57
Y2	PL45B	6	C		Y2	PL57B	6	C	
AA1	PL46A	6	T		AA1	PL58A	6	T	
AA2	PL46B	6	C		AA2	PL58B	6	C	
W4	PL47A	6	T		W4	PL59A	6	T	
V4	PL47B	6	C		V4	PL59B	6	C	
W3	PL48A	6	T	VREF1_6	W3	PL68A	6	T	VREF1_6
Y3	PL48B	6	C	VREF2_6	Y3	PL68B	6	C	VREF2_6
GND	GND6	6			GND	GND6	6		
GND	GND5	5			GND	GND6	6		
GND	-				GND	GND6	6		
GND	-				GND	GND5	5		
GND	GND5	5			GND	GND5	5		
V7	PB10A	5	T		V7	PB10A	5	T	
T6	PB10B	5	C		T6	PB10B	5	C	
V8	PB11A	5	T		V8	PB11A	5	T	
U7	PB11B	5	C		U7	PB11B	5	C	
W5	PB12A	5	T		W5	PB12A	5	T	
U6	PB12B	5	C		U6	PB12B	5	C	
AA3	PB13A	5	T		AA3	PB13A	5	T	
GND	GND5	5			GND	GND5	5		
AB3	PB13B	5	C		AB3	PB13B	5	C	
Y6	PB14A	5	T	BDQS14	Y6	PB14A	5	T	BDQS14
V6	PB14B	5	C		V6	PB14B	5	C	
AA5	PB15A	5	T		AA5	PB15A	5	T	
W6	PB15B	5	C		W6	PB15B	5	C	
Y5	PB16A	5	T		Y5	PB16A	5	T	
Y4	PB16B	5	C		Y4	PB16B	5	C	
AA4	PB17A	5	T		AA4	PB17A	5	T	
GND	GND5	5			GND	GND5	5		
AB4	PB17B	5	C		AB4	PB17B	5	C	
Y7	PB18A	5	T		Y7	PB18A	5	T	
W8	PB18B	5	C		W8	PB18B	5	C	
W7	PB19A	5	T		W7	PB19A	5	T	
U8	PB19B	5	C		U8	PB19B	5	C	
W9	PB20A	5	T		W9	PB20A	5	T	
U9	PB20B	5	C		U9	PB20B	5	C	

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

LFECP20/LFEC20					LFECP/LFEC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
Y8	PB21A	5	T		Y8	PB21A	5	T	
GND	GND5	5			GND	GND5	5		
Y9	PB21B	5	C		Y9	PB21B	5	C	
V9	PB22A	5	T	BDQS22	V9	PB22A	5	T	BDQS22
T9	PB22B	5	C		T9	PB22B	5	C	
W10	PB23A	5	T		W10	PB23A	5	T	
U10	PB23B	5	C		U10	PB23B	5	C	
V10	PB24A	5	T		V10	PB24A	5	T	
T10	PB24B	5	C		T10	PB24B	5	C	
AA6	PB25A	5	T		AA6	PB25A	5	T	
GND	GND5	5			GND	GND5	5		
AB5	PB25B	5	C		AB5	PB25B	5	C	
AA8	PB26A	5	T		AA8	PB26A	5	T	
AA7	PB26B	5	C		AA7	PB26B	5	C	
AB6	PB27A	5	T		AB6	PB27A	5	T	
AB7	PB27B	5	C		AB7	PB27B	5	C	
Y10	PB28A	5	T		Y10	PB28A	5	T	
W11	PB28B	5	C		W11	PB28B	5	C	
AB8	PB29A	5	T		AB8	PB29A	5	T	
GND	GND5	5			GND	GND5	5		
AB9	PB29B	5	C		AB9	PB29B	5	C	
AA10	PB30A	5	T	BDQS30	AA10	PB30A	5	T	BDQS30
AA9	PB30B	5	C		AA9	PB30B	5	C	
Y11	PB31A	5	T		Y11	PB31A	5	T	
AA11	PB31B	5	C		AA11	PB31B	5	C	
V11	PB32A	5	T	VREF2_5	V11	PB32A	5	T	VREF2_5
V12	PB32B	5	C	VREF1_5	V12	PB32B	5	C	VREF1_5
AB10	PB33A	5	T	PCLKT5_0	AB10	PB33A	5	T	PCLKT5_0
GND	GND5	5			GND	GND5	5		
AB11	PB33B	5	C	PCLKC5_0	AB11	PB33B	5	C	PCLKC5_0
Y12	PB34A	4	T	WRITEN	Y12	PB34A	4	T	WRITEN
U11	PB34B	4	C	CS1N	U11	PB34B	4	C	CS1N
W12	PB35A	4	T	VREF1_4	W12	PB35A	4	T	VREF1_4
U12	PB35B	4	C	CSN	U12	PB35B	4	C	CSN
W13	PB36A	4	T	VREF2_4	W13	PB36A	4	T	VREF2_4
U13	PB36B	4	C	D0/SPID7	U13	PB36B	4	C	D0/SPID7
AA12	PB37A	4	T	D2/SPID5	AA12	PB37A	4	T	D2/SPID5
GND	GND4	4			GND	GND4	4		
AB12	PB37B	4	C	D1/SPID6	AB12	PB37B	4	C	D1/SPID6
T13	PB38A	4	T	BDQS38	T13	PB38A	4	T	BDQS38
V13	PB38B	4	C	D3/SPID4	V13	PB38B	4	C	D3/SPID4
W14	PB39A	4	T		W14	PB39A	4	T	
U14	PB39B	4	C	D4/SPID3	U14	PB39B	4	C	D4/SPID3
Y13	PB40A	4	T		Y13	PB40A	4	T	

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

	LFECP20/LFEC20					LFECP/LFEC33			
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
V14	PB40B	4	C	D5/SPID2	V14	PB40B	4	C	D5/SPID2
AA13	PB41A	4	T		AA13	PB41A	4	T	
GND	GND4	4			GND	GND4	4		
AB13	PB41B	4	C	D6/SPID1	AB13	PB41B	4	C	D6/SPID1
AA14	PB42A	4	T		AA14	PB42A	4	T	
Y14	PB42B	4	C		Y14	PB42B	4	C	
Y15	PB43A	4	T		Y15	PB43A	4	T	
W15	PB43B	4	C		W15	PB43B	4	C	
V15	PB44A	4	T		V15	PB44A	4	T	
T14	PB44B	4	C		T14	PB44B	4	C	
AB14	PB45A	4	T		AB14	PB45A	4	T	
GND	GND4	4			GND	GND4	4		
AB15	PB45B	4	C		AB15	PB45B	4	C	
AB16	PB46A	4	T	BDQS46	AB16	PB46A	4	T	BDQS46
AA15	PB46B	4	C		AA15	PB46B	4	C	
AB17	PB47A	4	T		AB17	PB47A	4	T	
AA16	PB47B	4	C		AA16	PB47B	4	C	
AB18	PB48A	4	T		AB18	PB48A	4	T	
AA17	PB48B	4	C		AA17	PB48B	4	C	
AB19	PB49A	4	T		AB19	PB49A	4	T	
GND	GND4	4			GND	GND4	4		
AA18	PB49B	4	C		AA18	PB49B	4	C	
W16	PB50A	4	T		W16	PB50A	4	T	
U15	PB50B	4	C		U15	PB50B	4	C	
V16	PB51A	4	T		V16	PB51A	4	T	
U16	PB51B	4	C		U16	PB51B	4	C	
Y17	PB52A	4	T		Y17	PB52A	4	T	
V17	PB52B	4	C		V17	PB52B	4	C	
AB20	PB53A	4	T		AB20	PB53A	4	T	
GND	GND4	4			GND	GND4	4		
AA19	PB53B	4	C		AA19	PB53B	4	C	
Y16	PB54A	4	T	BDQS54	Y16	PB54A	4	T	BDQS54
W17	PB54B	4	C		W17	PB54B	4	C	
AA20	PB55A	4	T		AA20	PB55A	4	T	
Y19	PB55B	4	C		Y19	PB55B	4	C	
Y18	PB56A	4	T		Y18	PB56A	4	T	
W18	PB56B	4	C		W18	PB56B	4	C	
T17	PB57A	4	T		T17	PB57A	4	T	
U17	PB57B	4	C		U17	PB57B	4	C	
GND	-	-			GND	GND4	4		
GND	GND4	4			GND	GND4	4		
GND	GND3	3			GND	GND4	4		
GND	-	-			GND	GND3	3		
W20	PR48B	3	C	VREF2_3	W20	PR68B	3	C	VREF2_3

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

LFECP20/LFEC20					LFECP/LFEC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
Y20	PR48A	3	T	VREF1_3	Y20	PR68A	3	T	VREF1_3
GND	-	-			GND	GND3	3		
GND	-	-			GND	GND3	3		
AA21	PR47B	3	C		AA21	PR59B	3	C	
AB21	PR47A	3	T		AB21	PR59A	3	T	
W19	PR46B	3	C		W19	PR58B	3	C	
V19	PR46A	3	T		V19	PR58A	3	T	
Y21	PR45B	3	C		Y21	PR57B	3	C	
AA22	PR45A	3	T	RDQS45	AA22	PR57A	3	T	RDQS57
V20	PR44B	3	C	RLM0_PLLC_IN_A	V20	PR56B	3	C	RLM0_PLLC_IN_A
GND	GND3	3			GND	GND3	3		
U20	PR44A	3	T	RLM0_PLLT_IN_A	U20	PR56A	3	T	RLM0_PLLT_IN_A
W21	PR43B	3	C	RLM0_PLLC_FB_A	W21	PR55B	3	C	RLM0_PLLC_FB_A
Y22	PR43A	3	T	RLM0_PLLT_FB_A	Y22	PR55A	3	T	RLM0_PLLT_FB_A
V21	PR42B	3	C	DI/CSSPIN	V21	PR54B	3	C	DI/CSSPIN
W22	PR42A	3	T	DOUT/CSON	W22	PR54A	3	T	DOUT/CSON
U21	PR41B	3	C	BUSY/SISPI	U21	PR53B	3	C	BUSY/SISPI
V22	PR41A	3	T	D7/SPID0	V22	PR53A	3	T	D7/SPID0
T19	CFG2	3			T19	CFG2	3		
U19	CFG1	3			U19	CFG1	3		
U18	CFG0	3			U18	CFG0	3		
V18	PROGRAMN	3			V18	PROGRAMN	3		
T20	CCLK	3			T20	CCLK	3		
T21	INITN	3			T21	INITN	3		
R20	DONE	3			R20	DONE	3		
GND	GND3	3			GND	GND3	3		
T18	PR37B	3	C		T18	PR49B	3	C	
R17	PR37A	3	T		R17	PR49A	3	T	
R19	PR36B	3	C		R19	PR48B	3	C	
R18	PR36A	3	T	RDQS36	R18	PR48A	3	T	RDQS48
U22	PR35B	3	C		U22	PR47B	3	C	
GND	GND3	3			GND	GND3	3		
T22	PR35A	3	T		T22	PR47A	3	T	
R21	PR34B	3	C		R21	PR46B	3	C	
R22	PR34A	3	T		R22	PR46A	3	T	
P20	PR33B	3	C		P20	PR45B	3	C	
N20	PR33A	3	T		N20	PR45A	3	T	
P19	PR32B	3	C		P19	PR44B	3	C	
P18	PR32A	3	T		P18	PR44A	3	T	
P21	PR31B	3	C		P21	PR43B	3	C	
GND	GND3	3			GND	GND3	3		
P22	PR31A	3	T		P22	PR43A	3	T	
N21	PR30B	3	C		N21	PR42B	3	C	
N22	PR30A	3	T		N22	PR42A	3	T	

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

	LFECP20/LFEC20					LFECP/LFEC33			
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
N19	PR29B	3	C		N19	PR41B	3	C	
N18	PR29A	3	T		N18	PR41A	3	T	
M21	PR28B	3	C		M21	PR40B	3	C	
L20	PR28A	3	T	RDQS28	L20	PR40A	3	T	RDQS40
L21	PR27B	3	C		L21	PR39B	3	C	
GND	GND3	3			GND	GND3	3		
M20	PR27A	3	T		M20	PR39A	3	T	
M18	PR26B	3	C		M18	PR38B	3	C	
M19	PR26A	3	T		M19	PR38A	3	T	
M22	PR25B	3	C		M22	PR37B	3	C	
L22	PR25A	3	T		L22	PR37A	3	T	
K22	PR24B	3	C		K22	PR36B	3	C	
K21	PR24A	3	T		K21	PR36A	3	T	
J22	PR22B	2	C	PCLKC2_0	J22	PR34B	2	C	PCLKC2_0
GND	GND2	2			GND	GND2	2		
J21	PR22A	2	T	PCLKT2_0	J21	PR34A	2	T	PCLKT2_0
H22	PR21B	2	C		H22	PR33B	2	C	
H21	PR21A	2	T		H21	PR33A	2	T	
L19	PR20B	2	C		L19	PR32B	2	C	
L18	PR20A	2	T		L18	PR32A	2	T	
K20	PR19B	2	C		K20	PR31B	2	C	
J20	PR19A	2	T	RDQS19	J20	PR31A	2	T	RDQS31
K19	PR18B	2	C		K19	PR30B	2	C	
GND	GND2	2			GND	GND2	2		
K18	PR18A	2	T		K18	PR30A	2	T	
G22	PR17B	2	C		G22	PR29B	2	C	
F22	PR17A	2	T		F22	PR29A	2	T	
F21	PR16B	2	C		F21	PR28B	2	C	
E22	PR16A	2	T		E22	PR28A	2	T	
E21	PR15B	2	C		E21	PR27B	2	C	
D22	PR15A	2	T		D22	PR27A	2	T	
G21	PR14B	2	C		G21	PR26B	2	C	
G20	PR14A	2	T		G20	PR26A	2	T	
GND	GND2	2			GND	GND2	2		
J18	PR13B	2	C		J18	PR25B	2	C	
H19	PR13A	2	T		H19	PR25A	2	T	
J19	PR12B	2	C		J19	PR24B	2	C	
H20	PR12A	2	T		H20	PR24A	2	T	
H17	PR11B	2	C		H17	PR23B	2	C	
H18	PR11A	2	T		H18	PR23A	2	T	RDQS23
D21	PR9B	2	C	RUM0_PLLC_FB_A	D21	PR17B	2	C	RUM0_PLLC_FB_A
GND	GND2	2			GND	GND2	2		
GND	-	-			GND	GND2	2		
C22	PR9A	2	T	RUM0_PLLT_FB_A	C22	PR17A	2	T	RUM0_PLLT_FB_A

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

LFECP20/LFEC20					LFECP/LFEC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
G19	PR8B	2	C	RUM0_PLLC_IN_A	G19	PR16B	2	C	RUM0_PLLC_IN_A
G18	PR8A	2	T	RUM0_PLLT_IN_A	G18	PR16A	2	T	RUM0_PLLT_IN_A
F20	PR7B	2	C		F20	PR15B	2	C	
F19	PR7A	2	T		F19	PR15A	2	T	
E20	PR6B	2	C		E20	PR14B	2	C	
D20	PR6A	2	T	RDQS6	D20	PR14A	2	T	RDQS14
C21	PR5B	2	C		C21	PR13B	2	C	
GND	-	-			GND	GND2	2		
C20	PR5A	2	T		C20	PR13A	2	T	
F18	PR4B	2	C		F18	PR12B	2	C	
E18	PR4A	2	T		E18	PR12A	2	T	
B22	PR3B	2	C		B22	PR11B	2	C	
B21	PR3A	2	T		B21	PR11A	2	T	
GND	-	-			GND	GND2	2		
E19	PR2B	2	C	VREF1_2	E19	PR2B	2	C	VREF1_2
D19	PR2A	2	T	VREF2_2	D19	PR2A	2	T	VREF2_2
GND	GND2	2			GND	GND2	2		
GND	GND1	1			GND	GND1	1		
GND	-	-			GND	GND1	1		
G17	PT57B	1	C		G17	PT57B	1	C	
GND	-	-			GND	GND1	1		
F17	PT57A	1	T		F17	PT57A	1	T	
D18	PT56B	1	C		D18	PT56B	1	C	
C18	PT56A	1	T		C18	PT56A	1	T	
C19	PT55B	1	C		C19	PT55B	1	C	
B20	PT55A	1	T		B20	PT55A	1	T	
D17	PT54B	1	C		D17	PT54B	1	C	
C16	PT54A	1	T	TDQS54	C16	PT54A	1	T	TDQS54
B19	PT53B	1	C		B19	PT53B	1	C	
GND	GND1	1			GND	GND1	1		
A20	PT53A	1	T		A20	PT53A	1	T	
E17	PT52B	1	C		E17	PT52B	1	C	
C17	PT52A	1	T		C17	PT52A	1	T	
F16	PT51B	1	C		F16	PT51B	1	C	
E16	PT51A	1	T		E16	PT51A	1	T	
F15	PT50B	1	C		F15	PT50B	1	C	
D16	PT50A	1	T		D16	PT50A	1	T	
B18	PT49B	1	C		B18	PT49B	1	C	
GND	GND1	1			GND	GND1	1		
A19	PT49A	1	T		A19	PT49A	1	T	
B17	PT48B	1	C		B17	PT48B	1	C	
A18	PT48A	1	T		A18	PT48A	1	T	
B16	PT47B	1	C		B16	PT47B	1	C	
A17	PT47A	1	T		A17	PT47A	1	T	

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

	LFECP20/LFEC20					LFECP/LFEC33			
Ball Number	**Ball Function**	**Bank**	**LVDS**	**Dual Function**	**Ball Number**	**Ball Function**	**Bank**	**LVDS**	**Dual Function**
B15	PT46B	1	C		B15	PT46B	1	C	
A16	PT46A	1	T	TDQS46	A16	PT46A	1	T	TDQS46
A15	PT45B	1	C		A15	PT45B	1	C	
GND	GND1	1			GND	GND1	1		
A14	PT45A	1	T		A14	PT45A	1	T	
G14	PT44B	1	C		G14	PT44B	1	C	
E15	PT44A	1	T		E15	PT44A	1	T	
D15	PT43B	1	C		D15	PT43B	1	C	
C15	PT43A	1	T		C15	PT43A	1	T	
C14	PT42B	1	C		C14	PT42B	1	C	
B14	PT42A	1	T		B14	PT42A	1	T	
A13	PT41B	1	C		A13	PT41B	1	C	
GND	GND1	1			GND	GND1	1		
B13	PT41A	1	T		B13	PT41A	1	T	
E14	PT40B	1	C		E14	PT40B	1	C	
C13	PT40A	1	T		C13	PT40A	1	T	
F14	PT39B	1	C		F14	PT39B	1	C	
D14	PT39A	1	T		D14	PT39A	1	T	
E13	PT38B	1	C		E13	PT38B	1	C	
G13	PT38A	1	T	TDQS38	G13	PT38A	1	T	TDQS38
A12	PT37B	1	C		A12	PT37B	1	C	
GND	GND1	1			GND	GND1	1		
B12	PT37A	1	T		B12	PT37A	1	T	
F13	PT36B	1	C		F13	PT36B	1	C	
D13	PT36A	1	T		D13	PT36A	1	T	
F12	PT35B	1	C	VREF2_1	F12	PT35B	1	C	VREF2_1
D12	PT35A	1	T	VREF1_1	D12	PT35A	1	T	VREF1_1
F11	PT34B	1	C		F11	PT34B	1	C	
C12	PT34A	1	T		C12	PT34A	1	T	
A11	PT33B	0	C	PCLKC0_0	A11	PT33B	0	C	PCLKC0_0
GND	GND0	0			GND	GND0	0		
A10	PT33A	0	T	PCLKT0_0	A10	PT33A	0	T	PCLKT0_0
E12	PT32B	0	C	VREF1_0	E12	PT32B	0	C	VREF1_0
E11	PT32A	0	T	VREF2_0	E11	PT32A	0	T	VREF2_0
B11	PT31B	0	C		B11	PT31B	0	C	
C11	PT31A	0	T		C11	PT31A	0	T	
B9	PT30B	0	C		B9	PT30B	0	C	
B10	PT30A	0	T	TDQS30	B10	PT30A	0	T	TDQS30
A9	PT29B	0	C		A9	PT29B	0	C	
GND	GND0	0			GND	GND0	0		
A8	PT29A	0	T		A8	PT29A	0	T	
D11	PT28B	0	C		D11	PT28B	0	C	
C10	PT28A	0	T		C10	PT28A	0	T	
A7	PT27B	0	C		A7	PT27B	0	C	

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

	LFECP20/LFEC20					LFECP/LFEC33			
Ball Number	**Ball Function**	**Bank**	**LVDS**	**Dual Function**	**Ball Number**	**Ball Function**	**Bank**	**LVDS**	**Dual Function**
A6	PT27A	0	T		A6	PT27A	0	T	
B7	PT26B	0	C		B7	PT26B	0	C	
B8	PT26A	0	T		B8	PT26A	0	T	
A5	PT25B	0	C		A5	PT25B	0	C	
GND	GND0	0			GND	GND0	0		
B6	PT25A	0	T		B6	PT25A	0	T	
G10	PT24B	0	C		G10	PT24B	0	C	
E10	PT24A	0	T		E10	PT24A	0	T	
F10	PT23B	0	C		F10	PT23B	0	C	
D10	PT23A	0	T		D10	PT23A	0	T	
G9	PT22B	0	C		G9	PT22B	0	C	
E9	PT22A	0	T	TDQS22	E9	PT22A	0	T	TDQS22
C9	PT21B	0	C		C9	PT21B	0	C	
GND	GND0	0			GND	GND0	0		
C8	PT21A	0	T		C8	PT21A	0	T	
F9	PT20B	0	C		F9	PT20B	0	C	
D9	PT20A	0	T		D9	PT20A	0	T	
F8	PT19B	0	C		F8	PT19B	0	C	
D7	PT19A	0	T		D7	PT19A	0	T	
D8	PT18B	0	C		D8	PT18B	0	C	
C7	PT18A	0	T		C7	PT18A	0	T	
GND	GND0	0			GND	GND0	0		
A4	PT17B	0	C		A4	PT17B	0	C	
B4	PT17A	0	T		B4	PT17A	0	T	
C4	PT16B	0	C		C4	PT16B	0	C	
C5	PT16A	0	T		C5	PT16A	0	T	
D6	PT15B	0	C		D6	PT15B	0	C	
B5	PT15A	0	T		B5	PT15A	0	T	
E6	PT14B	0	C		E6	PT14B	0	C	
C6	PT14A	0	T	TDQS14	C6	PT14A	0	T	TDQS14
A3	PT13B	0	C		A3	PT13B	0	C	
GND	GND0	0			GND	GND0	0		
B3	PT13A	0	T		B3	PT13A	0	T	
F6	PT12B	0	C		F6	PT12B	0	C	
D5	PT12A	0	T		D5	PT12A	0	T	
F7	PT11B	0	C		F7	PT11B	0	C	
E8	PT11A	0	T		E8	PT11A	0	T	
G6	PT10B	0	C		G6	PT10B	0	C	
E7	PT10A	0	T		E7	PT10A	0	T	
GND	GND0	0			GND	GND0	0		
GND	GND0	0			GND	GND0	0		
A1	GND	-			A1	GND	-		
A22	GND	-			A22	GND	-		
AB1	GND	-			AB1	GND	-		

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

LFECP20/LFEC20					LFECP/LFEC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
AB22	GND	-			AB22	GND	-		
H15	GND	-			H15	GND	-		
H8	GND	-			H8	GND	-		
J10	GND	-			J10	GND	-		
J11	GND	-			J11	GND	-		
J12	GND	-			J12	GND	-		
J13	GND	-			J13	GND	-		
J14	GND	-			J14	GND	-		
J9	GND	-			J9	GND	-		
K10	GND	-			K10	GND	-		
K11	GND	-			K11	GND	-		
K12	GND	-			K12	GND	-		
K13	GND	-			K13	GND	-		
K14	GND	-			K14	GND	-		
K9	GND	-			K9	GND	-		
L10	GND	-			L10	GND	-		
L11	GND	-			L11	GND	-		
L12	GND	-			L12	GND	-		
L13	GND	-			L13	GND	-		
L14	GND	-			L14	GND	-		
L9	GND	-			L9	GND	-		
M10	GND	-			M10	GND	-		
M11	GND	-			M11	GND	-		
M12	GND	-			M12	GND	-		
M13	GND	-			M13	GND	-		
M14	GND	-			M14	GND	-		
M9	GND	-			M9	GND	-		
N10	GND	-			N10	GND	-		
N11	GND	-			N11	GND	-		
N12	GND	-			N12	GND	-		
N13	GND	-			N13	GND	-		
N14	GND	-			N14	GND	-		
N9	GND	-			N9	GND	-		
P10	GND	-			P10	GND	-		
P11	GND	-			P11	GND	-		
P12	GND	-			P12	GND	-		
P13	GND	-			P13	GND	-		
P14	GND	-			P14	GND	-		
P9	GND	-			P9	GND	-		
R15	GND	-			R15	GND	-		
R8	GND	-			R8	GND	-		
J16	VCC	-			J16	VCC	-		
J7	VCC	-			J7	VCC	-		
K16	VCC	-			K16	VCC	-		

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

LFECP20/LFEC20					LFECP/LFEC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
K17	VCC	-			K17	VCC	-		
K6	VCC	-			K6	VCC	-		
K7	VCC	-			K7	VCC	-		
L17	VCC	-			L17	VCC	-		
L6	VCC	-			L6	VCC	-		
M17	VCC	-			M17	VCC	-		
M6	VCC	-			M6	VCC	-		
N16	VCC	-			N16	VCC	-		
N17	VCC	-			N17	VCC	-		
N6	VCC	-			N6	VCC	-		
N7	VCC	-			N7	VCC	-		
P16	VCC	-			P16	VCC	-		
P7	VCC	-			P7	VCC	-		
G11	VCCIO0	0			G11	VCCIO0	0		
H10	VCCIO0	0			H10	VCCIO0	0		
H11	VCCIO0	0			H11	VCCIO0	0		
H9	VCCIO0	0			H9	VCCIO0	0		
G12	VCCIO1	1			G12	VCCIO1	1		
H12	VCCIO1	1			H12	VCCIO1	1		
H13	VCCIO1	1			H13	VCCIO1	1		
H14	VCCIO1	1			H14	VCCIO1	1		
J15	VCCIO2	2			J15	VCCIO2	2		
K15	VCCIO2	2			K15	VCCIO2	2		
L15	VCCIO2	2			L15	VCCIO2	2		
L16	VCCIO2	2			L16	VCCIO2	2		
M15	VCCIO3	3			M15	VCCIO3	3		
M16	VCCIO3	3			M16	VCCIO3	3		
N15	VCCIO3	3			N15	VCCIO3	3		
P15	VCCIO3	3			P15	VCCIO3	3		
R12	VCCIO4	4			R12	VCCIO4	4		
R13	VCCIO4	4			R13	VCCIO4	4		
R14	VCCIO4	4			R14	VCCIO4	4		
T12	VCCIO4	4			T12	VCCIO4	4		
R10	VCCIO5	5			R10	VCCIO5	5		
R11	VCCIO5	5			R11	VCCIO5	5		
R9	VCCIO5	5			R9	VCCIO5	5		
T11	VCCIO5	5			T11	VCCIO5	5		
M7	VCCIO6	6			M7	VCCIO6	6		
M8	VCCIO6	6			M8	VCCIO6	6		
N8	VCCIO6	6			N8	VCCIO6	6		
P8	VCCIO6	6			P8	VCCIO6	6		
J8	VCCIO7	7			J8	VCCIO7	7		
K8	VCCIO7	7			K8	VCCIO7	7		
L7	VCCIO7	7			L7	VCCIO7	7		

LFECP/EC20 and LFECP/EC33 Logic Signal Connections: 484 fpBGA

LFECP20/LFEC20					LFECP/LFEC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
L8	VCCIO7	7			L8	VCCIO7	7		
G15	VCCAUX	-			G15	VCCAUX	-		
G16	VCCAUX	-			G16	VCCAUX	-		
G7	VCCAUX	-			G7	VCCAUX	-		
G8	VCCAUX	-			G8	VCCAUX	-		
H16	VCCAUX	-			H16	VCCAUX	-		
H7	VCCAUX	-			H7	VCCAUX	-		
R16	VCCAUX	-			R16	VCCAUX	-		
R7	VCCAUX	-			R7	VCCAUX	-		
T15	VCCAUX	-			T15	VCCAUX	-		
T16	VCCAUX	-			T16	VCCAUX	-		
T7	VCCAUX	-			T7	VCCAUX	-		
T8	VCCAUX	-			T8	VCCAUX	-		
J6	VCC[1]	-			J6	VCCPLL	-		
J17	VCC[1]	-			J17	VCCPLL	-		
P6	VCC[1]	-			P6	VCCPLL	-		
P17	VCC[1]	-			P17	VCCPLL	-		
A2	NC	-			A2	NC	-		
AB2	NC	-			AB2	NC	-		
A21	NC	-			A21	NC	-		

1. Tied to V_{CCPLL}.

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA

	LFEC20/LFECP20					LFECP/EC33			
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
GND	GND7	7			GND	GND7	7		
E3	PL2A	7	T	VREF2_7	E3	PL2A	7	T	VREF2_7
E4	PL2B	7	C	VREF1_7	E4	PL2B	7	C	VREF1_7
E5	NC	-			E5	PL6A	7	T	LDQS6
D5	NC	-			D5	PL6B	7	C	
F4	NC	-			F4	PL7A	7	T	
F5	NC	-			F5	PL7B	7	C	
C3	NC	-			C3	PL8A	7	T	
D3	NC	-			D3	PL8B	7	C	
C2	NC	-			C2	PL9A	7	T	
-	-	-			GND	GND7	7		
B2	NC	-			B2	PL9B	7	C	
B1	PL3A	7	T		B1	PL10A	7	T	
C1	PL3B	7	C		C1	PL10B	7	C	
F3	PL4A	7	T		F3	PL11A	7	T	
G3	PL4B	7	C		G3	PL11B	7	C	
D2	PL5A	7	T		D2	PL12A	7	T	
E2	PL5B	7	C		E2	PL12B	7	C	
-	-	-			GND	GND7	7		
D1	PL6A	7	T	LDQS6	D1	PL14A	7	T	LDQS14
E1	PL6B	7	C		E1	PL14B	7	C	
F2	PL7A	7	T		F2	PL15A	7	T	
G2	PL7B	7	C		G2	PL15B	7	C	
F6	PL8A	7	T	LUM0_PLLT_IN_A	F6	PL16A	7	T	LUM0_PLLT_IN_A
G6	PL8B	7	C	LUM0_PLLC_IN_A	G6	PL16B	7	C	LUM0_PLLC_IN_A
H4	PL9A	7	T	LUM0_PLLT_FB_A	H4	PL17A	7	T	LUM0_PLLT_FB_A
GND	GND7	7			GND	GND7	7		
G4	PL9B	7	C	LUM0_PLLC_FB_A	G4	PL17B	7	C	LUM0_PLLC_FB_A
H6	NC	-			H6	PL19A	7	T	
J7	NC	-			J7	PL19B	7	C	
G5	NC	-			G5	PL20A	7	T	
H5	NC	-			H5	PL20B	7	C	
H3	NC	-			H3	PL21A	7	T	
J3	NC	-			J3	PL21B	7	C	
H2	NC	-			H2	PL22A	7	T	
-	-	-			GND	GND7	7		
J2	NC	-			J2	PL22B	7	C	
J4	PL11A	7	T		J4	PL23A	7	T	LDQS23
J5	PL11B	7	C		J5	PL23B	7	C	
K4	PL12A	7	T		K4	PL24A	7	T	
K5	PL12B	7	C		K5	PL24B	7	C	
J6	PL13A	7	T		J6	PL25A	7	T	

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
		LFEC20/LFECP20					LFECP/EC33		
K6	PL13B	7	C		K6	PL25B	7	C	
F1	PL14A	7	T		F1	PL26A	7	T	
GND	GND7	7			GND	GND7	7		
G1	PL14B	7	C		G1	PL26B	7	C	
H1	PL15A	7	T		H1	PL27A	7	T	
J1	PL15B	7	C		J1	PL27B	7	C	
K2	PL16A	7	T		K2	PL28A	7	T	
K1	PL16B	7	C		K1	PL28B	7	C	
K3	PL17A	7	T		K3	PL29A	7	T	
L3	PL17B	7	C		L3	PL29B	7	C	
L2	PL18A	7	T		L2	PL30A	7	T	
GND	GND7	7			GND	GND7	7		
L1	PL18B	7	C		L1	PL30B	7	C	
M3	PL19A	7	T	LDQS19	M3	PL31A	7	T	LDQS31
M4	PL19B	7	C		M4	PL31B	7	C	
M1	PL20A	7	T		M1	PL32A	7	T	
M2	PL20B	7	C		M2	PL32B	7	C	
L4	PL21A	7	T		L4	PL33A	7	T	
L5	PL21B	7	C		L5	PL33B	7	C	
N2	PL22A	7	T	PCLKT7_0	N2	PL34A	7	T	PCLKT7_0
GND	GND7	7			GND	GND7	7		
N1	PL22B	7	C	PCLKC7_0	N1	PL34B	7	C	PCLKC7_0
N3	XRES	6			N3	XRES	6		
P1	PL24A	6	T		P1	PL36A	6	T	
P2	PL24B	6	C		P2	PL36B	6	C	
L7	PL25A	6	T		L7	PL37A	6	T	
L6	PL25B	6	C		L6	PL37B	6	C	
N4	PL26A	6	T		N4	PL38A	6	T	
N5	PL26B	6	C		N5	PL38B	6	C	
R1	PL27A	6	T		R1	PL39A	6	T	
GND	GND6	6			GND	GND6	6		
R2	PL27B	6	C		R2	PL39B	6	C	
P4	PL28A	6	T	LDQS28	P4	PL40A	6	T	LDQS40
P3	PL28B	6	C		P3	PL40B	6	C	
M5	PL29A	6	T		M5	PL41A	6	T	
M6	PL29B	6	C		M6	PL41B	6	C	
T1	PL30A	6	T		T1	PL42A	6	T	
T2	PL30B	6	C		T2	PL42B	6	C	
R4	PL31A	6	T		R4	PL43A	6	T	
GND	GND6	6			GND	GND6	6		
R3	PL31B	6	C		R3	PL43B	6	C	
N6	PL32A	6	T		N6	PL44A	6	T	

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
		LFEC20/LFECP20					LFECP/EC33		
P5	PL32B	6	C		P5	PL44B	6	C	
P6	PL33A	6	T		P6	PL45A	6	T	
R5	PL33B	6	C		R5	PL45B	6	C	
U1	PL34A	6	T		U1	PL46A	6	T	
U2	PL34B	6	C		U2	PL46B	6	C	
T3	PL35A	6	T		T3	PL47A	6	T	
GND	GND6	6			GND	GND6	6		
T4	PL35B	6	C		T4	PL47B	6	C	
R6	PL36A	6	T	LDQS36	R6	PL48A	6	T	LDQS48
T5	PL36B	6	C		T5	PL48B	6	C	
T6	PL37A	6	T		T6	PL49A	6	T	
U5	PL37B	6	C		U5	PL49B	6	C	
U3	PL38A	6	T		U3	PL50A	6	T	
U4	PL38B	6	C		U4	PL50B	6	C	
V1	PL39A	6	T		V1	PL51A	6	T	
GND	GND6	6			GND	GND6	6		
V2	PL39B	6	C		V2	PL51B	6	C	
U7	TCK	6			U7	TCK	6		
V4	TDI	6			V4	TDI	6		
V5	TMS	6			V5	TMS	6		
V3	TDO	6			V3	TDO	6		
U6	VCCJ	6			U6	VCCJ	6		
W1	PL41A	6	T	LLM0_PLLT_IN_A	W1	PL53A	6	T	LLM0_PLLT_IN_A
W2	PL41B	6	C	LLM0_PLLC_IN_A	W2	PL53B	6	C	LLM0_PLLC_IN_A
V6	PL42A	6	T	LLM0_PLLT_FB_A	V6	PL54A	6	T	LLM0_PLLT_FB_A
W6	PL42B	6	C	LLM0_PLLC_FB_A	W6	PL54B	6	C	LLM0_PLLC_FB_A
Y1	PL43A	6	T		Y1	PL55A	6	T	
Y2	PL43B	6	C		Y2	PL55B	6	C	
W3	PL44A	6	T		W3	PL56A	6	T	
GND	GND6	6			GND	GND6	6		
W4	PL44B	6	C		W4	PL56B	6	C	
AA1	PL45A	6	T	LDQS45	AA1	PL57A	6	T	LDQS57
AB1	PL45B	6	C		AB1	PL57B	6	C	
Y4	PL46A	6	T		Y4	PL58A	6	T	
Y3	PL46B	6	C		Y3	PL58B	6	C	
AC1	PL47A	6	T		AC1	PL59A	6	T	
AB2	PL47B	6	C		AB2	PL59B	6	C	
AA2	NC	-			AA2	PL60A	6	T	
-	-	-			GND	GND6	6		
AA3	NC	-			AA3	PL60B	6	C	
W5	NC	-			W5	PL61A	6	T	
Y5	NC	-			Y5	PL61B	6	C	

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

LFEC20/LFECP20					LFECP/EC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
Y6	NC	-			Y6	PL62A	6	T	
W7	NC	-			W7	PL62B	6	C	
AA4	NC	-			AA4	PL63A	6	T	
AB3	NC	-			AB3	PL63B	6	C	
AC2	NC	-			AC2	PL64A	6	T	
-	-	-			GND	GND6	6		
AC3	NC	-			AC3	PL64B	6	C	
AA5	NC	-			AA5	PL65A	6	T	LDQS65
AB5	NC	-			AB5	PL65B	6	C	
AD3	NC	-			AD3	PL66A	6	T	
AD2	NC	-			AD2	PL66B	6	C	
AE1	NC	-			AE1	PL67A	6	T	
AD1	NC	-			AD1	PL67B	6	C	
AB4	PL48A	6	T	VREF1_6	AB4	PL68A	6	T	VREF1_6
AC4	PL48B	6	C	VREF2_6	AC4	PL68B	6	C	VREF2_6
GND	GND6	6			GND	GND6	6		
GND	GND5	5			GND	GND5	5		
AB6	PB2A	5	T		AB6	PB2A	5	T	
AA6	PB2B	5	C		AA6	PB2B	5	C	
AC7	PB3A	5	T		AC7	PB3A	5	T	
Y8	PB3B	5	C		Y8	PB3B	5	C	
AB7	PB4A	5	T		AB7	PB4A	5	T	
AA7	PB4B	5	C		AA7	PB4B	5	C	
AC6	PB5A	5	T		AC6	PB5A	5	T	
AC5	PB5B	5	C		AC5	PB5B	5	C	
AB8	PB6A	5	T	BDQS6	AB8	PB6A	5	T	BDQS6
AC8	PB6B	5	C		AC8	PB6B	5	C	
AE2	PB7A	5	T		AE2	PB7A	5	T	
AA8	PB7B	5	C		AA8	PB7B	5	C	
AF2	PB8A	5	T		AF2	PB8A	5	T	
Y9	PB8B	5	C		Y9	PB8B	5	C	
AD5	PB9A	5	T		AD5	PB9A	5	T	
GND	GND5	5			GND	GND5	5		
AD4	PB9B	5	C		AD4	PB9B	5	C	
AD8	PB10A	5	T		AD8	PB10A	5	T	
AC9	PB10B	5	C		AC9	PB10B	5	C	
AE3	PB11A	5	T		AE3	PB11A	5	T	
AB9	PB11B	5	C		AB9	PB11B	5	C	
AF3	PB12A	5	T		AF3	PB12A	5	T	
AD9	PB12B	5	C		AD9	PB12B	5	C	
AE4	PB13A	5	T		AE4	PB13A	5	T	
GND	GND5	5			GND	GND5	5		

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

	LFEC20/LFECP20					LFECP/EC33			
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
AF4	PB13B	5	C		AF4	PB13B	5	C	
AE5	PB14A	5	T	BDQS14	AE5	PB14A	5	T	BDQS14
AA9	PB14B	5	C		AA9	PB14B	5	C	
AF5	PB15A	5	T		AF5	PB15A	5	T	
Y10	PB15B	5	C		Y10	PB15B	5	C	
AD6	PB16A	5	T		AD6	PB16A	5	T	
AC10	PB16B	5	C		AC10	PB16B	5	C	
AF6	PB17A	5	T		AF6	PB17A	5	T	
GND	GND5	5			GND	GND5	5		
AE6	PB17B	5	C		AE6	PB17B	5	C	
AF7	PB18A	5	T		AF7	PB18A	5	T	
AB10	PB18B	5	C		AB10	PB18B	5	C	
AE7	PB19A	5	T		AE7	PB19A	5	T	
AD10	PB19B	5	C		AD10	PB19B	5	C	
AD7	PB20A	5	T		AD7	PB20A	5	T	
AA10	PB20B	5	C		AA10	PB20B	5	C	
AF8	PB21A	5	T		AF8	PB21A	5	T	
GND	GND5	5			GND	GND5	5		
AF9	PB21B	5	C		AF9	PB21B	5	C	
AD11	PB22A	5	T	BDQS22	AD11	PB22A	5	T	BDQS22
Y11	PB22B	5	C		Y11	PB22B	5	C	
AE8	PB23A	5	T		AE8	PB23A	5	T	
AC11	PB23B	5	C		AC11	PB23B	5	C	
AF10	PB24A	5	T		AF10	PB24A	5	T	
AB11	PB24B	5	C		AB11	PB24B	5	C	
AE10	PB25A	5	T		AE10	PB25A	5	T	
GND	GND5	5			GND	GND5	5		
AE9	PB25B	5	C		AE9	PB25B	5	C	
AA11	PB26A	5	T		AA11	PB26A	5	T	
Y12	PB26B	5	C		Y12	PB26B	5	C	
AE11	PB27A	5	T		AE11	PB27A	5	T	
AF11	PB27B	5	C		AF11	PB27B	5	C	
AF12	PB28A	5	T		AF12	PB28A	5	T	
AE12	PB28B	5	C		AE12	PB28B	5	C	
AD12	PB29A	5	T		AD12	PB29A	5	T	
GND	GND5	5			GND	GND5	5		
AC12	PB29B	5	C		AC12	PB29B	5	C	
AA12	PB30A	5	T	BDQS30	AA12	PB30A	5	T	BDQS30
AB12	PB30B	5	C		AB12	PB30B	5	C	
AE13	PB31A	5	T		AE13	PB31A	5	T	
AF13	PB31B	5	C		AF13	PB31B	5	C	
AD13	PB32A	5	T	VREF2_5	AD13	PB32A	5	T	VREF2_5

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

LFEC20/LFECP20					LFECP/EC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
AC13	PB32B	5	C	VREF1_5	AC13	PB32B	5	C	VREF1_5
AF14	PB33A	5	T	PCLKT5_0	AF14	PB33A	5	T	PCLKT5_0
GND	GND5	5			GND	GND5	5		
AE14	PB33B	5	C	PCLKC5_0	AE14	PB33B	5	C	PCLKC5_0
AA13	PB34A	4	T	WRITEN	AA13	PB34A	4	T	WRITEN
AB13	PB34B	4	C	CS1N	AB13	PB34B	4	C	CS1N
AD14	PB35A	4	T	VREF1_4	AD14	PB35A	4	T	VREF1_4
AA14	PB35B	4	C	CSN	AA14	PB35B	4	C	CSN
AC14	PB36A	4	T	VREF2_4	AC14	PB36A	4	T	VREF2_4
AB14	PB36B	4	C	D0/SPID7	AB14	PB36B	4	C	D0/SPID7
AF15	PB37A	4	T	D2/SPID5	AF15	PB37A	4	T	D2/SPID5
GND	GND4	4			GND	GND4	4		
AE15	PB37B	4	C	D1/SPID6	AE15	PB37B	4	C	D1/SPID6
AD15	PB38A	4	T	BDQS38	AD15	PB38A	4	T	BDQS38
AC15	PB38B	4	C	D3/SPID4	AC15	PB38B	4	C	D3/SPID4
AF16	PB39A	4	T		AF16	PB39A	4	T	
Y14	PB39B	4	C	D4/SPID3	Y14	PB39B	4	C	D4/SPID3
AE16	PB40A	4	T		AE16	PB40A	4	T	
AB15	PB40B	4	C	D5/SPID2	AB15	PB40B	4	C	D5/SPID2
AF17	PB41A	4	T		AF17	PB41A	4	T	
GND	GND4	4			GND	GND4	4		
AE17	PB41B	4	C	D6/SPID1	AE17	PB41B	4	C	D6/SPID1
Y15	PB42A	4	T		Y15	PB42A	4	T	
AA15	PB42B	4	C		AA15	PB42B	4	C	
AD17	PB43A	4	T		AD17	PB43A	4	T	
Y16	PB43B	4	C		Y16	PB43B	4	C	
AD18	PB44A	4	T		AD18	PB44A	4	T	
AC16	PB44B	4	C		AC16	PB44B	4	C	
AE18	PB45A	4	T		AE18	PB45A	4	T	
GND	GND4	4			GND	GND4	4		
AF18	PB45B	4	C		AF18	PB45B	4	C	
AD16	PB46A	4	T	BDQS46	AD16	PB46A	4	T	BDQS46
AB16	PB46B	4	C		AB16	PB46B	4	C	
AF19	PB47A	4	T		AF19	PB47A	4	T	
AA16	PB47B	4	C		AA16	PB47B	4	C	
AA17	PB48A	4	T		AA17	PB48A	4	T	
Y17	PB48B	4	C		Y17	PB48B	4	C	
AF21	PB49A	4	T		AF21	PB49A	4	T	
GND	GND4	4			GND	GND4	4		
AF20	PB49B	4	C		AF20	PB49B	4	C	
AE21	PB50A	4	T		AE21	PB50A	4	T	
AC17	PB50B	4	C		AC17	PB50B	4	C	

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

LFEC20/LFECP20					LFECP/EC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
AF22	PB51A	4	T		AF22	PB51A	4	T	
AB17	PB51B	4	C		AB17	PB51B	4	C	
AE22	PB52A	4	T		AE22	PB52A	4	T	
AA18	PB52B	4	C		AA18	PB52B	4	C	
AE19	PB53A	4	T		AE19	PB53A	4	T	
GND	GND4	4			GND	GND4	4		
AE20	PB53B	4	C		AE20	PB53B	4	C	
AA19	PB54A	4	T	BDQS54	AA19	PB54A	4	T	BDQS54
Y18	PB54B	4	C		Y18	PB54B	4	C	
AF23	PB55A	4	T		AF23	PB55A	4	T	
AA20	PB55B	4	C		AA20	PB55B	4	C	
AC18	PB56A	4	T		AC18	PB56A	4	T	
AB18	PB56B	4	C		AB18	PB56B	4	C	
AF24	PB57A	4	T		AF24	PB57A	4	T	
-	-	-			GND	GND4	4		
AE23	PB57B	4	C		AE23	PB57B	4	C	
AD19	NC	-			AD19	PB58A	4	T	
AD20	NC	-			AD20	PB58B	4	C	
AC19	NC	-			AC19	PB59A	4	T	
AB19	NC	-			AB19	PB59B	4	C	
AD21	NC	-			AD21	PB60A	4	T	
AC20	NC	-			AC20	PB60B	4	C	
AF25	NC	-			AF25	PB61A	4	T	
-	-	-			GND	GND4	4		
AE25	NC	-			AE25	PB61B	4	C	
AB21	NC	-			AB21	PB62A	4	T	BDQS62
AB20	NC	-			AB20	PB62B	4	C	
AE24	NC	-			AE24	PB63A	4	T	
AD23	NC	-			AD23	PB63B	4	C	
AD22	NC	-			AD22	PB64A	4	T	
AC21	NC	-			AC21	PB64B	4	C	
AC22	NC	-			AC22	PB65A	4	T	
AB22	NC	-			AB22	PB65B	4	C	
GND	GND4	4			GND	GND4	4		
GND	GND3	3			GND	GND3	3		
AC23	PR48B	3	C	VREF2_3	AC23	PR68B	3	C	VREF2_3
AC24	PR48A	3	T	VREF1_3	AC24	PR68A	3	T	VREF1_3
AD24	NC	-			AD24	PR67B	3	C	
AD25	NC	-			AD25	PR67A	3	T	
AE26	NC	-			AE26	PR66B	3	C	
AD26	NC	-			AD26	PR66A	3	T	
Y20	NC	-			Y20	PR65B	3	C	

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
Y19	NC	-			Y19	PR65A	3	T	RDQS65
AA23	NC	-			AA23	PR64B	3	C	
-	-	-			GND	GND3	3		
AA22	NC	-			AA22	PR64A	3	T	
AB23	NC	-			AB23	PR63B	3	C	
AB24	NC	-			AB24	PR63A	3	T	
Y21	NC	-			Y21	PR62B	3	C	
AA21	NC	-			AA21	PR62A	3	T	
Y23	NC	-			Y23	PR61B	3	C	
Y22	NC	-			Y22	PR61A	3	T	
AA24	NC	-			AA24	PR60B	3	C	
-	-	-			GND	GND3	3		
Y24	NC	-			Y24	PR60A	3	T	
AC25	PR47B	3	C		AC25	PR59B	3	C	
AC26	PR47A	3	T		AC26	PR59A	3	T	
AB25	PR46B	3	C		AB25	PR58B	3	C	
AA25	PR46A	3	T		AA25	PR58A	3	T	
AB26	PR45B	3	C		AB26	PR57B	3	C	
AA26	PR45A	3	T	RDQS45	AA26	PR57A	3	T	RDQS57
W23	PR44B	3	C	RLM0_PLLC_IN_A	W23	PR56B	3	C	RLM0_PLLC_IN_A
GND	GND3	3			GND	GND3	3		
W24	PR44A	3	T	RLM0_PLLT_IN_A	W24	PR56A	3	T	RLM0_PLLT_IN_A
W22	PR43B	3	C	RLM0_PLLC_FB_A	W22	PR55B	3	C	RLM0_PLLC_FB_A
W21	PR43A	3	T	RLM0_PLLT_FB_A	W21	PR55A	3	T	RLM0_PLLT_FB_A
Y25	PR42B	3	C	DI/CSSPIN	Y25	PR54B	3	C	DI/CSSPIN
Y26	PR42A	3	T	DOUT/CSON	Y26	PR54A	3	T	DOUT/CSON
W25	PR41B	3	C	BUSY/SISPI	W25	PR53B	3	C	BUSY/SISPI
W26	PR41A	3	T	D7/SPID0	W26	PR53A	3	T	D7/SPID0
V24	CFG2	3			V24	CFG2	3		
V21	CFG1	3			V21	CFG1	3		
V23	CFG0	3			V23	CFG0	3		
V22	PROGRAMN	3			V22	PROGRAMN	3		
V20	CCLK	3			V20	CCLK	3		
V25	INITN	3			V25	INITN	3		
U20	DONE	3			U20	DONE	3		
V26	PR39B	3	C		V26	PR51B	3	C	
GND	GND3	3			GND	GND3	3		
U26	PR39A	3	T		U26	PR51A	3	T	
U24	PR38B	3	C		U24	PR50B	3	C	
U25	PR38A	3	T		U25	PR50A	3	T	
U23	PR37B	3	C		U23	PR49B	3	C	
U22	PR37A	3	T		U22	PR49A	3	T	

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
				LFEC20/LFECP20					**LFECP/EC33**
U21	PR36B	3	C		U21	PR48B	3	C	
T21	PR36A	3	T	RDQS36	T21	PR48A	3	T	RDQS48
T25	PR35B	3	C		T25	PR47B	3	C	
GND	GND3	3			GND	GND3	3		
T26	PR35A	3	T		T26	PR47A	3	T	
T22	PR34B	3	C		T22	PR46B	3	C	
T23	PR34A	3	T		T23	PR46A	3	T	
T24	PR33B	3	C		T24	PR45B	3	C	
R23	PR33A	3	T		R23	PR45A	3	T	
R25	PR32B	3	C		R25	PR44B	3	C	
R24	PR32A	3	T		R24	PR44A	3	T	
R26	PR31B	3	C		R26	PR43B	3	C	
GND	GND3	3			GND	GND3	3		
P26	PR31A	3	T		P26	PR43A	3	T	
R21	PR30B	3	C		R21	PR42B	3	C	
R22	PR30A	3	T		R22	PR42A	3	T	
P25	PR29B	3	C		P25	PR41B	3	C	
P24	PR29A	3	T		P24	PR41A	3	T	
P23	PR28B	3	C		P23	PR40B	3	C	
P22	PR28A	3	T	RDQS28	P22	PR40A	3	T	RDQS40
N26	PR27B	3	C		N26	PR39B	3	C	
GND	GND3	3			GND	GND3	3		
M26	PR27A	3	T		M26	PR39A	3	T	
N21	PR26B	3	C		N21	PR38B	3	C	
P21	PR26A	3	T		P21	PR38A	3	T	
N23	PR25B	3	C		N23	PR37B	3	C	
N22	PR25A	3	T		N22	PR37A	3	T	
N25	PR24B	3	C		N25	PR36B	3	C	
N24	PR24A	3	T		N24	PR36A	3	T	
L26	PR22B	2	C	PCLKC2_0	L26	PR34B	2	C	PCLKC2_0
GND	GND2	2			GND	GND2	2		
K26	PR22A	2	T	PCLKT2_0	K26	PR34A	2	T	PCLKT2_0
M22	PR21B	2	C		M22	PR33B	2	C	
M23	PR21A	2	T		M23	PR33A	2	T	
M25	PR20B	2	C		M25	PR32B	2	C	
M24	PR20A	2	T		M24	PR32A	2	T	
M21	PR19B	2	C		M21	PR31B	2	C	
L21	PR19A	2	T	RDQS19	L21	PR31A	2	T	RDQS31
L22	PR18B	2	C		L22	PR30B	2	C	
GND	GND2	2			GND	GND2	2		
L23	PR18A	2	T		L23	PR30A	2	T	
L25	PR17B	2	C		L25	PR29B	2	C	

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

LFEC20/LFECP20					LFECP/EC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
L24	PR17A	2	T		L24	PR29A	2	T	
K25	PR16B	2	C		K25	PR28B	2	C	
J25	PR16A	2	T		J25	PR28A	2	T	
J26	PR15B	2	C		J26	PR27B	2	C	
H26	PR15A	2	T		H26	PR27A	2	T	
H25	PR14B	2	C		H25	PR26B	2	C	
GND	GND2	2			GND	GND2	2		
J24	PR14A	2	T		J24	PR26A	2	T	
K21	PR13B	2	C		K21	PR25B	2	C	
K22	PR13A	2	T		K22	PR25A	2	T	
K20	PR12B	2	C		K20	PR24B	2	C	
J20	PR12A	2	T		J20	PR24A	2	T	
K23	PR11B	2	C		K23	PR23B	2	C	
K24	PR11A	2	T		K24	PR23A	2	T	RDQS23
J21	NC	-			J21	PR22B	2	C	
-	-	-			GND	GND2	2		
J22	NC	-			J22	PR22A	2	T	
J23	NC	-			J23	PR21B	2	C	
H22	NC	-			H22	PR21A	2	T	
G26	NC	-			G26	PR20B	2	C	
F26	NC	-			F26	PR20A	2	T	
E26	NC	-			E26	PR19B	2	C	
E25	NC	-			E25	PR19A	2	T	
F25	PR9B	2	C	RUM0_PLLC_FB_A	F25	PR17B	2	C	RUM0_PLLC_FB_A
GND	GND2	2			GND	GND2	2		
G25	PR9A	2	T	RUM0_PLLT_FB_A	G25	PR17A	2	T	RUM0_PLLT_FB_A
H23	PR8B	2	C	RUM0_PLLC_IN_A	H23	PR16B	2	C	RUM0_PLLC_IN_A
H24	PR8A	2	T	RUM0_PLLT_IN_A	H24	PR16A	2	T	RUM0_PLLT_IN_A
H21	PR7B	2	C		H21	PR15B	2	C	
G21	PR7A	2	T		G21	PR15A	2	T	
D26	PR6B	2	C		D26	PR14B	2	C	
D25	PR6A	2	T	RDQS6	D25	PR14A	2	T	RDQS14
F21	PR5B	2	C		F21	PR13B	2	C	
-	-	-			GND	GND2	2		
G22	PR5A	2	T		G22	PR13A	2	T	
G24	PR4B	2	C		G24	PR12B	2	C	
G23	PR4A	2	T		G23	PR12A	2	T	
C26	PR3B	2	C		C26	PR11B	2	C	
C25	PR3A	2	T		C25	PR11A	2	T	
F24	NC	-			F24	PR9B	2	C	
-	-	-			GND	GND2	2		
F23	NC	-			F23	PR9A	2	T	

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
E24	NC	-			E24	PR8B	2	C	
D24	NC	-			D24	PR8A	2	T	
E22	NC	-			E22	PR7B	2	C	
F22	NC	-			F22	PR7A	2	T	
E21	NC	-			E21	PR6B	2	C	
D22	NC	-			D22	PR6A	2	T	RDQS6
E23	PR2B	2	C	VREF1_2	E23	PR2B	2	C	VREF1_2
D23	PR2A	2	T	VREF2_2	D23	PR2A	2	T	VREF2_2
GND	GND2	2			GND	GND2	2		
GND	GND1	1			GND	GND1	1		
G20	NC	-			G20	PT65B	1	C	
F20	NC	-			F20	PT65A	1	T	
D21	NC	-			D21	PT64B	1	C	
C21	NC	-			C21	PT64A	1	T	
C23	NC	-			C23	PT63B	1	C	
C22	NC	-			C22	PT63A	1	T	
B23	NC	-			B23	PT62B	1	C	
C24	NC	-			C24	PT62A	1	T	TDQS62
D20	NC	-			D20	PT61B	1	C	
-	-	-			GND	GND1	1		
E19	NC	-			E19	PT61A	1	T	
B25	NC	-			B25	PT60B	1	C	
B24	NC	-			B24	PT60A	1	T	
B26	NC	-			B26	PT59B	1	C	
A25	NC	-			A25	PT59A	1	T	
C20	NC	-			C20	PT58B	1	C	
C19	NC	-			C19	PT58A	1	T	
A24	PT57B	1	C		A24	PT57B	1	C	
-	-	-			GND	GND1	1		
A23	PT57A	1	T		A23	PT57A	1	T	
E18	PT56B	1	C		E18	PT56B	1	C	
D19	PT56A	1	T		D19	PT56A	1	T	
F19	PT55B	1	C		F19	PT55B	1	C	
B22	PT55A	1	T		B22	PT55A	1	T	
G19	PT54B	1	C		G19	PT54B	1	C	
B21	PT54A	1	T	TDQS54	B21	PT54A	1	T	TDQS54
D18	PT53B	1	C		D18	PT53B	1	C	
GND	GND1	1			GND	GND1	1		
C18	PT53A	1	T		C18	PT53A	1	T	
F18	PT52B	1	C		F18	PT52B	1	C	
A22	PT52A	1	T		A22	PT52A	1	T	
G18	PT51B	1	C		G18	PT51B	1	C	

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

LFEC20/LFECP20					LFECP/EC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
A21	PT51A	1	T		A21	PT51A	1	T	
E17	PT50B	1	C		E17	PT50B	1	C	
B17	PT50A	1	T		B17	PT50A	1	T	
C17	PT49B	1	C		C17	PT49B	1	C	
GND	GND1	1			GND	GND1	1		
D17	PT49A	1	T		D17	PT49A	1	T	
F17	PT48B	1	C		F17	PT48B	1	C	
E20	PT48A	1	T		E20	PT48A	1	T	
G17	PT47B	1	C		G17	PT47B	1	C	
B20	PT47A	1	T		B20	PT47A	1	T	
E16	PT46B	1	C		E16	PT46B	1	C	
A20	PT46A	1	T	TDQS46	A20	PT46A	1	T	TDQS46
A19	PT45B	1	C		A19	PT45B	1	C	
GND	GND1	1			GND	GND1	1		
B19	PT45A	1	T		B19	PT45A	1	T	
D16	PT44B	1	C		D16	PT44B	1	C	
C16	PT44A	1	T		C16	PT44A	1	T	
F16	PT43B	1	C		F16	PT43B	1	C	
A18	PT43A	1	T		A18	PT43A	1	T	
G16	PT42B	1	C		G16	PT42B	1	C	
B18	PT42A	1	T		B18	PT42A	1	T	
A17	PT41B	1	C		A17	PT41B	1	C	
GND	GND1	1			GND	GND1	1		
A16	PT41A	1	T		A16	PT41A	1	T	
D15	PT40B	1	C		D15	PT40B	1	C	
B16	PT40A	1	T		B16	PT40A	1	T	
E15	PT39B	1	C		E15	PT39B	1	C	
C15	PT39A	1	T		C15	PT39A	1	T	
F15	PT38B	1	C		F15	PT38B	1	C	
G15	PT38A	1	T	TDQS38	G15	PT38A	1	T	TDQS38
B15	PT37B	1	C		B15	PT37B	1	C	
GND	GND1	1			GND	GND1	1		
A15	PT37A	1	T		A15	PT37A	1	T	
E14	PT36B	1	C		E14	PT36B	1	C	
G14	PT36A	1	T		G14	PT36A	1	T	
D14	PT35B	1	C	VREF2_1	D14	PT35B	1	C	VREF2_1
E13	PT35A	1	T	VREF1_1	E13	PT35A	1	T	VREF1_1
F14	PT34B	1	C		F14	PT34B	1	C	
C14	PT34A	1	T		C14	PT34A	1	T	
B14	PT33B	0	C	PCLKC0_0	B14	PT33B	0	C	PCLKC0_0
GND	GND0	0			GND	GND0	0		
A14	PT33A	0	T	PCLKT0_0	A14	PT33A	0	T	PCLKT0_0

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
			LFEC20/LFECP20					LFECP/EC33	
D13	PT32B	0	C	VREF1_0	D13	PT32B	0	C	VREF1_0
C13	PT32A	0	T	VREF2_0	C13	PT32A	0	T	VREF2_0
A13	PT31B	0	C		A13	PT31B	0	C	
B13	PT31A	0	T		B13	PT31A	0	T	
F13	PT30B	0	C		F13	PT30B	0	C	
F12	PT30A	0	T	TDQS30	F12	PT30A	0	T	TDQS30
A12	PT29B	0	C		A12	PT29B	0	C	
GND	GND0	0			GND	GND0	0		
B12	PT29A	0	T		B12	PT29A	0	T	
A11	PT28B	0	C		A11	PT28B	0	C	
B11	PT28A	0	T		B11	PT28A	0	T	
D12	PT27B	0	C		D12	PT27B	0	C	
C12	PT27A	0	T		C12	PT27A	0	T	
B10	PT26B	0	C		B10	PT26B	0	C	
A10	PT26A	0	T		A10	PT26A	0	T	
G12	PT25B	0	C		G12	PT25B	0	C	
GND	GND0	0			GND	GND0	0		
A9	PT25A	0	T		A9	PT25A	0	T	
E12	PT24B	0	C		E12	PT24B	0	C	
B9	PT24A	0	T		B9	PT24A	0	T	
F11	PT23B	0	C		F11	PT23B	0	C	
A8	PT23A	0	T		A8	PT23A	0	T	
D11	PT22B	0	C		D11	PT22B	0	C	
C11	PT22A	0	T	TDQS22	C11	PT22A	0	T	TDQS22
B8	PT21B	0	C		B8	PT21B	0	C	
GND	GND0	0			GND	GND0	0		
B7	PT21A	0	T		B7	PT21A	0	T	
E11	PT20B	0	C		E11	PT20B	0	C	
A7	PT20A	0	T		A7	PT20A	0	T	
G11	PT19B	0	C		G11	PT19B	0	C	
C7	PT19A	0	T		C7	PT19A	0	T	
G10	PT18B	0	C		G10	PT18B	0	C	
C6	PT18A	0	T		C6	PT18A	0	T	
C10	PT17B	0	C		C10	PT17B	0	C	
GND	GND0	0			GND	GND0	0		
D10	PT17A	0	T		D10	PT17A	0	T	
F10	PT16B	0	C		F10	PT16B	0	C	
A6	PT16A	0	T		A6	PT16A	0	T	
E10	PT15B	0	C		E10	PT15B	0	C	
C9	PT15A	0	T		C9	PT15A	0	T	
G9	PT14B	0	C		G9	PT14B	0	C	
D9	PT14A	0	T	TDQS14	D9	PT14A	0	T	TDQS14

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

LFEC20/LFECP20					LFECP/EC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
A5	PT13B	0	C		A5	PT13B	0	C	
GND	GND0	0			GND	GND0	0		
A4	PT13A	0	T		A4	PT13A	0	T	
F9	PT12B	0	C		F9	PT12B	0	C	
B6	PT12A	0	T		B6	PT12A	0	T	
E9	PT11B	0	C		E9	PT11B	0	C	
C8	PT11A	0	T		C8	PT11A	0	T	
G8	PT10B	0	C		G8	PT10B	0	C	
B5	PT10A	0	T		B5	PT10A	0	T	
A3	PT9B	0	C		A3	PT9B	0	C	
GND	GND0	0			GND	GND0	0		
A2	PT9A	0	T		A2	PT9A	0	T	
F8	PT8B	0	C		F8	PT8B	0	C	
B4	PT8A	0	T		B4	PT8A	0	T	
E8	PT7B	0	C		E8	PT7B	0	C	
B3	PT7A	0	T		B3	PT7A	0	T	
D8	PT6B	0	C		D8	PT6B	0	C	
G7	PT6A	0	T	TDQS6	G7	PT6A	0	T	TDQS6
C4	PT5B	0	C		C4	PT5B	0	C	
C5	PT5A	0	T		C5	PT5A	0	T	
E7	PT4B	0	C		E7	PT4B	0	C	
D4	PT4A	0	T		D4	PT4A	0	T	
F7	PT3B	0	C		F7	PT3B	0	C	
D6	PT3A	0	T		D6	PT3A	0	T	
D7	PT2B	0	C		D7	PT2B	0	C	
E6	PT2A	0	T		E6	PT2A	0	T	
GND	GND0	0			GND	GND0	0		
K10	GND	-			K10	GND	-		
K11	GND	-			K11	GND	-		
K12	GND	-			K12	GND	-		
K13	GND	-			K13	GND	-		
K14	GND	-			K14	GND	-		
K15	GND	-			K15	GND	-		
K16	GND	-			K16	GND	-		
L10	GND	-			L10	GND	-		
L11	GND	-			L11	GND	-		
L12	GND	-			L12	GND	-		
L13	GND	-			L13	GND	-		
L14	GND	-			L14	GND	-		
L15	GND	-			L15	GND	-		
L16	GND	-			L16	GND	-		
L17	GND	-			L17	GND	-		

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

LFEC20/LFECP20					LFECP/EC33				
Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
M10	GND	-			M10	GND	-		
M11	GND	-			M11	GND	-		
M12	GND	-			M12	GND	-		
M13	GND	-			M13	GND	-		
M14	GND	-			M14	GND	-		
M15	GND	-			M15	GND	-		
M16	GND	-			M16	GND	-		
M17	GND	-			M17	GND	-		
N10	GND	-			N10	GND	-		
N11	GND	-			N11	GND	-		
N12	GND	-			N12	GND	-		
N13	GND	-			N13	GND	-		
N14	GND	-			N14	GND	-		
N15	GND	-			N15	GND	-		
N16	GND	-			N16	GND	-		
N17	GND	-			N17	GND	-		
P10	GND	-			P10	GND	-		
P11	GND	-			P11	GND	-		
P12	GND	-			P12	GND	-		
P13	GND	-			P13	GND	-		
P14	GND	-			P14	GND	-		
P15	GND	-			P15	GND	-		
P16	GND	-			P16	GND	-		
P17	GND	-			P17	GND	-		
R10	GND	-			R10	GND	-		
R11	GND	-			R11	GND	-		
R12	GND	-			R12	GND	-		
R13	GND	-			R13	GND	-		
R14	GND	-			R14	GND	-		
R15	GND	-			R15	GND	-		
R16	GND	-			R16	GND	-		
R17	GND	-			R17	GND	-		
T10	GND	-			T10	GND	-		
T11	GND	-			T11	GND	-		
T12	GND	-			T12	GND	-		
T13	GND	-			T13	GND	-		
T14	GND	-			T14	GND	-		
T15	GND	-			T15	GND	-		
T16	GND	-			T16	GND	-		
T17	GND	-			T17	GND	-		
U10	GND	-			U10	GND	-		
U11	GND	-			U11	GND	-		

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
U12	GND	-			U12	GND	-		
U13	GND	-			U13	GND	-		
U14	GND	-			U14	GND	-		
U15	GND	-			U15	GND	-		
U16	GND	-			U16	GND	-		
U17	GND	-			U17	GND	-		
H10	VCC	-			H10	VCC	-		
H11	VCC	-			H11	VCC	-		
H16	VCC	-			H16	VCC	-		
H17	VCC	-			H17	VCC	-		
H18	VCC	-			H18	VCC	-		
H19	VCC	-			H19	VCC	-		
H8	VCC	-			H8	VCC	-		
H9	VCC	-			H9	VCC	-		
J18	VCC	-			J18	VCC	-		
J9	VCC	-			J9	VCC	-		
K8	VCC	-			K8	VCC	-		
L19	VCC	-			L19	VCC	-		
M19	VCC	-			M19	VCC	-		
N7	VCC	-			N7	VCC	-		
R20	VCC	-			R20	VCC	-		
R7	VCC	-			R7	VCC	-		
T19	VCC	-			T19	VCC	-		
V18	VCC	-			V18	VCC	-		
V8	VCC	-			V8	VCC	-		
V9	VCC	-			V9	VCC	-		
W10	VCC	-			W10	VCC	-		
W11	VCC	-			W11	VCC	-		
W16	VCC	-			W16	VCC	-		
W17	VCC	-			W17	VCC	-		
W18	VCC	-			W18	VCC	-		
W19	VCC	-			W19	VCC	-		
W8	VCC	-			W8	VCC	-		
W9	VCC	-			W9	VCC	-		
H12	VCCIO0	0			H12	VCCIO0	0		
H13	VCCIO0	0			H13	VCCIO0	0		
J10	VCCIO0	0			J10	VCCIO0	0		
J11	VCCIO0	0			J11	VCCIO0	0		
J12	VCCIO0	0			J12	VCCIO0	0		
J13	VCCIO0	0			J13	VCCIO0	0		
H14	VCCIO1	1			H14	VCCIO1	1		
H15	VCCIO1	1			H15	VCCIO1	1		

Note: The table header spans LFEC20/LFECP20 for the left half and LFECP/EC33 for the right half.

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
J14	VCCIO1	1			J14	VCCIO1	1		
J15	VCCIO1	1			J15	VCCIO1	1		
J16	VCCIO1	1			J16	VCCIO1	1		
J17	VCCIO1	1			J17	VCCIO1	1		
K17	VCCIO2	2			K17	VCCIO2	2		
K18	VCCIO2	2			K18	VCCIO2	2		
L18	VCCIO2	2			L18	VCCIO2	2		
M18	VCCIO2	2			M18	VCCIO2	2		
N18	VCCIO2	2			N18	VCCIO2	2		
N19	VCCIO2	2			N19	VCCIO2	2		
P18	VCCIO3	3			P18	VCCIO3	3		
P19	VCCIO3	3			P19	VCCIO3	3		
R18	VCCIO3	3			R18	VCCIO3	3		
R19	VCCIO3	3			R19	VCCIO3	3		
T18	VCCIO3	3			T18	VCCIO3	3		
U18	VCCIO3	3			U18	VCCIO3	3		
V14	VCCIO4	4			V14	VCCIO4	4		
V15	VCCIO4	4			V15	VCCIO4	4		
V16	VCCIO4	4			V16	VCCIO4	4		
V17	VCCIO4	4			V17	VCCIO4	4		
W14	VCCIO4	4			W14	VCCIO4	4		
W15	VCCIO4	4			W15	VCCIO4	4		
V10	VCCIO5	5			V10	VCCIO5	5		
V11	VCCIO5	5			V11	VCCIO5	5		
V12	VCCIO5	5			V12	VCCIO5	5		
V13	VCCIO5	5			V13	VCCIO5	5		
W12	VCCIO5	5			W12	VCCIO5	5		
W13	VCCIO5	5			W13	VCCIO5	5		
P8	VCCIO6	6			P8	VCCIO6	6		
P9	VCCIO6	6			P9	VCCIO6	6		
R8	VCCIO6	6			R8	VCCIO6	6		
R9	VCCIO6	6			R9	VCCIO6	6		
T9	VCCIO6	6			T9	VCCIO6	6		
U9	VCCIO6	6			U9	VCCIO6	6		
K9	VCCIO7	7			K9	VCCIO7	7		
L9	VCCIO7	7			L9	VCCIO7	7		
M8	VCCIO7	7			M8	VCCIO7	7		
M9	VCCIO7	7			M9	VCCIO7	7		
N8	VCCIO7	7			N8	VCCIO7	7		
N9	VCCIO7	7			N9	VCCIO7	7		
G13	VCCAUX	-			G13	VCCAUX	-		
H20	VCCAUX	-			H20	VCCAUX	-		

LFECP/EC20, LFECP/EC33 Logic Signal Connections: 672 fpBGA (Cont.)

Ball Number	Ball Function	Bank	LVDS	Dual Function	Ball Number	Ball Function	Bank	LVDS	Dual Function
	LFEC20/LFECP20					LFECP/EC33			
H7	VCCAUX	-			H7	VCCAUX	-		
J19	VCCAUX	-			J19	VCCAUX	-		
J8	VCCAUX	-			J8	VCCAUX	-		
K7	VCCAUX	-			K7	VCCAUX	-		
L20	VCCAUX	-			L20	VCCAUX	-		
M20	VCCAUX	-			M20	VCCAUX	-		
M7	VCCAUX	-			M7	VCCAUX	-		
N20	VCCAUX	-			N20	VCCAUX	-		
P20	VCCAUX	-			P20	VCCAUX	-		
P7	VCCAUX	-			P7	VCCAUX	-		
T20	VCCAUX	-			T20	VCCAUX	-		
T7	VCCAUX	-			T7	VCCAUX	-		
T8	VCCAUX	-			T8	VCCAUX	-		
V19	VCCAUX	-			V19	VCCAUX	-		
V7	VCCAUX	-			V7	VCCAUX	-		
W20	VCCAUX	-			W20	VCCAUX	-		
Y13	VCCAUX	-			Y13	VCCAUX	-		
Y7	VCCAUX	-			Y7	VCCAUX	-		
K19	VCC[1]	-			K19	VCCPLL	-		
L8	VCC[1]	-			L8	VCCPLL	-		
U19	VCC[1]	-			U19	VCCPLL	-		
U8	VCC[1]	-			U8	VCCPLL	-		

1. Tied to V_{CCPLL}.

Thermal Management

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

For Further Information

For further information regarding Thermal Management, refer to the following located on the Lattice website at www.latticesemi.com.

- Thermal Management document

- Technical Note TN1052 - Power Estimation and Management for LatticeECP/EC and LatticeXP Devices

- Power Calculator tool included with Lattice's ispLEVER design tool, or as a standalone download from www.latticesemi.com/software

Part Number Description

LFXXX XX X – X XXXX X

Device Family
  Lattice EC (FPGA)
  Lattice ECP (EC FPGA + DSP Blocks)

Logic Capacity
  1* = 1.5K LUTs
  3* = 3K LUTs
  6 = 6K LUTs
  10 = 10K LUTs
  15 = 15K LUTs
  20 = 20K LUTs
  33 = 33K LUTs

Supply Voltage
  E = 1.2V

Grade
  C = Commercial
  I = Industrial

Package
  T100 = 100-pin TQFP*
  T144 = 144-pin TQFP
  Q208 = 208-pin PQFP
  F256 = 256-ball fpBGA
  F484 = 484-ball fpBGA
  F672 = 672-ball fpBGA

  TN100 = 100-pin Lead-free TQFP*
  TN144 = 144-pin Lead-free TQFP
  QN208 = 208-pin Lead-free PQFP
  FN256 = 256-ball Lead-free fpBGA
  FN484 = 484-ball Lead-free fpBGA
  FN672 = 672-ball Lead-free fpBGA

Speed
  3 = Slowest
  4
  5 = Fastest

*Not available in the LatticeECP Family.

Ordering Information

Note: LatticeECP/EC devices are dual marked. For example, the commercial speed grade LFEC20E-4F484C is also marked with industrial grade -3I (LFEC20E-3F484I). The commercial grade is one speed grade faster than the associated dual mark industrial grade. The slowest commercial speed grade does not have industrial markings. The markings appear as follows:

Lattice
EC

LFEC20E-
4F484C-3I

Datecode
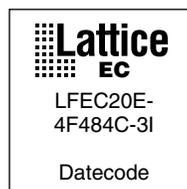
Conventional Packaging

LatticeEC Commercial

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC1E-3Q208C	112	-3	PQFP	208	COM	1.5K
LFEC1E-4Q208C	112	-4	PQFP	208	COM	1.5K
LFEC1E-5Q208C	112	-5	PQFP	208	COM	1.5K
LFEC1E-3T144C	97	-3	TQFP	144	COM	1.5K
LFEC1E-4T144C	97	-4	TQFP	144	COM	1.5K
LFEC1E-5T144C	97	-5	TQFP	144	COM	1.5K
LFEC1E-3T100C	67	-3	TQFP	100	COM	1.5K
LFEC1E-4T100C	67	-4	TQFP	100	COM	1.5K
LFEC1E-5T100C	67	-5	TQFP	100	COM	1.5K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC3E-3F256C	160	-3	fpBGA	256	COM	3.1K
LFEC3E-4F256C	160	-4	fpBGA	256	COM	3.1K
LFEC3E-5F256C	160	-5	fpBGA	256	COM	3.1K
LFEC3E-3Q208C	145	-3	PQFP	208	COM	3.1K
LFEC3E-4Q208C	145	-4	PQFP	208	COM	3.1K
LFEC3E-5Q208C	145	-5	PQFP	208	COM	3.1K
LFEC3E-3T144C	97	-3	TQFP	144	COM	3.1K
LFEC3E-4T144C	97	-4	TQFP	144	COM	3.1K
LFEC3E-5T144C	97	-5	TQFP	144	COM	3.1K
LFEC3E-3T100C	67	-3	TQFP	100	COM	3.1K
LFEC3E-4T100C	67	-4	TQFP	100	COM	3.1K
LFEC3E-5T100C	67	-5	TQFP	100	COM	3.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC6E-3F484C	224	-3	fpBGA	484	COM	6.1K
LFEC6E-4F484C	224	-4	fpBGA	484	COM	6.1K
LFEC6E-5F484C	224	-5	fpBGA	484	COM	6.1K
LFEC6E-3F256C	195	-3	fpBGA	256	COM	6.1K
LFEC6E-4F256C	195	-4	fpBGA	256	COM	6.1K
LFEC6E-5F256C	195	-5	fpBGA	256	COM	6.1K
LFEC6E-3Q208C	147	-3	PQFP	208	COM	6.1K
LFEC6E-4Q208C	147	-4	PQFP	208	COM	6.1K
LFEC6E-5Q208C	147	-5	PQFP	208	COM	6.1K
LFEC6E-3T144C	97	-3	TQFP	144	COM	6.1K
LFEC6E-4T144C	97	-4	TQFP	144	COM	6.1K
LFEC6E-5T144C	97	-5	TQFP	144	COM	6.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC10E-3F484C	288	-3	fpBGA	484	COM	10.2K
LFEC10E-4F484C	288	-4	fpBGA	484	COM	10.2K
LFEC10E-5F484C	288	-5	fpBGA	484	COM	10.2K
LFEC10E-3F256C	195	-3	fpBGA	256	COM	10.2K

LatticeEC Commercial (Continued)

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC10E-4F256C	195	-4	fpBGA	256	COM	10.2K
LFEC10E-5F256C	195	-5	fpBGA	256	COM	10.2K
LFEC10E-3Q208C	147	-3	PQFP	208	COM	10.2K
LFEC10E-4Q208C	147	-4	PQFP	208	COM	10.2K
LFEC10E-5Q208C	147	-5	PQFP	208	COM	10.2K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC15E-3F484C	352	-3	fpBGA	484	COM	15.3K
LFEC15E-4F484C	352	-4	fpBGA	484	COM	15.3K
LFEC15E-5F484C	352	-5	fpBGA	484	COM	15.3K
LFEC15E-3F256C	195	-3	fpBGA	256	COM	15.3K
LFEC15E-4F256C	195	-4	fpBGA	256	COM	15.3K
LFEC15E-5F256C	195	-5	fpBGA	256	COM	15.3K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC20E-3F672C	400	-3	fpBGA	672	COM	19.7K
LFEC20E-4F672C	400	-4	fpBGA	672	COM	19.7K
LFEC20E-5F672C	400	-5	fpBGA	672	COM	19.7K
LFEC20E-3F484C	360	-3	fpBGA	484	COM	19.7K
LFEC20E-4F484C	360	-4	fpBGA	484	COM	19.7K
LFEC20E-5F484C	360	-5	fpBGA	484	COM	19.7K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC33E-3F672C	496	-3	fpBGA	672	COM	32.8K
LFEC33E-4F672C	496	-4	fpBGA	672	COM	32.8K
LFEC33E-5F672C	496	-5	fpBGA	672	COM	32.8K
LFEC33E-3F484C	360	-3	fpBGA	484	COM	32.8K
LFEC33E-4F484C	360	-4	fpBGA	484	COM	32.8K
LFEC33E-5F484C	360	-5	fpBGA	484	COM	32.8K

LatticeECP Commercial

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP6E-3F484C	224	-3	fpBGA	484	COM	6.1K
LFECP6E-4F484C	224	-4	fpBGA	484	COM	6.1K
LFECP6E-5F484C	224	-5	fpBGA	484	COM	6.1K
LFECP6E-3F256C	195	-3	fpBGA	256	COM	6.1K
LFECP6E-4F256C	195	-4	fpBGA	256	COM	6.1K
LFECP6E-5F256C	195	-5	fpBGA	256	COM	6.1K
LFECP6E-3Q208C	147	-3	PQFP	208	COM	6.1K
LFECP6E-4Q208C	147	-4	PQFP	208	COM	6.1K
LFECP6E-5Q208C	147	-5	PQFP	208	COM	6.1K
LFECP6E-3T144C	97	-3	TQFP	144	COM	6.1K
LFECP6E-4T144C	97	-4	TQFP	144	COM	6.1K
LFECP6E-5T144C	97	-5	TQFP	144	COM	6.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP10E-3F484C	288	-3	fpBGA	484	COM	10.2K
LFECP10E-4F484C	288	-4	fpBGA	484	COM	10.2K
LFECP10E-5F484C	288	-5	fpBGA	484	COM	10.2K
LFECP10E-3F256C	195	-3	fpBGA	256	COM	10.2K
LFECP10E-4F256C	195	-4	fpBGA	256	COM	10.2K
LFECP10E-5F256C	195	-5	fpBGA	256	COM	10.2K
LFECP10E-3Q208C	147	-3	PQFP	208	COM	10.2K
LFECP10E-4Q208C	147	-4	PQFP	208	COM	10.2K
LFECP10E-5Q208C	147	-5	PQFP	208	COM	10.2K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP15E-3F484C	352	-3	fpBGA	484	COM	15.3K
LFECP15E-4F484C	352	-4	fpBGA	484	COM	15.3K
LFECP15E-5F484C	352	-5	fpBGA	484	COM	15.3K
LFECP15E-3F256C	195	-3	fpBGA	256	COM	15.3K
LFECP15E-4F256C	195	-4	fpBGA	256	COM	15.3K
LFECP15E-5F256C	195	-5	fpBGA	256	COM	15.3K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP20E-3F672C	400	-3	fpBGA	672	COM	19.7K
LFECP20E-4F672C	400	-4	fpBGA	672	COM	19.7K
LFECP20E-5F672C	400	-5	fpBGA	672	COM	19.7K
LFECP20E-3F484C	360	-3	fpBGA	484	COM	19.7K
LFECP20E-4F484C	360	-4	fpBGA	484	COM	19.7K
LFECP20E-5F484C	360	-5	fpBGA	484	COM	19.7K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP33E-3F672C	496	-3	fpBGA	672	COM	32.8K
LFECP33E-4F672C	496	-4	fpBGA	672	COM	32.8K
LFECP33E-5F672C	496	-5	fpBGA	672	COM	32.8K

LatticeECP Commercial (Continued)

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP33E-3F484C	360	-3	fpBGA	484	COM	32.8K
LFECP33E-4F484C	360	-4	fpBGA	484	COM	32.8K
LFECP33E-5F484C	360	-5	fpBGA	484	COM	32.8K

LatticeEC Industrial

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC1E-3Q208I	112	-3	PQFP	208	IND	1.5K
LFEC1E-4Q208I	112	-4	PQFP	208	IND	1.5K
LFEC1E-3T144I	97	-3	TQFP	144	IND	1.5K
LFEC1E-4T144I	97	-4	TQFP	144	IND	1.5K
LFEC1E-3T100I	67	-3	TQFP	100	IND	1.5K
LFEC1E-4T100I	67	-4	TQFP	100	IND	1.5K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC3E-3F256I	160	-3	fpBGA	256	IND	3.1K
LFEC3E-4F256I	160	-4	fpBGA	256	IND	3.1K
LFEC3E-3Q208I	145	-3	PQFP	208	IND	3.1K
LFEC3E-4Q208I	145	-4	PQFP	208	IND	3.1K
LFEC3E-3T144I	97	-3	TQFP	144	IND	3.1K
LFEC3E-4T144I	97	-4	TQFP	144	IND	3.1K
LFEC3E-3T100I	67	-3	TQFP	100	IND	3.1K
LFEC3E-4T100I	67	-4	TQFP	100	IND	3.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC6E-3F484I	224	-3	fpBGA	484	IND	6.1K
LFEC6E-4F484I	224	-4	fpBGA	484	IND	6.1K
LFEC6E-3F256I	195	-3	fpBGA	256	IND	6.1K
LFEC6E-4F256I	195	-4	fpBGA	256	IND	6.1K
LFEC6E-3Q208I	147	-3	PQFP	208	IND	6.1K
LFEC6E-4Q208I	147	-4	PQFP	208	IND	6.1K
LFEC6E-3T144I	97	-3	TQFP	144	IND	6.1K
LFEC6E-4T144I	97	-4	TQFP	144	IND	6.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC10E-3F484I	288	-3	fpBGA	484	IND	10.2K
LFEC10E-4F484I	288	-4	fpBGA	484	IND	10.2K
LFEC10E-3F256I	195	-3	fpBGA	256	IND	10.2K
LFEC10E-4F256I	195	-4	fpBGA	256	IND	10.2K
LFEC10E-3 P208I	147	-3	PQFP	208	IND	10.2K
LFEC10E-4 P208I	147	-4	PQFP	208	IND	10.2K

LatticeEC Industrial (Continued)

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC15E-3F484I	352	-3	fpBGA	484	IND	15.3K
LFEC15E-4F484I	352	-4	fpBGA	484	IND	15.3K
LFEC15E-3F256I	195	-3	fpBGA	256	IND	15.3K
LFEC15E-4F256I	195	-4	fpBGA	256	IND	15.3K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC20E-3F672I	400	-3	fpBGA	672	IND	19.7K
LFEC20E-4F672I	400	-4	fpBGA	672	IND	19.7K
LFEC20E-3F484I	360	-3	fpBGA	484	IND	19.7K
LFEC20E-4F484I	360	-4	fpBGA	484	IND	19.7K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC33E-3F672I	496	-3	fpBGA	672	IND	32.8
LFEC33E-4F672I	496	-4	fpBGA	672	IND	32.8
LFEC33E-3F484I	360	-3	fpBGA	484	IND	32.8
LFEC33E-4F484I	360	-4	fpBGA	484	IND	32.8

LatticeECP Industrial

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP6E-3F484I	224	-3	fpBGA	484	IND	6.1K
LFECP6E-4F484I	224	-4	fpBGA	484	IND	6.1K
LFECP6E-3F256I	195	-3	fpBGA	256	IND	6.1K
LFECP6E-4F256I	195	-4	fpBGA	256	IND	6.1K
LFECP6E-3Q208I	147	-3	PQFP	208	IND	6.1K
LFECP6E-4Q208I	147	-4	PQFP	208	IND	6.1K
LFECP6E-3T144I	97	-3	TQFP	144	IND	6.1K
LFECP6E-4T144I	97	-4	TQFP	144	IND	6.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP10E-3F484I	288	-3	fpBGA	484	IND	10.2K
LFECP10E-4F484I	288	-4	fpBGA	484	IND	10.2K
LFECP10E-3F256I	195	-3	fpBGA	256	IND	10.2K
LFECP10E-4F256I	195	-4	fpBGA	256	IND	10.2K
LFECP10E-3Q208I	147	-3	PQFP	208	IND	10.2K
LFECP10E-4Q208I	147	-4	PQFP	208	IND	10.2K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP15E-3F484I	352	-3	fpBGA	484	IND	15.3K
LFECP15E-4F484I	352	-4	fpBGA	484	IND	15.3K
LFECP15E-3F256I	195	-3	fpBGA	256	IND	15.3K
LFECP15E-4F256I	195	-4	fpBGA	256	IND	15.3K

LatticeECP Industrial (Continued)

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP20E-3F672I	400	-3	fpBGA	672	IND	19.7K
LFECP20E-4F672I	400	-4	fpBGA	672	IND	19.7K
LFECP20E-3F484I	360	-3	fpBGA	484	IND	19.7K
LFECP20E-4F484I	360	-4	fpBGA	484	IND	19.7K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP33E-3F672I	496	-3	fpBGA	672	IND	32.8K
LFECP33E-4F672I	496	-4	fpBGA	672	IND	32.8K
LFECP33E-3F484I	360	-3	fpBGA	484	IND	32.8K
LFECP33E-4F484I	360	-4	fpBGA	484	IND	32.8K

Lead-Free Packaging

LatticeEC Commercial

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC1E-3QN208C	112	-3	Lead-Free PQFP	208	COM	1.5K
LFEC1E-4QN208C	112	-4	Lead-Free PQFP	208	COM	1.5K
LFEC1E-5QN208C	112	-5	Lead-Free PQFP	208	COM	1.5K
LFEC1E-3TN144C	97	-3	Lead-Free TQFP	144	COM	1.5K
LFEC1E-4TN144C	97	-4	Lead-Free TQFP	144	COM	1.5K
LFEC1E-5TN144C	97	-5	Lead-Free TQFP	144	COM	1.5K
LFEC1E-3TN100C	67	-3	Lead-Free TQFP	100	COM	1.5K
LFEC1E-4TN100C	67	-4	Lead-Free TQFP	100	COM	1.5K
LFEC1E-5TN100C	67	-5	Lead-Free TQFP	100	COM	1.5K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC3E-3FN256C	160	-3	Lead-Free fpBGA	256	COM	3.1K
LFEC3E-4FN256C	160	-4	Lead-Free fpBGA	256	COM	3.1K
LFEC3E-5FN256C	160	-5	Lead-Free fpBGA	256	COM	3.1K
LFEC3E-3QN208C	145	-3	Lead-Free PQFP	208	COM	3.1K
LFEC3E-4QN208C	145	-4	Lead-Free PQFP	208	COM	3.1K
LFEC3E-5QN208C	145	-5	Lead-Free PQFP	208	COM	3.1K
LFEC3E-3TN144C	97	-3	Lead-Free TQFP	144	COM	3.1K
LFEC3E-4TN144C	97	-4	Lead-Free TQFP	144	COM	3.1K
LFEC3E-5TN144C	97	-5	Lead-Free TQFP	144	COM	3.1K
LFEC3E-3TN100C	67	-3	Lead-Free TQFP	100	COM	3.1K
LFEC3E-4TN100C	67	-4	Lead-Free TQFP	100	COM	3.1K
LFEC3E-5TN100C	67	-5	Lead-Free TQFP	100	COM	3.1K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC6E-3FN484C	224	-3	Lead-Free fpBGA	484	COM	6.1K
LFEC6E-4FN484C	224	-4	Lead-Free fpBGA	484	COM	6.1K
LFEC6E-5FN484C	224	-5	Lead-Free fpBGA	484	COM	6.1K
LFEC6E-3FN256C	195	-3	Lead-Free fpBGA	256	COM	6.1K
LFEC6E-4FN256C	195	-4	Lead-Free fpBGA	256	COM	6.1K
LFEC6E-5FN256C	195	-5	Lead-Free fpBGA	256	COM	6.1K
LFEC6E-3QN208C	147	-3	Lead-Free PQFP	208	COM	6.1K
LFEC6E-4QN208C	147	-4	Lead-Free PQFP	208	COM	6.1K
LFEC6E-5QN208C	147	-5	Lead-Free PQFP	208	COM	6.1K
LFEC6E-3TN144C	97	-3	Lead-Free TQFP	144	COM	6.1K
LFEC6E-4TN144C	97	-4	Lead-Free TQFP	144	COM	6.1K
LFEC6E-5TN144C	97	-5	Lead-Free TQFP	144	COM	6.1K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC10E-3FN484C	288	-3	Lead-Free fpBGA	484	COM	10.2K
LFEC10E-4FN484C	288	-4	Lead-Free fpBGA	484	COM	10.2K
LFEC10E-5FN484C	288	-5	Lead-Free fpBGA	484	COM	10.2K
LFEC10E-3FN256C	195	-3	Lead-Free fpBGA	256	COM	10.2K

LatticeEC Commercial (Continued)

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC10E-4FN256C	195	-4	Lead-Free fpBGA	256	COM	10.2K
LFEC10E-5FN256C	195	-5	Lead-Free fpBGA	256	COM	10.2K
LFEC10E-3QN208C	147	-3	Lead-Free PQFP	208	COM	10.2K
LFEC10E-4QN208C	147	-4	Lead-Free PQFP	208	COM	10.2K
LFEC10E-5QN208C	147	-5	Lead-Free PQFP	208	COM	10.2K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC15E-3FN484C	352	-3	Lead-Free fpBGA	484	COM	15.3K
LFEC15E-4FN484C	352	-4	Lead-Free fpBGA	484	COM	15.3K
LFEC15E-5FN484C	352	-5	Lead-Free fpBGA	484	COM	15.3K
LFEC15E-3FN256C	195	-3	Lead-Free fpBGA	256	COM	15.3K
LFEC15E-4FN256C	195	-4	Lead-Free fpBGA	256	COM	15.3K
LFEC15E-5FN256C	195	-5	Lead-Free fpBGA	256	COM	15.3K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC20E-3FN672C	400	-3	Lead-Free fpBGA	672	COM	19.7K
LFEC20E-4FN672C	400	-4	Lead-Free fpBGA	672	COM	19.7K
LFEC20E-5FN672C	400	-5	Lead-Free fpBGA	672	COM	19.7K
LFEC20E-3FN484C	360	-3	Lead-Free fpBGA	484	COM	19.7K
LFEC20E-4FN484C	360	-4	Lead-Free fpBGA	484	COM	19.7K
LFEC20E-5FN484C	360	-5	Lead-Free fpBGA	484	COM	19.7K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC33E-3FN672C	496	-3	Lead-Free fpBGA	672	COM	32.8K
LFEC33E-4FN672C	496	-4	Lead-Free fpBGA	672	COM	32.8K
LFEC33E-5FN672C	496	-5	Lead-Free fpBGA	672	COM	32.8K
LFEC33E-3FN484C	360	-3	Lead-Free fpBGA	484	COM	32.8K
LFEC33E-4FN484C	360	-4	Lead-Free fpBGA	484	COM	32.8K
LFEC33E-5FN484C	360	-5	Lead-Free fpBGA	484	COM	32.8K

LatticeECP Commercial

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP6E-3FN484C	224	-3	Lead-Free fpBGA	484	COM	6.1K
LFECP6E-4FN484C	224	-4	Lead-Free fpBGA	484	COM	6.1K
LFECP6E-5FN484C	224	-5	Lead-Free fpBGA	484	COM	6.1K
LFECP6E-3FN256C	195	-3	Lead-Free fpBGA	256	COM	6.1K
LFECP6E-4FN256C	195	-4	Lead-Free fpBGA	256	COM	6.1K
LFECP6E-5FN256C	195	-5	Lead-Free fpBGA	256	COM	6.1K
LFECP6E-3QN208C	147	-3	Lead-Free PQFP	208	COM	6.1K
LFECP6E-4QN208C	147	-4	Lead-Free PQFP	208	COM	6.1K
LFECP6E-5QN208C	147	-5	Lead-Free PQFP	208	COM	6.1K
LFECP6E-3TN144C	97	-3	Lead-Free TQFP	144	COM	6.1K
LFECP6E-4TN144C	97	-4	Lead-Free TQFP	144	COM	6.1K
LFECP6E-5TN144C	97	-5	Lead-Free TQFP	144	COM	6.1K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP10E-3FN484C	288	-3	Lead-Free fpBGA	484	COM	10.2K
LFECP10E-4FN484C	288	-4	Lead-Free fpBGA	484	COM	10.2K
LFECP10E-5FN484C	288	-5	Lead-Free fpBGA	484	COM	10.2K
LFECP10E-3FN256C	195	-3	Lead-Free fpBGA	256	COM	10.2K
LFECP10E-4FN256C	195	-4	Lead-Free fpBGA	256	COM	10.2K
LFECP10E-5FN256C	195	-5	Lead-Free fpBGA	256	COM	10.2K
LFECP10E-3QN208C	147	-3	Lead-Free PQFP	208	COM	10.2K
LFECP10E-4QN208C	147	-4	Lead-Free PQFP	208	COM	10.2K
LFECP10E-5QN208C	147	-5	Lead-Free PQFP	208	COM	10.2K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP15E-3FN484C	352	-3	Lead-Free fpBGA	484	COM	15.3K
LFECP15E-4FN484C	352	-4	Lead-Free fpBGA	484	COM	15.3K
LFECP15E-5FN484C	352	-5	Lead-Free fpBGA	484	COM	15.3K
LFECP15E-3FN256C	195	-3	Lead-Free fpBGA	256	COM	15.3K
LFECP15E-4FN256C	195	-4	Lead-Free fpBGA	256	COM	15.3K
LFECP15E-5FN256C	195	-5	Lead-Free fpBGA	256	COM	15.3K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP20E-3FN672C	400	-3	Lead-Free fpBGA	672	COM	19.7K
LFECP20E-4FN672C	400	-4	Lead-Free fpBGA	672	COM	19.7K
LFECP20E-5FN672C	400	-5	Lead-Free fpBGA	672	COM	19.7K
LFECP20E-3FN484C	400	-3	Lead-Free fpBGA	484	COM	19.7K
LFECP20E-4FN484C	400	-4	Lead-Free fpBGA	484	COM	19.7K
LFECP20E-5FN484C	400	-5	Lead-Free fpBGA	484	COM	19.7K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP33E-3FN672C	496	-3	Lead-Free fpBGA	672	COM	32.8K
LFECP33E-4FN672C	496	-4	Lead-Free fpBGA	672	COM	32.8K
LFECP33E-5FN672C	496	-5	Lead-Free fpBGA	672	COM	32.8K

LatticeECP Commercial (Continued)

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP33E-3FN484C	360	-3	Lead-Free fpBGA	484	COM	32.8K
LFECP33E-4FN484C	360	-4	Lead-Free fpBGA	484	COM	32.8K
LFECP33E-5FN484C	360	-5	Lead-Free fpBGA	484	COM	32.8K

LatticeEC Industrial

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC1E-3QN208I	112	-3	Lead-Free PQFP	208	IND	1.5K
LFEC1E-4QN208I	112	-4	Lead-Free PQFP	208	IND	1.5K
LFEC1E-3TN144I	97	-3	Lead-Free TQFP	144	IND	1.5K
LFEC1E-4TN144I	97	-4	Lead-Free TQFP	144	IND	1.5K
LFEC1E-3TN100I	67	-3	Lead-Free TQFP	100	IND	1.5K
LFEC1E-4TN100I	67	-4	Lead-Free TQFP	100	IND	1.5K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC3E-3FN256I	160	-3	Lead-Free fpBGA	256	IND	3.1K
LFEC3E-4FN256I	160	-4	Lead-Free fpBGA	256	IND	3.1K
LFEC3E-3QN208I	145	-3	Lead-Free PQFP	208	IND	3.1K
LFEC3E-4QN208I	145	-4	Lead-Free PQFP	208	IND	3.1K
LFEC3E-3TN144I	97	-3	Lead-Free TQFP	144	IND	3.1K
LFEC3E-4TN144I	97	-4	Lead-Free TQFP	144	IND	3.1K
LFEC3E-3TN100I	67	-3	Lead-Free TQFP	100	IND	3.1K
LFEC3E-4TN100I	67	-4	Lead-Free TQFP	100	IND	3.1K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC6E-3FN484I	224	-3	Lead-Free fpBGA	484	IND	6.1K
LFEC6E-4FN484I	224	-4	Lead-Free fpBGA	484	IND	6.1K
LFEC6E-3FN256I	195	-3	Lead-Free fpBGA	256	IND	6.1K
LFEC6E-4FN256I	195	-4	Lead-Free fpBGA	256	IND	6.1K
LFEC6E-3QN208I	147	-3	Lead-Free PQFP	208	IND	6.1K
LFEC6E-4QN208I	147	-4	Lead-Free PQFP	208	IND	6.1K
LFEC6E-3TN144I	97	-3	Lead-Free TQFP	144	IND	6.1K
LFEC6E-4TN144I	97	-4	Lead-Free TQFP	144	IND	6.1K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC10E-3FN484I	288	-3	Lead-Free fpBGA	484	IND	10.2K
LFEC10E-4FN484I	288	-4	Lead-Free fpBGA	484	IND	10.2K
LFEC10E-3FN256I	195	-3	Lead-Free fpBGA	256	IND	10.2K
LFEC10E-4FN256I	195	-4	Lead-Free fpBGA	256	IND	10.2K
LFEC10E-3QN208I	147	-3	Lead-Free PQFP	208	IND	10.2K
LFEC10E-4QN208I	147	-4	Lead-Free PQFP	208	IND	10.2K

LatticeEC Industrial (Continued)

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC15E-3FN484I	352	-3	Lead-Free fpBGA	484	IND	15.3K
LFEC15E-4FN484I	352	-4	Lead-Free fpBGA	484	IND	15.3K
LFEC15E-3FN256I	195	-3	Lead-Free fpBGA	256	IND	15.3K
LFEC15E-4FN256I	195	-4	Lead-Free fpBGA	256	IND	15.3K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC20E-3FN672I	400	-3	Lead-Free fpBGA	672	IND	19.7K
LFEC20E-4FN672I	400	-4	Lead-Free fpBGA	672	IND	19.7K
LFEC20E-3FN484I	400	-3	Lead-Free fpBGA	484	IND	19.7K
LFEC20E-4FN484I	400	-4	Lead-Free fpBGA	484	IND	19.7K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFEC33E-3FN672I	496	-3	Lead-Free fpBGA	672	IND	32.8K
LFEC33E-4FN672I	496	-4	Lead-Free fpBGA	672	IND	32.8K
LFEC33E-3FN484I	360	-3	Lead-Free fpBGA	484	IND	32.8K
LFEC33E-4FN484I	360	-4	Lead-Free fpBGA	484	IND	32.8K

LatticeECP Industrial

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP6E-3FN484I	224	-3	Lead-Free fpBGA	484	IND	6.1K
LFECP6E-4FN484I	224	-4	Lead-Free fpBGA	484	IND	6.1K
LFECP6E-3FN256I	195	-3	Lead-Free fpBGA	256	IND	6.1K
LFECP6E-4FN256I	195	-4	Lead-Free fpBGA	256	IND	6.1K
LFECP6E-3QN208I	147	-3	Lead-Free PQFP	208	IND	6.1K
LFECP6E-4QN208I	147	-4	Lead-Free PQFP	208	IND	6.1K
LFECP6E-3TN144I	97	-3	Lead-Free TQFP	144	IND	6.1K
LFECP6E-4TN144I	97	-4	Lead-Free TQFP	144	IND	6.1K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP10E-3FN484I	288	-3	Lead-Free fpBGA	484	IND	10.2K
LFECP10E-4FN484I	288	-4	Lead-Free fpBGA	484	IND	10.2K
LFECP10E-3FN256I	195	-3	Lead-Free fpBGA	256	IND	10.2K
LFECP10E-4FN256I	195	-4	Lead-Free fpBGA	256	IND	10.2K
LFECP10E-3QN208I	147	-3	Lead-Free PQFP	208	IND	10.2K
LFECP10E-4QN208I	147	-4	Lead-Free PQFP	208	IND	10.2K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP15E-3FN484I	352	-3	Lead-Free fpBGA	484	IND	15.3K
LFECP15E-4FN484I	352	-4	Lead-Free fpBGA	484	IND	15.3K
LFECP15E-3FN256I	195	-3	Lead-Free fpBGA	256	IND	15.3K
LFECP15E-4FN256I	195	-4	Lead-Free fpBGA	256	IND	15.3K

LatticeECP Industrial (Continued)

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP20E-3FN672I	400	-3	Lead-Free fpBGA	672	IND	19.7K
LFECP20E-4FN672I	400	-4	Lead-Free fpBGA	672	IND	19.7K
LFECP20E-3FN484I	400	-3	Lead-Free fpBGA	484	IND	19.7K
LFECP20E-4FN484I	400	-4	Lead-Free fpBGA	484	IND	19.7K

Part Number	I/Os	Grade	Package	Pins/Balls	Temp.	LUTs
LFECP33E-3FN672I	496	-3	Lead-Free fpBGA	672	IND	32.8K
LFECP33E-4FN672I	496	-4	Lead-Free fpBGA	672	IND	32.8K
LFECP33E-3FN484I	360	-3	Lead-Free fpBGA	484	IND	32.8K
LFECP33E-4FN484I	360	-4	Lead-Free fpBGA	484	IND	32.8K

For Further Information

A variety of technical notes for the LatticeECP/EC family are available on the Lattice web site at www.latticesemi.com.

- LatticeECP/EC sysIO Usage Guide (TN1056)
- LatticeECP/EC sysCLOCK PLL Design and Usage Guide (TN1049)
- Memory Usage Guide for LatticeECP/EC Devices (TN1051)
- LatticeECP/EC DDR Usage Guide (TN1050)
- Power Estimation and Management for LatticeECP/EC and LatticeXP Devices (TN1052)
- LatticeECP-DSP sysDSP Usage Guide (TN1057)
- LatticeECP/EC sysCONFIG Usage Guide (TN1053)
- IEEE 1149.1 Boundary Scan Testability in Lattice Devices

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

- JEDEC Standards (LVTTL, LVCMOS, SSTL, HSTL): www.jedec.org
- PCI: ww.pcisig.com

Revision History

Date	Version	Section	Change Summary
June 2004	01.0	—	Initial release.
August 2004	01.1	Introduction	Added new device LFECP/LFEC33 in Table 1-1.
		Architecture	Added New device LFECP/LFEC33 in Tables 2-9, 2-10 and 2-11.
		DC & Switching Characteristics	Added New device LFECP/LFEC33 on Supply current (Standby) tables.
			Added New device LFECP/LFEC33 on Initialization Supply current tables.
		Ordering Information	Added 33K Logic Capacity Device in Part Number Description section.
			Added EC33, ECP33 device: Industrial and Commercial to Part Number table.
			Corrected I/O counts in the part number tables for 100/144 TQFP and 208 PQFP packages to match Table 1-1 on page 1.
November 2004	01.3	Introduction	Changed DDR333 (166MHz) to DDR400 (200MHz)
			Added "RSDS" offering to the Features list: Flexible I/O Buffer
		Architecture	Added information about Secondary Clock Sources
			Added information about DCS
			Added a section on "Recommended Power-up Sequence"
			Updated Figure 2-24 "DQS Routing"
			Added DSP Block performance numbers to Table 2-11
			Added another row for RSDS in Table 2-13 and Table 2-14
		DC & Switching Characteristics	Updated new timing numbers
			Added numbers to derating table
			Added DC conditions to RSDS table
			Changed LVDS Max. V_{CCIO} to 2.625
			Added a row for RSDS in "Operating Condition" table
			Updated standby and initialization current table
			Added figure 3-12: sysConfig SPI port sequence
			Added DDR Timing Table and DDR Timings Figure 3-6
		Pinout Information	Added LFECP/EC6 to Pin Information
			Added LFECP/EC6 to Power Supply and NC Connections
			Added LFECP/EC6 144 TQFP Logic Signal Connections
			Added LFECP/EC6 208 PQFP Logic Signal Connections
			Added LFECP/EC6 256 fpBGA Logic Signal Connections
			Added LFECP/EC6 484 fpBGA Logic Signal Connections
		Ordering Information	Added 33K Logic Capacity Device in Part Number Description section.
			Added Part Number table for Commercial EC33.
			Added Part Number table for Commercial ECP33.
			Added Part Number table for Industrial EC33.
			Added Part Number table for Industrial ECP33.

Date	Version	Section	Change Summary
December 2004	01.4	Architecture	Updated Hot Socketing Recommended Power Up Sequence section.
		Pinout Information	Added LFEC1, LFEC3, LFECP/EC10, LFECP/EC15 to Pin Information
			Added LFEC1, LFEC3, LFECP/EC10, LFECP/EC15 to Power Supply and NC Connections
			Added LFEC1 and LFEC3 100 TQFP Pinout
			Added LFEC1 and LFEC3 144 TQFP Pinout
			Added LFEC1, LFEC3 and LFECP/EC10 208 PQFP Pinout
			Added LFEC3, LFECP/EC10 and LFECP/EC15 256 fpBGA Pinout
			Added LFECP/EC10 and LFECP/EC15 484 fpBGA Pinout
		Ordering Information	Added Lead-Free Package Designators
			Added Lead-Free Ordering Part Numbers
		Supplemental Information	Updated list of technical notes.
April 2005	01.5	Architecture	EBR memory support section has been updated with clarification.
			Updated sysIO buffer pair section.
		DC & Switching Characteristics	Hot Socketing Specification has been updated.
			DC Electrical Characteristics table (I_{IL}, I_{IH}) has been updated.
			Supply Current (Standby) table has been updated.
			Initialization Supply Current table has been updated.
			External Switching Characteristics section has been updated.
			Removed t_{RSTW} spec. from PLL Parameter table.
			t_{RST} specifications have been updated.
			sysCONFIG Port Timing Specifications (t_{BSCL}, t_{IODISS}, t_{PRGMRJ}) have been updated.
		Pinout Information	Added LFECP/EC33 Pinout Information
			Pin Information Summary table has been updated.
			Power Supply and NC Connection table has been updated.
			484-fpBGA logic connection has been updated (Ball # J6, J17, P6 and P17 for ECP/EC33 are now called VCCPLL).
			672-fpBGA logic connection has been updated (Ball # K19, L8, U19, U8 for ECP/EC33 are now called VCCPLL).
May 2005	01.6	Introduction	ECP/EC33 EBR SRAM Bits and Blocks have been updated to 498K and 54 respectively.
		Architecture	Table 2-10 has been updated (ECP/EC33 EBR SRAM Bits and Blocks have been updated to 498K and 54 respectively.)
			Recommended Power Up Sequence section has been removed.
		DC & Switching Characteristics	Supply Current (Standby) table has been updated.
			Initialization Supply Current table has been updated.
			Vos test condition has been updated to (VOP+VOM)/2.
			Register-to-Register performance table has been updated (rev. G 0.27).
			External switching characteristics have been updated (rev. G 0.27).
			Internal timing parameters have been updated (rev. G 0.27).
			Timing adders have been updated (rev. G 0.27).
			sysCONFIG port timing specifications have been updated.
		Pinout Information	Pin Information Summary table has been updated.
			Power Supply and NC Connection table has been updated.
		Ordering Information	OPN list has been updated.

Date	Version	Section	Change Summary
September 2005	02.0	Architecture	sysIO section has been updated.
		DC & Switching Characteristics	Recommended Operating Conditions has been updated with V_{CCPLL}.
			DC Electrical Characteristics table has been updated
			Removed 5V Tolerant Input Buffer section.
			Register-to-Register performance table has been updated (rev. G 0.28).
			LatticeECP/EC External Switching Characteristics table has been updated (rev. G 0.28).
			LatticeECP/EC Internal Switching Characteristics table has been updated (rev. G 0.28).
			LatticeECP/EC Family Timing Adders have been updated (rev. G 0.28).
			sysCLOCK PLL timing table has been updated (rev. G 0.28)
			LatticeECP/EC sysCONFIG Port Timing specification table has been updated (rev. G 0.28).
			Master Clock table has been updated (rev. G 0.28).
			JTAG Port Timing specification table has been updated (rev. G 0.28).
		Pinout Information	Signal Description table has been updated with V_{CCPLL}.
November 2005	02.1	DC & Switching Characteristics	Pin-to-Pin Performance table has been updated (G 0.30) - 4:1MUX, 8:1MUX, 16:1MUX, 32:1MUX Register-to-Register Performance (G 0.30) - No timing number changes.
			External Switching Characteristics (G 0.30) - No timing number changes.
			Internal Switching Characteristics (G 0.30) -t_{SUP_DSP}, t_{HP_DSP}, t_{SUO_DSP}, t_{HO_DSP}, t_{COI_DSP} t_{COD_DSP} numbers have been updated.
			Family Timing Adders (G 0.30) - No timing number changes.
			sysCLOCK PLL Timing (G 0.30) - No timing number changes.
			sysCONFIG Port Timing Specifications (G 0.30) - No timing number changes.
			Master Clock (G 0.30) - No timing number changes.
			JTAG Port Timing Specification (G 0.30) - No timing number changes.
		Ordering Information	Added 208-PQFP lead-free part numbers.
March 2006	02.2	DC & Switching Characteristics	Added footnote 3. to V_{CCAUX} in the Recommended Operating Conditions table.
January 2007	02.3	Architecture	EBR Asynchronous Reset section added.
February 2007	02.4	Architecture	Updated EBR Asynchronous Reset section.
			Updated Maximum Number of Elements in a Block table - MAC value for x9 changed to 2.
May 2007	02.5	Architecture	Updated text in Ripple Mode section.
November 2007	02.6	DC & Switching Characteristics	Added JTAG Port Waveforms diagram.
			Updated t_{RST} timing information in the sysCLOCK PLL Timing table.
		Pinout Information	Added Thermal Management text section.
		Supplemental Information	Updated title list.
February 2008	02.7	DC & Switching Characteristics	Read/Write Mode (Normal) and Read/Write Mode with Input and Output Registers waveforms in the EBR Memory Timing Diagrams section have been updated.

Section II. LatticeECP/EC Family Technical Notes

LatticeECP/EC and LatticeXP sysIO Usage Guide

Introduction

The LatticeECP™, LatticeEC™ and LatticeXP™ sysIO™ buffers give the designer the ability to easily interface with other devices using advanced system I/O standards. This technical note describes the sysIO standards available and how they can be implemented using Lattice's design software.

sysIO Buffer Overview

The LatticeECP/EC and LatticeXP sysIO interfaces contain multiple Programmable I/O Cells (PIC) blocks. In the case of the LatticeEC and LatticeECP devices, each PIC contains two Programmable I/Os (PIO), PIOA and PIOB, connected to their respective sysIO buffers. In the LatticeXP device, each PIC also contains two PIOs, PIOA and PIOB, but every fourth PIC will have only PIOA. Two adjacent PIOs can be joined to provide a differential I/O pair (labeled as "T" and "C").

Each Programmable I/O (PIO) includes a sysIO Buffer and I/O Logic (IOLOGIC). The LatticeECP/EC and LatticeXP sysIO buffers support a variety of single-ended and differential signaling standards. The sysIO buffer also supports the DQS strobe signal that is required for interfacing with the DDR memory. One of every 16 PIOs in the LatticeECP/EC and one of every 14 PIOs in the case of the LatticeXP contains a delay element to facilitate the generation of DQS signals. The DQS signal from the bus is used to strobe the DDR data from the memory into input register blocks. For more information on the architecture of the sysIO buffer, please refer to the device data sheets.

The IOLOGIC includes input, output and tristate registers that implement both single data rate (SDR) and double data rate (DDR) applications along with the necessary clock and data selection logic. Programmable delay lines and dedicated logic within the IOLOGIC are used to provide the required shift to incoming clock and data signals and the delay required by DQS inputs in DDR memory. The DDR implementation in the IOLOGIC and the DDR memory interface support are discussed in more details in Lattice technical note number TN1050, *LatticeECP/EC DDR Usage Guide.*

Supported sysIO Standards

The LatticeECP/EC and LatticeXP sysIO buffer supports both single-ended and differential standards. Single-ended standards can be further subdivided into LVCMOS, LVTTL, PCI and other standards. The buffers support the LVTTL, LVCMOS 1.2, 1.5, 1.8, 2.5 and 3.3V standards. In the LVCMOS and LVTTL modes, the buffer has individually configurable options for drive strength, bus maintenance (weak pull-up, weak pull-down, or a bus-keeper latch). Other single-ended standards supported include SSTL and HSTL. Differential standards supported include LVDS, RSDS, BLVDS, LVPECL, differential SSTL and differential HSTL. Table 8-1 lists the sysIO standards supported in the Lattice EC/ECP and LatticeXP devices.

Table 8-1. Supported sysIO Standards

Standard	V_{CCIO}			V_{REF} (V)		
	Min.	Typ.	Max.	Min.	Typ.	Max.
LVCMOS 3.3	3.135	3.3	3.465	—	—	—
LVCMOS 2.5	2.375	2.5	2.625	—	—	—
LVCMOS 1.8	1.71	1.8	1.89	—	—	—
LVCMOS 1.5	1.425	1.5	1.575	—	—	—
LVCMOS 1.2	1.14	1.2	1.26	—	—	—
LVTTL	3.135	3.3	3.465	—	—	—
PCI	3.135	3.3	3.465	—	—	—
SSTL18 Class I	1.71	2.5	1.89	0.833	0.9	0.969
SSTL2 Class I, II	2.375	2.5	2.625	1.15	1.25	1.35
SSTL3 Class I, II	3.135	3.3	3.465	1.3	1.5	1.7
HSTL15 Class I	1.425	1.5	1.575	0.68	0.75	0.9
HSTL15 Class III	1.425	1.5	1.575	—	0.9	—
HSTL 18 Class I, II	1.71	1.8	1.89	—	0.9	—
HSTL 18 Class III	1.71	1.8	1.89	—	1.08	—
LVDS	2.375	2.5	2.625	—	—	—
LVPECL[1]	3.135	3.3	3.465	—	—	—
BLVDS[1]	2.375	2.5	2.625	—	—	—
RSDS[1]	2.375	2.5	2.625	—	—	—

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

sysIO Banking Scheme

LatticeECP/EC and LatticeXP devices have eight programmable sysIO banks, two per side. Each sysIO bank has a V_{CCIO} supply voltage and two reference voltages, V_{REF1} and V_{REF2}. On the top and bottom banks, the sysIO buffer pair consists of two single-ended output drivers and two sets of single-ended input buffers (both ratioed and referenced). The left and right side sysIO buffer pair along with the two single-ended output and input drivers will also have a differential driver. The referenced input buffer can also be configured as a differential input. The two pads in the pair are described as "true" and "comp", where the true pad is associated with the positive side of the differential input buffer and the comp (complementary) pad is associated with the negative side of the differential input buffer. Figure 8-1 shows the eight banks and their associated supplies.

Figure 8-1. sysIO Banking
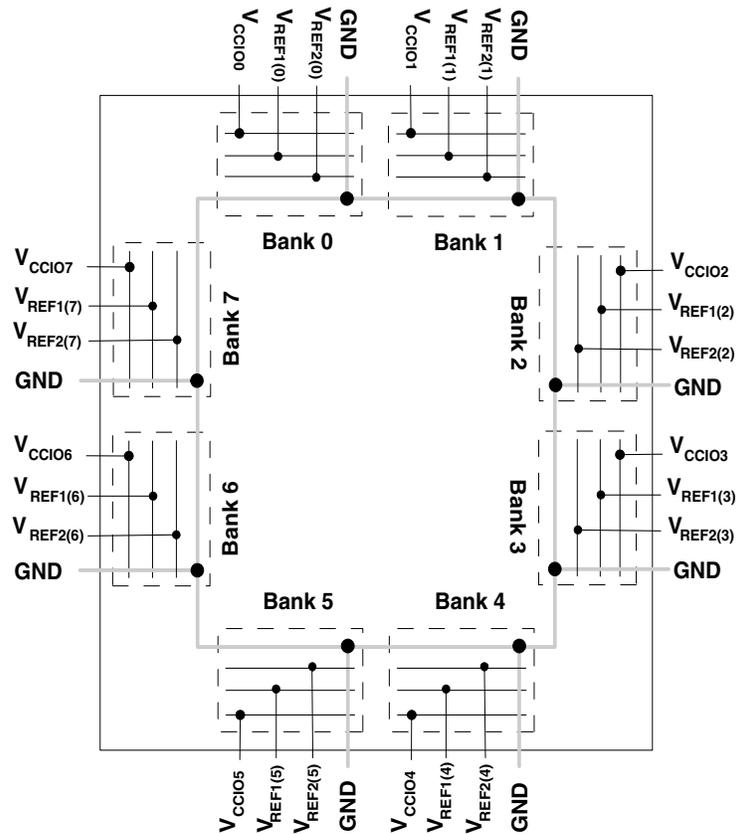
V_{CCIO} (1.2V/1.5V/1.8V/2.5V/3.3V)

Each bank has a separate V_{CCIO} supply that powers the single-ended output drivers and the ratioed input buffers such as LVTTL, LVCMOS, and PCI. LVTTL, LVCMOS3.3, LVCMOS2.5 and LVCMOS1.2 also have fixed threshold options allowing them to be placed in any bank. The VCCIO voltage applied to the bank determines the ratioed input standards that can be supported in that bank. It is also used to power the differential output drivers.

V_{CCAUX} (3.3V)

In addition to the bank V_{CCIO} supplies, devices have a V_{CC} core logic power supply, and a V_{CCAUX} auxiliary supply that powers the differential and referenced input buffers. V_{CCAUX} is required because V_{CC} does not have enough headroom to satisfy the common-mode range requirements of these drivers and input buffers.

V_{CCJ} (1.2V/1.5V/1.8V/2.5V/3.3V)

The JTAG pins have a separate V_{CCJ} power supply that is independent of the bank V_{CCIO} supplies. V_{CCJ} determines the electrical characteristics of the LVCMOS JTAG pins, both the output high level and the input threshold.

Input Reference Voltage (V_{REF1}, V_{REF2})

Each bank can support up to two separate V_{REF} input voltages, V_{REF1} and V_{REF2}, that are used to set the threshold for the referenced input buffers. The location of these V_{REF} pins is pre-determined within the bank. These pins can be used as regular I/Os if the bank does not require a V_{REF} voltage.

V_{REF1} for DDR Memory Interface

When interfacing to DDR memory, the V_{REF1} input must be used as the reference voltage for the DQS and DQ input from the memory. A voltage divider between V_{REF1} and GND is used to generate an on-chip reference volt-

age that is used by the DQS transition detector circuit. This voltage divider is only present on V_{REF1} it is not available on V_{REF2}. For more information on the DQS transition detect logic and its implementation please refer to Lattice technical note number TN1050, *LatticeECP/EC DDR Usage Guide.*

Mixed Voltage Support in a Bank

The LatticeECP/EC and LatticeXP sysIO buffer is connected to three parallel ratioed input buffers. These three parallel buffers are connected to V_{CCIO}, V_{CCAUX} and to V_{CC} giving support for thresholds that track with V_{CCIO} as well as fixed thresholds for 3.3V (V_{CCAUX}) and 1.2V (V_{CC}) inputs. This allows the input threshold for ratioed buffers to be assigned on a pin-by-pin basis, rather than tracking it with V_{CCIO}. This option is available for all 1.2V, 2.5V and 3.3V ratioed inputs and is independent of the bank V_{CCIO} voltage. For example, if the bank V_{CCIO} is 1.8V, it is possible to have 1.2V and 3.3V ratioed input buffers with fixed thresholds, as well as 2.5V ratioed inputs with tracking thresholds.

Prior to device configuration, the ratioed input thresholds always track the bank V_{CCIO}, this option only takes effect after configuration. Output standards within a bank are always set by V_{CCIO}. Table 8-2 shows the sysIO standards that the user can mix in the same bank.

Table 8-2. Mixed Voltage Support

V_{CCIO}	Input sysIO Standards					Output sysIO Standards				
	1.2V	1.5V	1.8V	2.5V	3.3V	1.2V	1.5V	1.8V	2.5V	3.3V
1.2V	Yes			Yes	Yes	Yes				
1.5V	Yes	Yes		Yes	Yes		Yes			
1.8V	Yes		Yes	Yes	Yes			Yes		
2.5V	Yes			Yes	Yes				Yes	
3.3V	Yes			Yes	Yes					Yes

sysIO Standards Supported in Each Bank

Table 8-3. I/O Standards Supported by Various Banks

Description	Top Side Banks 0-1	Right Side Banks 2-3	Bottom Side Banks 4-5	Left Side Banks 6-7
Types of I/O Buffers	Single-ended	Single-ended and Differential	Single-ended	Single-ended and Differential
Output standards supported	LVTTL LVCMOS33 LVCMOS25 LVCMOS18 LVCMOS15 LVCMOS12 SSTL18 Class I SSTL25 Class I, II SSTL33 Class I, II HSTL15 Class I, III HSTL18_I, II, III SSTL18D Class I, SSTL25D Class I, II SSTL33D Class I, II HSTL15D Class I, III, HSTL18D Class I, III PCI33 LVDS25E[1] LVPECL[1] BLVDS[1] RSDS[1]	LVTTL LVCMOS33 LVCMOS25 LVCMOS18 LVCMOS15 LVCMOS12 SSTL18 Class I SSTL25 Class I, II SSTL33 Class I, II HSTL15 Class I, III HSTL18 Class I, II, III SSTL18D Class I, SSTL25D Class I, II SSTL33D Class I, II HSTL15D Class I, III HSTL18D Class I, III PCI33 LVDS LVDS25E[1] LVPECL[1] BLVDS[1] RSDS[1]	LVTTL LVCMOS33 LVCMOS25 LVCMOS18 LVCMOS15 LVCMOS12 SSTL18 Class I SSTL2 Class I, II SSTL3 Class I, II HSTL15 Class I, III HSTL18 Class I, II, III SSTL18D Class I, SSTL25D Class I, II, SSTL33D Class I, II HSTL15D Class I, III HSTL18D Class I, III PCI33 LVDS25E[1] LVPECL[1] BLVDS[1] RSDS[1]	LVTTL LVCMOS33 LVCMOS25 LVCMOS18 LVCMOS15 LVCMOS12 SSTL18 Class I SSTL2 Class I, II SSTL3 Class I, II HSTL15 Class I, III HSTL18 Class I, II, III SSTL18D Class I, SSTL25D Class I, II, SSTL33D_I, II HSTL15D Class I, III HSTL18D Class I, III PCI33 LVDS LVDS25E[1] LVPECL[1] BLVDS[1] RSDS[1]
Inputs	All Single-ended, Differential	All Single-ended, Differential	All Single-ended, Differential	All Single-ended, Differential
Clock Inputs	All Single-ended, Differential	All Single-ended, Differential	All Single-ended, Differential	All Single-ended, Differential
PCI Support	PCI33 with clamp	PCI33 no clamp	PCI33 with clamp	PCI no clamp
LVDS Output Buffers		LVDS (3.5mA) Buffers		LVDS (3.5mA) Buffers

1. These differential standards are implemented by using complementary LVCMOS driver with external resistor pack.

LVCMOS Buffer Configurations

All LVCMOS buffers have programmable pull, programmable drive and programmable slew configurations that can be set in the software.

Programmable Pull-up/Pull-Down/Buskeeper

When configured as LVCMOS or LVTTL, each sysIO buffer has a weak pull-up, a weak pull-down resistor and a weak buskeeper (bus hold latch) available. Each I/O can independently be configured to have one of these features or none of them.

Programmable Drive

Each LVCMOS or LVTTL output buffer pin has a programmable drive strength option. This option can be set for each I/O independently. The drive strength setting available are 2mA, 4mA, 6mA, 8mA, 12mA, 16mA and 20mA. Actual options available vary by the I/O voltage. The user must consider the maximum allowable current per bank and the package thermal limit current when selecting the drive strength.

The programmable drive feature also allows the user to match to the impedance of the transmission line.

Table 8-4 shows the drive current setting required to match 50¾ transmission line with 50¾ and 200¾ terminations.

Table 8-4. Impedance Matching Using Programmable Drive Strength

50¾ Transmission Line Termination (¾)	I/O Standard	Drive Strength (mA)
200	LVCMOS18	8
	LVCMOS33	12
50	LVCMOS18	16
	LVCMOS33	20

The actual impedance matching may vary on the transmission line design and the load. To find the best matching, it is recommended to drive the transmission line with different combinations of I/O standards and drive strengths that best match the line impedance. Lattice provides IBIS buffer models for the users to further analyze the impedance matching.

The figure below shows how this impedance matching is done for a 50¾ transmission line with 200¾ termination using LVCMOS18 I/O buffers programmed to drive 16mA, 12mA, 8mA and 4mA. From this experiment it is empirical that the best matching is achieved with the 8mA drive setting.

Figure 8-2. Impedance Matching for a 50¾ Transmission Line with 200¾ Termination

LVCMOS18 16mA　　　　　　**LVCMOS18 12mA**
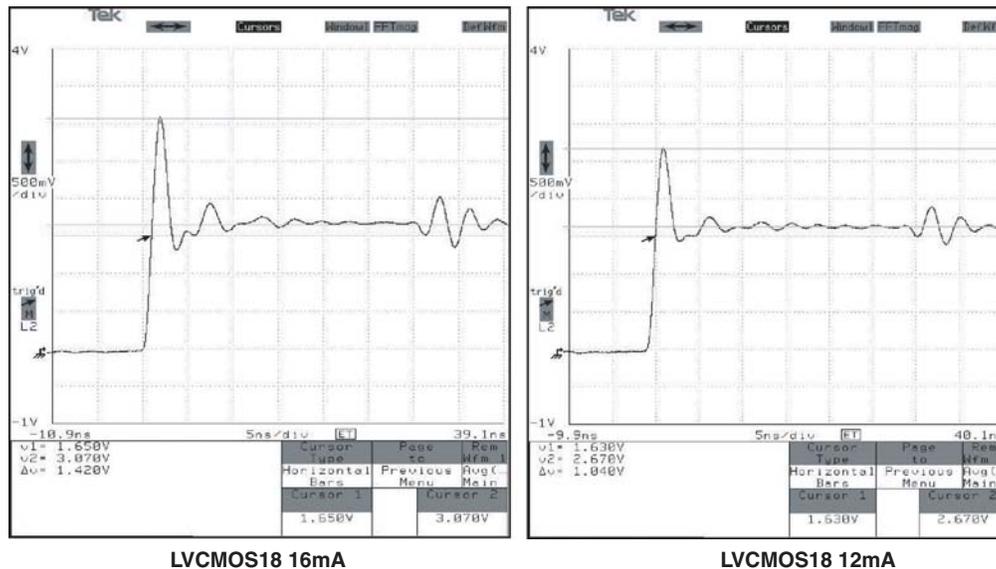
Figure 7-2. Impedance Matching for a 50¾ Transmission Line with 200¾ Termination (Cont.)

LVCMOS18 8mA	LVCMOS18 4mA
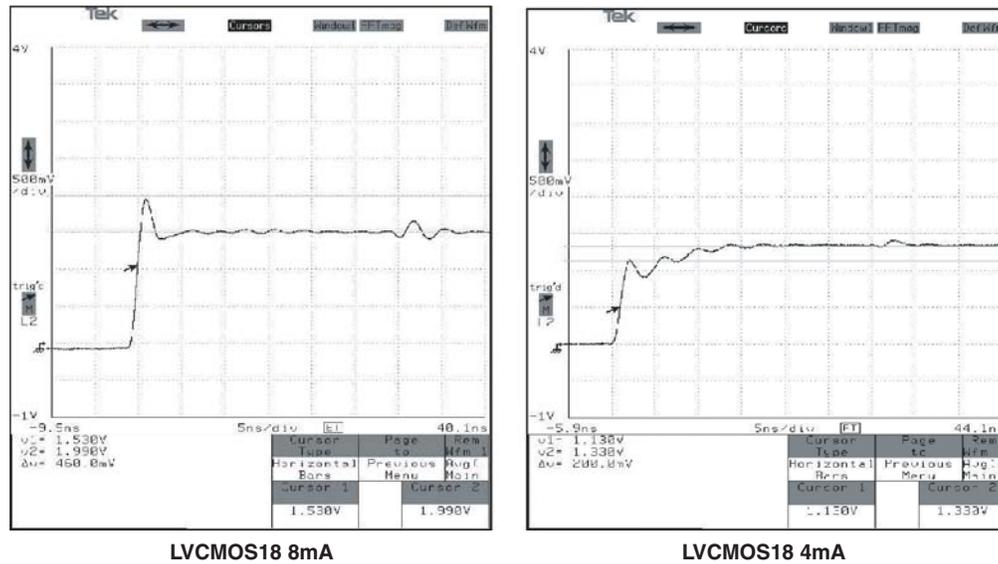
Programmable Slew Rate

Each LVCMOS or LVTTL output buffer pin also has a programmable output slew rate control that can be configured for either low noise or high-speed performance. Each I/O pin has an individual slew rate control. This allows slew rate control to be specified on pin-by-pin basis. This slew rate control affects both the rising edges and the falling edges.

Open Drain Control

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.

The software implements open drain in the LatticeECP/EC and LatticeXP devices by connecting the data and tristate input of the output buffer. Software will implement open drain using this method for simple output buffers. If the user wants to assign open drain functionality to a bidirectional I/O, a similar implementation is required in the HDL design. This can be accomplished by combining the equations for the output enable with the output data. The function of an open drain output is to drive a high Z when the data to the output buffer is driven high and drive a low when the data to the output buffer is driven low.

Differential SSTL and HSTL Support

The single-ended driver associated with the complementary 'C' pad can optionally be driven by the complement of the data that drives the single-ended driver associated with the true pad. This allows a pair of single-ended drivers to be used to drive complementary outputs with the lowest possible skew between the signals. This is used for driving complementary SSTL and HSTL signals (as required by the differential SSTL and HSTL clock inputs on synchronous DRAM and synchronous SRAM devices respectively). This capability is also used in conjunction with off-chip resistors to emulate LVPECL and BLVDS output drivers.

PCI Support with Programmable PCICLAMP

Each sysIO buffer can be configured to support PCI33. The buffers on the top and bottom of the device have an optional PCI clamp diode that may optionally be specified in the ispLEVER® design tool.

The programmable PCICLAMP can be turned ON or OFF. This option is available on each I/O independently on the top and bottom banks.

5V Interface with PCI Clamp Diode

All the I/Os on the top and bottom sides of the device (Banks 0, 1, 4, and 5) have a clamp diode that is used to clamp the voltage at the input to V_{CCIO}. This is especially used for PCI I/O standards. This clamp diode can be used along with an external resistor to make an input 5V tolerant.

Figure 8-3. 5V Tolerant Input Buffer
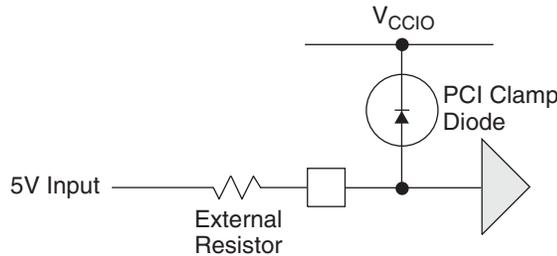
The value of this external resistor will depend on the PCI clamp diode characteristics. You can find the voltage vs. current data across this diode in the device IBIS model.

In order to interface to 5V input, it is recommended to set the V_{CCIO} between 2.5V to 3.3V.

Below is an example for calculating the value of this external resistor when V_{CCIO} is 2.75V.

- Maximum voltage at input pin, V_{INMAX} = 3.75V (see device data sheet for more details)

- Bank V_{CCIO} = 2.75V

- Maximum voltage drop across clamp diode, $V_D = V_{INMAX} - V_{CCIO} = 3.75 - 2.75 = 1V$

- The current across the clamp diode at V_D can be found in the power clamp data of the IBIS file. Below is the power clamp portion of the IBIS file for a LVCMOS3.3 input model with PCI Clamp turned on. When V_D is 1V, the clamp diode current is I_D = 27.4mA.

Table 8-5. Power Clamp Data from IBIS Model

Voltage	I (Max.)	Units
-1.40	72.5	mA
-1.30	61.2	mA
-1.20	49.9	mA
-1.10	38.6	mA
-1.00	27.4	mA
-0.90	16.9	mA
-0.80	9.52	mA
-0.70	5.35	mA
-0.60	2.31	mA
-0.50	550.8	µA
-0.40	58.0	µA
-0.30	3.61	µA
-0.20	0.07917	µA
-0.10	0.0009129	µA
0.00	0.0001432	µA

- Assume the maximum output voltage of the driving device is V_{EXT} = 5.25V. The value of the external resistor can then be calculated as follows:

$$R_{EXT} = (V_{EXT} - V_{INMAX})/I_D = (5.25V - 3.75V)/27.4 = 54.8 \text{ ohm}$$

If the V_{CCIO} of the bank is increased, it will also increase the value of the external resistor required. Changing the bank V_{CCIO} will also change the value of the input threshold voltage.

Programmable Input Delay

Each input can optionally be delayed before it is passed to the core logic or input registers. The primary use for the input delay is to achieve zero hold time for the input registers when using a direct drive primary clock. To arrive at zero hold time, the input delay will delay the data by at least as much as the primary clock injection delay. This option can be turned ON or OFF for each I/O independently in the software using the FIXEDDELAY attribute. This attribute is described in more detail in the Software sysIO Attributes section. Appendix A shows how this feature can be enabled in the software using HDL attributes.

Software sysIO Attributes

sysIO attributes can be specified in the HDL, using the Preference Editor GUI or in the ASCII Preference file (.prf) file directly. Appendices A, B and C list examples of how these can be assigned using each of the methods mentioned above. This section describes in detail each of these attributes.

IO_TYPE

This is used to set the sysIO standard for an I/O. The V_{CCIO} required to set these I/O standards are embedded in the attribute names itself. There is no separate attribute to set the V_{CCIO} requirements. Table 8-6 lists the available I/O types.

Table 8-6. I/O_TYPE Attribute Values

sysIO Signaling Standard	IO_TYPE
DEFAULT (for LatticeECP/EC)	LVCMOS12
DEFAULT (for LatticeXP)	LVCMOS25
LVDS 2.5V	LVDS25
RSDS	RSDS
Emulated LVDS 2.5V	LVDS25E[1]
Bus LVDS 2.5V	BLVDS25[1]
LVPECL 3.3V	LVPECL33[1]
HSTL18 Class I, II and III	HSTL18_I, HSTL18_II, HSTL18_III
Differential HSTL 18 Class I, II and III	HSTL18D_I HSTL18D_II HSTL18D_III
HSTL 15 Class I and III	HSTL15_I HSTL15_III
Differential HSTL 15 Class I and III	HSTL15D_I HSTL15D_III
SSTL 33 Class I and II	SSTL33_I, SSTL33_II
Differential SSTL 33 Class I and II	SSTL33D_I SSTL3D_II
SSTL 25 Class I and II	SSTL25_I SSTL25_II
Differential SSTL 25 Class I and II	SSTL25D_I SSTL25D_II
SSTL 18 Class I	SSTL18_I
Differential SSTL 18 Class I	SSTL18D_I
LVTTL	LVTTL33
3.3V LVCMOS	LVCMOS33
2.5V LVCMOS	LVCMOS25
1.8V LVCMOS	LVCMOS18
1.5V LVCMOS	LVCMOS15
1.2V LVCMOS	LVCMOS12
3.3V PCI	PCI33

1. These differential standards are implemented by using complementary
   LVCMOS driver with external resistor pack.

OPENDRAIN

LVCMOS and LVTTL I/O standards can be set to Open Drain configuration by using the OPENDRAIN attribute.

Values: ON, OFF
Default: OFF

DRIVE

The drive strength attribute is available for LVTTL and LVCMOS output standards. These can be set or each I/O pin individually.

Values: NA, 2, 4, 8, 12, 16, 20
LatticeECP/EC Default: 6
LatticeXP Default: 8

The programmable drive available on a pad will depend on the V_{CCIO}. Table 8-7 shows the drive strength available for different V_{CCIO}.

Table 8-7. Programmable Drive Strength Values at Various V_{CCIO} Voltages

Drive	V_{CCIO}				
	1.2 V	1.5 V	1.8 V	2.5 V	3.3 V
2	X				
4		X	X	X	X
6	X				
8		X	X	X	X
12			X	X	X
16			X	X	X
20				X	X

PULLMODE

The PULLMODE attribute is available for all the LVTLL and LVCMOS inputs and outputs. This attribute can be enabled for each I/O independently.

Values: UP, DOWN, NONE, KEEPER
Default: UP

PCICLAMP

PCI33 inputs and outputs on the top and bottom of the device have an optional PCI clamp that is enabled via the PCICLAMP attribute. The PCICLAMP is also available for all LVCMOS33 and LVTTL inputs and outputs.

Values: ON, OFF
Default: OFF

SLEWRATE

The SLEWRATE attribute is available for all LVTTL and LVCMOS output drivers. Each I/O pin has an individual slew rate control. This allows the designer to specify the slew rate control on a pin-by-pin basis.

Values: FAST, SLOW
Default: FAST

FIXEDDELAY

The FIXEDDELAY attribute is available to each input pin. When enabled, this attribute is used to achieve zero hold time for the input registers when using global clock.

Values: TRUE, FALSE
Default: FALSE

DIN/DOUT

This attribute can be used when I/O registers need to be assigned. Using DIN will assert an input register and using the DOUT attribute will assert an output register in the design. By default the software will try to assign the I/O registers if applicable. The user can turn this OFF by using the synthesis attribute or using the preference editor of the software. These attributes can only be applied on registers.

LOC

This attribute can be used to make pin assignments to the I/O ports in the design. This attribute is only used when the pin assignments are made in HDL source. Pins assignments can be made directly using the GUI in the Preference Editor of the software. The appendices explain this in more detail.

Design Considerations and Usage

This section discusses some of design rules and considerations that need to be taken into account when designing with the LatticeECP/ECP and LatticeXP sysIO buffer.

Banking Rules

- If V_{CCIO} or V_{CCJ} for any bank is set to 3.3V, it is recommended that it be connected to the same power supply as $V_{CCAUX,}$ thus minimizing leakage.
- If V_{CCIO} or V_{CCJ} for any bank is set to 1.2V, it is recommended that it be connected to the same power supply as $V_{CC,}$ thus minimizing leakage.
- When implementing DDR memory interfaces, the V_{REF1} of the bank is used to provide reference to the interface pins and cannot be used to power any other referenced inputs.
- Only the top and bottom banks (Banks 0, 1, 4, and 5) will support PCI clamps. The left and right side (Banks 2, 3, 6 and 7) do not support PCI Clamp, but will support True LVDS output.

Differential I/O Rules

- All the banks can support LVDS input buffers. Only the banks on the right and left side (Banks 2, 3, 6 and 7) will support True Differential output buffers. The banks on the top and bottom will support the LVDS input buffers but will not support True LVDS outputs. The user can use emulated LVDS output buffers on these banks.
- All banks support emulated differential buffers using external resistor pack and complementary LVCMOS drivers.
- In LatticeXP devices, not all PIOs have LVDS capability. Only four out of every seven I/Os can provide LVDS buffer capability. In LatticeECP/EC devices, there are no restrictions on the number of I/Os that can support LVDS. In both cases LVDS can only be assigned to the TRUE pad. Refer to the device data sheets to see the pin listing for all the LVDS pairs.

Assigning V_{REF}/ V_{REF} Groups for Referenced Inputs

Each bank has two dedicated V_{REF} input pins, V_{REF1} and $V_{REF2.}$ Buffers can be grouped to a particular V_{REF} rail, V_{REF1} or V_{REF2}. This grouping is done by assigning a PGROUP VREF preference along with the LOCATE PGROUP preference.

Preference Syntax

```
PGROUP <pgrp_name> [(VREF <vref_name>)+] (COMP <comp_name>)+;
LOCATE PGROUP <pgrp_name> BANK <bank_num>;
LOCATE VREF <vref_name> SITE <site_name>;
```

Example of VREF Groups

```
PGROUP "vref_pg1" VREF "ref1" COMP "ah(0)" COMP "ah(1)" COMP "ah(2)" COMP "ah(3)"
COMP "ah(4)" COMP "ah(5)" COMP "ah(6)" COMP "ah(7)";

PGROUP "vref_pg2" VREF "ref2" COMP "al(0)" COMP "al(1)" COMP "al(2)" COMP "al(3)"
COMP "al(4)" COMP "al(5)" COMP "al(6)" COMP "al(7)";

LOCATE VREF "ref1" SITE PR29C;
LOCATE VREF "ref2" SITE PR48B;
```

or

```
LOCATE PGROUP " vref_pg1" BANK 2;
LOCATE PGROUP " vref_pg2" BANK 2;
```

The second example show V_{REF} groups, "vref_pg1" assigned to V_{REF} "ref1" and "vref_pg2" assigned to "ref2". V_{REF} must then be locked to either V_{REF1} or V_{REF2} using LOCATE preference. Or, the user can simply designate to which bank V_{REF} group should be located. The software will then assign these to either V_{REF1} or V_{REF2} of the bank.

If the PGROUP VREF is not used, the software will automatically group all pins that need the same V_{REF} reference voltage. This preference is most useful when there is more than one bus using the same reference voltage and the user wants to associate each of these buses to different V_{REF} resources.

Differential I/O Implementation

The LatticeECP/EC and LatticeXP devices support a variety of differential standards as detailed in the following section.

LVDS

True LVDS (LVDS25) drivers are available on the left and right side of the devices. LVDS input support is provided on all sides of the device. All four sides support LVDS using complementary LVCMOS drivers with external resistors (LVDS25E).

Please refer to the LatticeECP/EC and LatticeXP data sheets for a more detailed explanation of these LVDS implementations.

BLVDS

All single-ended sysIO buffer pairs in the LatticeECP family support the Bus-LVDS standard using complementary LVCMOS drivers with external resistors.

Please refer to the LatticeECP/EC and LatticeXP data sheets to learn more about BLVDS implementation.

RSDS

All single-ended sysIO buffers pairs in the LatticeECP family support the RSDS standard using complementary LVCMOS drivers with external resistors. This mode uses LVDS25E with an alternative resistor pack.

Please refer to the LatticeECP/EC and LatticeXP data sheets for a detailed explanation of RSDS implementation.

LVPECL

All the sysIO buffers will support LVPECL inputs. LVPECL outputs are supported using a complementary LVCMOS driver with external resistors.

Please refer to the LatticeECP/EC and LatticeXP data sheets for further information on LVPECL implementation.

Differential SSTL and HSTL

All single-ended sysIO buffers pairs in the LatticeECP family support differential SSTL and HSTL. Please refer to the LatticeECP/EC and LatticeXP data sheets for a detailed explanation of Differential HSTL and SSTL implementation.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

+1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Appendix A. HDL Attributes for Synplify® and Precision® RTL Synthesis

Using these HDL attributes, you can assign sysIO attributes directly in your source. You will need to use the attribute definition and syntax for the synthesis vendor you are planning to use. Below are a list of all the sysIO attributes syntax and examples for Precision RTL Synthesis and Synplify. This section only lists the sysIO buffer attributes for these devices. You can refer to the Precision RTL Synthesis and Synplify user manuals for a complete list of synthesis attributes. These manuals are available through ispLEVER Software Help.

VHDL Synplify/Precision RTL Synthesis

This section lists syntax and examples for all the sysIO attributes in VHDL when using Precision RTL Synthesis or Synplicity synthesis tools.

Syntax

Table 8-8. VHDL Attribute Syntax for Synplify and Precision RTL Synthesis

Attribute	Syntax
IO_TYPE	attribute IO_TYPE: string; attribute IO_TYPE of Pinname: signal is "IO_TYPE Value";
OPENDRAIN	attribute OPENDRAIN: string; attribute OPENDRAIN of Pinname: signal is "OpenDrain Value";
DRIVE	attribute DRIVE: string; attribute DRIVE of Pinname: signal is "Drive Value";
PULLMODE	attribute PULLMODE: string; attribute PULLMODE of Pinname: signal is "Pullmode Value";
PCICLAMP	attribute PCICLAMP: string; attribute PCICLAMP of Pinname: signal is "PCIClamp Value";
SLEWRATE	attribute PULLMODE: string; attribute PULLMODE of Pinname: signal is "Slewrate Value";
FIXEDDELAY	attribute FIXEDDELAY: string; attribute FIXEDDELAY of Pinname: signal is "Fixeddelay Value";
DIN	attribute DIN: string; attribute DIN of Pinname: signal is " ";
DOUT	attribute DOUT: string; attribute DOUT of Pinname: signal is " ";
LOC	attribute LOC: string; attribute LOC of Pinname: signal is "pin_locations";

Examples

IO_TYPE
--*Attribute Declaration*****

ATTRIBUTE IO_TYPE: string;

--*IO_TYPE assignment for I/O Pin*****

ATTRIBUTE IO_TYPE OF portA: SIGNAL IS "PCI33";

ATTRIBUTE IO_TYPE OF portB: SIGNAL IS "LVCMOS33";

ATTRIBUTE IO_TYPE OF portC: SIGNAL IS "LVDS25";

OPENDRAIN

--***Attribute Declaration***

ATTRIBUTE OPENDRAIN: string;

--***DRIVE assignment for I/O Pin***

ATTRIBUTE OPENDRAIN OF portB: SIGNAL IS "ON";

DRIVE
--***Attribute Declaration***

ATTRIBUTE DRIVE: string;

--***DRIVE assignment for I/O Pin***

ATTRIBUTE DRIVE OF portB: SIGNAL IS "20";

PULLMODE

--***Attribute Declaration***

ATTRIBUTE PULLMODE : string;

--***PULLMODE assignment for I/O Pin***

ATTRIBUTE PULLMODE OF portA: SIGNAL IS "DOWN";

ATTRIBUTE PULLMODE OF portB: SIGNAL IS "UP";

PCICLAMP

--***Attribute Declaration***

ATTRIBUTE PCICLAMP: string;

--***PULLMODE assignment for I/O Pin***

ATTRIBUTE PCICLAMP OF portA: SIGNAL IS "ON";

SLEWRATE
--***Attribute Declaration***

ATTRIBUTE SLEWRATE : string;

--*** SLEWRATE assignment for I/O Pin***

ATTRIBUTE SLEWRATE OF portB: SIGNAL IS "FAST";

FIXEDDELAY
--***Attribute Declaration***

ATTRIBUTE FIXEDDELAY: string;

--*** SLEWRATE assignment for I/O Pin***

ATTRIBUTE FIXEDDELAY OF portB: SIGNAL IS "TRUE";

DIN/DOUT
--***Attribute Declaration***

ATTRIBUTE din : string;

ATTRIBUTE dout : string;

--*** din/dout assignment for I/O Pin***

ATTRIBUTE din OF input_vector: SIGNAL IS " ";

ATTRIBUTE dout OF output_vector: SIGNAL IS " ";

LOC
--***Attribute Declaration***

ATTRIBUTE LOC : string;

--*** LOC assignment for I/O Pin***

ATTRIBUTE LOC OF input_vector: SIGNAL IS "E3,B3,C3 ";

Verilog for Synplify

This section lists syntax and examples for all the sysIO Attributes in Verilog using the Synplify synthesis tool.

Syntax

Table 8-9. Verilog Synplify Attribute Syntax

Attribute	Syntax
IO_TYPE	PinType PinName /* synthesis IO_TYPE="IO_Type Value"*/;
OPENDRAIN	PinType PinName /* synthesis OPENDRAIN ="OpenDrain Value"*/;
DRIVE	PinType PinName /* synthesis DRIVE="Drive Value"*/;
PULLMODE	PinType PinName /* synthesis PULLMODE="Pullmode Value"*/;
PCICLAMP	PinType PinName /* synthesis PCICLAMP =" PCIClamp Value"*/;
SLEWRATE	PinType PinName /* synthesis SLEWRATE="Slewrate Value"*/;
FIXEDDELAY	PinType PinName /* synthesis FIXEDDELAY="Fixeddelay Value"*/;
DIN	PinType PinName /* synthesis DIN=" "*/;
DOUT	PinType PinName /* synthesis DOUT=" "*/;
LOC	PinType PinName /* synthesis LOC="pin_locations "*/;

Examples

//IO_TYPE, PULLMODE, SLEWRATE and DRIVE assignment

output portB /*synthesis IO_TYPE="LVCMOS33" PULLMODE ="UP" SLEWRATE ="FAST"

DRIVE ="20"*/;

output portC /*synthesis IO_TYPE="LVDS25" */;

//OPENDRAIN

output portA /*synthesis OPENDRAIN ="ON"*/;

//PCICLAMP

output portA /*synthesis IO_TYPE="PCI33" PULLMODE ="PCICLAMP"*/;

// Fixeddelay

input load /* synthesis FIXEDDELAY="TRUE" */;

// Place the flip-flops near the load input

input load /* synthesis din="" */;

// Place the flip-flops near the outload output

output outload /* synthesis dout="" */;

//I/O pin location

input [3:0] DATA0 /* synthesis loc="E3,B1,F3"*/;

//Register pin location

reg data_in_ch1_buf_reg3 /* synthesis loc="R40C47" */;

//Vectored internal bus

reg [3:0] data_in_ch1_reg /*synthesis loc ="R40C47,R40C46,R40C45,R40C44" */;

Verilog for Precision RTL Synthesis

This section lists syntax and examples for all the sysIO Attributes in Verilog using the Precision RTL Synthesis synthesis tool.

Syntax

Table 8-10. Verilog Precision RTL Synthesis Attribute Syntax

ATTRIBUTE	SYNTAX
IO_TYPE	//pragma attribute PinName IO_TYPE IO_TYPE Value
OPENDRAIN	//pragma attribute PinName OPENDRAIN OpenDrain Value
DRIVE	//pragma attribute PinName DRIVE Drive Value
PULLMODE	//pragma attribute PinName IO_TYPE Pullmode Value
PCICLAMP	//pragma attribute PinName PCICLAMP PCIClamp Value
SLEWRATE	//pragma attribute PinName IO_TYPE Slewrate Value
FIXEDDELAY	//pragma attribute PinName IO_TYPE Fixeddelay Value
LOC	//pragma attribute PinName LOC pin_location

Example

//**IO_TYPE ***

//pragma attribute portA IO_TYPE PCI33

//pragma attribute portB IO_TYPE LVCMOS33

//pragma attribute portC IO_TYPE SSTL25_II

//* Opendrain ***

//pragma attribute portB OPENDRAIN ON

//pragma attribute portD OPENDRAIN OFF

//* Drive ***

//pragma attribute portB DRIVE 20

//pragma attribute portD DRIVE 8

//* Pullmode***

//pragma attribute portB PULLMODE UP

//* PCIClamp***

//pragma attribute portB PCICLAMP ON

//* Slewrate ***

//pragma attribute portB SLEWRATE FAST

//pragma attribute portD SLEWRATE SLOW

// ***Fixeddelay***

// pragma attribute load FIXEDDELAY TRUE

//***LOC***

//pragma attribute portB loc E3

Appendix B. sysIO Attributes Using Preference Editor User Interface

You can also assign the sysIO buffer attributes using the Pre Map Preference Editor GUI available in the ispLEVER tools. The Pin Attribute Sheet list all the ports in your design and all the available sysIO attributes as preferences. Clicking on each of these cells will produce a list of all the valid I/O preference for that port. Each column takes precedence over the next. Hence, when a particular IO_TYPE is chosen, the DRIVE, PULLMODE and SLEWRATE columns will only list the valid combinations for that IO_TYPE. The user can lock the pin locations using the pin location column of the Pin Attribute sheet. Right-clicking on a cell will list all the available pin locations. The Preference Editor will also conduct a DRC check to look for incorrect pin assignments.

You can enter the DIN/ DOUT preferences using the Cell Attributes Sheet of the Preference Editor. All the preferences assigned using the Preference Editor are written into the logical preference file (.lpf).

Figure 8-4 and Figure 8-5 show the Pin Attribute Sheet and the Cell Attribute Sheet views of the Preference Editor. For further information on how to use the Preference Editor, refer to the ispLEVER Help documentation located in the Help menu option of the software.

Figure 8-4. Pin Attributes Tab

Figure 8-5. Cell Attributes Tab

Appendix C. sysIO Attributes Using Preference File (ASCII File)

You can also enter the sysIO attributes directly in the preference (.prf) file as sysIO buffer preferences. The PRF file is an ASCII file containing two sections: a schematic section for preferences created by the Mapper or translator, and a user section for preferences entered by the user. You can write user preferences directly into this file. The synthesis attributes appear between the schematic start and schematic end of the file. You can enter the sysIO buffer preferences after the schematic end line using the preference file syntax. Below are a list of sysIO buffer preference syntax and examples.

IOBUF

This preference is used to assign the attribute IO_TYPE, PULLMODE, SLEWRATE and DRIVE.

Syntax
IOBUF [ALLPORTS | PORT <port_name> | GROUP <group_name>] (keyword=<value>)+;

where:

<port_name> = These are not the actual top-level port names, but should be the signal name attached to the port. PIOs in the physical design (.ncd) file are named using this convention. Any multiple listings or wildcarding should be done using GROUPs

Keyword = IO_TYPE, OPENDRAIN, DRIVE, PULLMODE, PCICLAMP, SLEWRATE.

Example
IOBUF PORT "port1" IO_TYPE=LVTTL33 OPENDRAIN=ON DRIVE=8 PULLMODE=UP

PCICLAMP =OFF SLEWRATE=FAST;

DEFINE GROUP "bank1" "in*" "out_[0-31]";

IOBUF GROUP "bank1" IO_TYPE=SSTL18_II;

LOCATE

When this preference is applied to a specified component it places the component at a specified site and locks the component to the site. If applied to a specified macro instance it places the macro's reference component at a specified site, places all of the macro's pre-placed components (that is, all components that were placed in the macro's library file) in sites relative to the reference component, and locks all of these placed components at their sites. This can also be applied to a specified PGROUP.

Syntax
LOCATE [COMP <comp_name> | MACRO <macro_name>] SITE <site_name>;

LOCATE PGROUP <pgroup_name> [SITE <site_name>; | REGION <region_name>;]

LOCATE PGROUP <pgroup_name> RANGE <site_1> [<site_2> | <count>] [<direction>] | RANGE <chip_side> [<direction>];

LOCATE BUS < bus_name> ROW|COL <number>;

<bus_name> := string

<number> := integer

Note: If the comp_name, macro_name, or site_name begins with anything other than an alpha character (for example, "11C7"), you must enclose the name in quotes. Wildcard expressions are allowed in <comp_name>.

Example
This command places the port Clk0 on the site A4:

LOCATE COMP "Clk0" SITE "A4";

This command places the component PFU1 on the site named R1C7:

LOCATE COMP "PFU1" SITE "R1C7";

This command places bus1 on ROW 3 and bus2 on COL4

LOCATE BUS "bus1" ROW 3;

LOCATE BUS "bus2" COL 4;

USE DIN CELL

This preference specifies the given register to be used as an input Flip Flop.

Syntax
USE DIN CELL <cell_name>;

where:

<cell_name> := string

Example

USE DIN CELL "din0";

USE DOUT CELL

Specifies the given register to be used as an output Flip Flop.

Syntax
USE DOUT CELL <cell_name>;

where:

<cell_name> := string

Examples
USE DOUT CELL "dout1";

PGROUP VREF

This preference is used to group all the components that need to be associated to one VREF pin within a bank.

Syntax
PGROUP <pgrp_name> [(VREF <vref_name>)+] (COMP <comp_name>)+;

LOCATE PGROUP <pgrp_name> BANK <bank_num>;

LOCATE VREF <vref_name> SITE <site_name>;

Example
PGROUP "vref_pg1" VREF "ref1" COMP "ah(0)" COMP "ah(1)" COMP "ah(2)" COMP "ah(3)" COMP "ah(4)"
COMP "ah(5)" COMP "ah(6)" COMP "ah(7)";

PGROUP "vref_pg2" VREF "ref2" COMP "al(0)" COMP "al(1)" COMP "al(2)" COMP "al(3)" COMP "al(4)" COMP
"al(5)" COMP "al(6)" COMP "al(7)";

LOCATE VREF "ref1" SITE PR29C;

LOCATE VREF "ref2" SITE PR48B;

or

LOCATE PGROUP " vref_pg1" BANK 2;

LOCATE PGROUP " vref_pg2" BANK 2;

Introduction

This technical note discusses memory usage in the LatticeEC™, LatticeECP™ and LatticeXP™ device families. It is intended to be used as a guide for integrating the EBR and PFU based memories for these device families using the ispLEVER® design tool.

The architecture of the LatticeECP/EC and LatticeXP devices provides a large amount of 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. LUTs and PFU can implement Distributed Single-Port RAM, Dual-Port RAM and ROM. The internal logic of the device can be used to configure the memory elements as FIFO and other storage types.

The capabilities of the EBR Block RAM and PFU RAM are referred to as primitives and described later in this document. Designers can generate the memory primitives using the IPexpress™ tool in the ispLEVER software. The IPexpress GUI allows users to specify the memory type and size required. IPexpress takes this specification and constructs a netlist to implement the desired memory by using one or more of the memory primitives.

The remainder of this document discusses how to utilize IPexpress, memory modules and memory primitives.

Memories in LatticeECP/EC and LatticeXP Devices

The LatticeECP/EC and LatticeXP architectures contain an array of logic blocks called PFUs or PFFs surrounded by Programmable I/O Cells (PICs). Interspersed between the rows of logic blocks are rows of sysMEM Embedded Block RAM (EBR) as shown in Figures 9-1, 9-2 and 9-3.

The PFU contains the building blocks for logic, and Distributed RAM and ROM. The PFF provides the logic building blocks without the distributed RAM

This document describes the memory usage and implementation for both embedded memory blocks (EBR) and distributed RAM of the PFU. Refer to the device data sheet for details on the hardware implementation of the EBR and Distributed RAM.

The logic blocks are arranged in a two-dimensional grid with rows and columns as shown in the figures below. The physical location of the EBR and Distributed RAM follows the row and column designation. The Distributed RAM, since it is part of the PFU resource, follows the PFU/PFF row and column designation. The EBR occupies two columns per block to account for the wider port interface.

Figure 9-1. Simplified Block Diagram, LatticeEC Device (Top Level)

Programmable I/O Cell
(PIC) includes sysIO
Interface

sysMEM Embedded
Block RAM (EBR)

JTAG Port

sysCONFIG Programming
Port (includes dedicated
and dual use pins)

PFF (Fast PFU
without RAM/ROM)

sysCLOCK PLL

Programmable
Functional Unit (PFU)
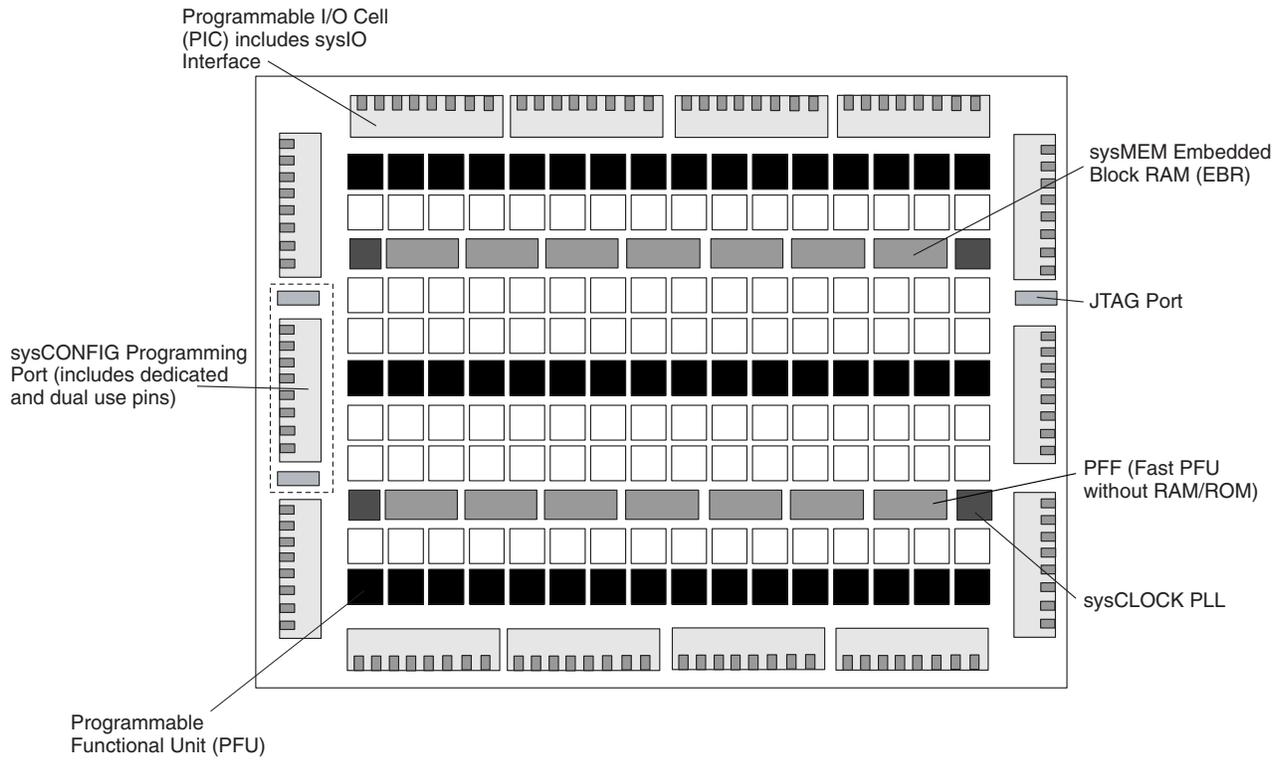
Figure 9-2. Simplified Block Diagram, LatticeECP Device (Top Level)

Programmable I/O Cell
(PIC) includes sysIO
Interface

sysMEM Embedded
Block RAM (EBR)

JTAG Port

sysCONFIG Programming
Port (includes dedicated
and dual use pins)

sysDSP Block

PFF (Fast PFU
without RAM/ROM)

sysCLOCK PLL

Programmable
Functional Unit (PFU)
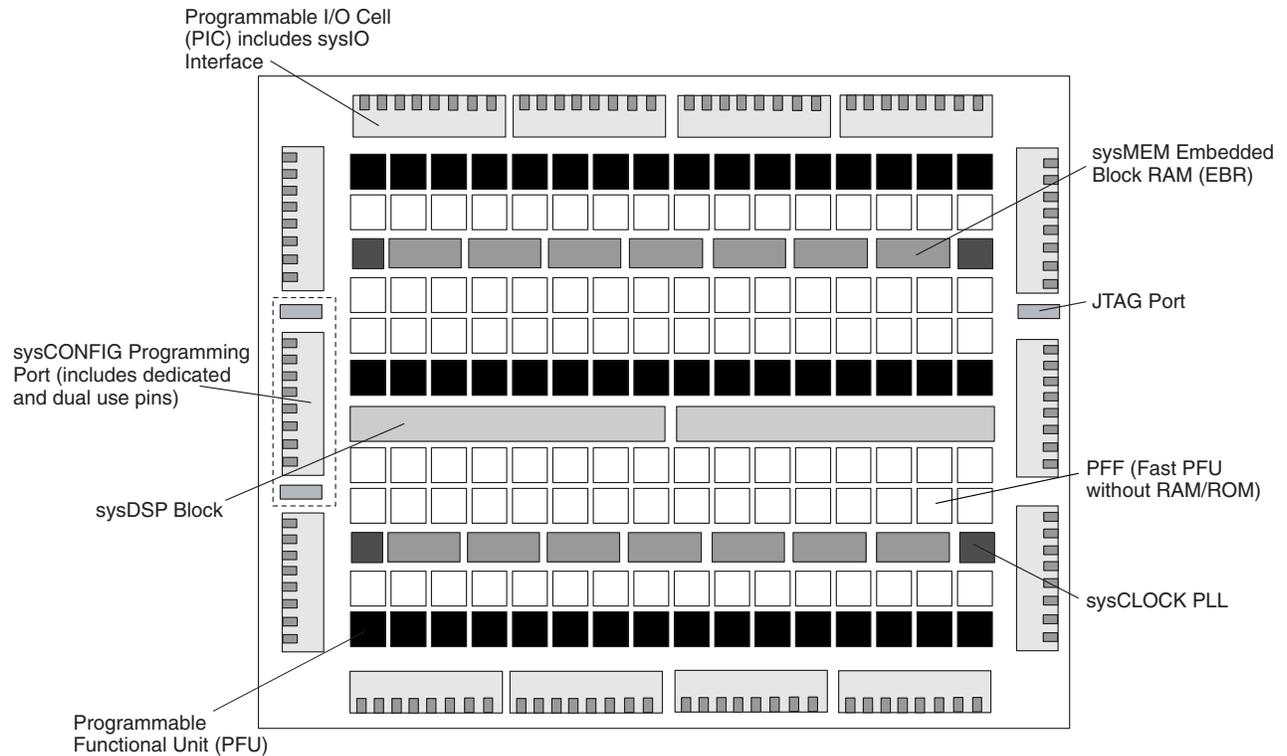
Figure 9-3. Simplified Block Diagram, LatticeXP Device (Top Level)
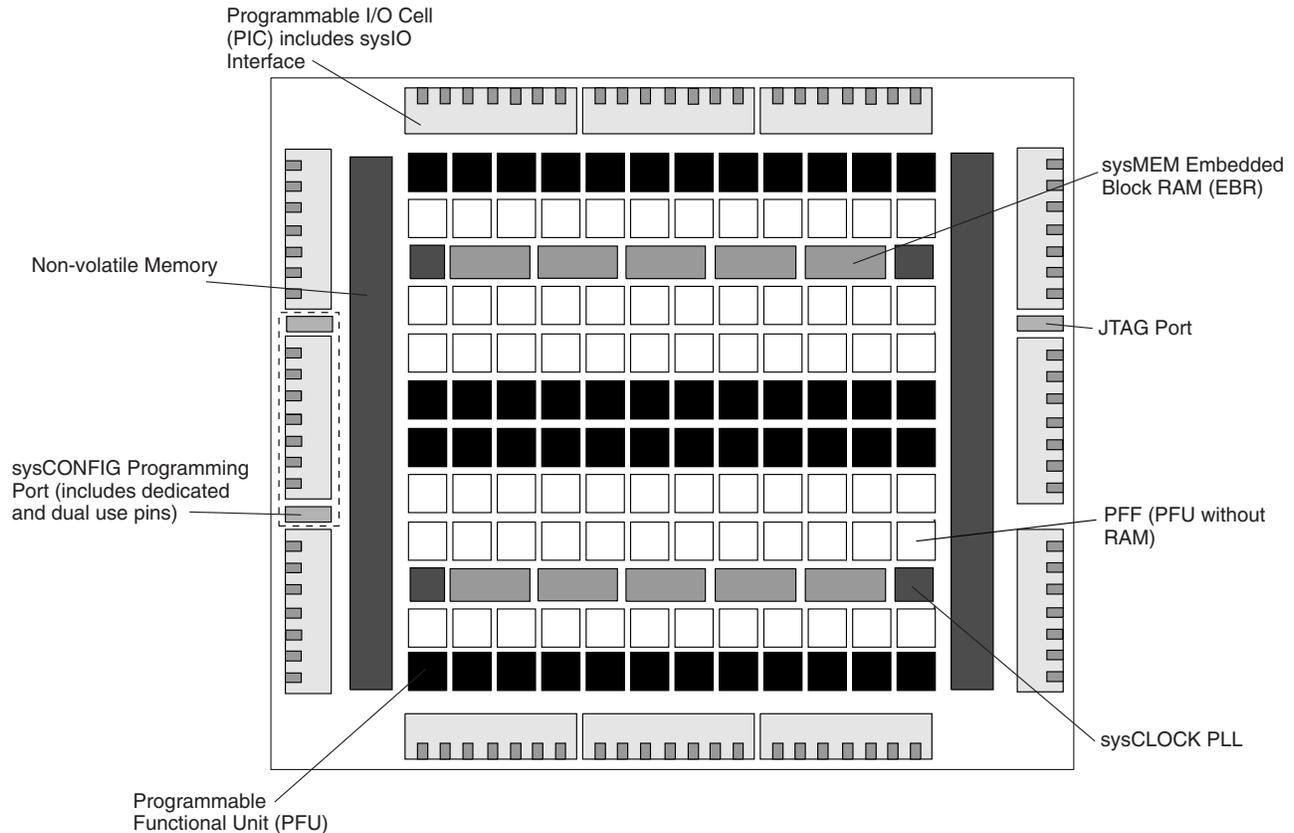
Utilizing IPexpress

Designers can utilize IPexpress to easily specify a variety of memories in their designs. These modules will be constructed using one or more memory primitives along with general purpose routing and LUTs as required. The available modules are:

- Single Port RAM (RAM_DQ) – EBR based
- Dual PORT RAM (RAM_DP_TRUE) – EBR based
- Pseudo Dual Port RAM (RAM_DP) – EBR based
- Read Only Memory (ROM) – EBR Based
- First In First Out Memory (FIFO and FIFO_DC) – EBR Based
- Distributed Single Port RAM (Distributed_SPRAM) – PFU based
- Distributed Dual Port RAM (Distributed_DPRAM) – PFU based
- Distributed ROM (Distributed_ROM) – PFU/PFF based

IPexpress Flow

For generating any of these memories, create (or open) a project for the LatticeECP/EC or LatticeXP devices.

From the Project Navigator, select **Tools > IPexpress**. Alternatively, users can also click on the button in the toolbar when the LatticeECP/EC and LatticeXP devices are targeted in the project.

This opens the IPexpress window as shown in Figure 9-4.

Figure 9-4. IPexpress - Main Window
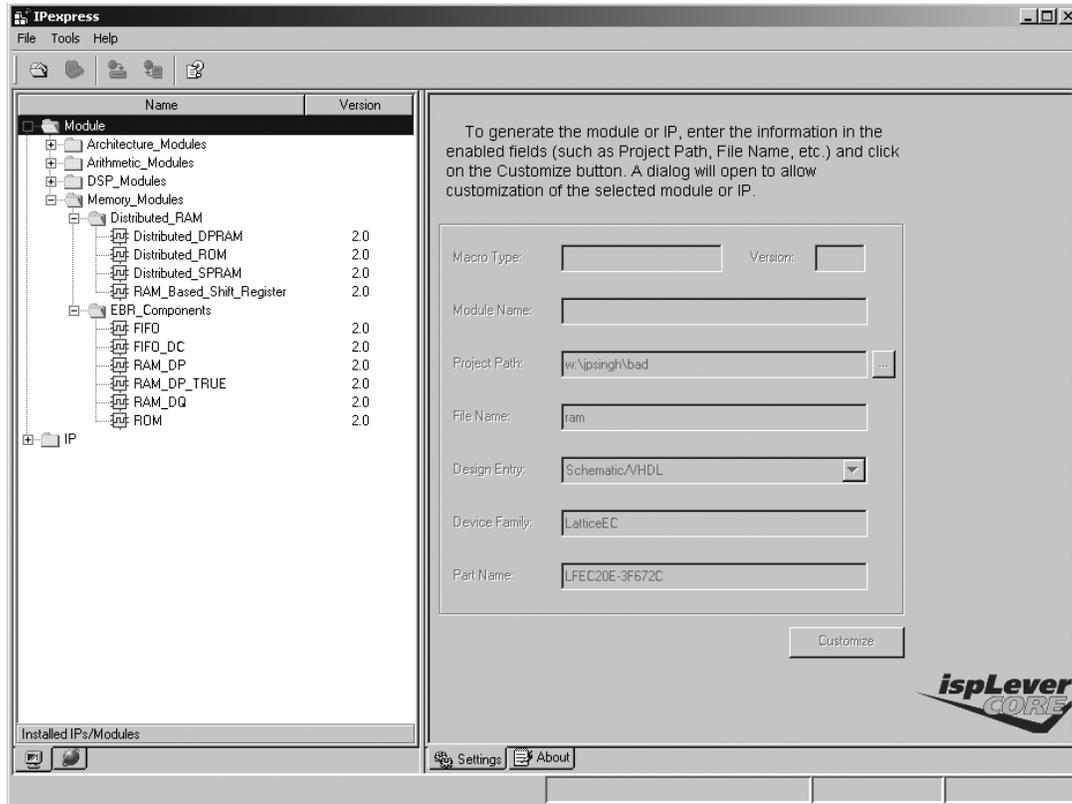
The left pane of this window has the Module Tree. The EBR-based Memory Modules are under the **Module > Memory- Module > Distributed RAM and EBR_Components** and the PFU-based Distributed Memory Modules are under **Storage_Components** as shown in Figure 9-4.

Let us look at an example of the generating an EBR-based Pseudo Dual Port RAM of size 512 x 16. Select RAM_DP under the EBR_Components. The right pane changes, as shown in Figure 9-5.

Figure 9-5. Example Generating Pseudo Dual Port RAM (RAM_DP) Using IPexpress
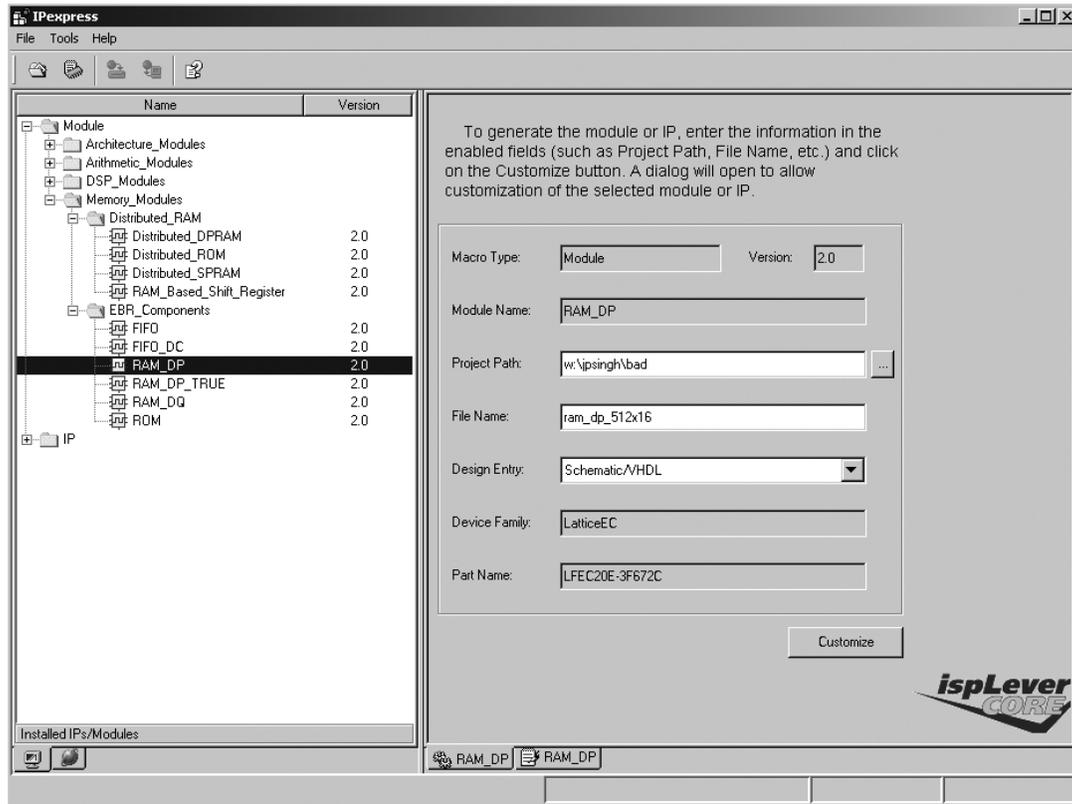
In the right-hand pane, options like **Macro Type**, **Version**, and **Module_Name** are device and selected module dependent. These cannot be changed in IPexpress.

Users can change the directory where the generated module files will be placed by clicking the browse button in the **Project Path**.

The **File Name** text box allows users to specify the entity and file name for the module they are about to generate. Users must provide this name.

Design Entry, Verilog or VHDL, by default is the same as the project type. If the project is a VHDL project, the selected Design Entry option will be "Schematic/ VHDL", and "Schematic/ Verilog-HDL" if the project type is Verilog-HDL.

Then click the **Customize** button. This opens another window where the RAM can be customized.

The the left-hand side of this window shows the block diagram of the module. The right-hand side includes the Configuration tab.

Figure 9-6. Generating Pseudo Dual Port RAM (RAM_DP) Module Customization – Configuration Tab
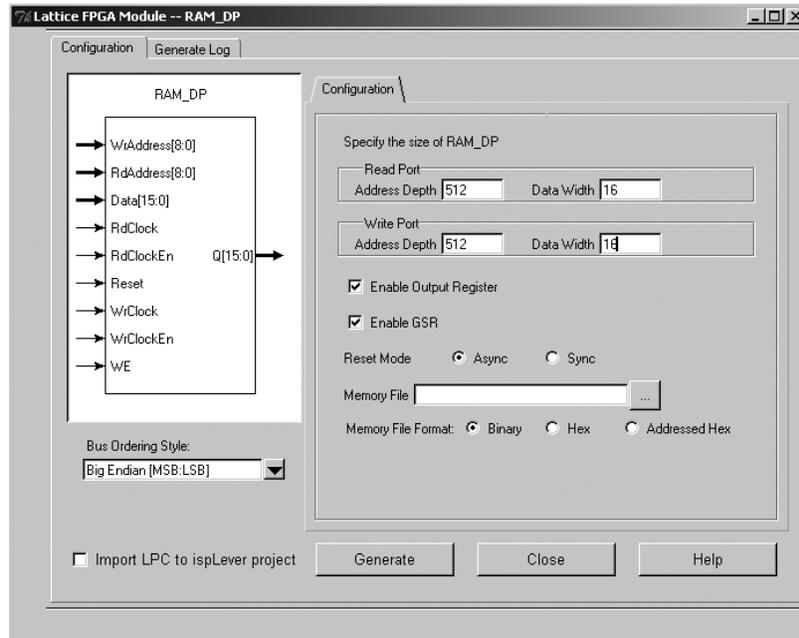
Users can specify the Address Depth and Data width for the **Read Port** and the **Write Port** in the text boxes provided. In this example we are generating a Pseudo Dual Port RAM of size 512 x 16. Users can also create RAMs of different port widths in the case of Pseudo Dual Port and True Dual Port RAMs.

The check box **Enable Output Registers** inserts the output registers in the Read Data Port, as the output registers are optional for the EBR-based RAMs.

The **Reset Mode** can be selected to be Asynchronous Reset or Synchronous Reset. **GSR** or Global Set Reset can be checked to be Enabled or Disabled.

The Input Data and the Address Control is always registered, as the hardware only supports synchronous operation for the EBR based RAMs

Users can also pre-initialize their memory with the contents they specify in the Memory file. It is optional to provide this file in the RAMs. However, in the case of ROM, it is required to provide the Memory file. These files can be of Binary, Hex or Addresses Hex format. The details of these formats are discussed in the Initialization File section of this technical note.

At this point, users can click the **Generate** button to generate the module that they have customized. A netlist in the desired format is then generated and placed in the specified location. Users can incorporate this netlist in their designs.

Users can check the Import LPC to ispLEVER project check box to automatically import the file in the Project Navigator.

Once the Module is generated, users can either instantiate the *.lpc or the Verilog-HDL/ VHDL file in the top level module of their design.

The various memory modules, both EBR and Distributed, are discussed in detail later in this document.

Memory Modules

Single Port RAM (RAM_DQ) – EBR Based

The EBR blocks in the LatticeECP/EC and LatticeXP devices can be configured as Single Port RAM or RAM_DQ. IPexpress allows users to generate the Verilog-HDL or VHDL along an EDIF netlist for the memory size as per the design requirements.

IPexpress generates the memory module as shown in Figure 9-7.

Figure 9-7. Single Port Memory Module generated by IPexpress
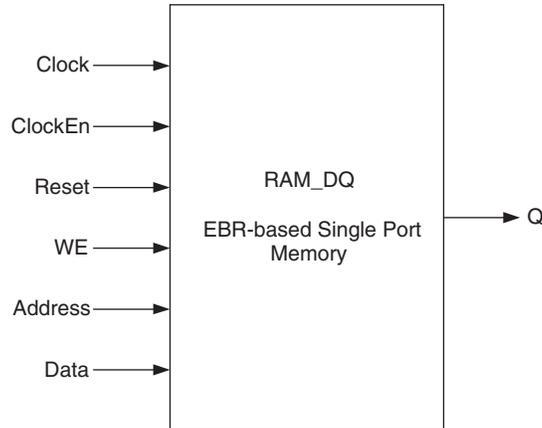
Since the device has a number of EBR blocks, the generated module makes use of these EBR blocks or primitives and cascades them to create the memory sizes specified by the user in the IPexpress GUI. For memory sizes smaller than an EBR block, the module will be created in one EBR block. In cases where the specified memory is larger than one EBR block, multiple EBR block can be cascaded, in depth or width (as required to create these sizes).

The memory primitive for RAM_DQ for LatticeECP/EC and LatticeXP devices is shown in Figure 9-8.

Figure 9-8. Single Port RAM Primitive or RAM_DQ for LatticeECP/EC and LatticeXP Devices
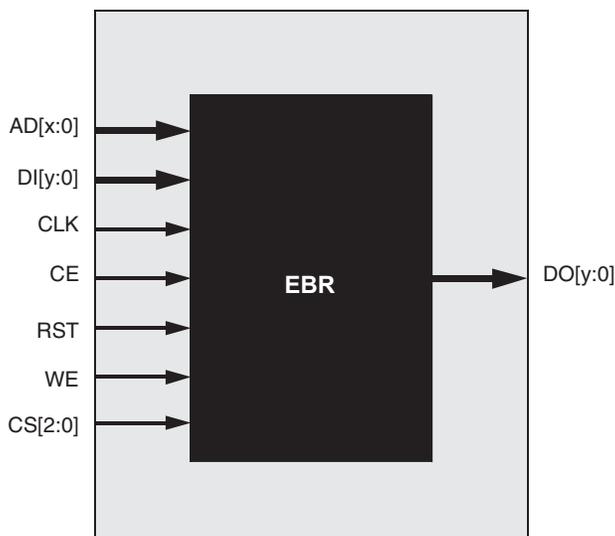
In Single Port RAM mode the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered.

The various ports and their definitions for the Single Port Memory are included in Table 9-1. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DQ primitive.

Table 9-1. EBR-based Single Port Memory Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
Clock	CLK	Clock	Rising Clock Edge
ClockEn	CE	Clock Enable	Active High
Address	AD[x:0]	Address Bus	—
Data	DI[y:0]	Data In	—
Q	DO[y:0]	Data Out	—
WE	WE	Write Enable	Active High
Reset	RST	Reset	Active High
—	CS[2:0]	Chip Select	—

Reset (or RST) only resets the input and output registers of the RAM. It does not reset the contents of the memory.

CS, or Chip Select, a port available in the EBR primitive, 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 easily cascade eight memories. If the memory size specified by the user requires more than eight EBR blocks, the software automatically generates the additional address decoding logic which is implemented in the PFU (external to the EBR blocks).

Each EBR block consists of 9,216 bits of RAM. The values for x (for Address) and y (Data) for each EBR block for the devices are included in Table 9-2.

Table 9-2. Single Port Memory Sizes for 9K Memories for LatticeECP/EC Devices

Single Port Memory Size	Input Data	Output Data	Address [MSB:LSB]
8K x 1	DI	DO	AD[12:0]
4K x 2	DI[1:0]	DO[1:0]	AD[11:0]
2K x 4	DI[3:0]	DO[3:0]	AD[10:0]
1K x 9	DI[8:0]	DO[8:0]	AD[9:0]
512 x 18	DI[17:0]	DO[17:0]	AD[8:0]
256 x 36	DI[35:0]	DO[35:0]	AD[7:0]

Table 9-3 shows the various attributes available for the Single Port Memory (RAM_DQ). Some of these attributes are user selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

Table 9-3. Single Port RAM Attributes for LatticeECP/EC Devices

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
DATA_WIDTH	Data Word Width	1, 2, 4, 9, 18, 36	1	YES
REGMODE	Register Mode (Pipelining)	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
CSDECODE	Chip Select Decode	000, 001, 010, 011, 100, 101, 110, 111	000	NO
WRITEMODE	Read / Write Mode	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	YES
GSR	Global Set Reset	ENABLED, DISABLED	ENABLED	YES

The Single Port RAM (RAM_DQ) can be configured as NORMAL, READ BEFORE WRITE or WRITE THROUGH modes. Each of these modes affects what data comes out of the port Q of the memory during the write operation followed by the read operation at the same memory location. The READ BEFORE WRITE attribute is supported for x9, x18 and x36 data widths.

Additionally users can select to enable the output registers for RAM_DQ. Figures 8-7 through 8-12 show the internal timing waveforms for the Single Port RAM (RAM_DQ) with these options.

Figure 9-9. Single Port RAM Timing Waveform – NORMAL Mode, without Output Registers
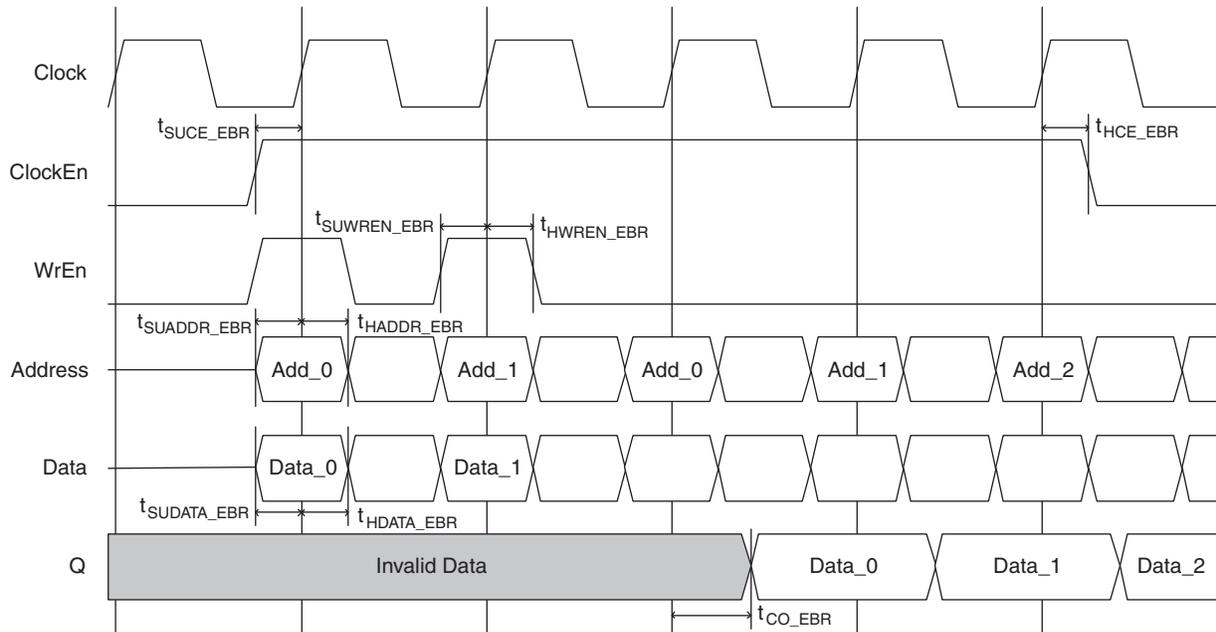
Figure 9-10. Single Port RAM Timing Waveform – NORMAL Mode, with Output Registers
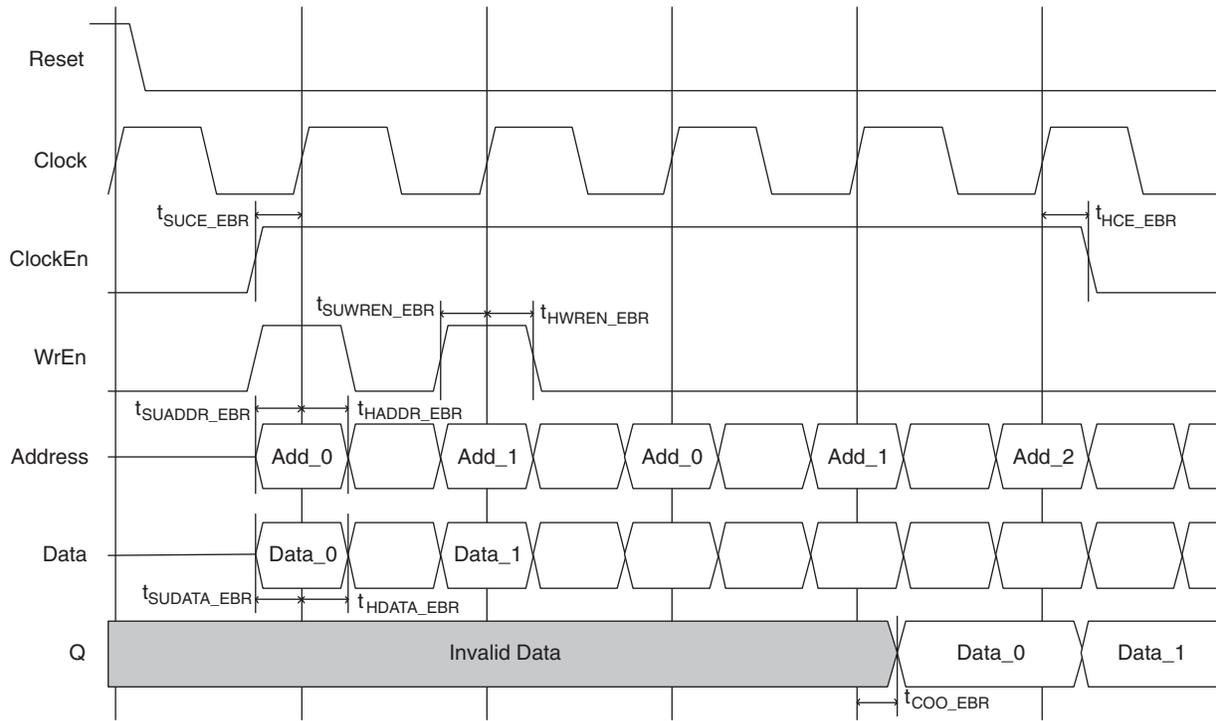
Figure 9-11. Single Port RAM Timing Waveform – READ BEFORE WRITE Mode, without Output Registers
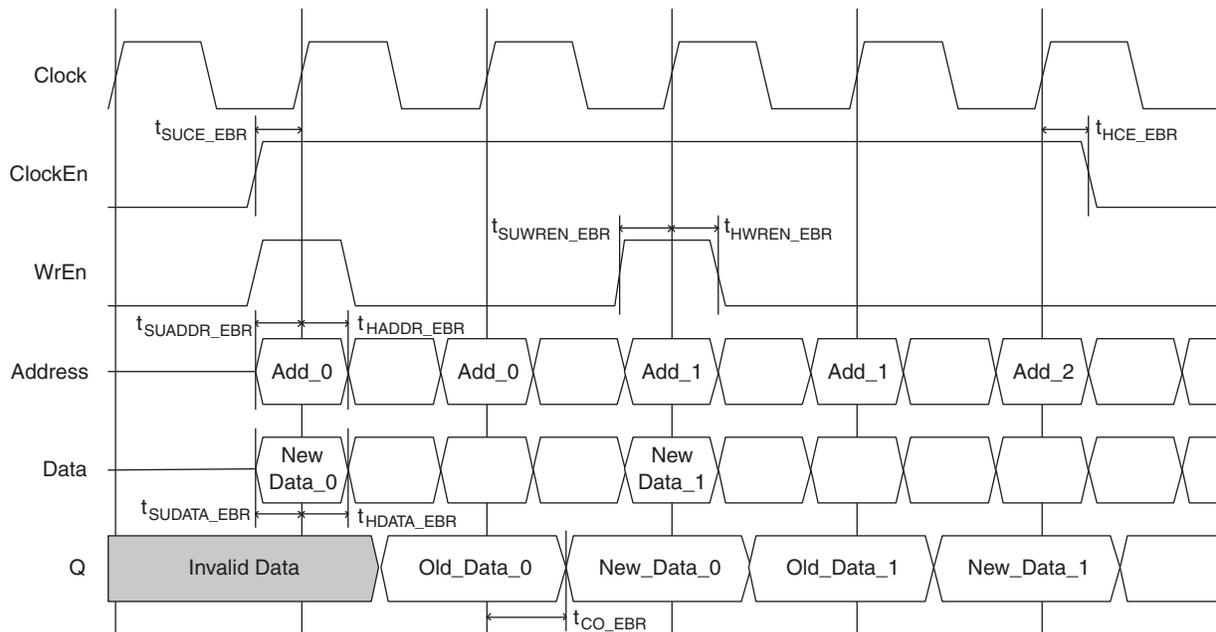
Figure 9-12. Single Port RAM Timing Waveform – READ BEFORE WRITE Mode, with Output Registers
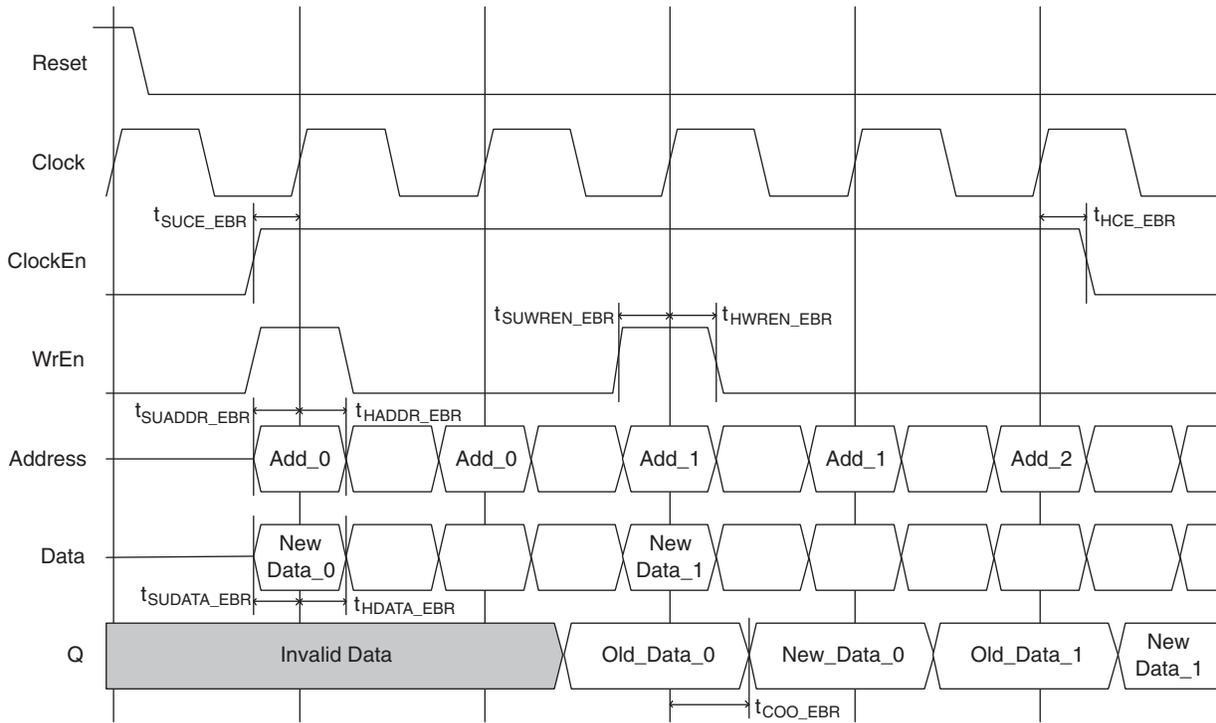
Figure 9-13. Single Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers
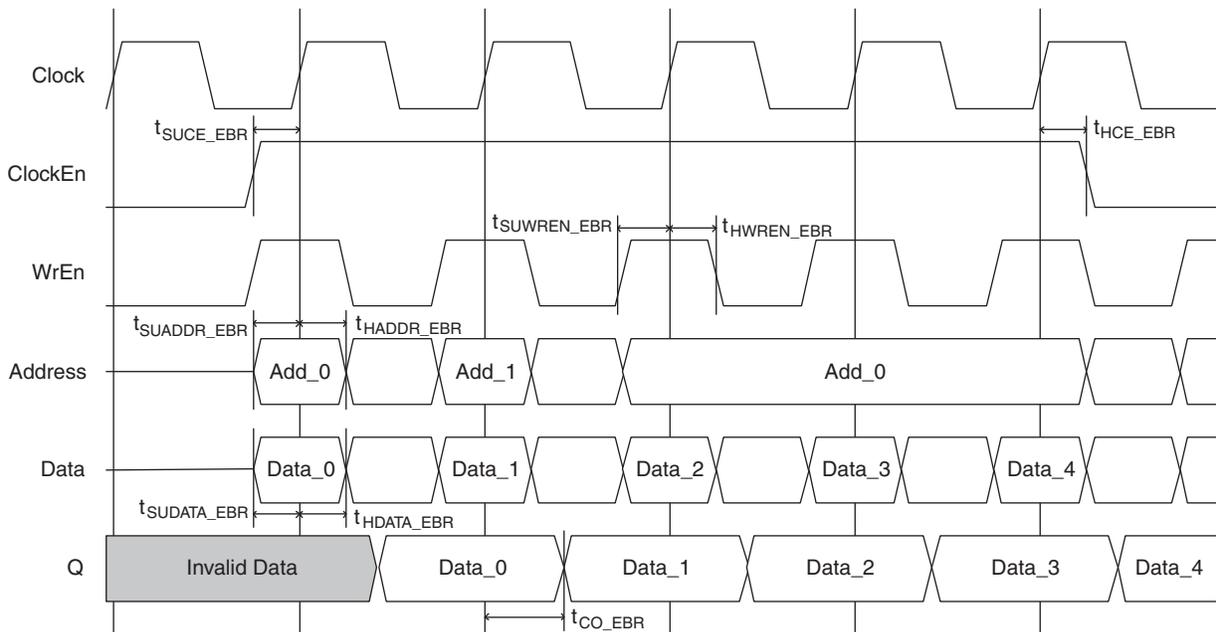
Figure 9-14. Single Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers
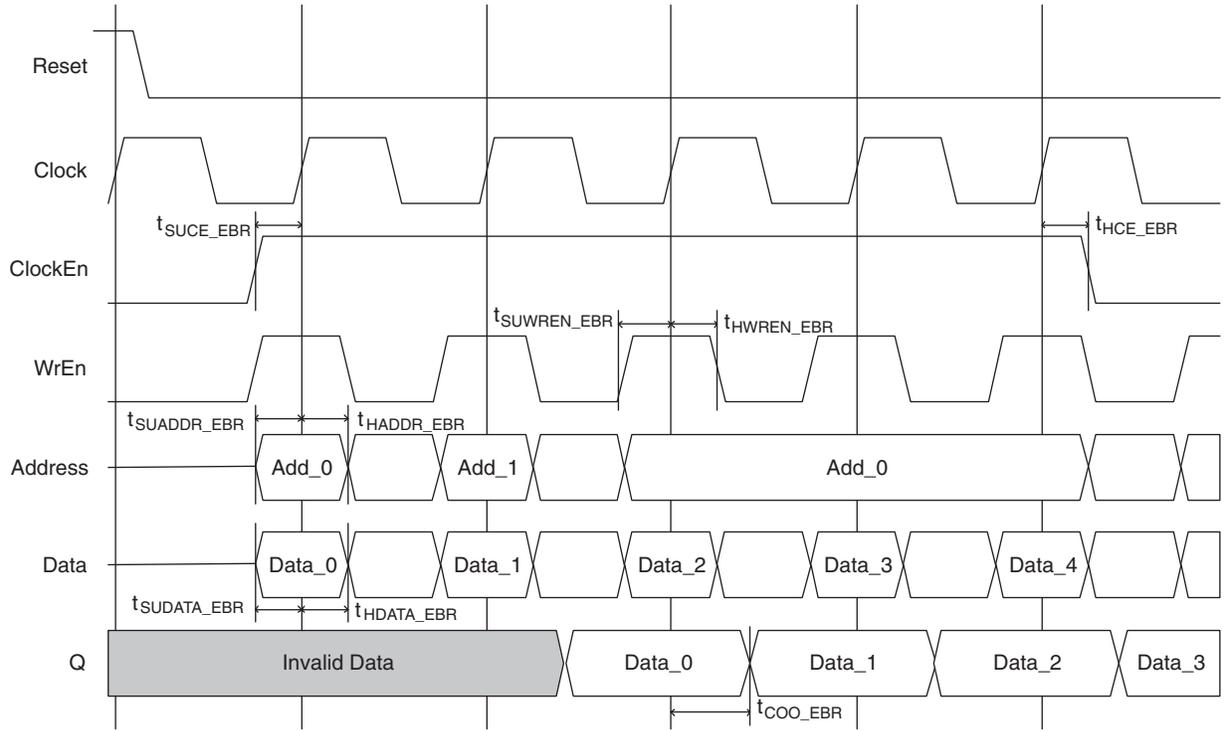
True Dual Port RAM (RAM_DP_TRUE) – EBR Based

The EBR blocks in the LatticeECP/EC and LatticeXP devices can be configured as True-Dual Port RAM or RAM_DP_TRUE. IPexpress allows users to generate the Verilog-HDL, VHDL or EDIF netlists for the memory size as per design requirements.

IPexpress generates the memory module as shown in Figure 9-15.

Figure 9-15. True Dual Port Memory Module Generated by IPexpress
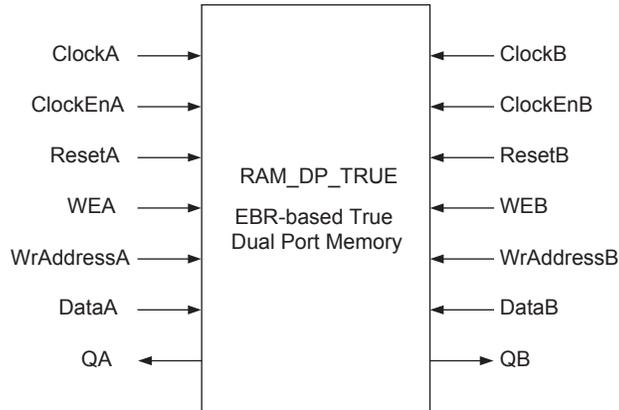
The generated module makes use of the RAM_DP_TRUE primitive. For memory sizes smaller than one EBR block, the module will be created in one EBR block. In cases where the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width (as required to create these sizes).

The basic memory primitive for the LatticeECP/EC and LatticeXP devices, RAM_DP_TRUE, is shown in Figure 9-16.

Figure 9-16. True Dual Port RAM Primitive or RAM_DP_TRUE for LatticeECP/EC and LatticeXP Devices
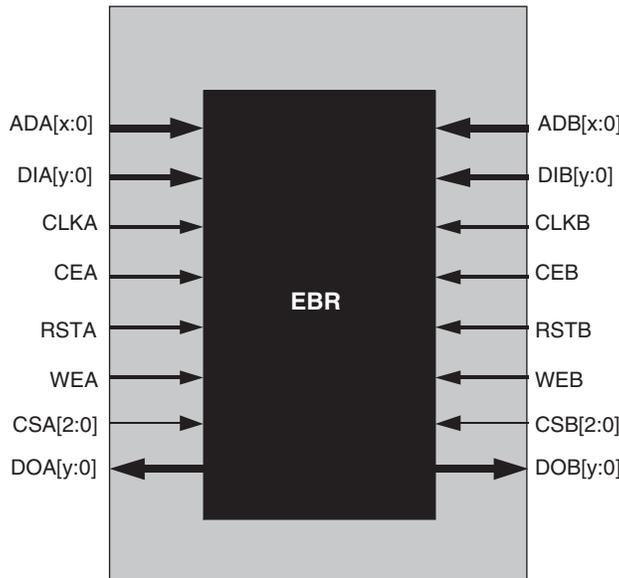
In True Dual Port RAM mode, the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions for the True Dual Memory are included in Table 9-4. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DP_TRUE primitive.

Table 9-4. EBR-based True Dual Port Memory Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
ClockA, ClockB	CLKA, CLKB	Clock for PortA and PortB	Rising Clock Edge
ClockEnA, ClockEnB	CEA, CEB	Clock Enables for Port CLKA and CLKB	Active High
AddressA, AddressB	ADA[x:0], ADB[x:0]	Address Bus Port A and Port B	—
DataA, DataB	DIA[y:0], DIB[y:0]	Input Data Port A and Port B	—
QA, QB	DOA[y:0], DOB[y:0]	Output Data Port A and Port B	—
WEA, WEB	WEA, WEB	Write Enable Port A and Port B	Active High
ResetA, ResetB	RSTA, RSTB	Reset for Port A and Port B	Active High
—	CSA[2:0], CSB[2:0]	Chip Selects for Each Port	—

Reset (or RST) only resets the input and output registers of the RAM. It does not reset the contents of the memory.

CS, or Chip Select, a port available in the EBR primitive, is useful when memory requires multiple EBR blocks to be cascaded. The CS signal would form the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can easily cascade eight memories. However, if the memory size specified by the user requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU external to the EBR blocks.

Each EBR block consists of 9,216 bits of RAM. The values for x (for Address) and y (Data) for each EBR block for the devices are included in Table 9-5.

Table 9-5. True Dual Port Memory Sizes for 9K Memory for LatticeECP/EC and LatticeXP Devices

Dual Port Memory Size	Input Data Port A	Input Data Port B	Output Data Port A	Output Data Port B	Address Port A [MSB:LSB]	Address Port B [MSB:LSB]
8K x 1	DIA	DIB	DOA	DOB	ADA[12:0]	ADB[12:0]
4K x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	ADA[11:0]	ADB[11:0]
2K x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	ADA[10:0]	ADB[10:0]
1K x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	ADA[9:0]	ADB[9:0]
512 x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	ADA[8:0]	ADB[8:0]

Table 9-6 shows the various attributes available for True Dual Port Memory (RAM_DP_TRUE). Some of these attributes are user selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

Table 9-6. True Dual Port RAM Attributes for LatticeECP/EC and LatticeXP

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
DATA_WIDTH_A	Data Word Width Port A	1, 2, 4, 9, 18	1	YES
DATA_WIDTH_B	Data Word Width Port B	1, 2, 4, 9, 18	1	YES
REGMODE_A	Register Mode (Pipelining) for Port A	NOREG, OUTREG	NOREG	YES
REGMODE_B	Register Mode (Pipelining) for Port B	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
CSDECODE_A	Chip Select Decode for Port A	000, 001, 010, 011, 100, 101, 110, 111	000	NO
CSDECODE_B	Chip Select Decode for Port B	000, 001, 010, 011, 100, 101, 110, 111	000	NO
WRITEMODE_A	Read / Write Mode for Port A	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	YES
WRITEMODE_B	Read / Write Mode for Port B	NORMAL, WRITETHROUGH, READBEFOREWRITE	NORMAL	YES
GSR	Global Set Reset	ENABLED, DISABLED	ENABLED	YES

The True Dual Port RAM (RAM_DP_TRUE) can be configured as NORMAL, READ BEFORE WRITE or WRITE THROUGH modes. Each of these modes affects what data comes out of the port Q of the memory during the write operation followed by the read operation at the same memory location. The READ BEFORE WRITE attribute is supported for x9 and x18 data widths. Detailed discussions of the WRITE modes and the constraints of the True Dual Port can be found in Appendix A.

Additionally users can select to enable the output registers for RAM_DP_TRUE. Figures 8-15 through 8-20 show the internal timing waveforms for the True Dual Port RAM (RAM_DP_TRUE) with these options.

Figure 9-17. True Dual Port RAM Timing Waveform – NORMAL Mode, without Output Registers

Figure 9-18. True Dual Port RAM Timing Waveform – NORMAL Mode with Output Registers

Figure 9-19. True Dual Port RAM Timing Waveform – READ BEFORE WRITE Mode, without Output Registers
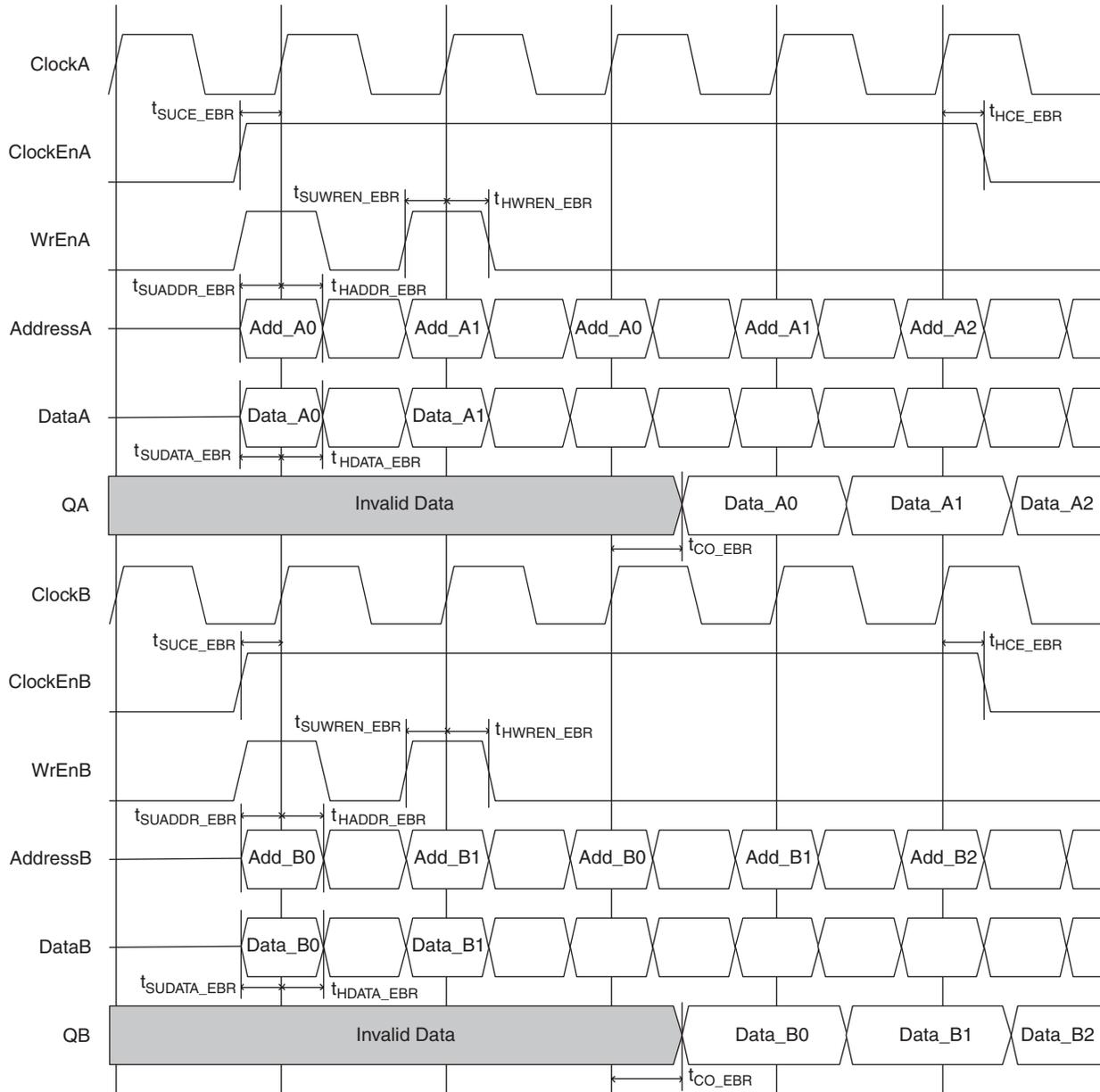
Figure 9-20. True Dual Port RAM Timing Waveform – READ BEFORE WRITE Mode, with Output Registers

Figure 9-21. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

Figure 9-22. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers
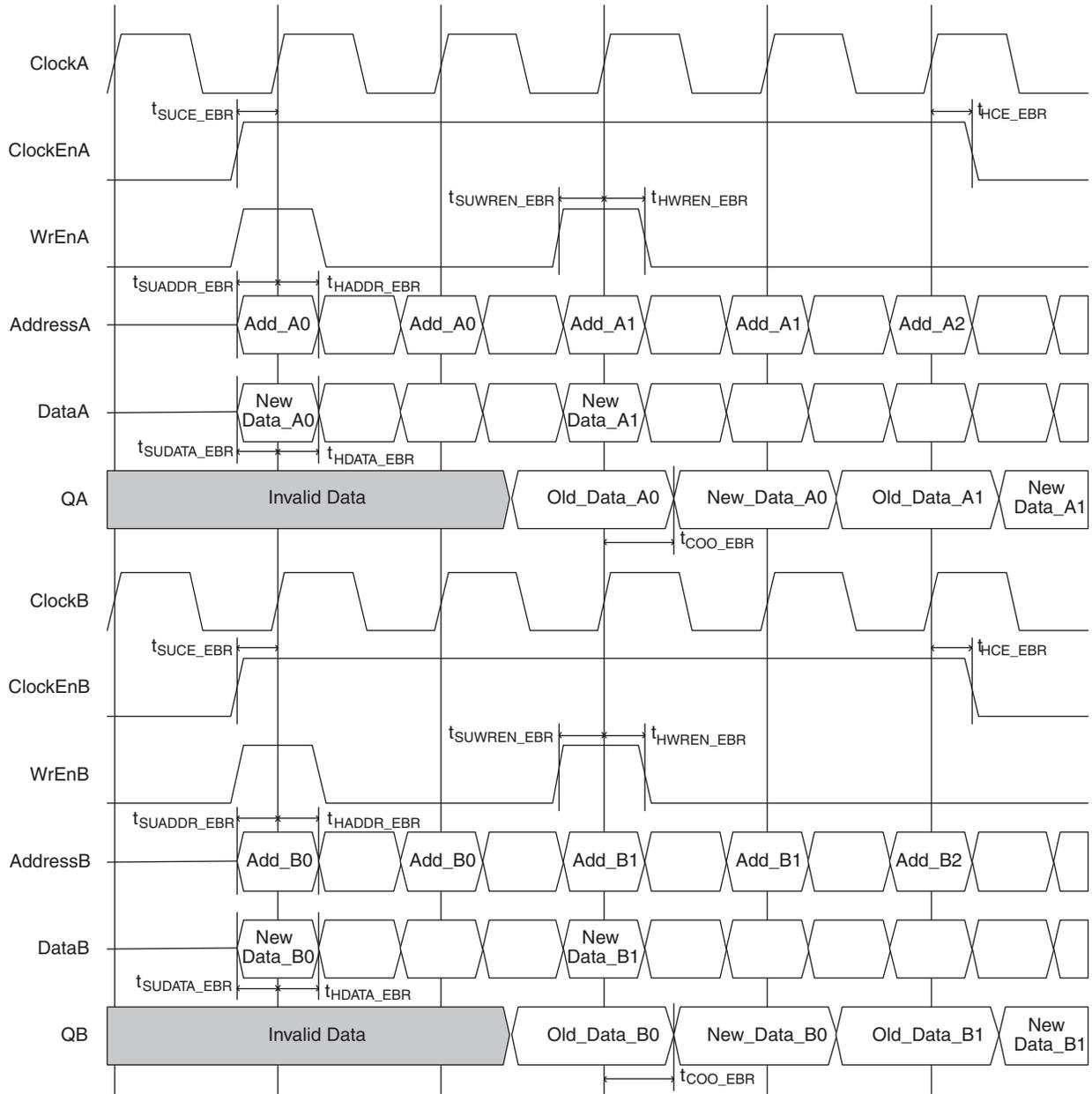
Pseudo Dual Port RAM (RAM_DP) – EBR-Based

The EBR blocks in the LatticeECP/EC and LatticeXP devices can be configured as Pseudo-Dual Port RAM or RAM_DP. IPexpress allows users to generate the Verilog-HDL or VHDL along with an EDIF netlist for the memory size as per design requirements.

IPexpress generates the memory module, as shown in Figure 9-23.

Figure 9-23. Pseudo Dual Port Memory Module Generated by IPexpress
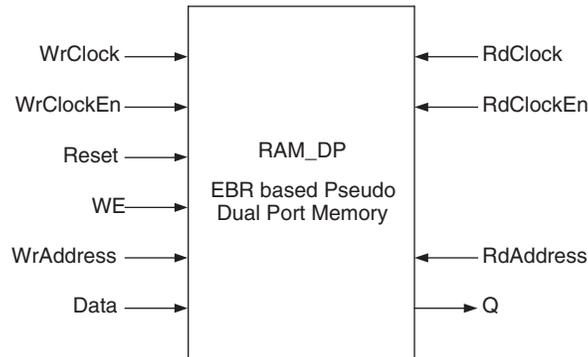
The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR block can be cascaded, in depth or width (as required to create these sizes).

The basic Pseudo Dual Port memory primitive for the LatticeECP/EC and LatticeXP devices is shown in Figure 9-24.

Figure 9-24. Pseudo Dual Port RAM primitive or RAM_DP for LatticeECP/EC and LatticeXP Devices
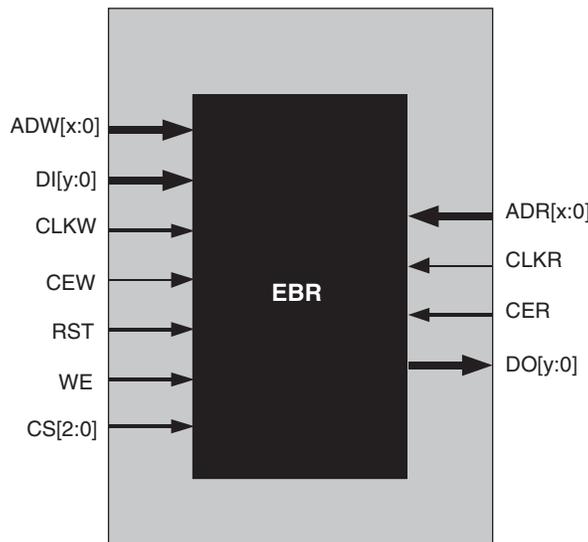
In the Pseudo Dual Port RAM mode, the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions for the Single Port Memory are included in Table 9-7. The table lists the corresponding ports for the module generated by IPexpress and for the EBR RAM_DP primitive.

Table 9-7. EBR based Pseudo-Dual Port Memory Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
RdAddress	ADR[x:0]	Read Address	—
WrAddress	ADW[x:0]	Write Address	—
RdClock	CLKR	Read Clock	Rising Clock Edge
WrClock	CLKW	Write Clock	Rising Clock Edge
RdClockEn	CER	Read Clock Enable	Active High
WrClockEn	CEW	Write Clock Enable	Active High
Q	DO[y:0]	Read Data	—
Data	DI[y:0]	Write Data	—
WE	WE	Write Enable	Active High
Reset	RST	Reset	Active High
—	CS[2:0]	Chip Select	—

Reset (or RST) only resets the input and output registers of the RAM. It does not reset the contents of the memory.

CS, or Chip Select, a port available in the EBR primitive, is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade eight memories easily. However, if the memory size specified by the user requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

Each EBR block consists of 9,216 bits of RAM. The values for x (for Address) and y (Data) for each EBR block for the devices are included in Table 9-8.

Table 9-8. Pseudo-Dual Port Memory Sizes for 9K Memory for LatticeECP/EC and LatticeXP Devices

Pseudo-Dual Port Memory Size	Input Data Port A	Input Data Port B	Output Data Port A	Output Data Port B	Read Address Port A [MSB:LSB]	Write Address Port B [MSB:LSB]
8K x 1	DIA	DIB	DOA	DOB	RAD[12:0]	WAD[12:0]
4K x 2	DIA[1:0]	DIB[1:0]	DOA[1:0]	DOB[1:0]	RAD[11:0]	WAD[11:0]
2K x 4	DIA[3:0]	DIB[3:0]	DOA[3:0]	DOB[3:0]	RAD[10:0]	WAD[10:0]
1K x 9	DIA[8:0]	DIB[8:0]	DOA[8:0]	DOB[8:0]	RAD[9:0]	WAD[9:0]
512 x 18	DIA[17:0]	DIB[17:0]	DOA[17:0]	DOB[17:0]	RAD[9:0]	WAD[9:0]

Table 9-9 shows the various attributes available for the Pseudo Dual Port Memory (RAM_DP). Some of these attributes are user selectable through the IPexpress GUI. For detailed attribute definitions, refer to Appendix A.

Table 9-9. Pseudo-Dual Port RAM Attributes for LatticeECP/EC and LatticeXP Devices

Attribute	Description	Values	Default Value	User Selectable Through IPexpress
DATA_WIDTH_W	Write Data Word Width	1, 2, 4, 9, 18, 36	1	YES
DATA_WIDTH_R	Read Data Word Width	1, 2, 4, 9, 18, 36	1	YES
REGMODE	Register Mode (Pipelining)	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
CSDECODE_W	Chip Select Decode for Write	000, 001, 010, 011, 100, 101, 110, 111	000	NO
CSDECODE_R	Chip Select Decode for Read	000, 001, 010, 011, 100, 101, 110, 111	000	NO
GSR	Global Set Reset	ENABLED, DISABLED	ENABLED	YES

Users have the option of enabling the output registers for Pseudo-Dual Port RAM (RAM_DP). Figures 8-23 and 8-24 show the internal timing waveforms for the Pseudo-Dual Port RAM (RAM_DP) with these options.

Figure 9-25. PSEUDO DUAL PORT RAM Timing Diagram – without Output Registers
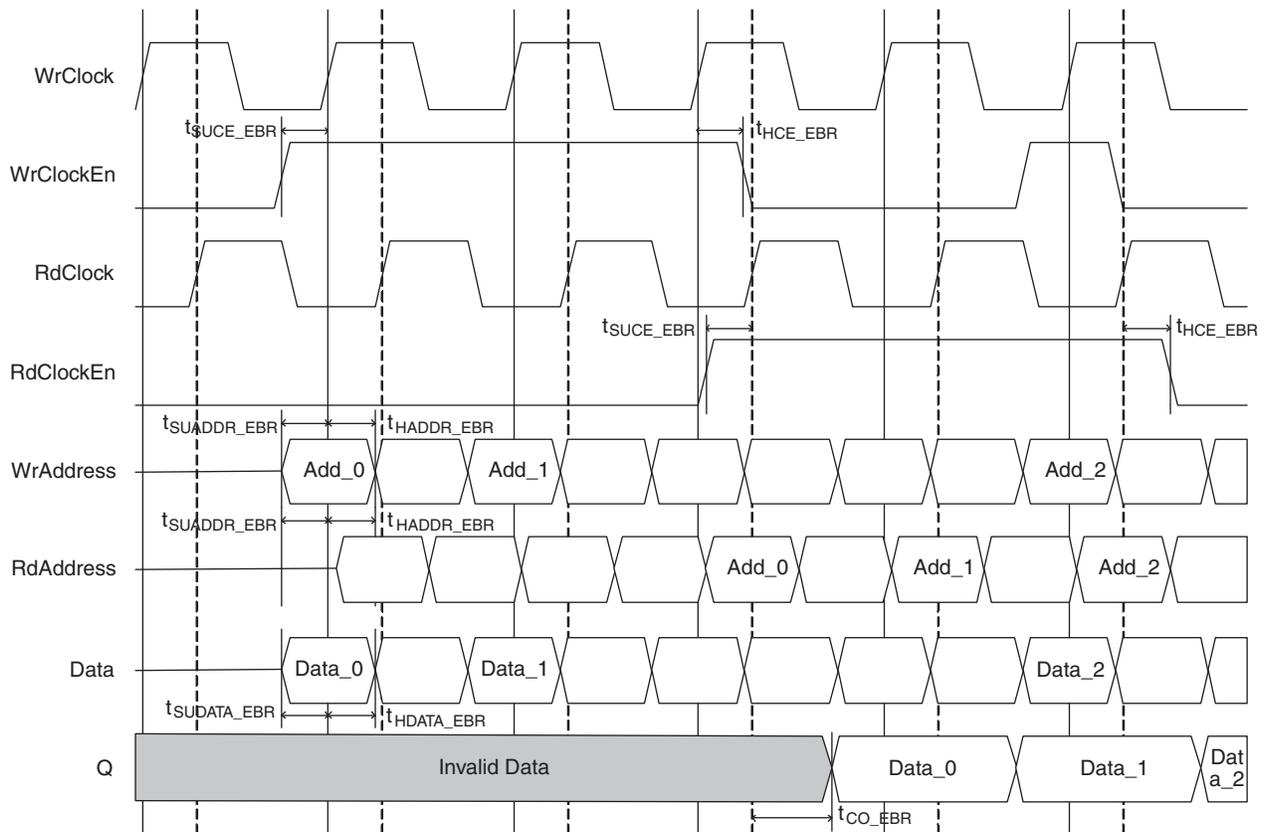
Figure 9-26. PSEUDO DUAL PORT RAM Timing Diagram – with Output Registers
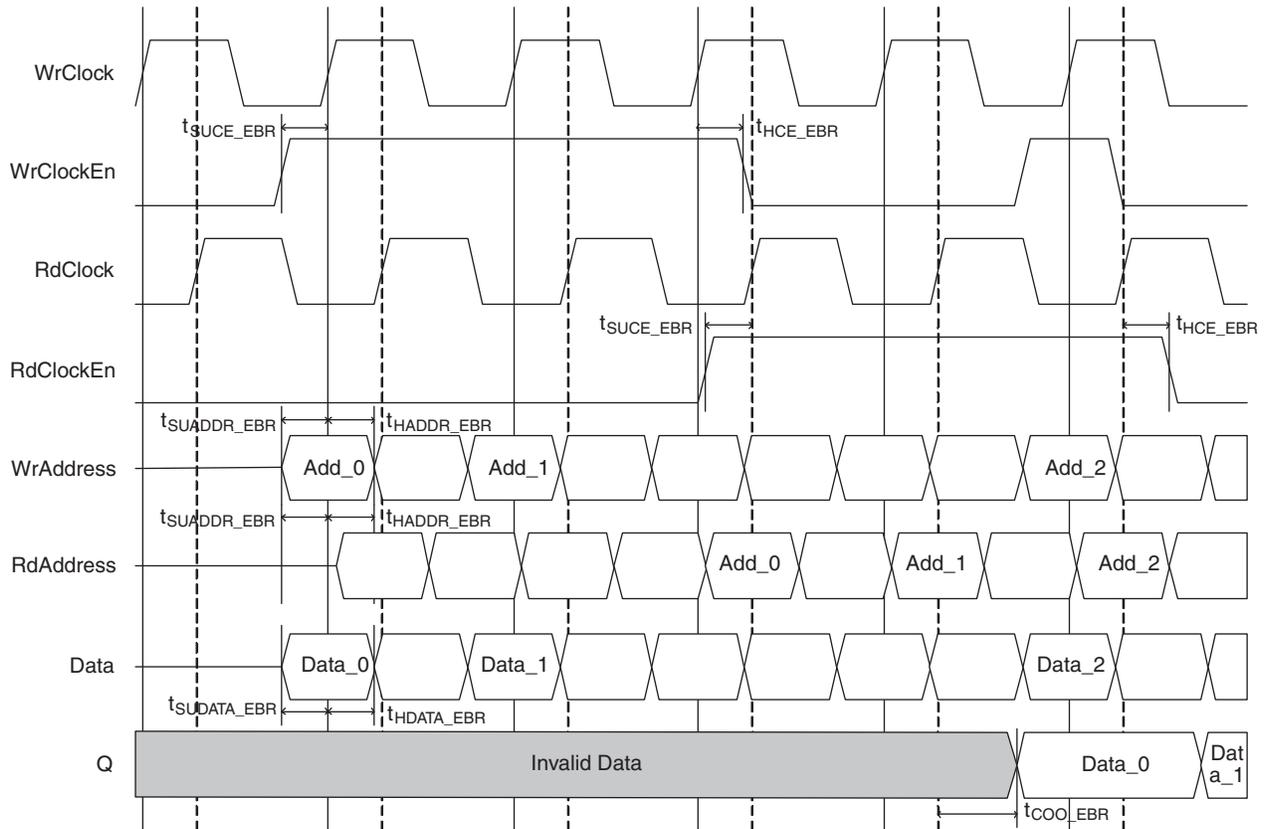
Read Only Memory (ROM) – EBR Based

The EBR blocks in the LatticeECP/EC and LatticeXP devices can be configured as Read Only Memory or ROM. IPexpress allows users to generate the Verilog-HDL or VHDL along with an EDIF netlist for the memory size as per design requirements. Users are required to provide the ROM memory content in the form of an initialization file.

IPexpress generates the memory module as shown in Figure 9-27.

Figure 9-27. ROM - Read Only Memory Module Generated by IPexpress
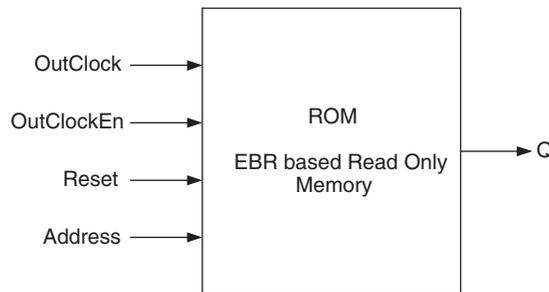
The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width (as required to create these sizes).

The basic ROM primitive for the LatticeECP/EC and LatticeXP devices is as shown in Figure 9-28.

Figure 9-28. ROM Primitive for LatticeECP/EC and LatticeXP Devices
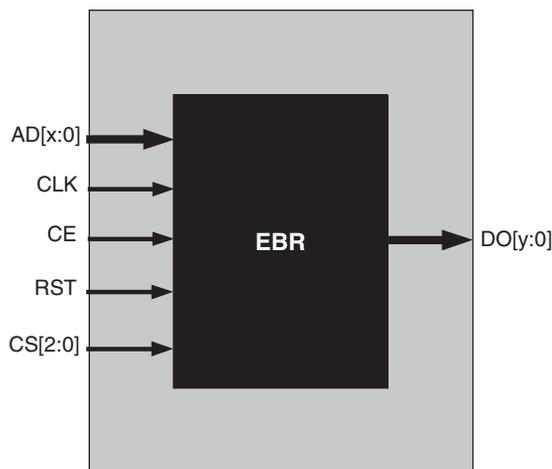
In the ROM mode the address for the port is registered at the input of the memory array. The output data of the memory is optionally registered at the output.

The various ports and their definitions for the ROM are included in Table 9-10. The table lists the corresponding ports for the module generated by IPexpress and for the ROM primitive.

Table 9-10. EBR-based ROM Port Definitions

Port Name in generated Module	Port Name in the EBR block primitive	Description	Active State
Address	AD[x:0]	Read Address	—
OutClock	CLK	Clock	Rising Clock Edge
OutClockEn	CE	Clock Enable	Active High
Reset	RST	Reset	Active High
—	CS[2:0]	Chip Select	—

Reset (or RST) only resets the input and output registers of the RAM. It does not reset the contents of the memory.

CS, or Chip Select, a port available in the EBR primitive, is useful when memory requires multiple EBR blocks to be cascaded. The CS signal forms the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade eight memories easily. However, if the memory size specified by the user requires more than eight EBR blocks, the software automatically generates the additional address decoding logic, which is implemented in the PFU (external to the EBR blocks).

While generating the ROM using IPexpress, the user is required to provide an initialization file to pre-initialize the contents of the ROM. These file are the *.mem files and they can be of Binary, Hex or the Addressed Hex formats. The initialization files are discussed in detail in the Initializing Memory section of this technical note.

Users have the option of enabling the output registers for Read Only Memory (ROM). Figures 8-27 and 8-28 show the internal timing waveforms for the Read Only Memory (ROM) with these options.

Figure 9-29. ROM Timing Waveform – without Output Registers
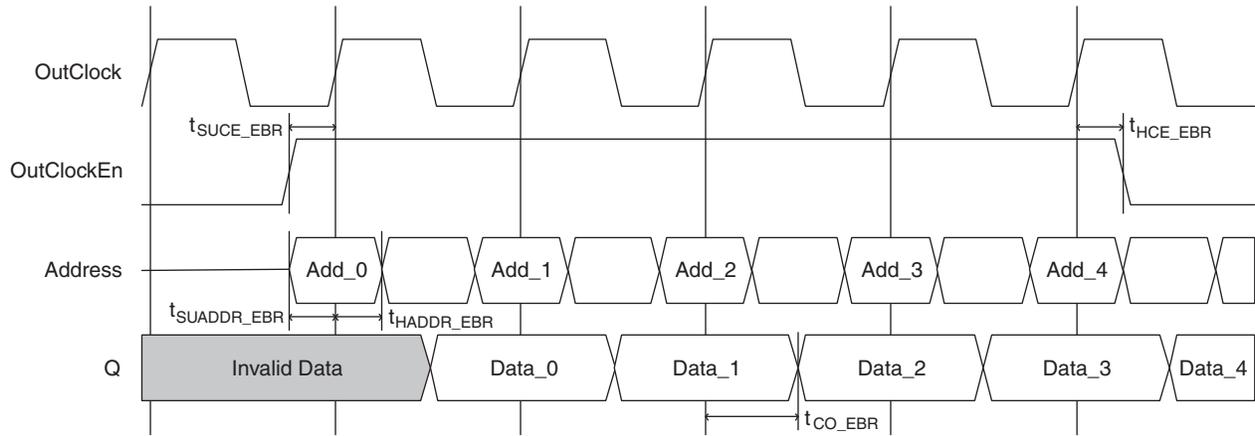
Figure 9-30. ROM Timing Waveform – with Output Registers
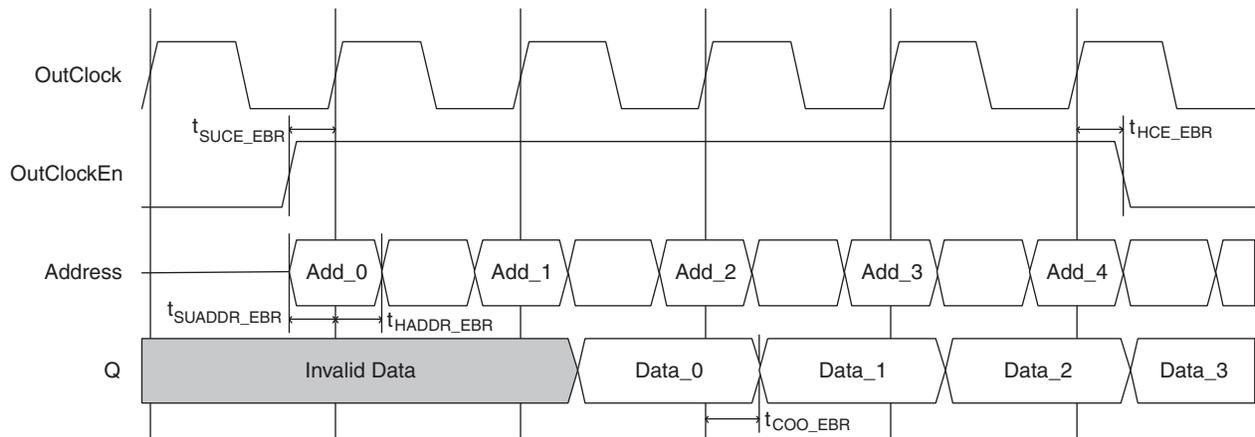
First In First Out (FIFO, FIFO_DC) – EBR Based

The EBR blocks in the LatticeECP/EC and LatticeXP devices can be configured as First In First Out Memories – FIFO and FIFO_DC. FIFO has a common clock for both read and write ports and FIFO_DC (or Dual Clock FIFO) has separate clocks for these ports. IPexpress allows users to generate the Verilog-HDL or VHDL along with an EDIF netlist for the memory size as per design requirement.

IPexpress generates the FIFO and FIFO_DC memory module as shown in Figures 9-31 and 9-32.

Figure 9-31. FIFO Module Generated by IPexpress
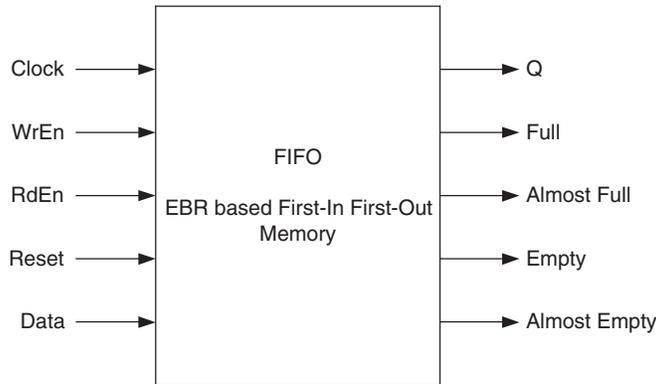
Figure 9-32. FIFO_DC Module Generated by IPexpress
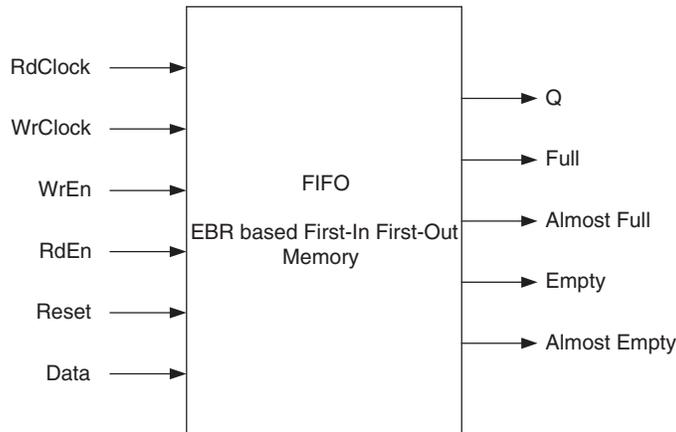
LatticeECP/EC and LatticeXP devices do not have a built in FIFO. These devices have an emulated FIFO and FIFO_DC. These are emulated by creating a wrapper around the existing RAMs (like RAM_DP). This wrapper also includes address pointer generation and FIFO flag generation logic which will be implemented external to the EBR block. Therefore, in addition to the regular EBR usage, there is extra logic for the address pointer generation and FIFO flag generation.

A clock is always required as only synchronous write is supported. The various ports and their definitions for the FIFO and FIFO_DC are included in Table 11.

Table 9-11. EBR-based FIFO and FIFO_DC Memory Port Definitions

Port Name in Generated Module	Description	
CLK	Clock (FIFO)	Rising Clock Edge
CLKR	Read Port Clock (FIFO_DC)	Rising Clock Edge
CLKW	Write Port Clock (FIFO_DC)	Rising Clock Edge
WE	Write Enable	Active High
RE	Read Enable	Active High
RST	Reset	Active High
DI	Data Input	—
DO	Data Output	—
FF	Full Flag	Active High
AF	Almost Full Flag	Active High
EF	Empty Flag	Active High
AE	Almost Empty	Active High

Reset (or RST) only resets the output registers of the FIFO and FIFO_DC. It does not reset the contents of the memory.

The various supported sizes for the FIFO and FIFO_DC in LatticeECP/EC and LatticeXP devices are shown in Table 9-12.

Table 9-12. FIFO and FIFO_DC Data Widths Sizes for LatticeECP/EC and LatticeXP Devices

FIFO Size	Input Data	Output Data
8K x 1	DI	DO
4K x 2	DI[1:0]	DO[1:0]
2K x 4	DI[3:0]	DO[3:0]
1K x 9	DI[8:0]	DO[8:0]
512 x 18	DI[17:0]	DO[17:0]
256 x 36	DI[35:0]	DO[35:0]

FIFO Flags

The FIFO and FIFO_DC have four flags available: Empty, Almost Empty, Almost Full and Full. The Almost Empty and Almost Full flags have a programmable range.

The program ranges for the four FIFO flags are specified in Table 9-13.

Table 9-13. FIFO Flag Settings

FIFO Attribute Name	Description	Programming Range	Program Bits
FF	Full flag setting	2N - 1	14
AFF	Almost full setting	1 to (FF-1)	14
AEF	Almost empty setting	1 to (FF-1)	14
EF	Empty setting	0	5

The only restriction on the flag setting is that the values must be in a specific order (Empty=0, Almost Empty next, followed by Almost Full and Full, respectively). The value of Empty is not equal to the value of Almost Empty (or Full is equal to Almost Full). In this case, a warning is generated and the value of Empty (or Full) is used in place of Almost Empty (or Almost Full). When coming out of reset, the Active High Flags empty and Almost Empty are set to high, since they are true.

The user should specify the absolute value of the address at which the Almost Empty and Almost Full Flags will go true. For example, if the Almost Full Flag is required to go true at the address location 500 for a FIFO of depth 512, the user should specify the value 500 in the IPexpress.

The Empty and Almost Empty Flags are always registered with the read clock and the Full and Almost Full Flags are always registered to the write clock.

FIFO Operation
FIFOs are not supported in the hardware. The hardware has Embedded block RAMs (EBR) which can be configured in Single Port (RAM_DQ), Pseudo-Dual Port (RAM_DP) and True Dual Port (RAM_DP_TRUE) RAMs. The FIFOs in these devices are emulated FIFOs that are built around these RAMs.

Each of these FIFOs can be configured with (pipelined) and without (non-pipelined) output registers. In the pipelined mode users have an extra option for these output registers to be enabled by the RdEn signal. We will discuss the operation in the following sections.

Let us take a look at the operation of these FIFOs.

First In First Out (FIFO) Memory: The FIFO or the single clock FIFO is an emulated FIFO. The address logic and the flag logic is implemented in the FPGA fabric around the RAM.

The ports available on the FIFO are:

- Reset
- Clock
- WrEn
- RdEn
- Data
- Q
- Full Flag
- Almost Full Flag
- Empty Flag
- Almost Empty Flag

Let us first discuss the non-pipelined or the FIFO without output registers. Figure 9-33 shows the operation of the FIFO when it is empty and the data starts to get written into it.

Figure 9-33. FIFO Without Output Registers, Start of Data Write Cycle
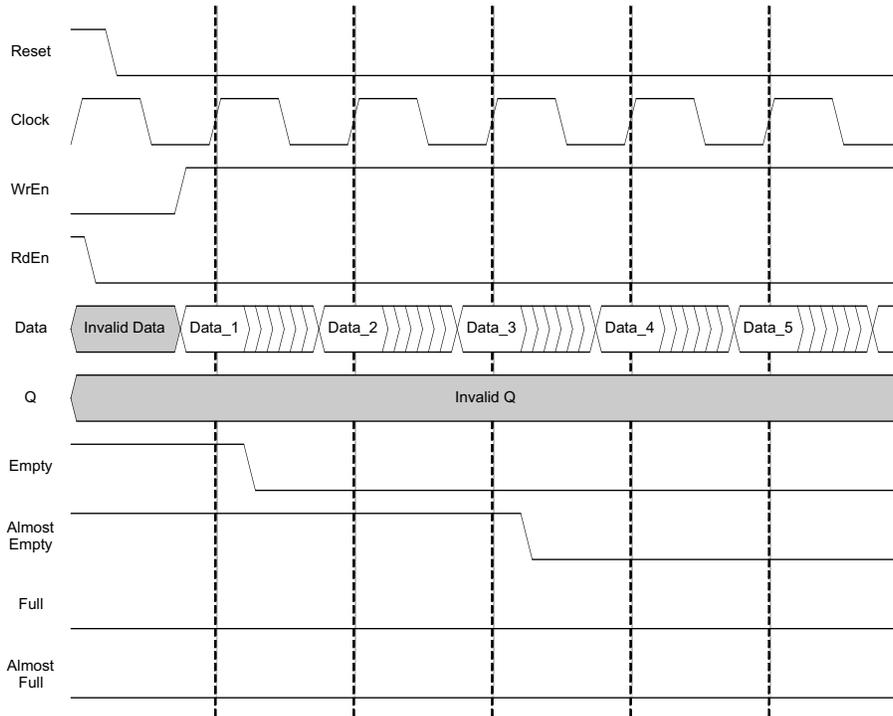
The WrEn signal has to be high to start writing into the FIFO. The Empty and Almost Empty flags are high to begin and Full and Almost full are low.

When the first data gets written into the FIFO, the Empty flag de-asserts (or goes low), as the FIFO is no longer empty. In this figure we are assuming that the Almost Empty setting flag setting is 3 (address location 3). So the Almost Empty flag gets de-asserted when the 3rd address location gets filled.

Now let is 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 9-34 shows the behavior of these flags. In this figure we assume that FIFO depth is 'N'.

Figure 9-34. FIFO Without Output Registers, End of Data Write Cycle
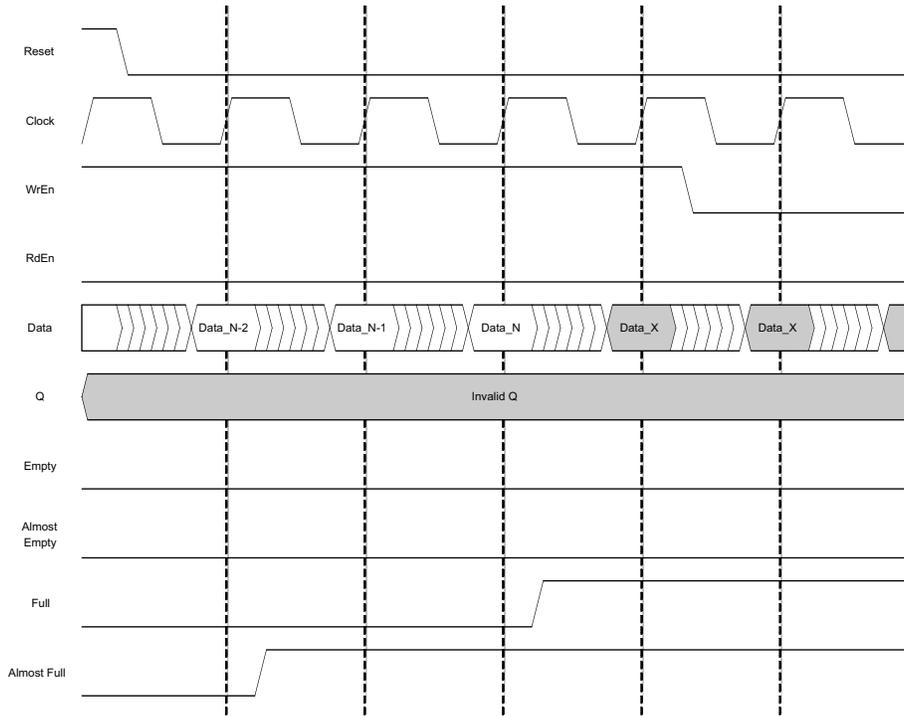
In this case, as seen above, the Almost Full flag is IN location 2 before the FIFO is filled. The Almost Full flag is asserted when N-2 location is written, and Full flag is asserted when the last word is written into the FIFO.

Data_X data inputs do not get written as 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 9-35 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags gets de-asserted as shown.

Figure 9-35. FIFO Without Output Registers, Start of Data Read Cycle
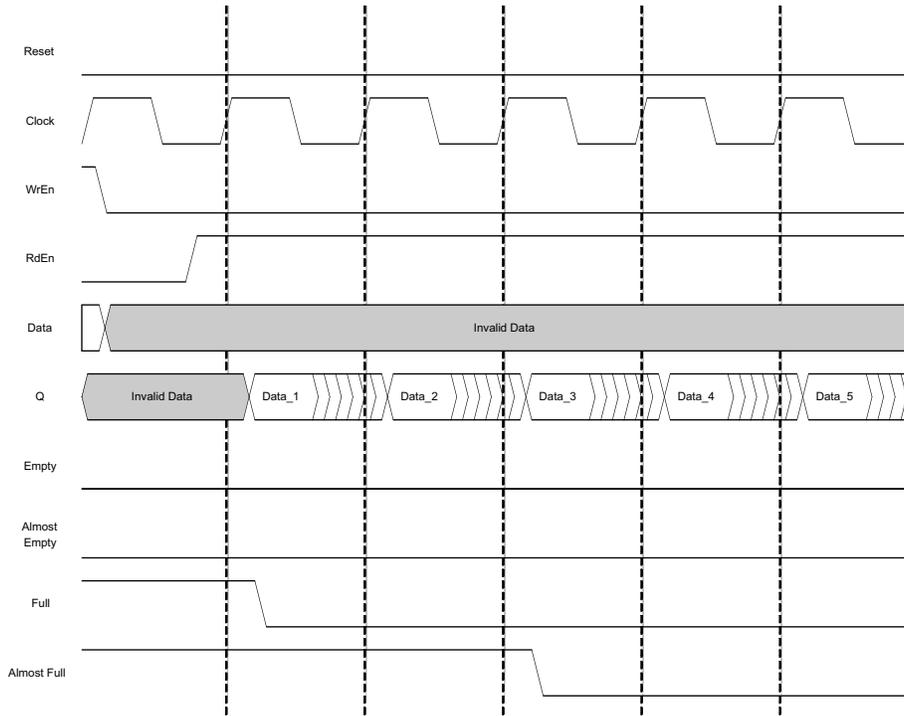
Similarly as the data is read out, and FIFO is emptied, the Almost Empty and Empty flags are asserted. Below is the

Figure 9-36. FIFO Without Output Registers, End of Data Read Cycle
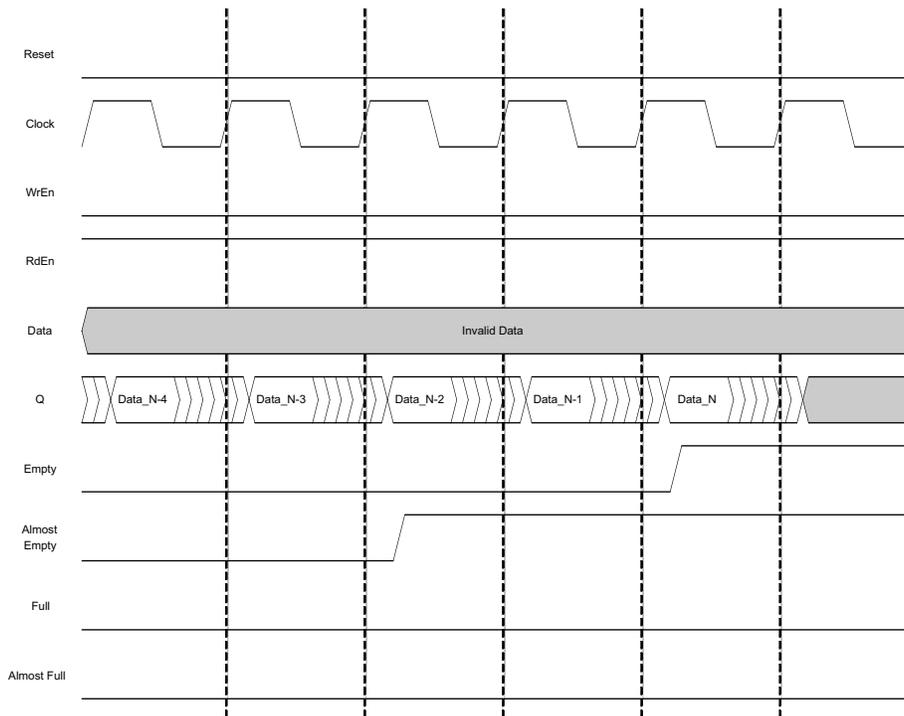
Figures 9-33 to 9-36 show the behavior of non-pipelined FIFO or FIFO without output registers. When we pipeline the registers, the output data is delayed by one clock cycle. There is an extra option of output registers being enabled by RdEn signal.

Figures 9-37 to 9-40 show the similar waveforms for the FIFO with output register and without output register enable with RdEn. It should be noted that flags are asserted and de-asserted with similar timing to the FIFO without output registers. However it is only the data out 'Q' that gets delayed by one clock cycle.

Figure 9-37. FIFO with Output Registers, Start of Data Write Cycle
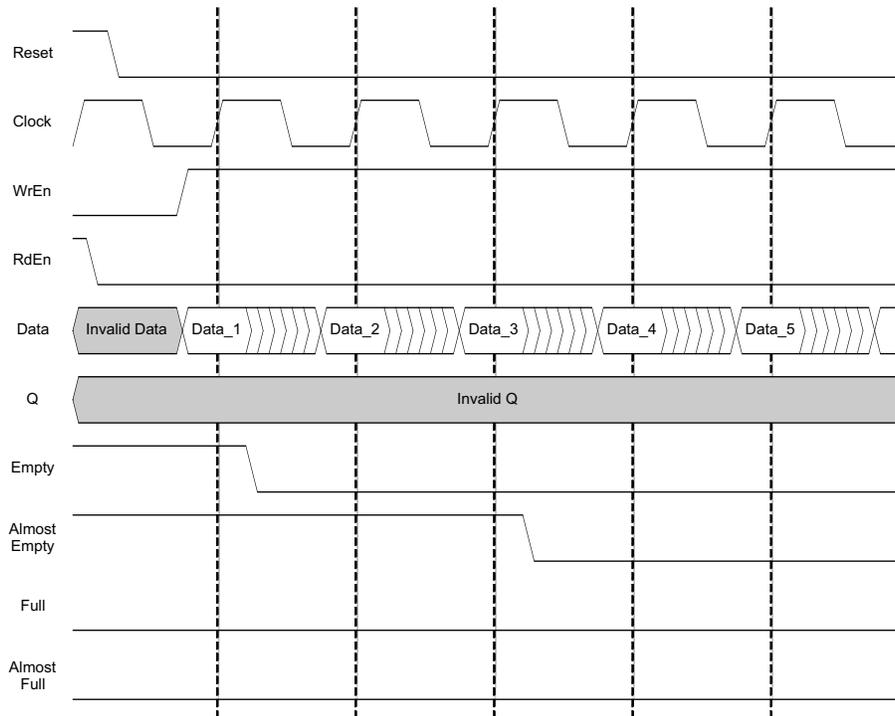
Figure 9-38. FIFO with Output Registers, End of Data Write Cycle
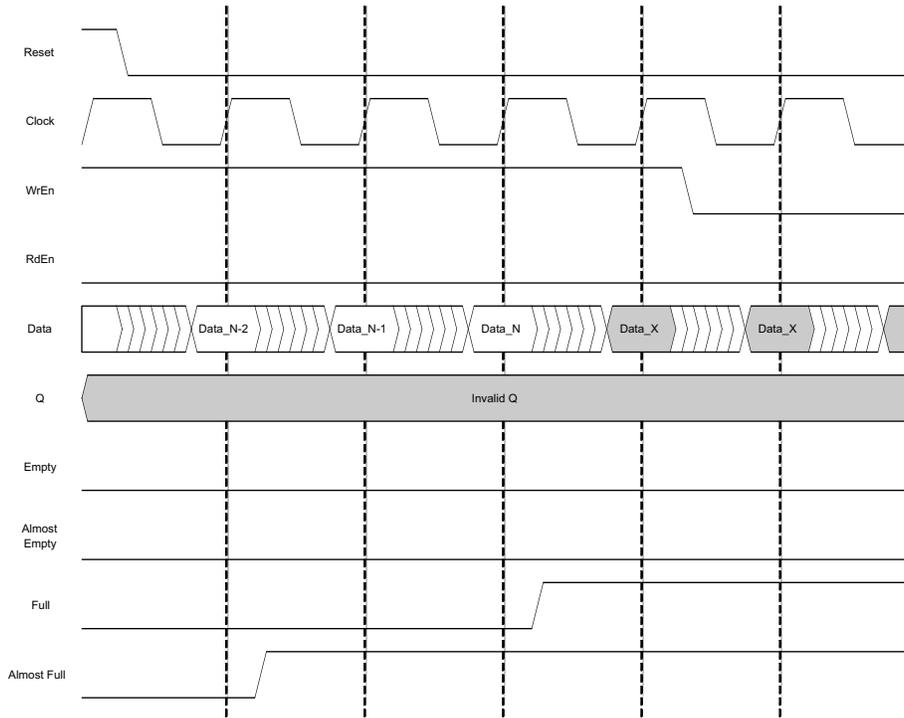
Figure 9-39. FIFO with Output Registers, Start of Data Read Cycle
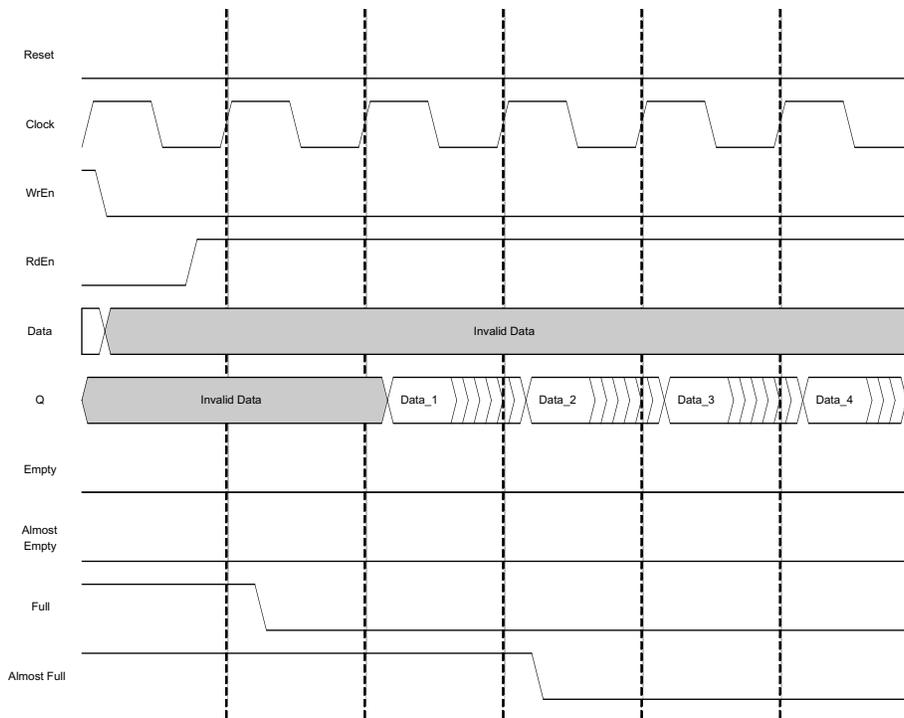
Figure 9-40. FIFO with Output Registers, End of Data Read Cycle
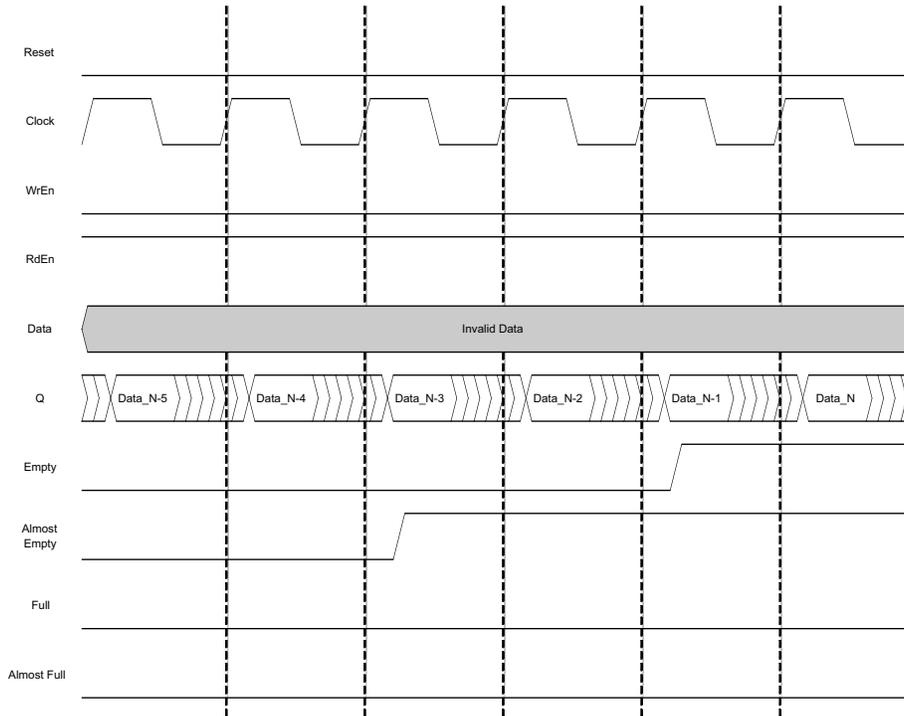
And finally, if you select the option enable output register with RdEn, it still delays the data out by one clock cycle (as compared to the non-pipelined FIFO), and the RdEn should be high also during that clock cycle, otherwise the data takes an extra clock cycle when the RdEn goes true.

Figure 9-41. FIFO with Output Registers and RdEn on Output Registers
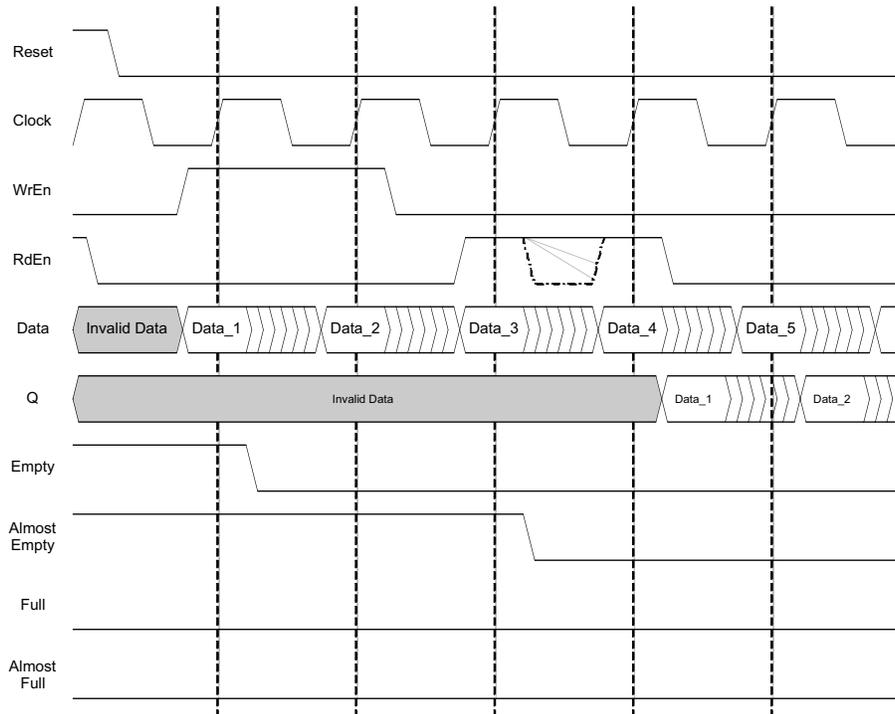
Dual Clock First In First Out (FIFO_DC) Memory: The FIFO_DC or the dual clock FIFO is also an emulated FIFO. Again the address logic and the flag logic is implemented in the FPGA fabric around the RAM.

The ports available on the FIFO_DC are:

• Reset

• RPReset

• WrClock

• RdClock

• WrEn

• RdEn

• Data

• Q

• Full Flag

• Almost Full Flag

• Empty Flag

• Almost Empty Flag

FIFO_DC Flags: FIFO_DC, as an emulated FIFO, required 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 during de-assertion. The latency can be avoided only in one of the cases (either assertion or de-assertion).

In the current emulated FIFO, there is no latency during assertion of these flags. Thus, when these flag go true, there is no latency. However this causes the latency during the 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 de-assertion of Empty and Almost Empty flags.

On the other hand, de-assertion of Full and Almost Full result in reading out the data from the FIFO_DC. It takes extra WrClock cycles after reading the data for these flags to come out.

With this in mind, let us look at the FIFO_DC without output register waveforms. Figure 9-42 shows the operation of the FIFO_DC when it is empty and the data starts to get written into it.

Figure 9-42. FIFO_DC Without Output Registers, Start of Data Write Cycle
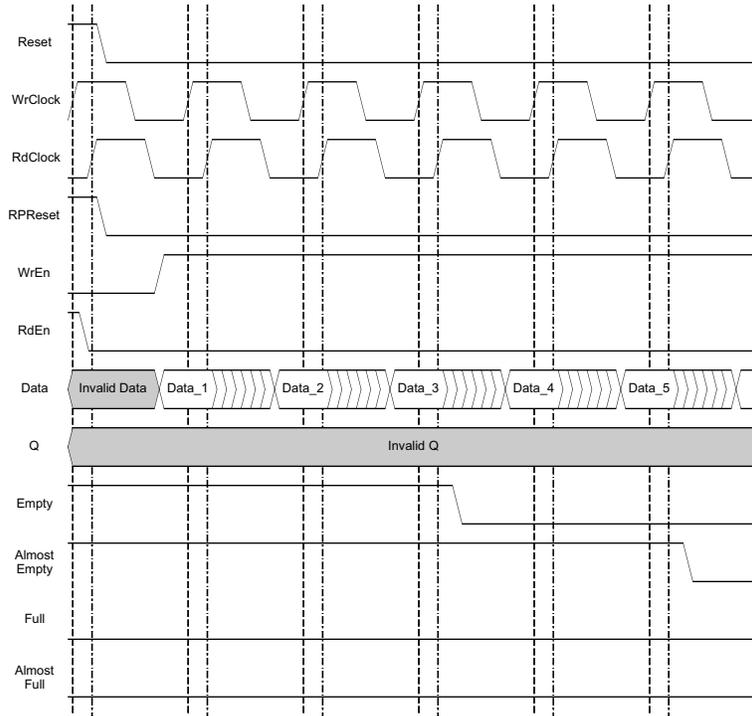
The WrEn signal has to be high to start writing into the FIFO_DC. The Empty and Almost Empty flags are high to begin and Full and Almost full are low.

When the first data gets 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 are assuming that the Almost Empty setting flag setting is 3 (address location 3). So the Almost Empty flag gets de-asserted when the third address location gets filled.

Now let is 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 9-43 shows the behavior of these flags. In this figure we assume that FIFO_DC depth is 'N'.

Figure 9-43. FIFO_DC Without Output Registers, End of Data Write Cycle
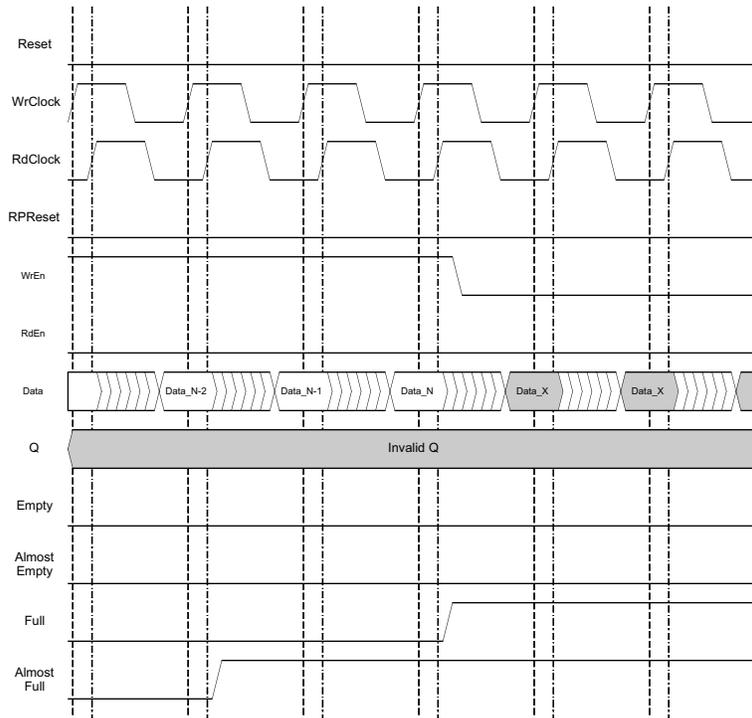
In this case, the Almost Full flag is in location 2 before the FIFO_DC is filled. The Almost Full flag is asserted when N-2 location is written, and Full flag is asserted when the last word is written into the FIFO_DC.

Data_X data inputs do not get written as 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 9-44 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags get de-asserted as shown. In this case, note that the de-assertion is delayed by two clock cycles.

Figure 9-44. FIFO_DC Without Output Registers, Start of Data Read Cycle
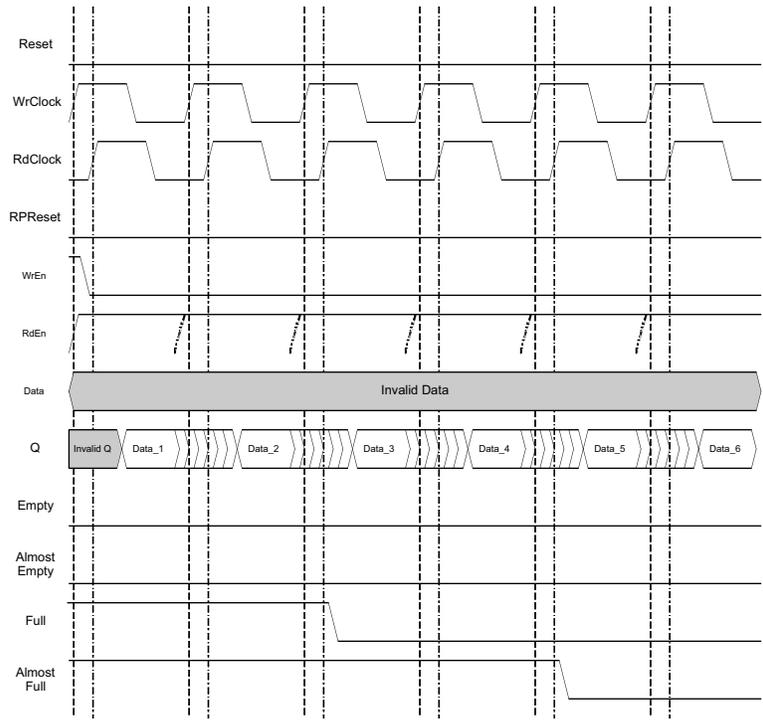
Similarly, as the data is read out and FIFO_DC is emptied, the Almost Empty and Empty flags are asserted. Below is the

Figure 9-45. FIFO_DC Without Output Registers, End of Data Read Cycle
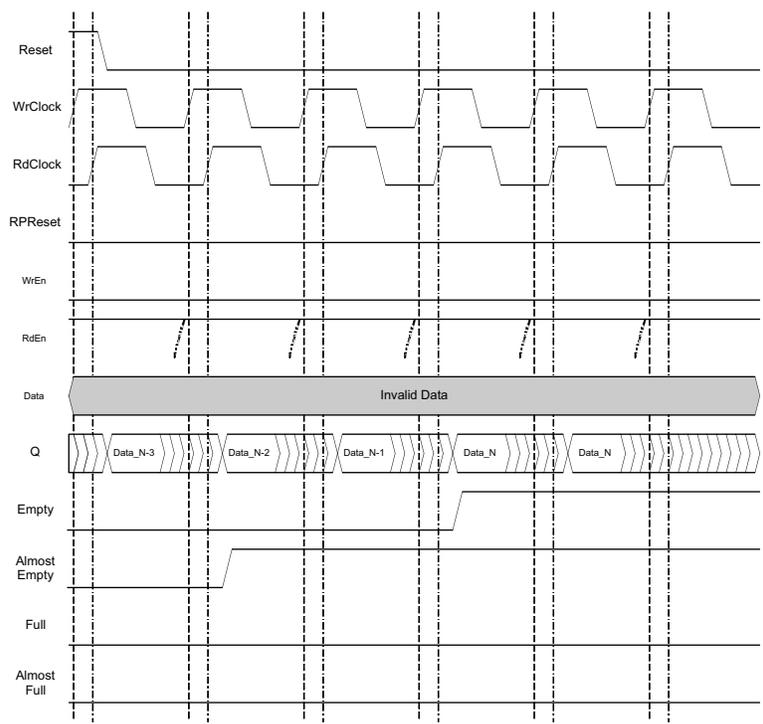
Figures 9-42 to 9-45 show the behavior of non-pipelined FIFO_DC or FIFO_DC without output registers. When we pipeline the registers, the output data is delayed by one clock cycle. There is an extra option for output registers to be enabled by the RdEn signal.

Figures 9-46 to 9-49 show similar waveforms for the FIFO_DC with output register and without output register enable with RdEn. It should be noted that flags are asserted and de-asserted with similar timing to the FIFO_DC without output registers. However it is only the data out 'Q' that is delayed by one clock cycle.

Figure 9-46. FIFO_DC With Output Registers, Start of Data Write Cycle
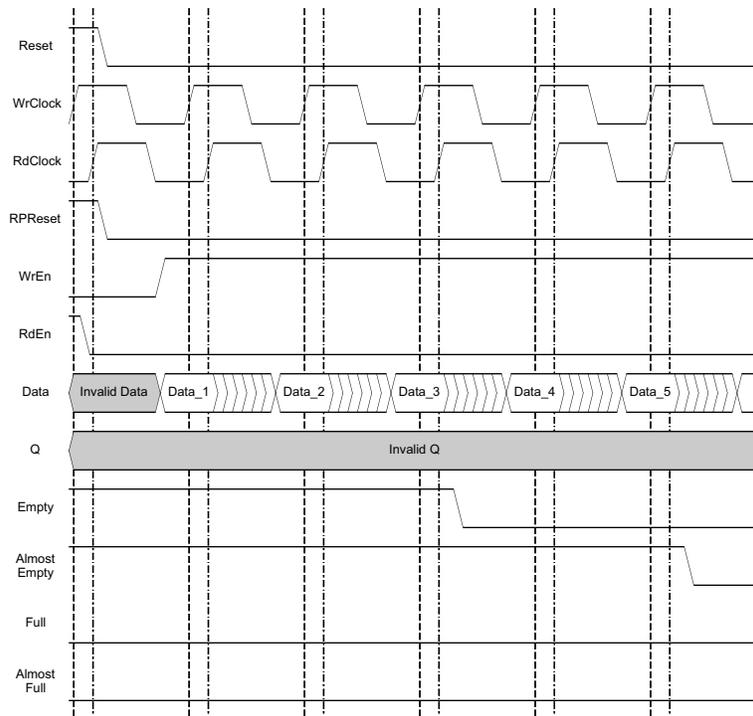
Figure 9-47. FIFO_DC With Output Registers, End of Data Write Cycle
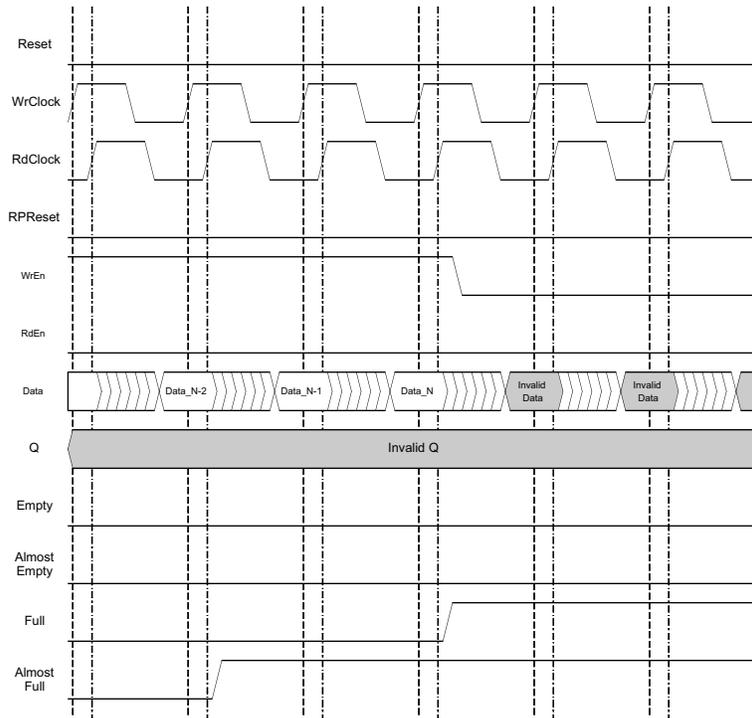
Figure 9-48. FIFO_DC With Output Registers, Start of Data Read Cycle
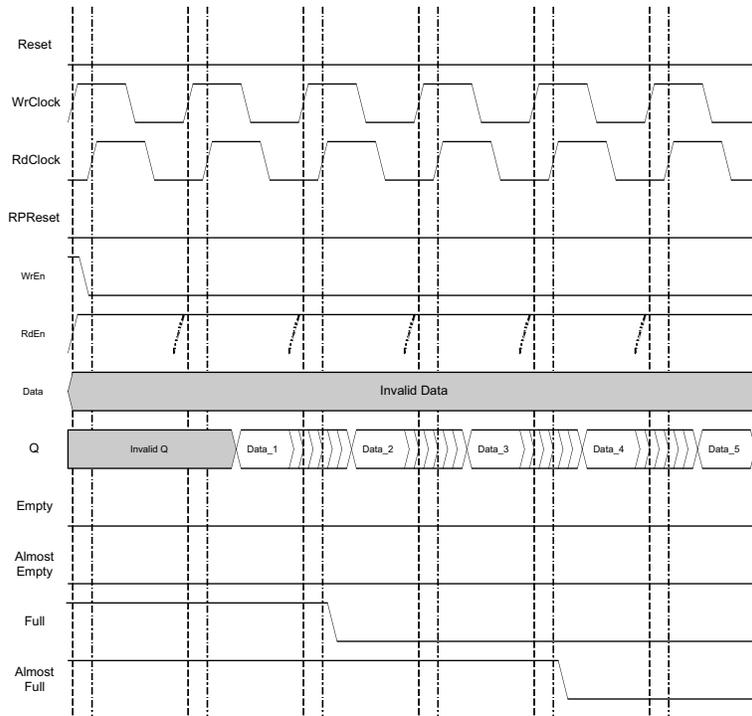
Figure 9-49. FIFO_DC With Output Registers, End of Data Read Cycle
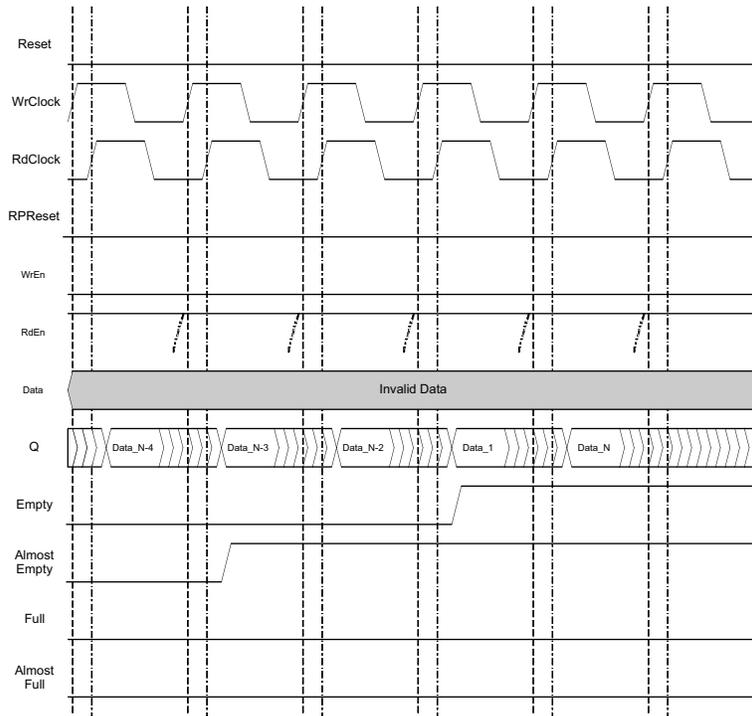
Finally, if you select the option enable output register with RdEn, it still delays the data out by one clock cycle (as compared to the non-pipelined FIFO_DC), and the RdEn should be high also during that clock cycle. Otherwise the data takes an extra clock cycle when the RdEn is goes true.

Figure 9-50. FIFO_DC With Output Registers and RdEn on Output Registers
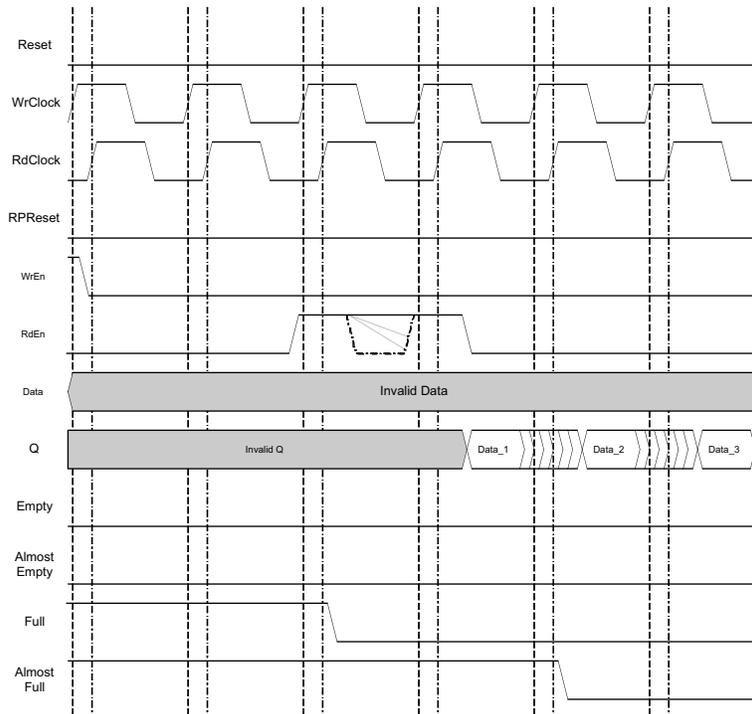
Distributed Single Port RAM (Distributed_SPRAM) – PFU Based

PFU-based Distributed Single Port RAM is created using the 4-input LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes. The memory's address and output registers are optional.

Figure 9-51 shows the Distributed Single Port RAM module as generated by the IPexpress.

Figure 9-51. Distributed Single Port RAM Module Generated by IPexpress
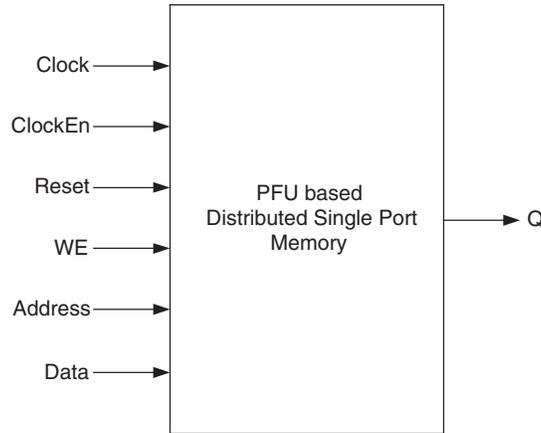
The generated module makes use of the 4-input LUT available in the PFU. Additional logic like Clock, ClockEn and Reset is generated by utilizing the resources available in the PFU. The basic Distributed Single Port RAM primitive for the LatticeECP/EC and LatticeXP devices is shown in Figure 9-52.

Figure 9-52. Distributed Single Port RAM (Distributed_SPRAM) for LatticeECP/EC and LatticeXP Devices
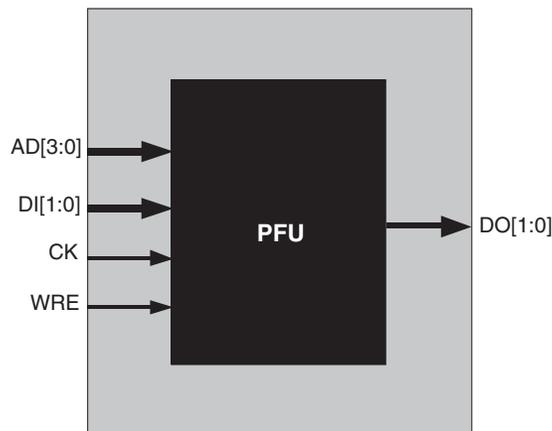
Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants to enable the output registers in the IPexpress configuration.

The various ports and their definitions for the memory are included in Table 9-14. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 9-14. PFU based Distributed Single port RAM Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
Clock	CK	Clock	Rising Clock Edge
ClockEn	-	Clock Enable	Active High
Reset	-	Reset	Active High
WE	WRE	Write Enable	Active High
Address	AD[3:0]	Address	—
Data	DI[1:0]	Data In	—
Q	DO[1:0]	Data Out	—

Users have an option of enabling the output registers for Distributed Single Port RAM (Distributed_SPRAM). Figures 8-35 and 8-36 show the internal timing waveforms for the Distributed Single Port RAM (Distributed_SPRAM) with these options.

Figure 9-53. PFU Based Distributed Single Port RAM Timing Waveform - Without Output Registers

Figure 9-54. PFU Based Distributed Single Port RAM Timing Waveform – with Output Registers
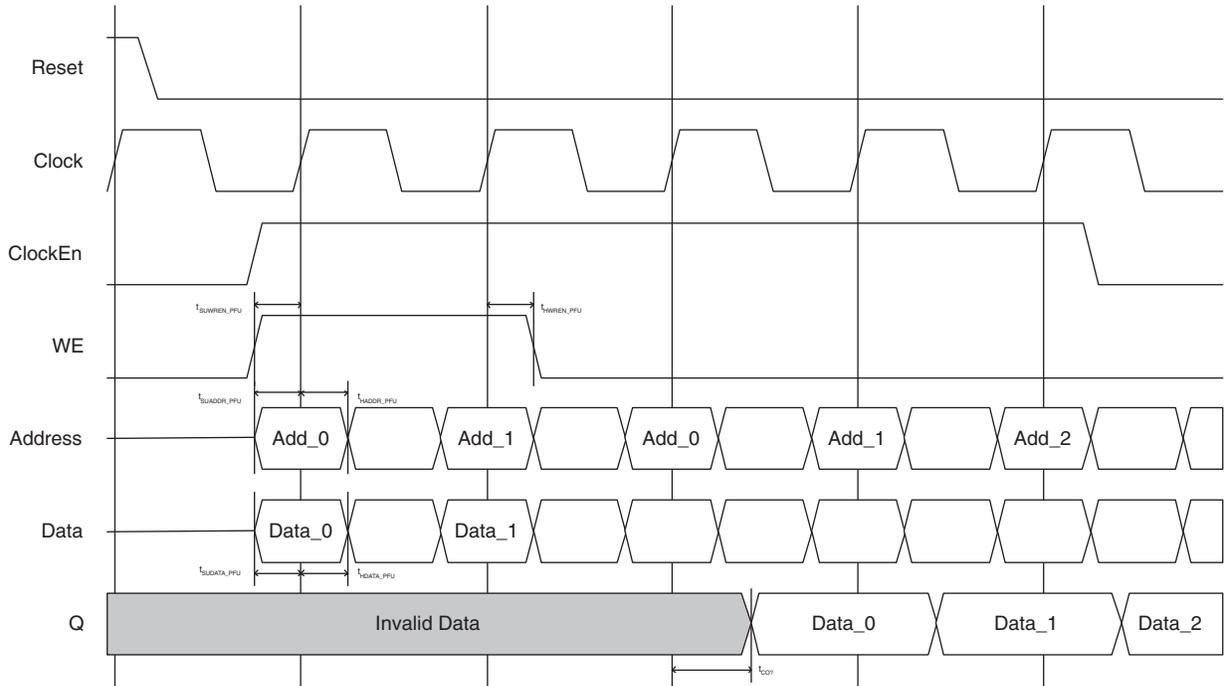
Distributed Dual Port RAM (Distributed_DPRAM) – PFU Based

PFU-based Distributed Dual Port RAM is also created using the four input LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 9-55 shows the Distributed Single Port RAM module as generated by IPexpress.

Figure 9-55. Distributed Dual Port RAM Module Generated by IPexpress
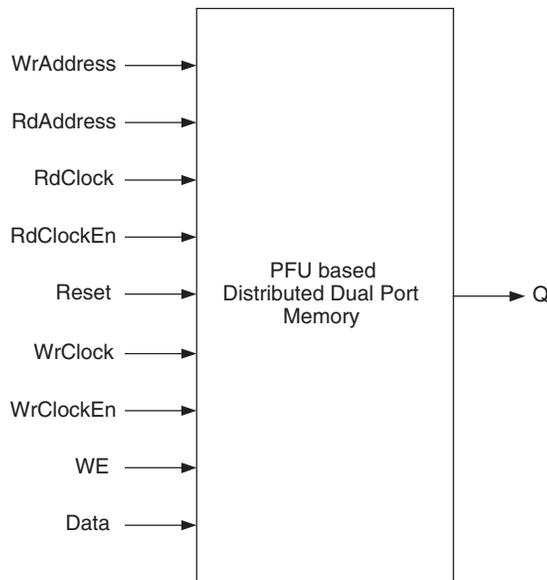
The generated module makes use of a 4-input LUT available in the PFU. Additional logic for Clocks, Clock Enables and Reset is generated by utilizing the resources available in the PFU. The basic Distributed Dual Port RAM primitive for the LatticeECP/EC and LatticeXP devices is shown in Figure 9-56.

Figure 9-56. PFU-based Distributed Dual Port RAM for LatticeECP/EC and LatticeXP Devices
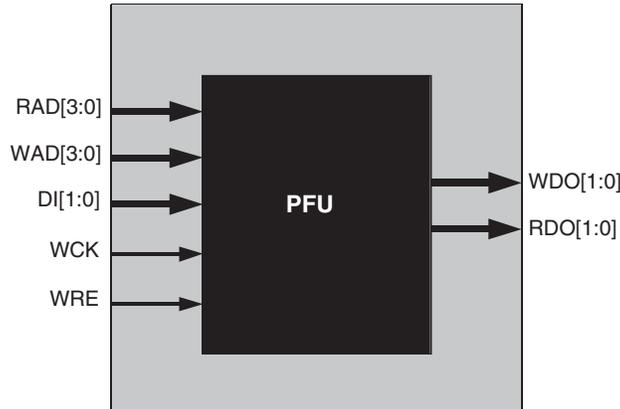
Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants the to enable the output registers in the IPexpress configuration.

The various ports and their definitions for the memory are included in Table 9-15. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 9-15. PFU-based Distributed Dual-Port RAM Port Definitions

Port Name in Generated Module	Port Name in EBR Block Primitive	Description	Active State
WrAddress	WAD[23:0]	Write Address	—
RdAddress	RAD[3:0]	Read Address	—
RdClock	—	Read Clock	Rising Clock Edge
RdClockEn	—	Read Clock Enable	Active High
WrClock	WCK	Write Clock	Rising Clock Edge
WrClockEn	—	Write Clock Enable	Active High
WE	WRE	Write Enable	Active High
Data	DI[1:0]	Data Input	—
Q	RDO[1:0]	Data Out	—

Users have the option of enabling the output registers for Distributed Dual Port RAM (Distributed_DPRAM). Figures 8-39 and 8-40 show the internal timing waveforms for the Distributed Dual Port RAM (Distributed_DPRAM) with these options.

Figure 9-57. PFU Based Distributed Dual Port RAM Timing Waveform - Without Output Registers (Non-Pipelined)

Figure 9-58. PFU Based Distributed Dual Port RAM Timing Waveform – With Output Registers

Distributed ROM (Distributed_ROM) – PFU Based

PFU-based Distributed ROM is also created using the 4-input LUT (Look-Up Table) available in the PFU. These LUTs can be cascaded to create larger distributed memory sizes.

Figure 9-59 shows the Distributed Single Port RAM module as generated by IPexpress.

Figure 9-59. Distributed ROM Generated by IPexpress

The generated module makes use of the 4-input LUT available in the PFU. The basic Distributed Dual Port RAM primitive for the LatticeECP/EC and LatticeXP devices is shown in Figure 9-60.

Figure 9-60. PFU-based Distributed ROM (Sync_ROM) for LatticeECP/EC and LatticeXP Devices
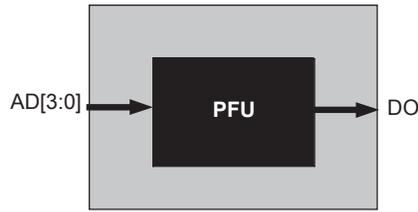
Ports such as Out Clock (OutClock) and Out Clock Enable (OutClockEn) are not available in the hardware primitive. These are generated by IPexpress when the user wants the to enable the output registers in the IPexpress configuration.

The various ports and their definitions for the memory are included in Table 9-16. The table lists the corresponding ports for the module generated by IPexpress and for the primitive.

Table 9-16. PFU-based Distributed ROM Port Definitions

Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
Address	AD[3:0]	Address	—
OutClock	—	Out Clock	Rising Clock Edge
OutClockEn	—	Out Clock Enable	Active High
Reset	—	Reset	Active High
Q	DO	Data Out	—

Users have the option of enabling the output registers for Distributed ROM (Distributed_ROM). Figures 8-43 and 8-44 show the internal timing waveforms for the Distributed ROM with these options.

Figure 9-61. PFU Based ROM Timing Waveform – without Output Registers
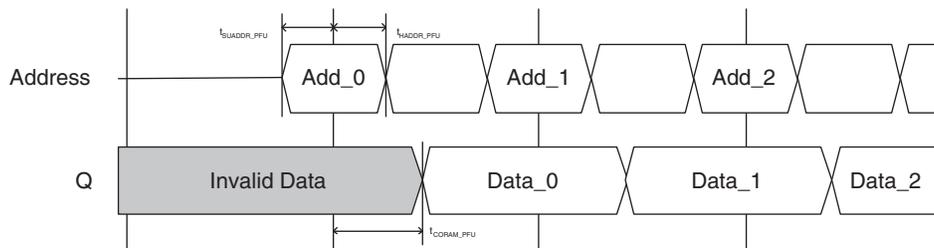
Figure 9-62. PFU Based ROM Timing Waveform – with Output Registers
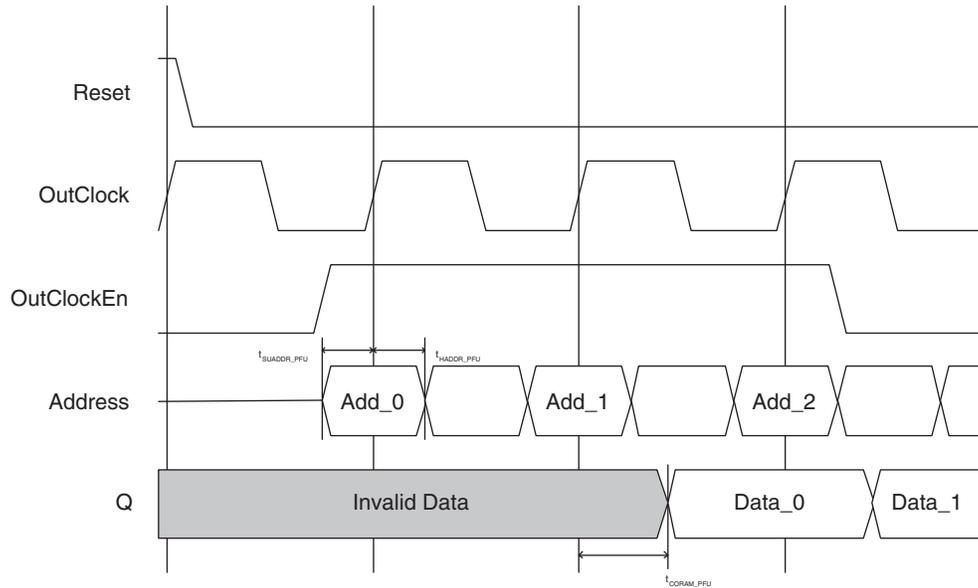
Initializing Memory

In the EBR based ROM or RAM memory modes and the PFU based ROM memory mode, it is possible to specify the power-on state of each bit in the memory array. Each bit in the memory array can have one of two values: 0 or 1.

Initialization File Format

The initialization file is an ASCII file, which a user can create or edit using any ASCII editor. IPexpress supports three types of 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 depicts the value to be stored in a particular memory location. The number of characters (or the number of columns) represents the number of bits for each address (or the width of the memory module).

The Initialization File is primarily used for configuring the ROMs. The EBR in RAM mode can optionally use this Initialization File also to preload the memory contents.

Binary File

The file is essentially a text file of 0's and 1's. The rows indicate the number of words and columns indicate the width of the memory.

```
Memory Size 20x32
00100000010000000010000001000000
00000001000000010000000100000001
00000010000000100000001000000010
00000011000000110000001100000011
00000100000001000000010000000100
00000101000001010000010100000101
00000110000001100000011000000110
00000111000001110000011100000111
00001000010010000000100001001000
00001001010010010000100101001001
00001010010010100000101001001010
00001011010010110000101101001011
00001100000011000000110000001100
00001101001011010000110100101101
00001110001111100000111000111110
00001111001111110000111100111111
00010000000100000001000000010000
00010001000100010001000100010001
00010010000100100001001000010010
00010011000100110001001100010011
```

Hex File

The Hex file is essentially a text file of Hex characters arranged in a similar row-column arrangement. The number of rows in the file is same as the number of address locations, with each row indicating the content of the memory location.

```
Memory Size 8x16
A001
0B03
1004
CE06
0007
040A
0017
02A4
```

Addressed Hex (ORCA)

Addressed Hex consists of lines of address and data. Each line starts with an address, followed by a colon, and any number of data. The format of memfile is address: data data data data ... where address and data are hexadecimal numbers.

```
-A0 : 03 F3 3E 4F
-B2 : 3B 9F
```

The first line puts 03 at address A0, F3 at address A1, 3E at address A2,and 4F at address A3. The second line puts 3B at address B2 and 9F at address B3.

There is no limitation on the values of address and data. The value range is automatically checked based on the values of addr_width and data_width. If there is an error in an address or data value, an error message is printed. Users need not specify data at all address locations. If data is not specified at a certain address, the data at that

location is initialized to 0. IPexpress makes memory initialization possible both through the synthesis and simulation flows.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

+1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
June 2004	01.0	Initial release.
July 2004	01.1	Minor updates only.
October 2004	01.2	Minor updates only.
February 2005	01.3	Replace LatticeEC™ and LatticeECP™ LatticeEC™, LatticeECP™ and LatticeXP™
		Replace LatticeECP/ECLatticeEC/ECP and LatticeXP
		Replace LatticeEC or LatticeECP LatticeEC, LatticeECP or LatticeXP
		Added Hardware related information for the LatticeEC/ECP/XP devices
		Update Figure 8-4
		Figure 8-40 Replace "_EBR" with "_PFU" in the figure's timing parameters.
April 2005	01.4	Updated the Trual Dual Port RAM and Module Manager Flow sections.
October 2005	01.5	Updated the block diagrams of modules generated by the Module Manager. Added section for Module Manager flow example. Added section for PMI flow.
February 2006	01.6	Removed the PMI support section
April 2006	01.7	Updated the Initializing Memory section
October 2006	01.8	Updated the FIFO section. Added dual port memory access notes in Appendix A.

Appendix A. Attribute Definitions

DATA_WIDTH

Data width is associated with the RAM and FIFO elements. The DATA_WIDTH attribute will define the number of bits in each word. It takes the values as defined in the RAM size tables in each memory module.

REGMODE

REGMODE or the Register mode attribute is used to enable pipelining in the memory. This attribute is associated with the RAM and FIFO elements. The REGMODE attribute takes the NOREG or OUTREG mode parameter that disables and enables the output pipeline registers.

RESETMODE

The RESETMODE attribute allows users to select the mode of reset in the memory. This attribute is associated with the block RAM elements. RESETMODE takes two parameters: SYNC and ASYNC. SYNC means that the memory reset is synchronized with the clock. ASYNC means that the memory reset is asynchronous to clock.

CSDECODE

CSDECODE or the Chip select decode attributes are associated to block RAM elements. CS, or Chip Select, is the port available in the EBR primitive that is useful when memory requires multiple EBR blocks cascaded. The CS signal would form the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade 8 memories easily. CSDECODE takes the following parameters: "000", "001", "010", "011", "100", "101", "110", and "111". CSDECODE values determine the decoding value of CS[2:0]. CSDECODE_W is chip select decode for write and CSDECODE_R is chip select decode for read for Pseudo Dual Port RAM. CSDECODE_A and CSDECODE_B are used for true dual port RAM elements and refer to the A and B ports.

WRITEMODE

The WRITEMODE attribute is associated with the block RAM elements. It takes the NORMAL, WRITETHROUGH, and READBEFOREWRITE mode parameters.

In NORMAL mode, the output data does not change or get updated, during the write operation. This mode is supported for all data widths.

In WRITETHROUGH mode, the output data is updated with the input data during the write cycle. This mode is supported for all data widths.

In READBEFOREWRITE mode, the output data port is updated with the existing data stored in the write address, during a write cycle. This mode is supported for x9, x18 and x36 data widths.

WRITEMODE_A and WRITEMODE_B are used for dual port RAM elements and refer to the A and B ports in case of a True Dual Port RAM.

For all modes (of the True Dual Port module), simultaneous read access from one port and write access from the other port to the same memory address is not recommended. The read data may be unknown in this situation. Also, simultaneous write access to the same address from both ports is not recommended. (When this occurs, the data stored in the address becomes undetermined when one port tries to write a 'H' and the other tries to write a 'L'.)

It is recommended that the designer implements control logic to identify this situation if it occurs and either:

1. Implement status signals to flag the read data as possibly invalid, or

2. Implement control logic to prevent the simultaneous access from both ports.

GSR

GSR or Global Set/ Reset attribute is used to enable or disable the global set/reset for RAM element.

LatticeECP/EC and LatticeXP DDR Usage Guide

Introduction

LatticeECP™, LatticeEC™ and LatticeXP™ devices support various Double Data Rate (DDR) and Single Data Rate (SDR) interfaces using the logic built into the Programmable I/O (PIO). SDR applications capture data on one edge of a clock while the DDR interfaces capture data on both the rising and falling edges of the clock, thus doubling the performance. This document will address in detail how to utilize the capabilities of the LatticeECP/EC and LatticeXP devices to implement both generic DDR and DDR memory interfaces.

DDR SDRAM Interfaces Overview

DDR SDRAM interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. When reading data from the external memory device, data coming into the device is edge aligned with respect to the DQS signal. This DQS strobe signal needs to be phase shifted 90 degrees before FPGA logic can sample the read data. When writing to a DDR SDRAM the memory controller (FPGA) must shift the DQS by 90 degrees to center align with the data signals (DQ). DQ and DQS are bi-directional ports. The same two signals are used for both write and read operations. A clock signal is also provided to the memory. This clock is provided as a differential clock (CLKP and CLKN) to minimize duty cycle variations. The memory also uses these clock signals to generate the DQS signal during a read via a DLL inside the memory. The skew between CLKP or CLKN and the SDRAM-generated DQS signal is specified in the DDR SDRAM data sheet. Figures 10-1 and 10-2 show DQ and DQS relationships for read and write cycles.

During read, the DQS signal is LOW for some duration after it comes out of tristate. This state is called Preamble. The state when the DQS is LOW before it goes into Tristate is the Postamble state. This is the state after the last valid data transition.

DDR SDRAM also require a Data Mask (DM) signals to mask data bits during write cycles. SDRAM interfaces typically are implemented with x8, x16 and x32 bits for each DQS signal. Note that the ratio of DQS to data bits is independent of the overall width of the memory. An 8-bit interface will have one strobe signal.

Figure 10-1. Typical DDR Interface
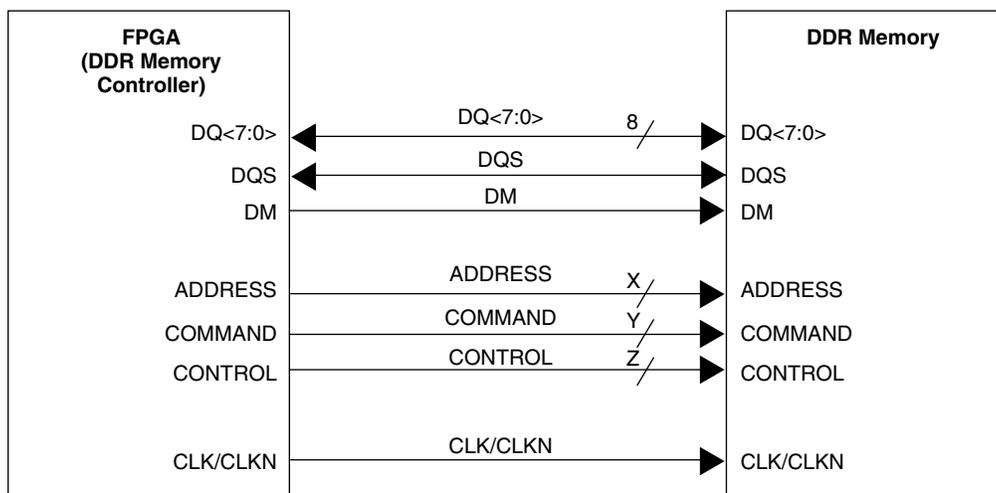
Figure 10-2. DQ-DQS During READ
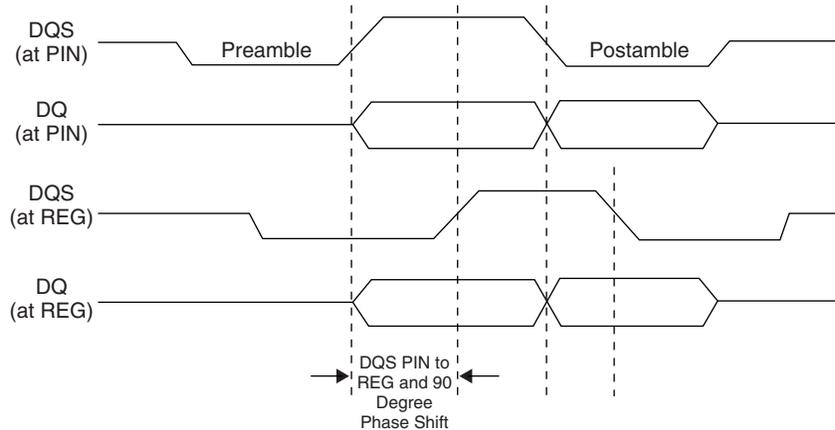
Figure 10-3. DQ-DQS During WRITE
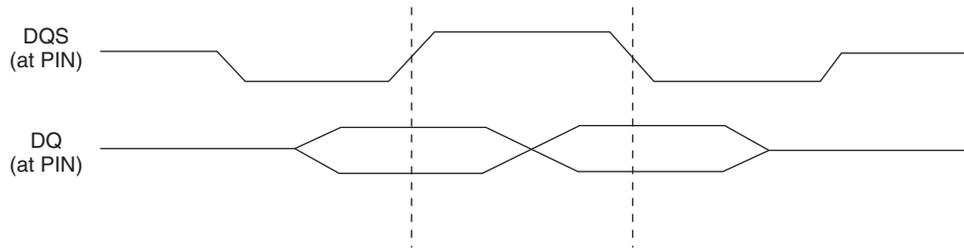
Implementing DDR Memory Interfaces with the LatticeECP/EC Devices

This section describes how to implement the read and write sections of a DDR memory interface. It also provides details of the DQ and DQS grouping rules associated with the LatticeECP/EC and LatticeXP devices.

DQS Grouping

Each DQS group generally consists of at least 10 I/Os (1DQS, 8DQ and 1DM) to implement a complete 8-bit DDR memory interface. In the LatticeECP/EC devices each DQS signal will span across 16 I/Os and in the LatticeXP devices the DQS will span 14 I/Os. Any 10 of these 16 I/Os can be used to implement an 8-bit DDR memory interface. In addition to the DQS grouping, the user must also assign one reference voltage VREF1 for a given I/O bank.

The tables below show the total number of DQS groups available per I/O bank for each device and package.

Table 10-1. Number of DQS Banks in the LatticeECP/EC Device

Package	Device	Total x8 DQS Groups	Number of DQS Groups per I/O Bank							
			0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
100-pin TQFP	LFEC1	3	1	1	0	0	1	0	0	0
	LFEC3	3	1	1	0	0	1	0	0	0
144-pin TQFP	LFEC1	6	1	1	0	1	1	1	1	0
	LFEC3	6	1	1	0	1	1	1	1	0
	LFEC6/LFECP6	6	1	1	0	1	1	1	1	0
208-pin TQFP	LFEC1	6	1	1	0	1	1	1	1	0
	LFEC3	8+2[1]	1+1[1]	1	1	1	1	1+1[1]	1	1
	LFEC6/LFECP6	8+2[1]	1+1[1]	1	1	1	1	1+1[1]	1	1
	LFEC10/LFECP10	8+2[1]	1+1[1]	1	1	1	1	1+1[1]	1	1
256-ball fpBGA	LFEC3	10	2	1	1	1	1	2	1	1
	LFEC6/LFECP6	12	2	1	1	2	1	2	2	1
	LFEC10/LFECP10	12	2	1	1	2	1	2	2	1
	LFEC15/LFECP15	12	2	1	1	2	1	2	2	1
484-ball fpBGA	LFEC6/LFECP6	14	2	2	1	2	2	2	2	1
	LFEC10/LFECP10	18	3	2	2	2	2	3	2	2
	LFEC15/LFECP15	20	3	3	2	2	3	3	2	2
	LFEC20/LFECP20	22	3	3	2	3	3	3	3	2
	LFEC33/LFECP33	22	3	3	2	3	3	3	3	2
672-ball fpBGA	LFEC20/LFECP20	24	4	3	2	3	3	4	3	2
	LFEC33/LFECP33	30	4	4	3	4	4	4	4	3

1. 10 I/Os (1 DQS + 8 DQs + Bank VREF1) can function as a DDR interface in which the FPGA can have a DM output but not a DQS aligned input (in the same DDR bank as the rest of the system).

Table 10-2. Number of DQS Banks in the LatticeXP Device

Package	Device	Total x8 DQS Groups	Number of DQS Groups per I/O Bank							
			0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
100-pin TQFP	LFXP3C/LFXP3E	2	0	0	0	0	0	1	0	1
	LFXP6C LFXP6E	2	0	0	0	0	0	1	0	1
144-pin TQFP	LFXP3C/LFXP3E	7	1	0	1	1	1	1	1	1
	LFXP6C/LFXP6E	7	1	0	1	1	1	1	1	1
208-pin PQFP	LFXP3C/LFXP3E	8	1	1	1	1	1	1	1	1
	LFXP6C/LFXP6E	8	1	1	1	1	1	1	1	1
256-ball fpBGA	LFXP3C/LFXP3E	12	2	2	1	1	2	2	1	1
	LFXP6C/LFXP6E	12	2	2	1	1	2	2	1	1
	LFXP10C/LFXP10E	16	2	2	2	2	2	2	2	2
	LFXP15C/LFXP15E	16	2	2	2	2	2	2	2	2
	LFXP20C/LFXP20E	20	3	3	2	2	3	3	2	2
388-ball fpBGA	LFXP10C/LFXP10E	16	2	2	2	2	2	2	2	2
	LFXP15C/LFXP15E	20	3	3	2	2	3	3	2	2
	LFXP20C/LFXP20E	20	3	3	2	2	3	3	2	2
484-ball fpBGA	LFXP15C/LFXP15E	20	3	3	2	2	3	3	2	2
	LFXP20C/LFXP20E	20	3	3	2	2	3	3	2	2
672-ball fpBGA	LFXP20C/LFXP20E	24	4	4	2	2	4	4	2	2

Figure 10-4. DQ-DQS Grouping

n* I/O PADS

DQ, DM or VREF1

DQS PAD
(Ninth I/O Pad)
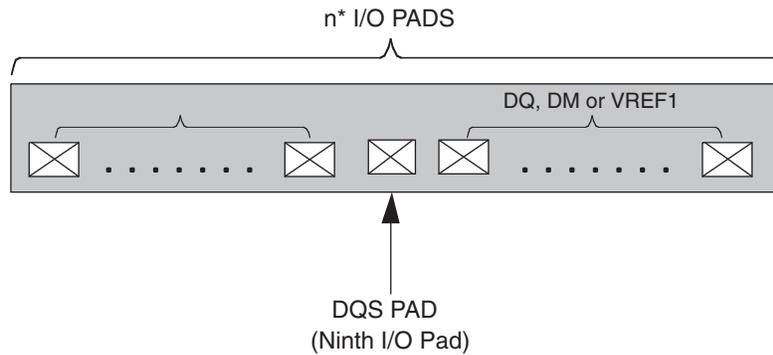
*For LatticeECP/EC: n = 16, for LatticeXP: n = 14.

Figure 10-4 shows a typical DQ-DQS group for both the LatticeECP/EC device and the LatticeXP device. The ninth I/O of this group of 16 I/Os (for LatticeECP/EC) or 14 I/Os (for LatticeXP) is the dedicated DQS pin. All eight pads before the DQS and seven (for LatticeECP/EC) or four (for LatticeXP) pads after the DQS are covered by this DQS bus span. The user can assign any eight of these I/O pads to be DQ data pins. Hence, to implement a 32-bit wide memory interface you would need to use four such DQ-DQS groups.

When not interfacing with the memory, the dedicated DQS pin can be used as a general purpose I/O. Each of the dedicated DQS pin is internally connected to the DQS phase shift circuitry. The pinout information contained in the LatticeECP/EC and LatticeXP device data sheets shows pin locations for the DQS pads. Table 10-2 shows an extract from the LatticeECP/EC data sheet. In this case, the DQS is marked as LDQS6 (L=left side, 6 =associated PFU row/column). Since DQS is always the fifth true pad in the DQ-DQS group, counting from low to high PFU row/column number, LDQS6 will cover PL2A to PL9B. Following this convention, there are eight pads before and seven pads after DQS for DQ available following counter-clockwise for the left and bottom sides of the device, and following clockwise for the top and right sides of the device. The user can assign any eight of these pads to be DQ data signals. The LatticeXP device follows the same method.

Table 10-3. EC20 Pinout (from LatticeECP/EC Family Data Sheet)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PL2A	7	T	VREF2_7	D4	E3
PL2B	7	C	VREF1_7	E4	E4
PL3A	7	T	—	C3	B1
PL3B	7	C	—	B2	C1
PL4A	7	T	—	E5	F3
PL4B	7	C	—	F5	G3
PL5A	7	T	—	D3	D2
PL5B	7	C	—	C2	E2
PL6A	7	T	LDQS6	F4	D1
PL6B	7	C	—	G4	E1
PL7A	7	T	—	E3	F2
PL7B	7	C	—	D2	G2
PL8A	7	T	LUM0_PLLT_IN_A	B1	F6
PL8B	7	C	LUM0_PLLC_IN_A	C1	G6
PL9A	7	T	LUM0_PLLT_FB_A	F3	H4
PL9B	7	C	LUM0_PLLC_FB_A	E2	G4
PL11A	7	T	—	G5	J4

Table 10-3. EC20 Pinout (from LatticeECP/EC Family Data Sheet)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PL11B	7	C	—	H6	J5
PL12A	7	T	—	G3	K4

DDR Software Primitives

This section describes the software primitives that can be used to implement DDR interfaces and provides details about how to instantiate them in the software. The primitives described include:

- DQSDLL   The DQS delay calibration DLL
- DQSBUF   The DQS delay function and the clock polarity selection logic
- INDDRXB   The DDR input and DQS to system clock transfer registers
- ODDRXB   The DDR output registers

An HDL usage example for each of these primitives is listed in Appendices B and C.

DQSDLL
The DQSDLL will generate a 90-degree phase shift required for the DQS signal. This primitive will implement the on-chip DQSDLL. Only one DQSDLL should be instantiated for all the DDR implementations on one half of the device. The clock input to this DLL should be at the same frequency as the DDR interface. The DLL will generate the delay based on this clock frequency and the update control input to this block. The DLL will update the dynamic delay control to the DQS delay block when this update control (UDDCNTL) input is asserted. Figure 10-5 shows the primitive symbol. The active low signal on UDDCNTL updates the DQS phase alignment and should be initiated at the beginning of READ cycles.

Figure 10-5. DQSDLL Symbol
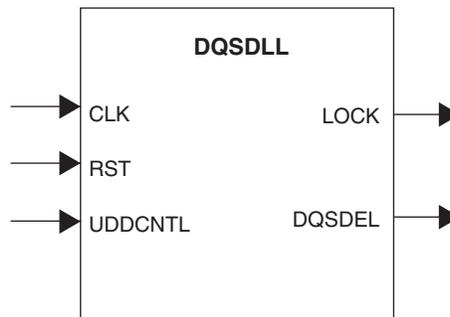
Table 10-4 provides a description of the ports.

Table 10-4. DQSDLL Ports

Port Name	I/O	Definition
CLK	I	System CLK should be at frequency of the DDR interface, from the FPGA core.
RST	I	Resets the DQSDLL
UDDCNTL	I	Provides an update signal to the DLL that will update the dynamic delay. When held low this signal will update the DQSDEL.
LOCK	O	Indicates when the DLL is in phase
DQSDEL	O	The digital delay generated by the DLL should be connected to the DQSBUF primitive.

DQSDLL Configuration Attributes
By default this DLL will generate a 90-degree phase shift for the DQS strobe based on the frequency of the input reference clock to the DLL. The user can control the sensitivity to jitter by using the LOCK_SENSITIVITY attribute. This configuration bit can be programmed to be either "HIGH" or "LOW".

The DLL Lock Detect circuit has two modes of operation controlled by the LOCK_SENSITIVITY bit, which selects more or less sensitivity to jitter. If this DLL is operated at or above 150 MHz, it is recommended that the LOCK_SENSITIVITY bit be programmed "HIGH" (more sensitive). For operation running at or under 100 MHz it is recommended that the bit be programmed "LOW" (more tolerant). For 133 MHz, the LOCK_SENSITIVITY bit can go either way.

DQSBUF

This primitive implements the DQS Delay and the DQS transition detector logic. Figure 10-6 shows the DQSBUFB function. The preamble detect signal is also generated within this primitive.

Figure 10-6. DQSBUFB Function
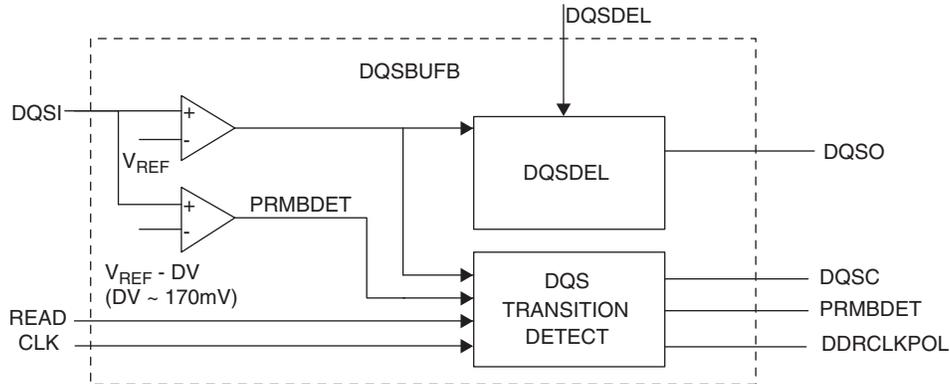
Figure 10-7 shows the primitive symbol and its ports. DQSI is the DQS signal from the memory. PRMBDET is the preamble detect signal that is generated from the DQSI input. READ and CLK are user interface signals coming from the FPGA logic. The DQSDLL block sends digital control line DQSDEL to this block. The DQS is delayed based on this input from the DQSDLL. DQSO is the delayed DQS and is connected to the clock input of the first set of DDR registers.

Figure 10-7. DQSBUFB Symbol
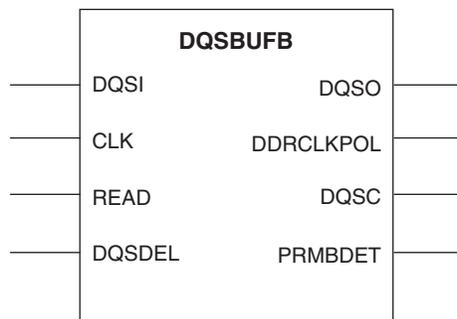
Table 10-5 provides a description of the I/O ports associated with the DQSBUFB primitive.

Table 10-5. DQSBUFB Ports

Port Name	I/O	Definition
DQSI	I	DQS strobe signal from memory
CLK	I	System CLK
READ	I	Read generated from the FPGA core
DQSDEL	I	DQS delay from the DQSDLL primitive
DQSO	O	Delayed DQS Strobe signal, to the input capture register block
DQSC	O	DQS Strobe signal before delay, going to the FPGA core logic
DDRCLKPOL	O	DDR Clock Polarity signal
PRMBDET	O	Preamble detect signal, going to the FPGA core logic

Notes:

1. The DDR Clock Polarity output from this block should be connected to the DDCLKPOL inputs of the input register blocks (IDDRXB).

READ Pulse Generation

The READ signal to the DQSBUFB block is internally generated in the FPGA core. The Read signal will go high when the READ command to control the DDR SDRAM is initially asserted. This should normally precede the DQS preamble by one cycle yet may overlap the trailing bits of a prior read cycle. The DQS Detect circuitry of the LatticeECP/EC and LatticeXP devices require the falling edge of the READ signal to be placed within the preamble stage.

The preamble state of the DQS can be detected using the CAS latency and the round trip delay for the signals between the FPGA and the memory device. Note that the internal FPGA core generates the READ pulse. The rise of the READ pulse needs to coincide with the initial READ Command of the Read Burst and needs to go low before the Preamble goes high.

Figure 10-8 shows the READ Pulse Timing Example with respect to the PRMBDET signal.

Figure 10-8. READ Pulse Generation
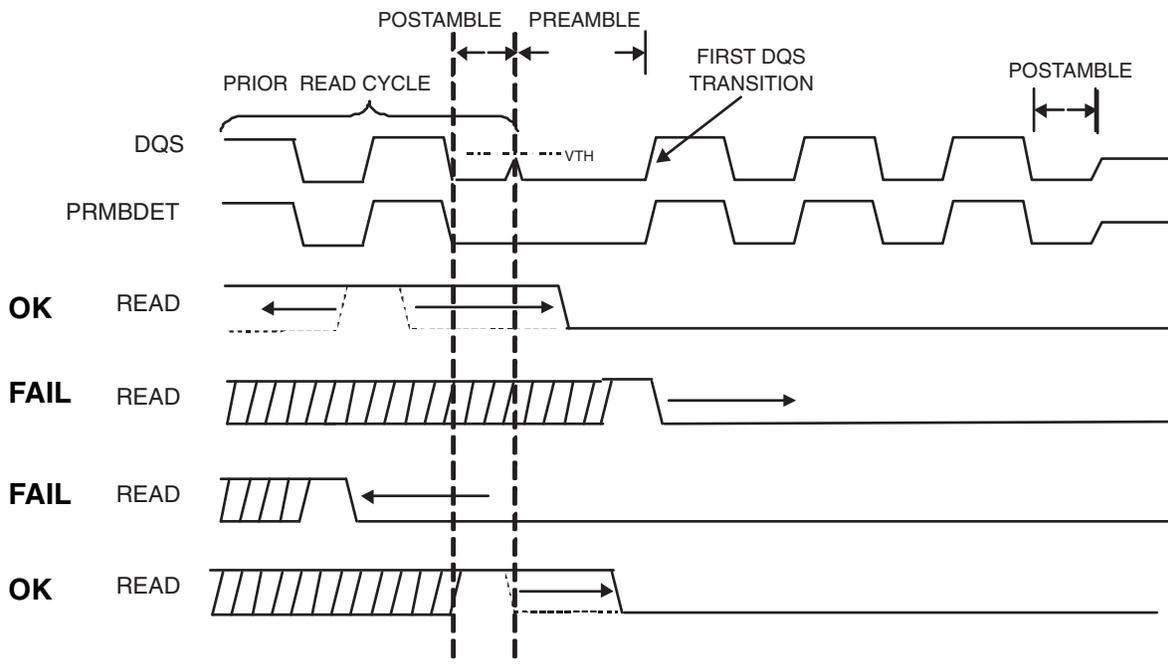
IDDRXB

This primitive will implement the input register block. The software defaults to CE Enabled unless otherwise specified. The ECLK input is used to connect to the DQS strobe coming from the DQS delay block (DQSBUFB primitive). The SCLK input should be connected to the system (FPGA) clock. The SCLK and CE inputs to this primitive will be used primarily to synchronize the DDR inputs. DDRCLKPOL is an input from the DQS Clock Polarity tree. This signal is generated by the DQS Transition detect circuit in the hardware. Figure 10-9 shows the primitive symbol and the I/O ports.

Figure 10-9. IDDRXB Symbol

```
              ┌─────────────────────────┐
              │          IDDRXB         │
          ────│ D                       │
          ────│ ECLK                    │
          ────│ LSR              QA     │────
          ────│ SCLK             QB     │────
          ────│ CE                      │
          ────│ DDRCLKPOL               │
              └─────────────────────────┘
```
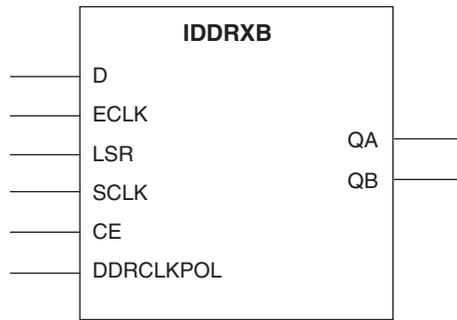
Table 10-6 provides a description of all I/O ports associated with the IDDRXB primitive.

Table 10-6. IDDRXB Ports

Port Name	I/O	Definition
D	I	DDR data
ECLK	I	The phase shifted DQS should be connected to this input
LSR	I	Reset
SCLK	I	System CLK
CE	I	Clock enable
DDRCLKPOL	I	DDR clock polarity signal
QA	O	Data at the positive edge of the CLK
QB	O	Data at the negative edge of the CLK

Note:

1. The DDRCLKPOL input to IDDRXB should be connected to the DDRCLKPOL output of DQSBUFB.

ODDRXB

The ODDRXB primitive implements both the write and the tristate functions. This primitive is used to output DDR data and the DQS strobe to the memory. The CKP and CKN can also be generated using this primitive. All the DDR output tristate implementations are also implemented using the same primitive.

Figure 10-10 shows the ODDRXB primitive symbol and its I/O ports.

Figure 10-10. ODDRXB Symbol
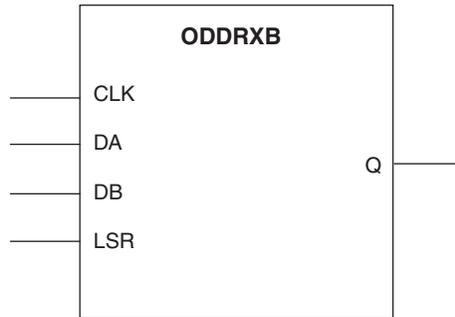
Table 10-7 provides a description of all I/O ports associated with the ODDRXB primitive.

Table 10-7. ODDRXB Ports

Port Name	I/O	Definition
CLK	I	System CLK
DA	I	Data at the positive edge of the clock
DB	I	Data at the negative edge of the clock
LSR	I	Reset
Q	I	DDR data to the memory

Notes:

1. LSR should be held low during DDR Write operation. By default, the software will be implemented CE High and LSR low.

2. DDR output and tristate registers do not have CE support. LSR is available for the tristate DDRX mode (while reading). The LSR will default to set when used in the tristate mode.

3. CE and LSR support is available for the regular (non-DDR) output mode.

4. When asserting reset during DDR writes, it is important to keep in mind that this would only reset the FFs and not the latches.

Memory Read Implementation

The LatticeECP/EC and LatticeXP devices contain a variety of features to simplify implementation of the read portion of a DDR interface:

- DLL compensated DQS delay elements
- DDR input registers
- Automatic DQS to system clock domain transfer circuitry

The LatticeECP/EC and LatticeXP device data sheets detail these circuit elements.

Three primitives in the Lattice ispLEVER® design tools represent the capability of these three elements. The DQS-DLL represents the DLL used for calibration. The IDDRXB primitive represents the DDR input registers and clock domain transfer registers. Finally, the DQSBUFB represents the DQS delay block and the clock polarity control logic. These primitives are explained in more detail in the following sections of this document. Figure 10-11 illustrates how to hook these primitives together to implement the read portion of a DDR memory interface. The DDR Software Primitives section describes each of the primitives and its instantiation in more detail. Appendices A and B provide example code to implement the complete I/O section of a memory interface within a LatticeECP/EC or LatticeXP device.

Figure 10-11. Software Primitive Implementation for Memory READ
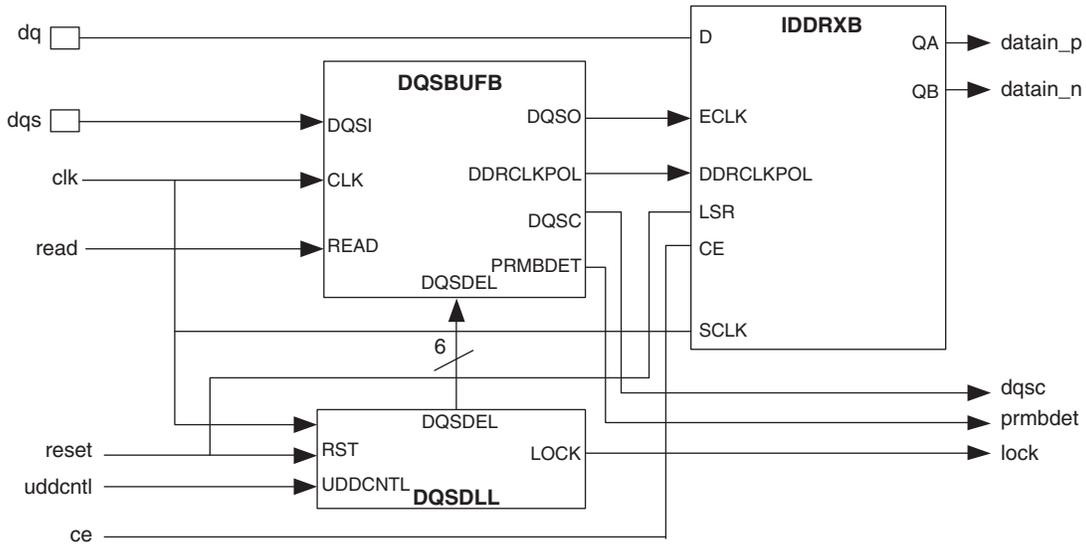
Read Timing Waveforms

Figure 10-12 and Figure 10-13 show READ data transfer for two cases based on the results of the DQS Transition detector logic. This circuitry decides whether or not to invert the phase of FPGA system CLK to the synchronization registers based on the relative phases of PRMBDET and CLK.

- Case 1 – If CLK = 0 on the 1st PRMBDET transition, then DDRCLKPOL = 0, hence no inversion required. (Figure 10-12)
- Case 2 – If CLK=1 on the 1st PRMBDET then DDRCLKPOL = 1, the system clock (CLK) needs to be inverted before it is used for synchronization. (Figure 10-13)

The signals A, B and C illustrate the Read Cycle half clock transfer at different stages of IDDRX registers. The first stage of the register captures data on the positive edge as shown by signal A and negative edge as shown by signal B. The data stream A goes through an additional half clock cycle transfers shown by signal C. Phase aligned data streams B and C are presented to the next stage registers clocked by the FPGA CLK

Figure 10-12. READ Data Transfer When DDRCLKPOL=0
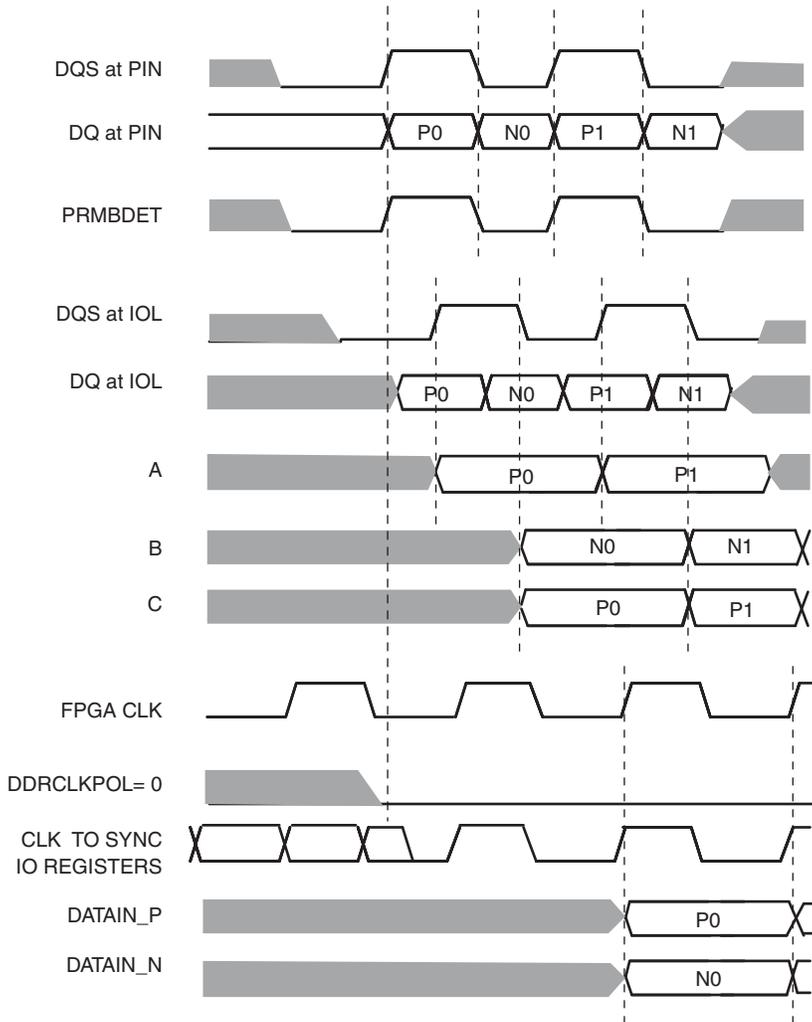
Notes -

(1) DDR memory sends DQ aligned to DQS strobe.
(2) The DQS Strobe is delayed by 90 degree using the dedicated DQS logic.
(3) DQ is now center aligned to DQS Strobe.
(4) PRMBDET is the Preamble detect signal generated using the DQSBUFB primitive. This is used to generate the DDRCLKPOL signal.
(5) The first set of IO registers A and B, capture data on the positive edge and negative edge of DQS.
(6) IO register C transfers data so that both data are now aligned to negative edge of DQS.
(7) DDCLKPOL signal generated will determine if the CLK going into the synchronization registers need to be inverted. In this case, the DDRCLKPOL=0 as the CLK is LOW at the 1st rising edge of PRMBDET.
(8) The IO Synchronization registers capture data at on positive edge of the FPGA CLK.

Figure 10-13. Read Data Transfer When DDRCLKPOL=1
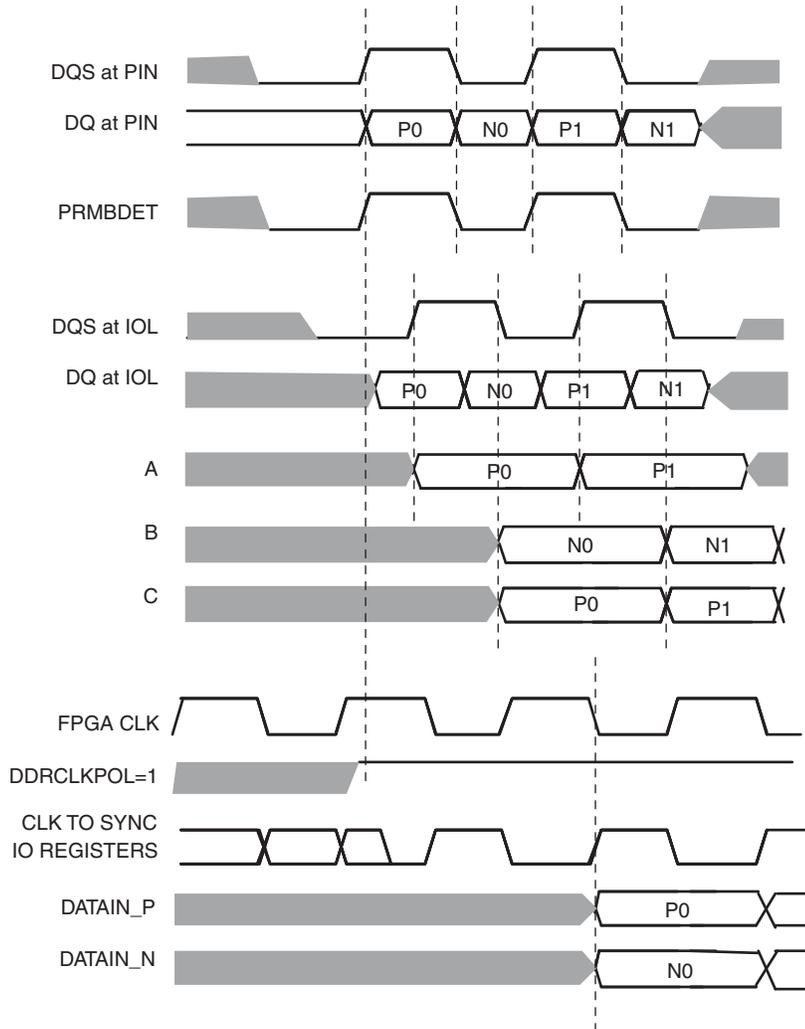
Notes -

(1) DDR memory sends DQ aligned to DQS strobe.
(2) The DQS Strobe is delayed by 90 degree using the dedicated DQS logic.
(3) DQ is now center aligned to DQS Strobe.
(4) PRMBDET is the Preamble detect signal generated using the DQSBUFB primitive. This is used to generate the DDRCLKPOL signal.
(5) The first set of IO registers A and B, capture data on the positive edge and negative edge of DQS.
(6) IO register C transfers data so that both data are now aligned to negative edge of DQS.
(7) DDCLKPOL signal generated will determine if the CLK going into the synchronization registers need to be inverted. In this case, the DDRCLKPOL=1 as the CLK is HIGH at the 1st rising edge of PRMBDET.
(8) The IO Synchronization registers capture data at on negative edge of the FPGA CLK.

Data Read Critical Path

Data in the second stage DDR registers can be registered either on the positive edge or on the falling edge of FPGA clock depending on the DDRCLKPOL signal. In order to ensure that the data transferred to the FPGA core registers is aligned to the rising edge of system CLK, this path should be constrained with a half clock transfer. This half clock transfer can be forced in the software by assigning a multicycle constraint (multicycle of 0.5 X) on all the data paths to the first PFU register.

DQS Postamble

At the end of a READ cycle, the DDR SDRAM device executes the READ cycle postamble and then immediately tristates both the DQ and DQS output drivers. Since neither the memory controller (FPGA) nor the DDR SDRAM device are driving DQ or DQS at that time, these signals float to a level determined by the off-chip termination resistors. While these signals are floating, noise on the DQS strobe may be interpreted as a valid strobe signal by the FPGA input buffer. This can cause the last READ data captured in the IOL input DDR registers to be overwritten before the data has been transferred to the free running resynchronization registers inside the FPGA.

Figure 10-14. Postamble Effect on READ
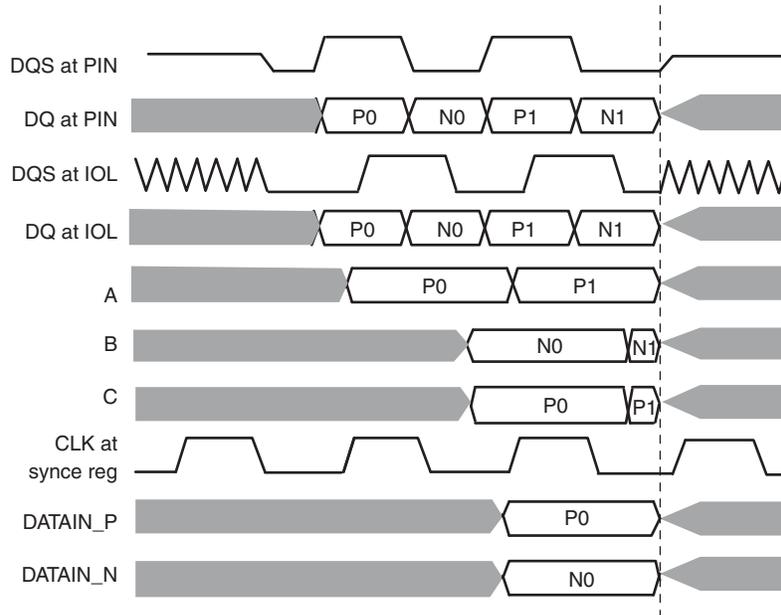
LatticeECP/EC and LatticeXP devices have extra dedicated logic in the in the DQS Delay Block that will prevent this postamble problem. The DQS postamble logic is automatically implemented when the user instantiates the DQS Delay logic (DQSBUFB software primitive) in a design.

This postamble solution was implemented in all the devices of the LatticeECP/EC and LatticeXP families except the LFEC20/LFECP20 device. For this device, it is recommended that the user issue an extra READ command to assure correct data has been transferred to the synchronization registers.

The circumstances under which the extended READ cycle is issued are given in Table 10-8.

Table 10-8. DDR Read Postamble

Current Command	Next Command	Action	Lost Cycles
Read (Row x, Bank y)	Read (Row x, Bank y)	None.	None
Read (any address)	NOP	Extend the current read command.[1]	3
Read (Row x, Bank y)	Read (Row n, Bank y)	Extend the current read to (Row x, Bank y) consecutive to current command	3
Read (Row x, Bank y)	Read (Row x, Bank n)	If the Row x, Bank n was open, do nothing. Else, extend the current read to Row x, Bank y	3
Read (any address)	Write/LMR	Extend the current read command.	3

1. Current read is extended one or more additional clock cycles.

Figure 10-15. Postamble Solution with Extra READ Command
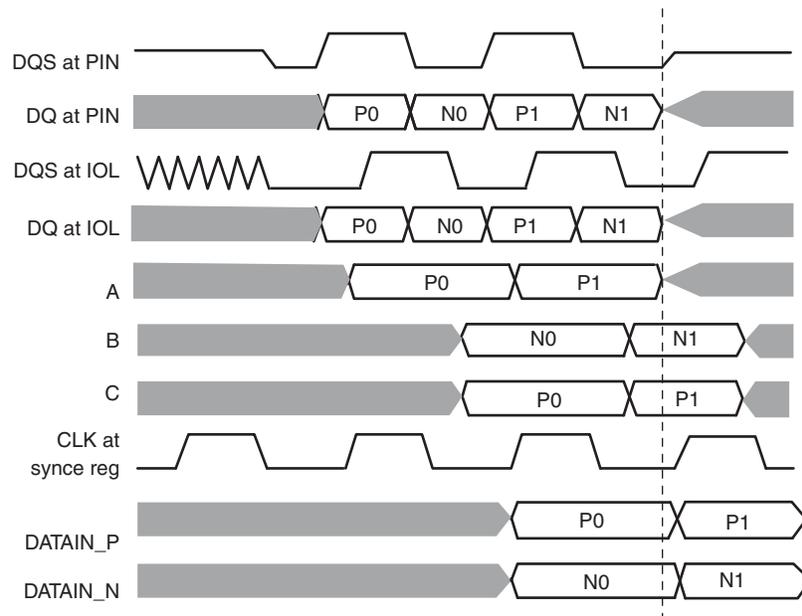
Memory Write Implementation

To implement the write portion of a DDR memory interface, two streams of single data rate data must be multiplexed together with data transitioning on both edges of the clock. In addition, during a write cycle, DQS must arrive at the memory pins center-aligned with data, DQ. Along with the strobe and data this portion of the interface provides the CLKP, CLKN Address/Command and Data Mask (DM) signals to the memory.

LatticeECP/EC and LatticeXP devices contain DDR output and tri-state registers along with PLLs that allow the easy implementation of the write portion of the DDR memory interfaces. The DDR output registers can be accessed in the design tools via the ODDRXB primitive.

All DDR output signals ("ADDR, CMD", DQS, DQ, DM) are initially aligned to the rising edge of CLK inside the FPGA core. These signals are used for the entire DDR write interface or the controls of DDR read interface. The relative phase of the signals may be adjusted in the IOL logic before departing the FPGA. The adjustments are shown in Figure 16

The adjustments are as follows:

The PLL is used to generate a 90 degree phase shifted clock. This 90 degree phase shifted clock will be used to generate DQS and the differential clocks going to the memory.

The CLKP needs to be centered relative to the ADDR,CMD signal, which is an SDR signal. This is accomplished by inverting the CLKP signal relative to the PLL's 90 degree phase shifted CLK.

The DDR clock can be generated by assigning "0" to the DA input and "1" to the DB inputs of the ODDRXB primitive as shown in Figure 10-16. This is then fed into a SSTL25 differential output buffer to generate CLKP and CLKN differential clocks. Generating the CLKN in this manner would prevent any skew between the two signals.

The DDR interface specification for t_{DSS} and t_{DSH} parameters, defined as DQS falling to CLKP rising setup and hold times must be met. This is met by making CLKP and DQS identical in phase. DQS is inverted to match CLKP (= CLK + 270). This is accomplished by routing the positive DQS data in core logic to DB, and negative DQS data in core logic to DA inputs of the ODDRXB primitive.

Internally the DQS and ADDR/CMD signals are clocked using the primary FPGA clock. Therefore, the user will need to do a 1/4 (one-quarter) clock transfer from the core logic to the DDR registers. Timing can be hard to meet, so it is recommended that the user first register these signals with the inverted Clock, so that the transfer from the core logic to I/O registers will only require a 1/2 (half) clock transfer.

The data DQ and DM needs to be delayed by 90° as it leaves the FPGA. This is to center the data and data mask relative to the DQS when it reaches the DDR memory. This can be accomplished by inverting the CLK to the DQ and DM data.

The DM signal is generated using the same clock as the DQ data pin. The memory masks the DQ signals if the DM pins are driven high.

The tristate control for the data output can also be implemented using the ODDRXB primitive.

Figure 10-16 illustrates how to hook up the ODDRXB primitives and the PLL. The DDR Software Primitives section describes each of the primitives and its instantiation in more detail. Appendix A and Appendix B provide example code for implementing the complete I/O section of a memory interface for a LatticeECP/EC or LatticeXP device.

Figure 10-16. Software Primitive Implementation for Memory Write
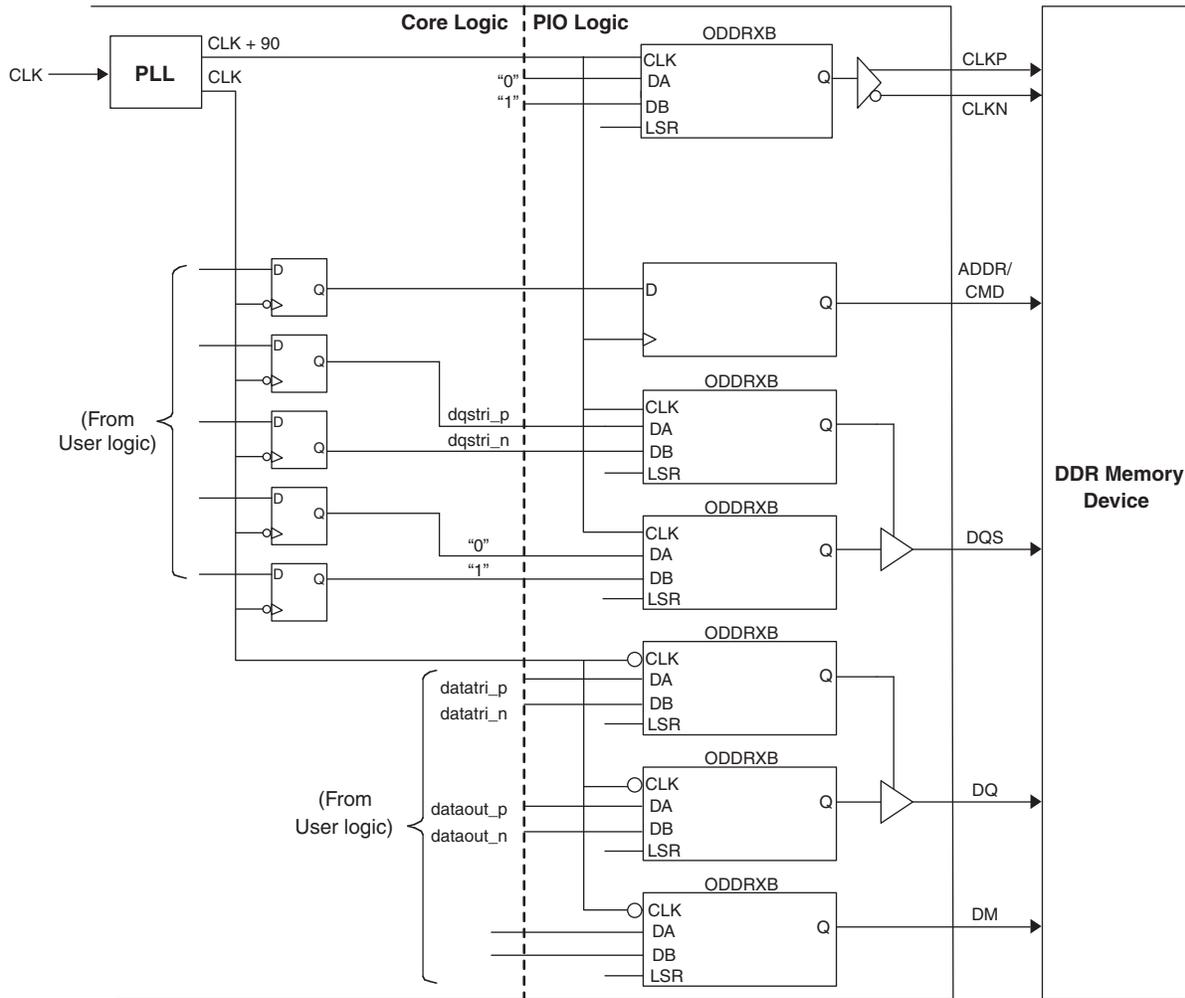
Write Timing Waveforms

Figure 10-17 shows DDR write side data transfer timing for the DQ Data pad and the DQS Strobe Pad. When writing to the DDR memory device, the DM (Data Mask) and the ADDR/ CMD (Address and Command) signals are also sent to the memory device along with the data and strobe signals.

Figure 10-17. DDR Write Data Transfer for DQ Data
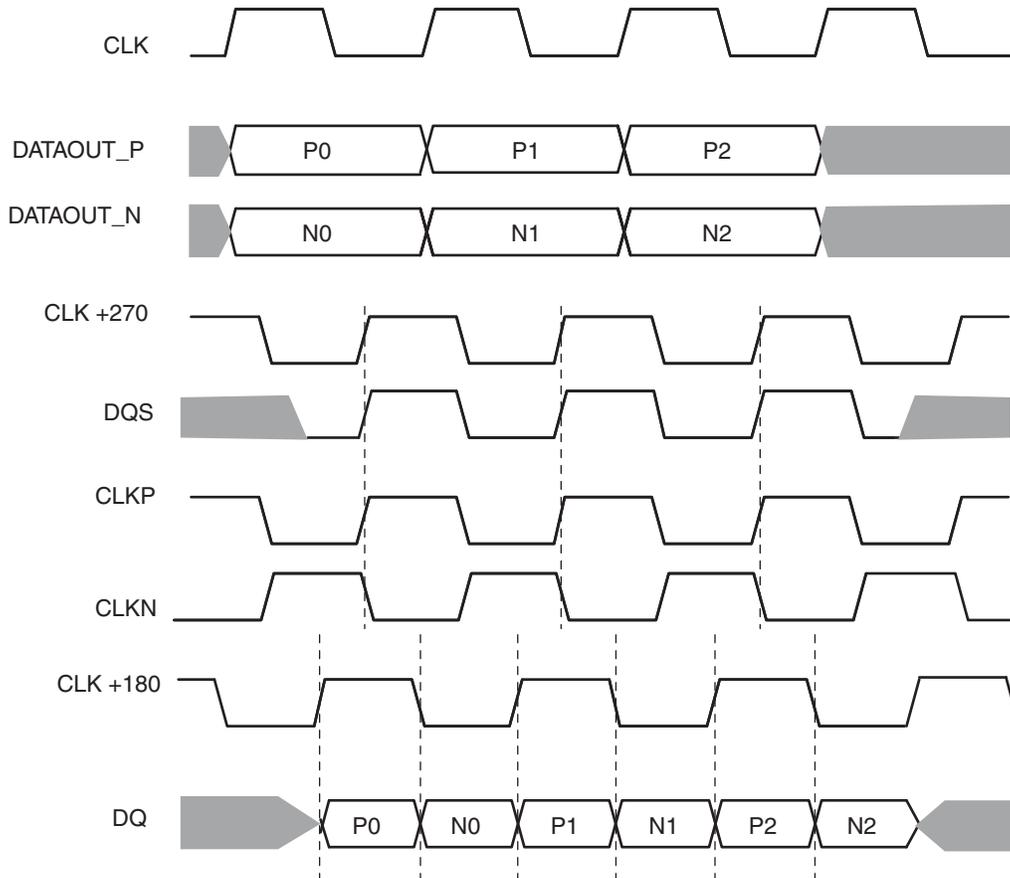
Notes -
(1) DATAOUT_P and DATAOUT_N are inputs to the DDR output registers.
(2) DQS is generated at 270 degree phase of CLK.
(3) CLKP is generated simular to DQS and CLKN is the inverted CLKP.
(4) DQ is generated at 180 degree phase of CLK.
(5) DQ is center aligned with the DQS strobe signal when it reaches the memory.

Design Rules/Guidelines

Listed below are some rules and guidelines to keep in mind when implementing DDR memory interfaces in the LatticeECP/EC and LatticeXP devices.

- The LatticeECP/EC and LatticeXP devices have dedicated DQ-DQS banks. Please refer to the logical signal connections of the groups in the LatticeECP/EC and LatticeXP data sheets before locking these pins.

- There are two DQSDLLs on the device, one for the top half and one for the bottom half. Hence, only one DQSDLL primitive should be instantiated for each half of the device. Since there is only one DQSDLL on each half of the device, all the DDR memory interfaces on that half of the device should run at the same frequency. Each DQSDLL will generate 90 degree digital delay bits for all the DQS delay blocks on that half of the device based on the reference clock input to the DLL.

- The DDR SDRAM interface supports the SSTL25 I/O standard, therefore the interface pins should be assigned as SSTL25 I/O type.

- When implementing a DDR interface, the VREF1 of the bank is used to provide the reference voltage for the interface pins.

- Appendix F shows DDR400 implementation results of the LatticeEC Advanced Evaluation Board.

QDR II Interface

QDR II SRAM is a new memory technology defined by a number of leading memory vendors for high-performance and high-bandwidth communication applications. QDR is a synchronous pipelined burst SRAM with two separate unidirectional data buses dedicated for read and write operations running at double data rate. Both the QDR II read and write interfaces use HSTL 1.8V I/O standard.

A QDR II memory controller can be easily implemented using the LatticeECP/EC and LatticeXP devices by taking advantage of the DDR I/O registers. For LatticeECP/EC and LatticeXP devices, ODDRXB primitives are used on the QDR outputs and PFU registers are used on the QDR inputs to implement the DDR interface. To see the details of this implementation refer to Lattice reference design RD1019, *QDR Memory Controller* on the Lattice web site at www.latticesemi.com.

FCRAM (Fast Cycle Random Access Memory) Interface

FCRAM is a DDR-type DRAM, which performs data output at both the rising and the falling edges of the clock. FCRAM devices operate at a core voltage of 2.5V with SSTL Class II I/O. It has enhanced both the core and peripheral logic of the SDRAM. In FCRAM the address and command signals are synchronized with the clock input, and the data pins are synchronized with the DQS signal. Data output takes place at both the rising and falling edges of the DQS. DQS is in phase with the clock input of the device. The DDR SDRAM and DDR FCRAM controller will have different pin outs.

LatticeECP/EC and LatticeXP devices can implement an FCRAM interface using the dedicated DQS logic, input DDR registers and output DDR registers as described in the Implementing Memory Interfaces section of this document. Generation of address and control signals for FCRAM are different compared to the DDR SDRAM devices. Please refer to the FCRAM data sheets to see detailed specifications. Toshiba, Inc. and Fujitsu, Inc. offer FCRAM devices in 256Mb densities. They are available in x8 or x16 configurations.

Generic High Speed DDR Implementation

In addition to the DDR memory interface, users can use the I/O logic registers to implement a high speed DDR interface. DDR data write operations can be implemented using the DDR output registers similar to the memory interface implementation using the ODDRXB primitives.

On the input side, the read interface can be implemented using the core logic PFU registers. The PFU register next to the I/O cells can be used to de-mux the DDR data to single data rate data. This method of implementing DDR can run at 300 MHz when accompanied by proper preferences in the software. The HDL and the preferences to implement this DDR interface are listed in Appendix D of this document.

Board Design Guidelines

The most common challenge associated with implementing DDR memory interfaces is the board design and layout. Users must strictly follow the guidelines recommended by memory device vendors.

Some common recommendations include matching trace lengths of interface signals to avoid skew, proper DQ-DQS signal grouping, proper termination of the SSTL2 I/O standard, proper VREF and VTT generation decoupling and proper PCB routing.

Some reference documents that discuss board layout guidelines:

- www.idt.com, IDT, PCB Design for Double Data Rate Memory.
- www.motorola.com, AN2582, Hardware and Layout Design Considerations for DDR Interfaces.
- www.micron.com, TN4607, DDR 333 Design Guide for Two DIMM Systems

References

- www.jedec.org – JEDEC Standard 79, Double Data Rate (DDR) SDRAM Specification
- www.micron.com – DDR SDRAM Data Sheets
- www.infinion.com – DDR SDRAM Data Sheets
- www.samsung.com – DDR SDRAM Data Sheets
- www.latticesemi.com – RD1019 *QDR Memory Controller* Reference Design for LatticeECP/EC devices
- www.toshiba.com – DDR FCRAM Data Sheet
- www.fujitsu.com – DDR FCRAM Data Sheet
- www.latticesemi.com – LatticeEC Advanced Evaluation Board User's Guide
- www.latticesemi.com – DDR SDRAM Controller (Pipelined Version for LatticeECP/EC Devices) User's Guide

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

+1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
—	—	Previous Lattice releases.
February 2007	03.2	Updated Generic High Speed DDR Implementation section.
		Updated Appendix D.

Appendix A. Using IPexpress™ to Generate DDR Modules

The input and output DDR module can be generated using IPexpress. The I/O section under the Architecture modules provides two options to the user:

1. DDR_GENERIC – The option allows generation of a Generic DDR interface, which in the case of LatticeECP/EC and LatticeXP devices, is only the output side DDR. The input side for a Generic DDR interface must be implemented using PFU registers. Appendix D provides the example code for the input side generic DDR.
2. DDR_MEM – This option allows the user to generate a complete DDR memory interface. It will generate both the read and write side interface required to interface with the memory.

IPexpress generates only the modules that are implemented within the IOLOGIC. Any logic required in the FPGA core to complete the memory interface must be implemented by the user.

Figure 10-18. IPexpress I/O Section
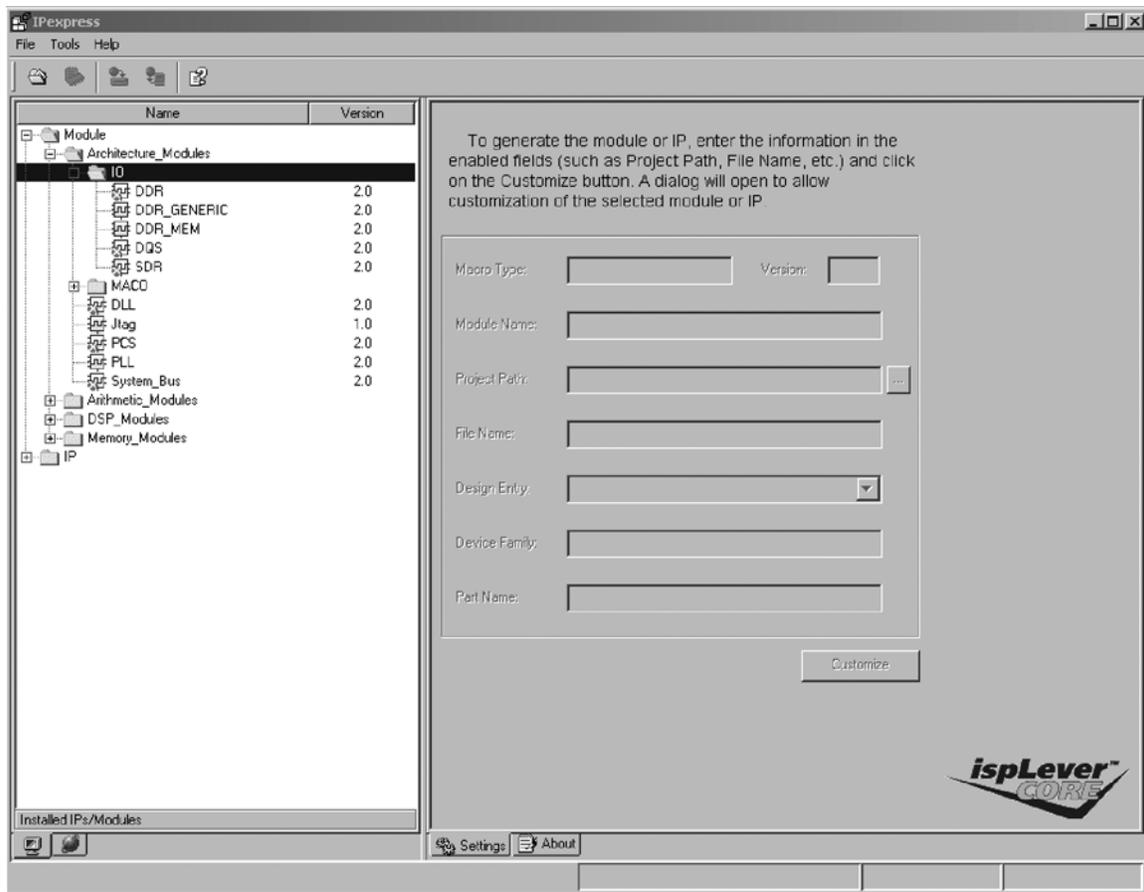
DDR Generic

DDR Generic will generate the output DDR (ODDRXB) primitives for a given bus width. The user has the option to enable or disable tristate control to the output DDR registers. Figure 10-19 shows the DDR Generic views of IPexpress.

Figure 10-19. DDR Generic Configuration Options
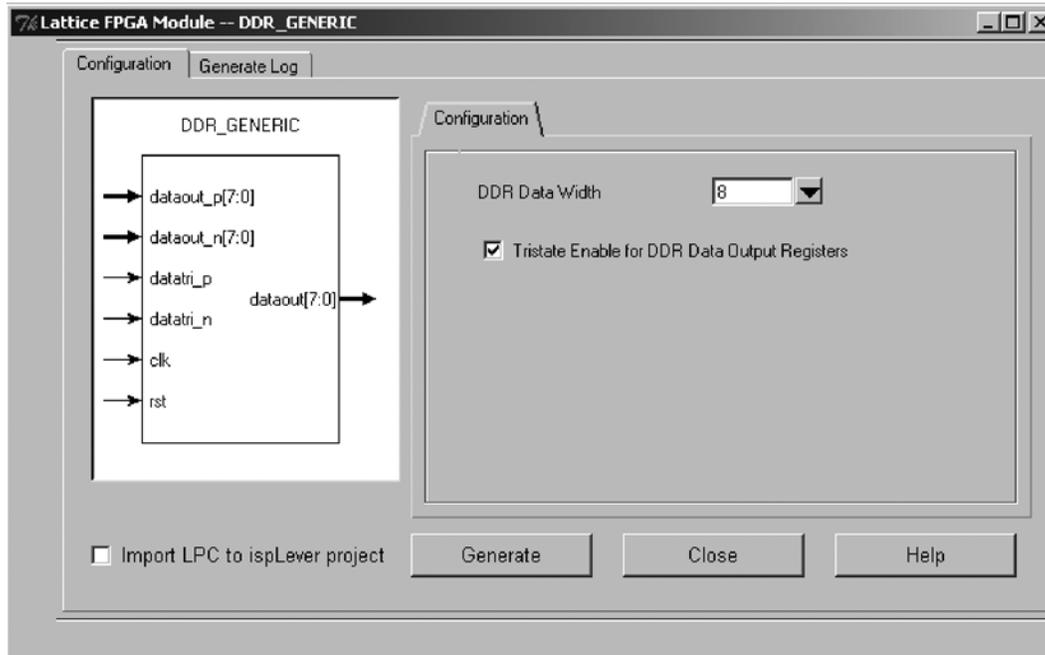
DDR Memory Interface

This IPexpress option generates both read and write interfaces using all DDR primitives for a given bus width. Figure 10-20 shows the options under this section.

Figure 10-20. Configuration Options for DDR Memory Interface
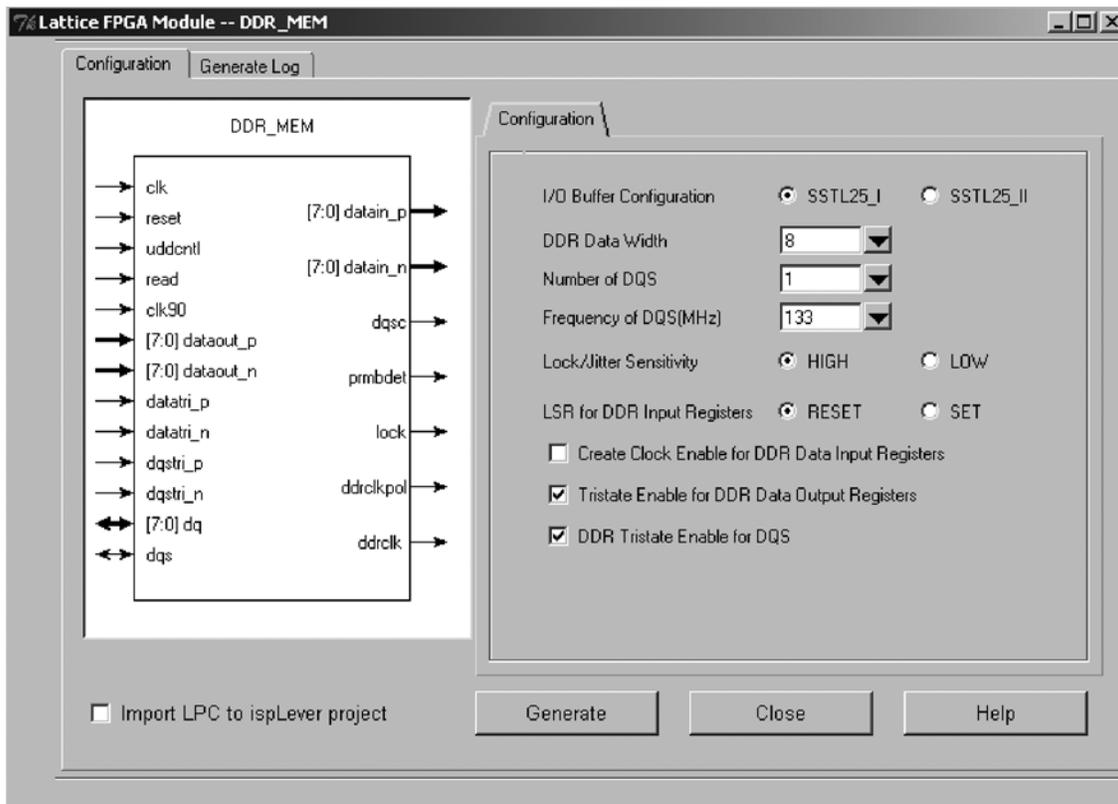
Appendix B. Verilog Example for DDR Input and Output Modules

```verilog
module ddr_mem (dq, dqs, clk, reset, uddcntl, read, datain_p, datain_n, dqsc, prmbdet, lock,
ddrclkpol, clk90, dqstri_p, dqstri_n, datatri_p, datatri_n, dataout_p, dataout_n, ddrclk);

    inout [7:0] dq/* synthesis IO_TYPE="SSTL25_II"*/;
    inout dqs/* synthesis IO_TYPE="SSTL25_II"*/;

    --clk is the core clock and clk90 is the 90 degree phase shifted clock coming from the PLL
    input clk, clk90;
    input reset, uddcntl, read;
    input [7:0] dataout_p, dataout_n;
    input [7:0] datatri_p, datatri_n;
    input dqstri_p, dqstri_n;

    output [7:0] datain_p;
    output[7:0] datain_n;
    output dqsc, prmbdet, lock, ddrclkpol;

    output ddrclk /* synthesis IO_TYPE="SSTL25D_II"*/ ;
    wire vcc_net,gnd_net;
    wire dqsbuf, dqsdel, clk, ddrclkpol_sig;
    wire [7:0] ddrin, ddrout, tridata;
    wire dqsout, tridqs, dqsin, ddrclk;


    assign vcc_net = 1'b1;
    assign gnd_net = 1'b0;
    assign ddrclkpol = ddrclkpol_sig;


    //-------Bidirectional Buffers ----------------------------------------------------
    BB bidiInst0 (.I(ddrout[0]), .T(tridata[0]), .O(ddrin[0]), .B(dq[0]));
    BB bidiInst1 (.I(ddrout[1]), .T(tridata[1]), .O(ddrin[1]), .B(dq[1]));
    BB bidiInst2 (.I(ddrout[2]), .T(tridata[2]), .O(ddrin[2]), .B(dq[2]));
    BB bidiInst3 (.I(ddrout[3]), .T(tridata[3]), .O(ddrin[3]), .B(dq[3]));
    BB bidiInst4 (.I(ddrout[4]), .T(tridata[4]), .O(ddrin[4]), .B(dq[4]));
    BB bidiInst5 (.I(ddrout[5]), .T(tridata[5]), .O(ddrin[5]), .B(dq[5]));
    BB bidiInst6 (.I(ddrout[6]), .T(tridata[6]), .O(ddrin[6]), .B(dq[6]));
    BB bidiInst7 (.I(ddrout[7]), .T(tridata[7]), .O(ddrin[7]), .B(dq[7]));

    //Bidirectional Strobe, DQS
    BB bidiInst8(.I(dqsout), .T(tridqs), .O(dqsin), .B(dqs));
    //-------------------------------------------------------------------------------

    //-----------DDR Input -------------------------------------------------------


    DQSBUFB   U8   (.DQSI(dqsin),   .CLK(clk),   .READ(read),   .DQSDEL(dqsdel),   .DDRCLK-
POL(ddrclkpol_sig),
                .DQSC(dqsc), .PRMBDET(prmbdet), .DQSO(dqsbuf));

    DQSDLL U9 (.CLK(clk), .UDDCNTL(uddcntl), .RST(reset), .DQSDEL(dqsdel), .LOCK(lock));

    IDDRXB   UL0   (.D(ddrin[0]),   .ECLK(dqsbuf),   .SCLK(clk),   .CE(vcc_net),   .DDRCLK-
POL(ddrclkpol_sig),
                .LSR(reset), .QA(datain_p[0]), .QB(datain_n[0]));
```

```
    IDDRXB    UL1    (.D(ddrin[1]),    .ECLK(dqsbuf),    .SCLK(clk),    .CE(vcc_net),    .DDRCLK-
POL(ddrclkpol_sig),
                .LSR(reset), .QA(datain_p[1]), .QB(datain_n[1]));

    IDDRXB    UL2    (.D(ddrin[2]),    .ECLK(dqsbuf),    .SCLK(clk),    .CE(vcc_net),    .DDRCLK-
POL(ddrclkpol_sig),
                .LSR(reset), .QA(datain_p[2]), .QB(datain_n[2]));

    IDDRXB    UL3    (.D(ddrin[3]),    .ECLK(dqsbuf),    .SCLK(clk),    .CE(vcc_net),    .DDRCLK-
POL(ddrclkpol_sig),
                .LSR(reset), .QA(datain_p[3]), .QB(datain_n[3]));

    IDDRXB    UL4    (.D(ddrin[4]),    .ECLK(dqsbuf),    .SCLK(clk),    .CE(vcc_net),    .DDRCLK-
POL(ddrclkpol_sig),
                .LSR(reset), .QA(datain_p[4]), .QB(datain_n[4]));

    IDDRXB    UL5    (.D(ddrin[5]),    .ECLK(dqsbuf),    .SCLK(clk),    .CE(vcc_net),    .DDRCLK-
POL(ddrclkpol_sig),
                .LSR(reset), .QA(datain_p[5]), .QB(datain_n[5]));

    IDDRXB    UL6    (.D(ddrin[6]),    .ECLK(dqsbuf),    .SCLK(clk),    .CE(vcc_net),    .DDRCLK-
POL(ddrclkpol_sig),
                .LSR(reset), .QA(datain_p[6]), .QB(datain_n[6]));

    IDDRXB    UL7    (.D(ddrin[7]),    .ECLK(dqsbuf),    .SCLK(clk),    .CE(vcc_net),    .DDRCLK-
POL(ddrclkpol_sig),
                .LSR(reset), .QA(datain_p[7]), .QB(datain_n[7]));


  //-------------------------------------------------------------------------------------
  //----TRISTATE Instantiations----------------------------------------------------------
  // DDR Trisate for data, DQ

  ODDRXB T0 (.DA(datatri_p[0]), .DB(datatri_n[0]), .LSR(reset), .CLK(~clk), .Q(tridata[0]));
  ODDRXB T1 (.DA(datatri_p[1]), .DB(datatri_n[1]), .LSR(reset), .CLK(~clk), .Q(tridata[1]));
  ODDRXB T2 (.DA(datatri_p[2]), .DB(datatri_n[2]), .LSR(reset), .CLK(~clk), .Q(tridata[2]));
  ODDRXB T3 (.DA(datatri_p[3]), .DB(datatri_n[3]), .LSR(reset), .CLK(~clk), .Q(tridata[3]));
  ODDRXB T4 (.DA(datatri_p[4]), .DB(datatri_n[4]), .LSR(reset), .CLK(~clk), .Q(tridata[4]));
  ODDRXB T5 (.DA(datatri_p[5]), .DB(datatri_n[5]), .LSR(reset), .CLK(~clk), .Q(tridata[5]));
  ODDRXB T6 (.DA(datatri_p[6]), .DB(datatri_n[6]), .LSR(reset), .CLK(~clk), .Q(tridata[6]));
  ODDRXB T7 (.DA(datatri_p[7]), .DB(datatri_n[7]), .LSR(reset), .CLK(~clk), .Q(tridata[7]));

  // DDR Trisate for strobe, DQS
  ODDRXB T8 (.DA(dqstri_p), .DB(dqstri_n), .LSR(reset), .CLK(clk90), .Q(tridqs));

  //-------------------------------------------------------------------------------------
-

  //-----------DQ output-----------------------------------------------------------------
-
  ODDRXB O0 (.DA(dataout_p[0]), .DB(dataout_n[0]), .LSR(reset), .CLK(~clk), .Q(ddrout[0]));
  ODDRXB O1 (.DA(dataout_p[1]), .DB(dataout_n[1]), .LSR(reset), .CLK(~clk), .Q(ddrout[1]));
  ODDRXB O2 (.DA(dataout_p[2]), .DB(dataout_n[2]), .LSR(reset), .CLK(~clk), .Q(ddrout[2]));
  ODDRXB O3 (.DA(dataout_p[3]), .DB(dataout_n[3]), .LSR(reset), .CLK(~clk), .Q(ddrout[3]));
  ODDRXB O4 (.DA(dataout_p[4]), .DB(dataout_n[4]), .LSR(reset), .CLK(~clk), .Q(ddrout[4]));
  ODDRXB O5 (.DA(dataout_p[5]), .DB(dataout_n[5]), .LSR(reset), .CLK(~clk), .Q(ddrout[5]));
  ODDRXB O6 (.DA(dataout_p[6]), .DB(dataout_n[6]), .LSR(reset), .CLK(~clk), .Q(ddrout[6]));
  ODDRXB O7 (.DA(dataout_p[7]), .DB(dataout_n[7]), .LSR(reset), .CLK(~clk), .Q(ddrout[7]));
```

```
  //-------------------------------------------------------------------------------
--
  //-------------- DQS output-----------------------------------------------------
--
  ODDRXB O8 (.DA(gnd_net), .DB(vcc_net), .LSR(reset), .CLK(clk90), .Q(dqsout));
  //-------------------------------------------------------------------------------
--

  //-------------- CLKOUTP and CLKOUTN Generation----------------------------------
--
  ODDRXB O9 (.DA(gnd_net), .DB(vcc_net), .LSR(reset), .CLK(clk90), .Q(ddrclk));
  //-------------------------------------------------------------------------------
--
endmodule
```

Appendix C. VHDL Example for DDR Input and Output Modules

```
library IEEE;
use IEEE.std_logic_1164.all;
library ec;
use ec.components.all;

entity ddr_mem is
        port( dq      : inout std_logic_vector(7 downto 0 );
              dqs     : inout std_logic;
              clk     : in std_logic; -- core clock
              clk90     : in std_logic; -- 90 degree phase shifted clock from the pll
              reset     : in std_logic;
              uddcntl   : in std_logic;
              read      : in std_logic;
              dataout_p : in std_logic_vector(7 downto 0);
              dataout_n : in std_logic_vector(7 downto 0);
              datatri_p : in std_logic_vector(7 downto 0);
              datatri_n : in std_logic_vector(7 downto 0);
              dqstri_p  : in std_logic;
              dqstri_n  : in std_logic;
              ddrclk    : out std_logic;
              datain_p  : out std_logic_vector(7 downto 0);
              datain_n  : out std_logic_vector(7 downto 0);
              dqsc      : out std_logic;
              prmbdet   : out std_logic;
              lock      : out std_logic;
              ddrclkpol : out std_logic);

--*****DDR interface signals assigned SSTL25 IO Standard *************
ATTRIBUTE IO_TYPE              : string;
ATTRIBUTE IO_TYPE OF ddrclk  : SIGNAL IS "SSTL25D_II";
ATTRIBUTE IO_TYPE OF dq      : SIGNAL IS "SSTL25_II";
ATTRIBUTE IO_TYPE OF dqs     : SIGNAL IS "SSTL25_II";

end ddr_mem;

architecture structure of ddr_mem is

--*****DDR Input register*********************************************
 component IDDRXB
    port(
        D    : in STD_LOGIC;
        ECLK : in STD_LOGIC;
        SCLK : in STD_LOGIC;
        CE   : in STD_LOGIC;
        LSR  : in STD_LOGIC;
        DDRCLKPOL  : in STD_LOGIC;
        QA   : out STD_LOGIC;
        QB   : out STD_LOGIC);
 end component;
```

```
--*******DDR Output register *******************************************
 component ODDRXB
    port(
        CLK  : in STD_LOGIC;
        DA   : in STD_LOGIC;
        DB   : in STD_LOGIC;
        LSR  : in STD_LOGIC;
        Q    : out STD_LOGIC);
 end component;


--*******Bidirectional Buffer*******************************************
 component BB
    port(
        I  : in STD_LOGIC;
        T  : in STD_LOGIC;
        O  : out STD_LOGIC;
        B  : inout STD_LOGIC);
 end component;


--******DQS DLL Component***********************************************
 component DQSDLL
    port(
        CLK        : in STD_LOGIC;
        RST        : in STD_LOGIC;
        UDDCNTL    : in STD_LOGIC;
        LOCK       : out STD_LOGIC;
        DQSDEL     : out STD_LOGIC);
 end component;


--****** DQS Delay block***********************************************
 component DQSBUFB
     port(
        DQSI       : in STD_LOGIC;
        CLK        : in STD_LOGIC;
        READ       : in STD_LOGIC;
        DQSDEL     : in STD_LOGIC;
        DQSO       : out STD_LOGIC;
        DDRCLKPOL  : out STD_LOGIC;
        DQSC       : out STD_LOGIC;
        PRMBDET    : out STD_LOGIC);
 end component;


 signal dqsbuf : std_logic;
 signal dqsdel : std_logic;
 signal ddrclkpol_sig : std_logic;
 signal ddrin : std_logic_vector(7 downto 0 );
 signal ddrout : std_logic_vector(7 downto 0 );
 signal tridata : std_logic_vector(7 downto 0 );
 signal dqsout : std_logic;
 signal tridqs : std_logic;
 signal dqsin : std_logic;
 signal vcc_net : std_logic;
 signal gnd_net : std_logic;
```

```
signal clkinv : std_logic;
signal ddrclk : std_logic;


begin
  vcc_net <= '1';
  gnd_net <= '0';
  clkinv<= not clk;
  ddrclkpol<=ddrclkpol_sig;

--*************BIDIRECTIONAL BUFFERS****************************************************
    bidiInst0 : BB PORT MAP( I => ddrout(0),T => tridata(0),O => ddrin(0),B => dq(0));
    bidiInst1 : BB PORT MAP( I => ddrout(1),T => tridata(1),O => ddrin(1),B => dq(1));
    bidiInst2 : BB PORT MAP( I => ddrout(2),T => tridata(2),O => ddrin(2),B => dq(2));
    bidiInst3 : BB PORT MAP( I => ddrout(3),T => tridata(3),O => ddrin(3),B => dq(3));
    bidiInst4 : BB PORT MAP( I => ddrout(4),T => tridata(4),O => ddrin(4),B => dq(4));
    bidiInst5 : BB PORT MAP( I => ddrout(5),T => tridata(5),O => ddrin(5),B => dq(5));
    bidiInst6 : BB PORT MAP( I => ddrout(6),T => tridata(6),O => ddrin(6),B => dq(6));
    bidiInst7 : BB PORT MAP( I => ddrout(7),T => tridata(7),O => ddrin(7),B => dq(7));

    bidiInst8 : BB PORT MAP( I=> dqsout, T=> tridqs, O=> dqsin, B=> dqs);

--*********************************************************************************

--*************DDRInput**********************************************************
--DQSDLL, generates the DQS delay
    I0: DQSDLL PORT MAP(CLK=>clk, UDDCNTL=> uddcntl, RST=> reset, DQSDEL=> dqsdel,
                        LOCK => lock);

    I1: DQSBUFB PORT MAP( DQSI=> dqsin, CLK=>clk, READ=> read, DQSDEL=> dqsdel,
                        DDRCLKPOL=> ddrclkpol_sig, DQSC=> dqsc, PRMBDET=> prmbdet,
                        DQSO=> dqsbuf);

--DDR INPUT primitives
    I2 : IDDRXB PORT MAP(D=> ddrin(0), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(0),
                        QB => datain_n(0));
    I3 : IDDRXB PORT MAP(D=> ddrin(1), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(1),
                        QB => datain_n(1));
    I4 : IDDRXB PORT MAP(D=> ddrin(2), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(2),
                        QB => datain_n(2));
    I5 : IDDRXB PORT MAP(D=> ddrin(3), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(3),
                        QB => datain_n(3));
    I6 : IDDRXB PORT MAP(D=> ddrin(4), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(4),
                        QB => datain_n(4));
    I7 : IDDRXB PORT MAP(D=> ddrin(5), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(5),
                        QB => datain_n(5));
```

```
    I8 : IDDRXB PORT MAP(D=> ddrin(6), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                         DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(6),
                         QB => datain_n(6));
    I9 : IDDRXB PORT MAP(D=> ddrin(7), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                         DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(7),
                         QB => datain_n(7));
--***********************************************************************************

--*****TRISTATE Instantiations******************************************************
-- DDR Trisate for data, DQ
    T0 : ODDRXB PORT MAP( DA => datatri_p(0), DB => datatri_n(0), LSR => reset,
                         CLK => clkinv, Q => tridata(0));
    T1 : ODDRXB PORT MAP( DA => datatri_p(1), DB => datatri_n(1), LSR => reset,
                         CLK => clkinv, Q => tridata(1));
    T2 : ODDRXB PORT MAP( DA=>  datatri_p(2), DB => datatri_n(2), LSR => reset,
                         CLK => clkinv, Q => tridata(2));
    T3 : ODDRXB PORT MAP( DA => datatri_p(3), DB => datatri_n(3), LSR => reset,
                         CLK => clkinv, Q => tridata(3));
    T4 : ODDRXB PORT MAP( DA => datatri_p(4), DB => datatri_n(4), LSR => reset,
                         CLK => clkinv, Q => tridata(4));
    T5 : ODDRXB PORT MAP( DA => datatri_p(5), DB => datatri_n(5), LSR => reset,
                         CLK => clkinv, Q => tridata(5));
    T6 : ODDRXB PORT MAP( DA => datatri_p(6), DB => datatri_n(6), LSR => reset,
                         CLK => clkinv, Q => tridata(6));
    T7 : ODDRXB PORT MAP( DA => datatri_p(7), DB => datatri_n(7), LSR => reset,
                         CLK => clkinv, Q => tridata(7));

--DDR Trisate for strobe, DQS
    T8: ODDRXB PORT MAP( DA =>dqstri_p, DB=> dqstri_n, LSR=> reset, CLK=> clk90,
                         Q => tridqs);
--***********************************************************************************

--**************DDR Output**********************************************************
--DQ OUTPUT
    O0 : ODDRXB PORT MAP( DA => dataout_p(0), DB  => dataout_n(0), LSR => reset,
                         CLK  => clkinv, Q => ddrout(0));
    O1 : ODDRXB PORT MAP( DA => dataout_p(1), DB  => dataout_n(1), LSR => reset,
                         CLK  => clkinv, Q => ddrout(1));
    O2 : ODDRXB PORT MAP( DA => dataout_p(2), DB  => dataout_n(2), LSR => reset,
                         CLK  => clkinv, Q => ddrout(2));
    O3 : ODDRXB PORT MAP( DA => dataout_p(3), DB  => dataout_n(3), LSR => reset,
                         CLK  => clkinv, Q => ddrout(3));
    O4 : ODDRXB PORT MAP( DA => dataout_p(4), DB  => dataout_n(4), LSR => reset,
                         CLK  => clkinv, Q => ddrout(4));
    O5 : ODDRXB PORT MAP( DA => dataout_p(5), DB  => dataout_n(5), LSR => reset,
                         CLK  => clkinv, Q => ddrout(5));
    O6 : ODDRXB PORT MAP( DA => dataout_p(6), DB  => dataout_n(6), LSR => reset,
                         CLK  => clkinv, Q => ddrout(6));
    O7 : ODDRXB PORT MAP( DA => dataout_p(7), DB  => dataout_n(7), LSR => reset,
                         CLK  => clkinv, Q => ddrout(7));
```

```
--DQS output
    O8: ODDRXB PORT MAP( DA => gnd_net, DB => vcc_net, LSR => reset, CLK => clk90,
                         Q => dqsout);

--clkp and clkn Generation

    O9 : ODDRXB PORT MAP( DA => vcc_net, DB => gnd_net, LSR => reset, CLK => clk90,
                         Q => ddrclk);
--********************************************************************************

end structure;
```

Appendix D. Generic (Non-Memory) High-Speed DDR Interface

The following HDL implements the DDR input interface using PFU registers for non-memory DDR applications.

VHDL Implementation

```
library IEEE;
use IEEE.std_logic_1164.all;

library ec;
use ec.components.all;

entity ddrin is
 port (rst : in std_logic;
       ddrclk: in std_logic;
       ddrdata: in std_logic_vector(7 downto 0);
       datap: out std_logic_vector(7 downto 0);
       datan: out std_logic_vector(7 downto 0));


end ddrin;
architecture structure of ddrin is
-- parameterized module component declaration
component pll90
    port (CLK: in std_logic; RESET: in std_logic; CLKOP: out std_logic;
        CLKOS: out std_logic; LOCK: out std_logic);
end component;


        signal pos0 : std_logic_vector( 7 downto 0 );
        signal pos1 : std_logic_vector( 7 downto 0 );
        signal neg0 : std_logic_vector( 7 downto 0 );

        signal clklock : std_logic;
        signal ddrclk0: std_logic;
        signal ddrclk90: std_logic;
        signal vcc_net : std_logic;
        signal gnd_net: std_logic;

attribute syn_useioff : boolean;
attribute syn_useioff of structure : architecture is false;

begin
        vcc_net <= '1';
        gnd_net <= '0';


        -- parameterized module component instance
        I0 : pll90
        port map (CLK=>ddrclk, RESET=>rst, CLKOP=>ddrclk0, CLKOS=>ddrclk90, LOCK=>clklock);

demux: process (rst, ddrclk90)
begin
        if  rst = '1' then
              pos0    <= (others => '0');
              neg0    <= (others => '0');
              pos1    <= (others => '0');
        elsif rising_edge(ddrclk90) then
              pos0    <= ddrdata;
```

```vhdl
       elsif falling_edge(ddrclk90) then
          neg0  <=ddrdata;
         pos1  <=pos0;
       end if;
    end  process demux;

    synch: process (rst, ddrclk90)
    begin
        if  rst = '1' then
           datap  <= (others => '0');
           datan  <= (others => '0');
        elsif rising_edge(ddrclk90) then
           datap<= pos1;
        elsif falling_edge(ddrclk90) then
              datan<= neg0;
        end if;
    end  process synch;

end structure;
```

Verilog Example

```
module ddrin (rst, ddrclk, ddrdata, datap, datan)/*synthesis syn_useioff = 0*/;
// Inputs
input           rst;
input           ddrclk;
input   [7:0]   ddrdata;
// Outputs
output  [7:0]   datap, datan;


reg [7:0] pos0/*synthesis syn_keep=1*/;
reg [7:0] pos1/*synthesis syn_keep=1*/;
reg [7:0] neg0/*synthesis syn_keep=1*/;
reg [7:0] datap, datan/*synthesis syn_keep=1*/;

//PLL signals
wire ddrclk0;
wire ddrclk90;

pll I0 (.CLK(ddrclk), .RESET(rst), .CLKOP(ddrclk0), .CLKOS(ddrclk90), .LOCK(clklock));

always @ ( posedge ddrclk90)
begin
    if (rst)
    begin
    pos0 <= 0;
    end
    else
    begin
        pos0 <= ddrdata;
    end
end

always@ (negedge ddrclk90)
begin
    if (rst)
    begin
        neg0<=0;
        pos1<=0;
    end
    else
    begin
        neg0<=ddrdata;
        pos1<=pos0;
    end
end
always @ (posedge ddrclk90)
begin
    if (rst)
    begin
        datap<= 0;
        datan<= 0;
    end
    else
    begin
        datap<= pos1;
        datan<= neg0;
    end
```

```
end
endmodule
```

Preference File

In order to run the above DDR PFU Implementation at 300MHZ, the following preferences were added to the software preference file.

```
COMMERCIAL;
FREQUENCY NET "ddrclk90" 300.000000 MHz ;
INPUT_SETUP PORT "ddrdata_0" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_1" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_2" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_3" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_4" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_5" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_6" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_7" 0.800000 ns CLKNET "ddrclk90" ;
BLOCK ASYNCPATHS ;
```

Appendix E. List of Compatible DDR SDRAM

Below are the criteria used to list the DDR SDRAM part numbers.

1. The memory device should support one DQS strobe for every eight DQ data bits.
2. 4-bit, 8-bit and 16-bit configurations. For 16-bit configurations, each data byte must have an independent DQS strobe.
3. The memory device uses SSTL2 I/O interface standard.
4. Data transfer rate DDR400, DDR333 or DDR266 for LatticeECP/EC devices and DDR333 or DDR266 for LatticeXP devices.
5. Clock transfer rate of 200MHz, 167MHz or 133MHz for LatticeECP/EC devices and 167MHz or 133MHz for LatticeXP devices.

Table 10-9 lists the DDR SDRAM part numbers that can be used with the LatticeECP/EC and LatticeXP devices.

Please note these part numbers are chosen based on the criteria stated above and have not necessary been validated in hardware.

Table 10-9. List of Compatible DDR SDRAM

DDR SDRAM Vendor	Part Number	Configuration	Max Data Rate	Clock Speed
Micron 128MB	MT46V32M4TG	32Mx4	DDR266	133MHz
Micron 128MB	MT46V16M8TG	16Mx8	DDR333 DDR266	167MHz 133MHz
Micron 128MB	MT46V8M16TG	8Mx16	DDR266	133MHz
Micron 256MB	MT46V64M4FG	64Mx4	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 256MB	MT46V64M4TG	64Mx4	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 256MB	MT46V32M8FG	32Mx8	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 256MB	MT46V32M8TG	32Mx8	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 256MB	MT46V16M16FG	16Mx16	DDR266	133MHz
Micron 256MB	MT46V16M16TG	16Mx16	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 512MB	MT46V128M4FN	128Mx4	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 512MB	MT46V128M4TG	128Mx4	DDR266	133MHz
Micron 512MB	MT46V64M8FN	64Mx8	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 512MB	MT46V64M8TG	64Mx8	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 512MB	MT46V32M16FN	32Mx16	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz

Table 10-9. List of Compatible DDR SDRAM (Continued)

DDR SDRAM Vendor	Part Number	Configuration	Max Data Rate	Clock Speed
Micron 512MB	MT46V32M16TG	32Mx16	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 1GB	MT46V256M4TG	256Mx4	DDR266	133MHz
Micron 1GB	MT46V128M8TG	128Mx8	DDR266	133MHz
Micron 1GB	MT46V64M16TG	64Mx16	DDR333 DDR266	167MHz 133MHz
Samsung 128MB E die	K4H280438E-TC/LB3	32Mx4	DDR333	167MHz
	K4H280838E-TC/LB3	16Mx8	DDR333	167MHz
	K4H281638E-TC/LB3	8Mx16	DDR333	167MHz
Samsung 256 Mb E-die	K4H560438E-TC/LB3	64Mx4	DDR333	167MHz
	K4H560438E-NC/LB3	64Mx4	DDR333	167MHz
	K4H560438E-GC/LB3, CC	64Mx4	DDR333 DDR400	167MHz 200MHz
	K4H560838E-TC/LB3, CC	32Mx8	DDR333 DDR400	167MHz 200MHz
	K4H560838E-NC/LB3, CC	32Mx8	DDR333 DDR400	167MHz 200MHz
	K4H560838E-GC/LB3, CC	32Mx8	DDR333 DDR400	167MHz 200MHz
Samsung 512Mb B die	K4H510838B-TC/LB3, CC	64Mx8	DDR333 DDR400	167MHz 200MHz
	K4H510838B-NC/LB3, CC	64Mx8	DDR333 DDR400	167MHz 200MHz
	K4H511638B-TC/LB3, CC	32Mx16	DDR333 DDR400	167MHz 200MHz
	K4H510438B-TC/LB3	128Mx4	DDR333	167MHz
	K4H510438B-NC/LB3	128Mx4	DDR333	167MHz
Infineon 128Mb	HYB25D128400AT	32Mx4	DDR266	133MHz
	HYB25D128400CT	32Mx4	DDR266	133MHz
	HYB25D128400CE	32Mx4	DDR266	133MHz
	HYB25D128800AT	16Mx8	DDR266	133MHz
	HYB25D128800CT	16Mx8	DDR333	167MHz
	HYB25D128800CE	16Mx8	DDR333	167MHz
	HYB25D128160AT	8Mx16	DDR333 DDR266	167MHz 133MHz
	HYB25D128160CT	8Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D128160CE	8Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D128160CC	8Mx16	DDR333	167MHz

Table 10-9. List of Compatible DDR SDRAM (Continued)

DDR SDRAM Vendor	Part Number	Configuration	Max Data Rate	Clock Speed
Infineon 256Mb	HYB25D256400BT	64Mx4	DDR266	133MHz
	HYB25D256400CT	64Mx4	DDR266	133MHz
	HYB25D256400CE	64Mx4	DDR266	133MHz
	HYB25D256800BT	32Mx8	DDR333	167MHz
	HYB25D256800CT	32Mx8	DDR333	167MHz
	HYB25D256800CE	32Mx8	DDR333 DDR400	167MHz 200MHz
	HYB25D256160BT	16Mx16	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
	HYB25D256160CT	16Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D256160CE	16Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D256400BC	64Mx16	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
	HYB25D256400CF	64Mx16	DDR333	167MHz
	HYB25D256400CC	64Mx16	DDR333	167MHz
	HYB25D256160BC	16Mx16	DDR333 DDR266	167MHz 133MHz
	HYB25D256160CC	16Mx16	DDR333 DDR400	167MHz 200MHz
Infineon 512Mb	HYB25D512400AT	128Mx4	DDR266	133MHz
	HYB25D512400BT	128Mx4	DDR333	167MHz
	HYB25D512400BE	128Mx4	DDR333	167MHz
	HYB25D1G400BG	256Mx4	DDR266	133MHz
	HYB25D512800AT	64Mx8	DDR333 DDR266	167MHz 133MHz
	HYB25D512800BT	64Mx8	DDR333 DDR400	167MHz 200MHz
	HYB25D512800BE	64Mx8	DDR333 DDR400	167MHz 200MHz
	HYB25D512160AT	32Mx16	DDR333 DDR266	167MHz 133MHz
	HYB25D512160BT	32Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D512160BE	32Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D512400BC	128Mx4	DDR333 DDR400	167MHz 200MHz
	HYB25D512400BF	128Mx4	DDR333 DDR400	167MHz 200MHz
	HYB25D512800BC	64Mx8	DDR333	167MHz
	HYB25D512800BF	64Mx8	DDR333 DDR400	167MHz 200MHz
	HYB25D512160BC	32Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D512160BF	32Mx16	DDR333 DDR400	167MHz 200MHz

Appendix F. DDR400 Interface using the LatticeEC Evaluation Board

The DDR400 interface was implemented using the LatticeEC20 device on the LatticeEC Advanced Evaluation Board. Figures 10-21, 10-22 and 10-23 show the READ, WRITE and WRITE to READ transition operations running at 200MHz. For more information on the evaluation board, refer to *LatticeEC Advanced Evaluation Board User's Guide* available on the Lattice web site at www.latticesemi.com.

Figure 10-21. READ Function Running at 200MHz
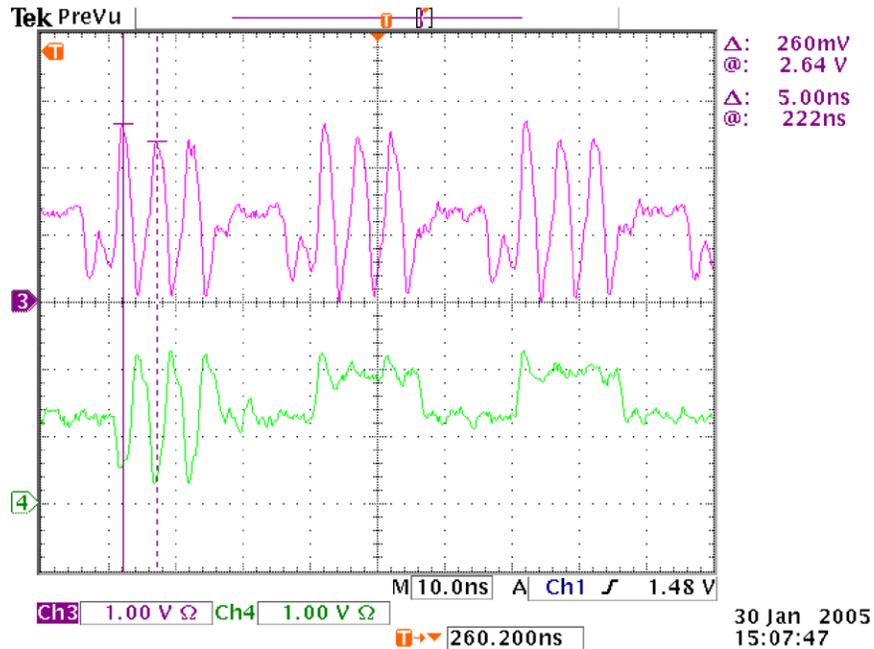
Note: An extra READ command is implemented in the LatticeEC20 device to protect the data during postamble. This extra READ is not required for other LatticeEC devices. Refer to the DQS Postamble section of this document for more information.

Figure 10-22. WRITE Function Running at 200MHz
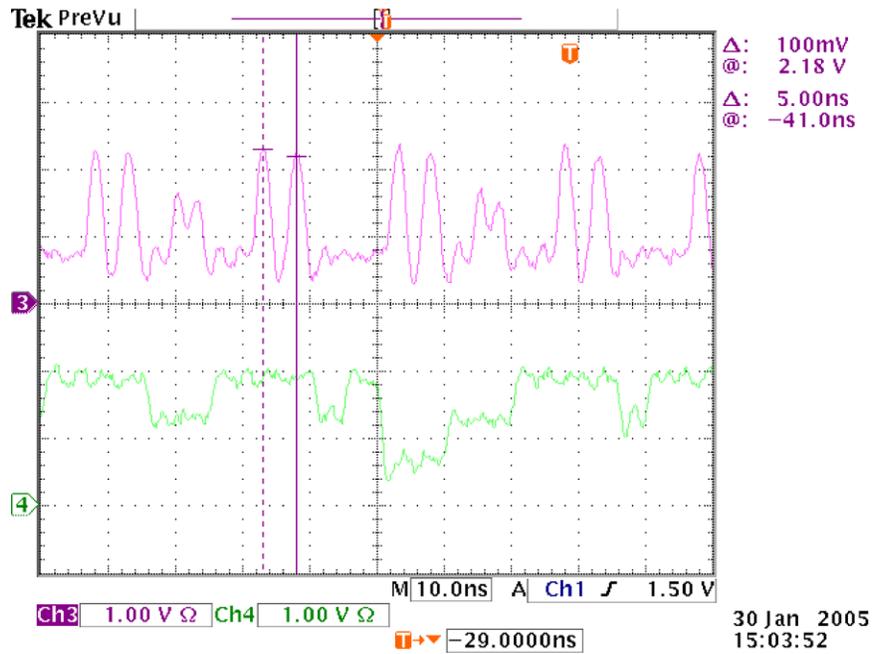
Figure 10-23. WRITE to READ Transition Running at 200MHz
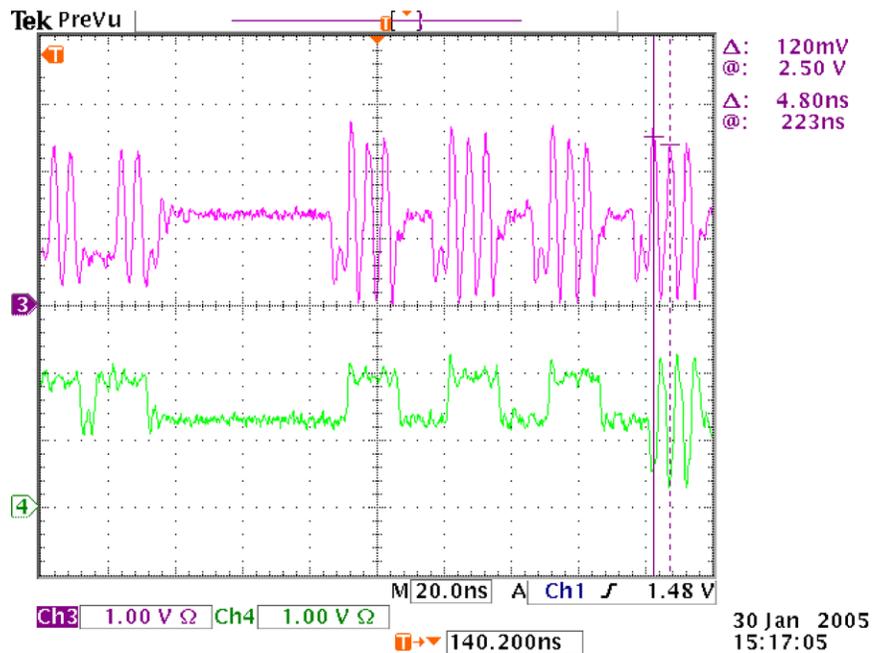
Note: An extra READ command is implemented in the LatticeEC20 device to protect the data during postamble. This extra READ is not required for other LatticeEC devices. Refer to the DQS Postamble section of this document for more information.

LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide

Introduction

As clock distribution and clock skew management become critical factors in overall system performance, the Phase Locked Loop (PLL) is increasing in importance for digital designers. Lattice incorporates its sysCLOCK™ PLL technology in the LatticeECP™, LatticeEC™ and LatticeXP™ device families to help designers manage clocks within their designs. The PLL components in the LatticeECP/EC and LatticeXP device families share the same architecture. This technical note describes the features and functionalities of the PLLs and their configuration in the isp-LEVER® design tool. Figure 11-1 shows the block diagram of the PLL.

Figure 11-1. LatticeECP/EC and LatticeXP sysCLOCK PLL Block Diagram
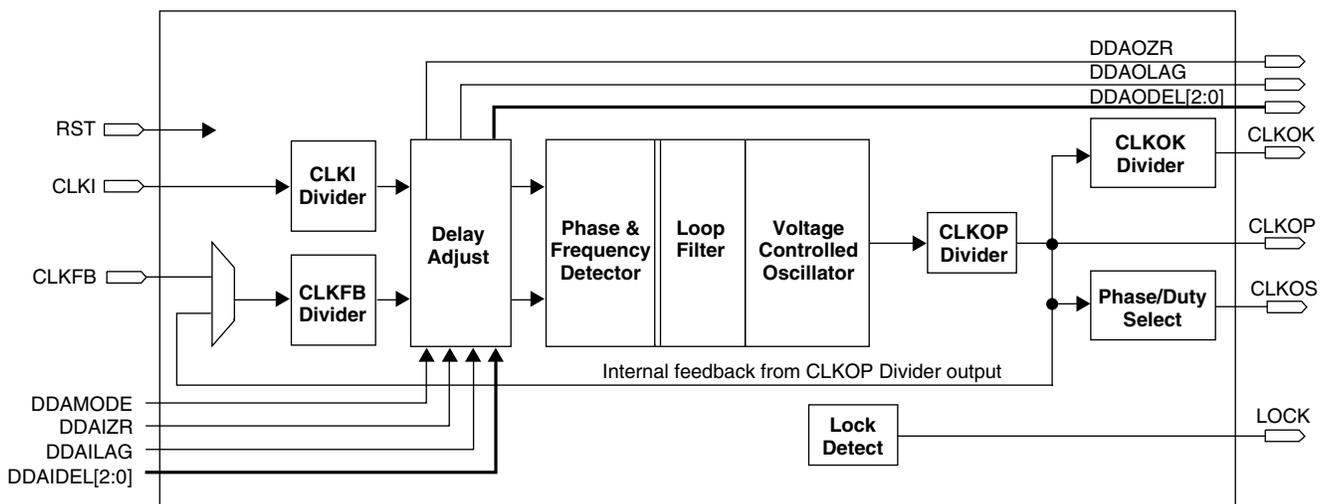
Features

- Clock synthesis
- Phase shift/duty cycle selection
- Internal and external feedback
- Dynamic delay adjustment
- No external components required
- Lock detect output

Functional Description

PLL Divider and Delay Blocks

Input Clock (CLKI) Divider
The CLKI divider is used to control the input clock frequency into the PLL block. It can be set to an integer value of 1 to 16. The divider setting directly corresponds to the divisor of the output clock. The input and output of the input divider must be within the input and output frequency ranges specified in the device data sheet.

Feedback Loop (CLKFB) Divider
The CLKFB divider is used to divide the feedback signal. Effectively, this multiplies the output clock, because the divided feedback must speed up to match the input frequency into the PLL block. The PLL block increases the output frequency until the divided feedback frequency equals the input frequency. Like the input divider, the feedback

loop divider can be set to an integer value of 1 to 16. The input and output of the feedback divider must be within the input and output frequency ranges specified in the device data sheet.

Delay Adjustment

The delay adjust circuit provides programmable clock delay. The programmable clock delay allows for step delays in increments of 250ps (nominal) for a total of 2.00ns lagging or leading. The time delay setting has a tolerance. See device data sheet for details. Under this mode, CLKOP, CLKOS and CLKOK are identically affected. The delay adjustment has two modes of operation:

- **Static Delay Adjustment** – In this mode, the user-selected delay is configured at power-up.
- **Dynamic Delay Adjustment (DDA)** – In this mode, a simple bus is used to configure the delay. The bus signals are available to the general purpose FPGA.

Output Clock (CLKOP) Divider

The CLKOP divider serves the dual purposes of squaring the duty cycle of the VCO output and scaling up the VCO frequency into the 420MHz to 840MHz range to minimize jitter. Refer to Table 11-3 for CLKOP Divider value.

CLKOK Divider

The CLKOK divider feeds the global clock net. It divides the CLKOP signal of the PLL by the value of the divider. It can be set to values of 2, 4, 6,....126,128.

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 can be derived from a dedicated dual-purpose pin or from routing.

RST Input

The PLL reset occurs under two conditions. At power-up an internal power-up reset signal from the configuration block resets the PLL. The user controlled PLL reset signal RST is provided as part of the PLL module that can be driven by an internally generated reset function or a pin. This RST signal resets all internal PLL counters. When RST goes inactive, the PLL will start the lock-in process, and will take the t_{LOCK} time to complete the PLL lock.

Note: For LatticeECP/EC, RST must be asserted to re-start the locking process after losing lock. Refer to the LatticeECP/EC Family Data Sheet for the RST pulse width requirement. For LatticeXP, RST may be tied to GND.

Figure 11-2 shows the timing diagram of the RST Input.

Figure 11-2. RST Input Timing Diagram
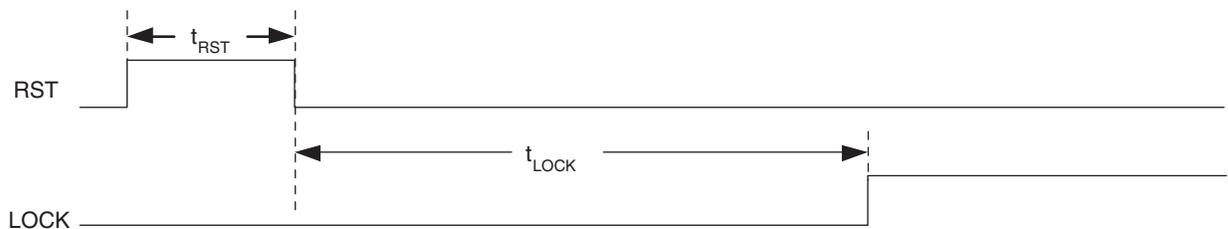
CLKFBK Input

The feedback signal to the PLL, which is fed through the feedback divider can be derived from the global clock net, a dedicated dual-purpose pin, or directly from the CLKOP divider. Feedback must be supplied in order for the PLL to synchronize the input and output clocks. External feedback allows the designer to compensate for board-level clock alignment.

CLKOP Output

The sysCLOCK PLL main clock output, CLKOP, is a signal available for selection as a primary clock.

CLKOS Output with Phase and Duty Cycle Select
The sysCLOCK PLL auxiliary clock output, CLKOS, is a signal available for selection as a primary clock. The CLKOS is used when phase shift and/or duty cycle adjustment is desired. The programmable phase shift allows for different phase in increments of 45° to 315°. The duty select feature provides duty select in 1/8th of the clock period.

CLKOK Output with Lower Frequency
The CLKOK is used when a lower frequency is desired. It is a signal available for selection as a primary clock.

Dynamic Delay Control I/O Ports
Refer to Table 11-1 and Table 11-3 for detailed information.

LOCK Output
The LOCK output provides information about the status of the PLL. After the device is powered up and the input clock is valid, the PLL will achieve lock within the specified lock time. Once lock is achieved, the PLL lock signal will be asserted. If, during operation, the input clock or feedback signals to the PLL become invalid, the PLL will lose lock. The LOCK signal is available to the FPGA routing to implement generation of RST.

Note: For LatticeECP/EC, RST must be asserted to restart the locking process after losing lock. Refer to the LatticeECP/EC Family Data Sheet for the RST pulse width requirement. For LatticeXP, RST may be tied to GND.

PLL Attributes

The PLL utilizes several attributes that allow the configuration of the PLL through source constraints. The following section details these attributes and their usage.

FIN
The input frequency can be any value within the specified frequency range based on the divider settings.

CLKI_DIV, CLKFB_DIV, CLKOP_DIV, CLKOK_DIV
These dividers determine the output frequencies of each output clock. The user is not allowed to input an invalid combination; determined by the input frequency, the dividers, and the PLL specifications.

Frequency_Pin_CLKI, Frequency_Pin_CLKOP and Frequency_Pin_CLKOK
These output clock frequencies determine the divider values.

FDEL
The FDEL attribute is used to pass the Delay Adjustment step associated with the Output Clock of the PLL. This allows the user to advance or retard the Output Clock by the step value passed multiplied by 250ps(nominal). The step ranges from -8 to +8 resulting the total delay range to +/- 2ns.

PHASEADJ
The PHASEADJ attribute is used to select Coarse Phase Shift for CLKOS output. The phase adjustment is programmable in 45° increments.

DUTY
The DUTY attribute is used to select the Duty Cycle for CLKOS output. The Duty Cycle is programmable at 1/8 of the period increment.

FB_MODE
There are three sources of feedback signals that can drive the CLKFB Divider: internal, clocktree and external feedback. Clocktree feedback is used by default. Internal feedback takes the CLKOP output at CLKOP Divider output before the Clocktree to minimize the feedback path delay. The external feedback is driven from the pin.

DELAY_CNTL

This attribute is designed to select the Delay Adjustment mode. If the attribute is set to "DYNAMIC" the delay control switches between Dynamic and Static depending upon the input logic of DDAMODE pin. If the attribute is set to "STATIC", Dynamic Delay inputs are ignored in this mode.

LatticeECP/EC and LatticeXP PLL Primitive Definitions

The PLL primitive name is EHXPLLB. Figure 11-3 shows the LatticeECP/EC and LatticeXP PLL primitive library symbol. Some features and I/Os are optional as described in Table 11-1 and Table 11-2.

Figure 11-3. LatticeECP/EC and LatticeXP PLL Primitive Symbol

```
                        EHXPLLB
          ┌──────────────────────────────┐
    ──────┤ RST                     CLKOP ├──────
    ──────┤ CLKI                    CLKOS ├──────
    ──────┤ CLKFB                   CLKOK ├──────
    ──────┤ DDAMODE                  LOCK ├──────
    ──────┤ DDAIZR                 DDAOZR ├──────
    ──────┤ DDAILAG               DDAOLAG ├──────
    ──────┤ DDAIDEL2             DDAODEL2 ├──────
    ──────┤ DDAIDEL1             DDAODEL1 ├──────
    ──────┤ DDAIDEL0             DDAODEL0 ├──────
          └──────────────────────────────┘
```
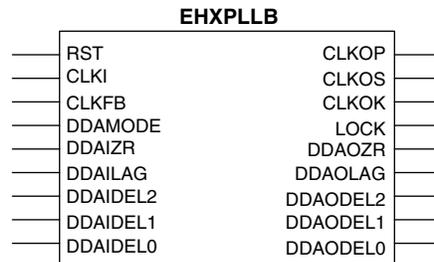
Table 11-1. LatticeECP/EC and LatticeXP PLL I/O Definitions

Signal	I/O	Description	Optional
CLKI	I	PLL reference clock input. From internal logic or dedicated clock pin.	No
CLKFB[1]	I	Feedback clock input. From internal node, CLKOP or dedicated pin.	No
RST	I	"1" to reset PLL	No
CLKOP	O	PLL output clock to clock tree	No
CLKOS	O	PLL output clock to clock tree with optional phase shift/duty cycle	Yes
CLKOK	O	PLL output clock to clock tree through K-divider for lower frequency	Yes
LOCK[2]	O	"1" indicates PLL locked to CLKI	Yes
DDAMODE	I	DDA Mode. "1": Pin Control (dynamic), "0": fuse control (static)	Yes
DDAIZR	I	DDA Delay Zero. "1" delay=0, "0": delay=[DDILAG+DDAIDEL].	Yes
DDAILAG	I	DDA Lag/Lead. "1": Lead, "0": Lag.	Yes
DDAIDEL[2:0]	I	DDA Delay	Yes
DDAOZR	O	DDA Delay Zero Output	Yes
DDAOLAG	O	DDA Lag/Lead Output	Yes
DDAODEL[2:0]	O	DDA Delay Output	Yes

1. When internal feedback or clocktree feedback is selected in the IPexpress™ GUI, software uses CLKOP as the source of CLKFB. CLKOS is not recommended as the source of CLKFB even in external feedback mode.
2. ModelSim® simulation models take two to four clock cycles from RST release to LOCK high.

PLL Attributes Definitions

The EHXPLLB can be configured through attributes in the source code. The following section details these attributes and their usage.

Table 11-2. LatticeECP/EC and LatticeXP PLL Attributes

User Accessible	IPexpress GUI Access	Attribute Name	Preference Language Support	Preference Editor Support	Value	Default Value	Units
CLKI Frequency	Y	FREQUENCY_PIN_CLKI	N	N	Note 5	100	MHz
CLKOP Frequency	Y	FREQUENCY_PIN_CLKOP	N	N	Note 5	100	MHz
CLKOK Frequency	Y	FREQUENCY_PIN_CLKOK	N	N	Note 5	50	MHz
CLKOP Frequency Tolerance	Y		N	N	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	%

Table 11-2. LatticeECP/EC and LatticeXP PLL Attributes (Continued)

User Accessible	IPexpress GUI Access	Attribute Name	Preference Language Support	Preference Editor Support	Value	Default Value	Units
CLKOP Actual Frequency	Y		N	N			MHz
CLKOK Frequency Tolerance	Y		N	N	0.0, 0.1, 0.2, 0.5,1.0, 2.0, 5.0, 10.0	0.0	%
CLKOK Actual Frequency	Y		N	N			MHz
CLKI Divider Setting	Y	CLKI_DIV[4, 6]	Y	N	1 to 16 (1 to 15)	1	
CLKFB Divider Setting	Y	CLKFB_DIV[6]	Y	N	1 to 16 (1 to 15)	1	
CLKOP Divider Setting	Y	CLKOP_DIV[6]	Y	N	Note 3	8^2 (4 or 6)	
CLKOK Divider Setting	Y	CLKOK_DIV	Y	N	2, 4, 6,..,126, 128	2	
Fine Delay Adjust	N	FDEL	Y	Y	-8 to 8	0	ps
Coarse Phase Shift Selection (O)	Y	PHASEADJ	Y	N	0, 45, 90...315	0	Degrees
Duty Cycle Selection (1/8 increment)	Y	DUTY	Y	N	1 to 7	4	
Delay Control	Y	DELAY_CNTL1	Y	N	DYNAMIC/STATIC	STATIC	
Feedback Mode	Y	FB_MODE	N	N	INTERNAL/CLOCKTREE/EXTERNAL	CLOCKTREE	
CLKOS Select	Y		N	N			
CLKOK Select	Y		N	N			

1. DYNAMIC: This mode switches delay control between Dynamic and Static depending upon the input logic of the DDAMODE pin. STATIC: This is Static Control Only mode.
2. The CLKOP_DIV value is calculated to maximize the f_{VCO} within the specified range. For LatticeXP devices, if CLKOS is not used, the default value is 6. If CLKOS is used, the value is 4.
3. The CLKOP Divider values are 2, 4, 6, 8,..32 (2, 4, 6, 8..30 for LatticeXP devices) if CLKOS is not used. The CLKOP Divider values are 2, 4, 8, 16, 32 (2, 4, 8, 16 for LatticeXP devices) if CLKOS is used.
4. All divider settings are user transparent in Frequency Mode. These are user attributes in Divider Mode.
5. Refer to data sheet for frequency limits.
6. Values in parentheses are for LatticeXP devices.
7. This attribute is not available in the IPexpress GUI. After reviewing the trace report file, users can determine the amount of delay that will best fit the clocking in their design. Further information on FDEL settings is described in the following section.

FDEL Settings

There are four ways the user can enter the desired FDEL value.

1. Although the FDEL entry is not available in the IPexpress GUI, the module generated by IPexpress includes the attribute with default value, "0". Users can replace it with a desired value.

   Example of source code with default FDEL value:

   ```
   attribute FDEL of ehxpll_mod_0_0 : label is "0";
   generic map (…
       FDEL=>"0",
       …
       ")
   ```

2. Preference File: User may specify the preference in the Preference file.

   Example:

   ```
   ASIC "FDEL_CODE_0_0" TYPE "EHXPLLB" FDEL="-2"  ;
   ```

3. Pre-Map Preference Editor: Users can enter the FDEL value in the Pre-Map Preference Editor as shown in Figure 11-4.

Figure 11-4. Pre-Map Preference Editor
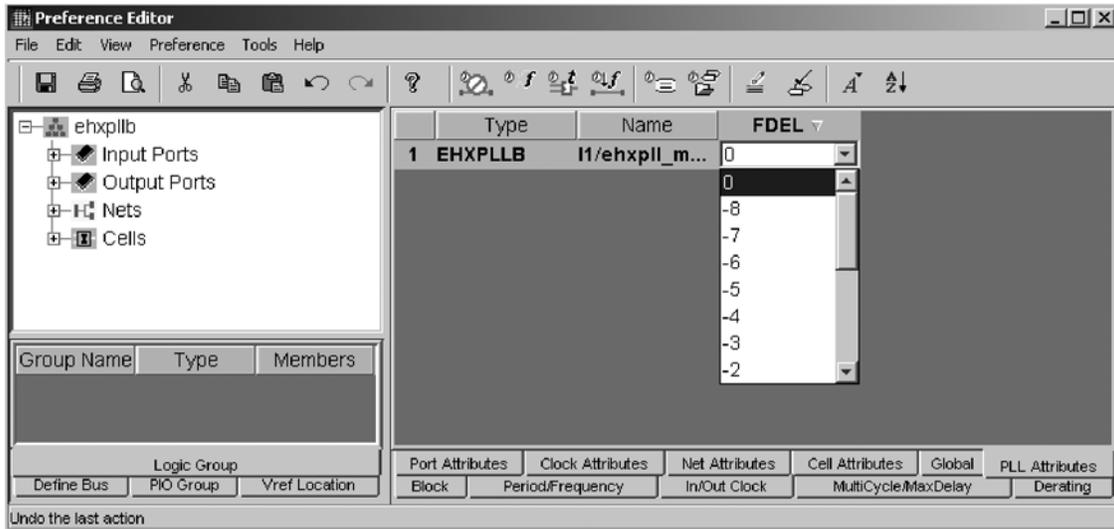
4. EPIC Device Editor: Users can edit their preferences in the EPIC Device Editor as shown in Figure 11-5.

Figure 11-5. EPIC Preferences Edit Window
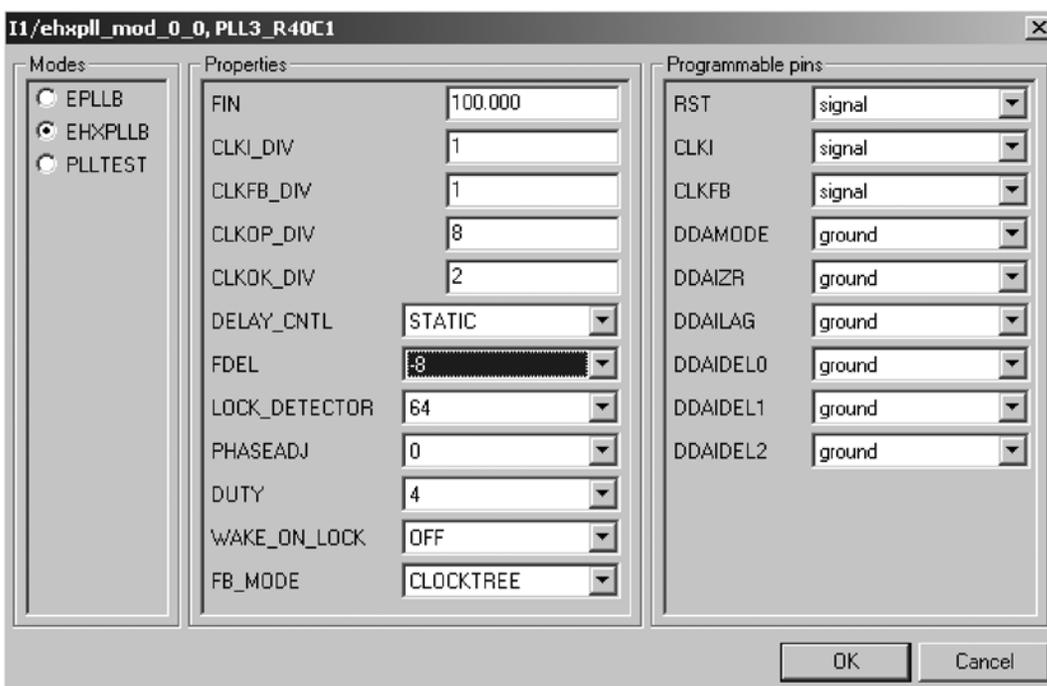
Dynamic Delay Adjustment

The Dynamic Delay Adjustment is controlled by the DDAMODE input. When the DDAMODE input is set to "1", the delay control is handled through the inputs, DDAIZR, DDAILAG and DDAIDEL(2:0). For this mode, the attribute "DELAY_CNTL" must be set to "DYNAMIC". Table 11-3 shows the delay adjustment values based on the attribute/input settings.

In this mode, the PLL may come out of lock due to the abrupt change of phase. RST must be asserted to re-lock the PLL. Upon de-assertion of RST, the PLL will start the lock-in process and will take the t_{LOCK} time to complete the PLL lock.

Table 11-3. Delay Adjustment

DDAMODE = 1: Dynamic Delay Adjustment			DELAY 1 t_{DLY} = 250ps (nominal)	DDAMODE = 0
DDAIZR	**DDAILAG**	**DDAIDEL[2:0]**		**Equivalent FDEL Value**
0	1	111	Lead 8 t_{DLY}	-8
0	1	110	Lead 7 t_{DLY}	-7
0	1	101	Lead 6 t_{DLY}	-6
0	1	100	Lead 5 t_{DLY}	-5
0	1	011	Lead 4 t_{DLY}	-4
0	1	010	Lead 3 t_{DLY}	-3
0	1	001	Lead 2 t_{DLY}	-2
0	1	000	Lead 1 t_{DLY}	-1
1	Don't Care	Don't Care	No delay	0
0	0	000	Lag 1 t_{DLY}	1
0	0	001	Lag 2 t_{DLY}	2
0	0	010	Lag 3 t_{DLY}	3
0	0	011	Lag 4 t_{DLY}	4
0	0	100	Lag 5 t_{DLY}	5
0	0	101	Lag 6 t_{DLY}	6
0	0	110	Lag 7 t_{DLY}	7
0	0	111	Lag 8 t_{DLY}	8

Note: t_{DLY} = Unit Delay Time = 250 ps (nominal). See the data sheet for the tolerance of this delay

PLL Usage in IPexpress

Including sysCLOCK PLLs in a Design

The sysCLOCK PLL capability can be accessed through the IPexpress GUI. The following section describes the usage of IPexpress.

IPexpress Usage

The LatticeECP/EC and LatticeXP PLL is fully supported in IPexpress in the ispLEVER software. IPexpress allows the user to define the desired PLL using a simple, easy-to-use GUI. Following definition, a VHDL or Verilog module that instantiates the desired PLL is created. This module can be included directly in the user's design.

Figure 11-6 shows the main window when PLL is selected. The only entry required in this window is the module name. After entering the module name, clicking on "Customize" will open the "Configuration" window as shown in Figure 11-7.

Figure 11-6. IPexpress Main Window
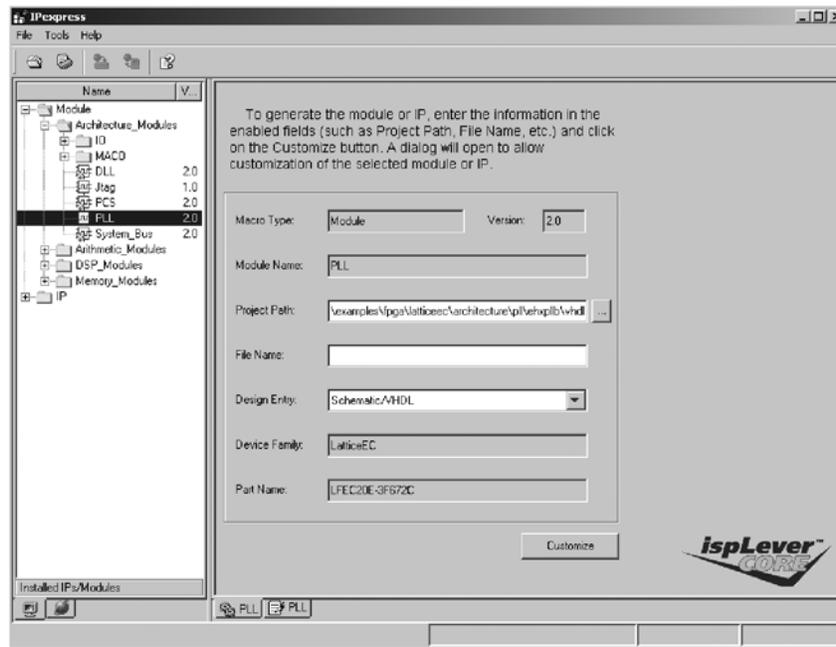
Configuration Tab

The Configuration Tab lists all user accessible attributes. Default values are set initially.

There are two modes in the Configuration Tab which can be used to configure the PLL, Frequency Mode and Divider Mode.

Frequency Mode: In this mode, the user enters input and output clock frequencies and the software calculates the divider settings for the user. If the output frequency the user entered is not achievable, the nearest frequency will be displayed in the 'Actual' text box. After input and output frequencies are entered, clicking the 'Calculate' button will display the divider values. If the desired output frequency is not achievable with the given frequency tolerance, the software generates an error. Users may increase the frequency tolerance or change the output frequencies. Figure 11-7 shows the Configuration Tab window.

Figure 11-7. Configuration Tab
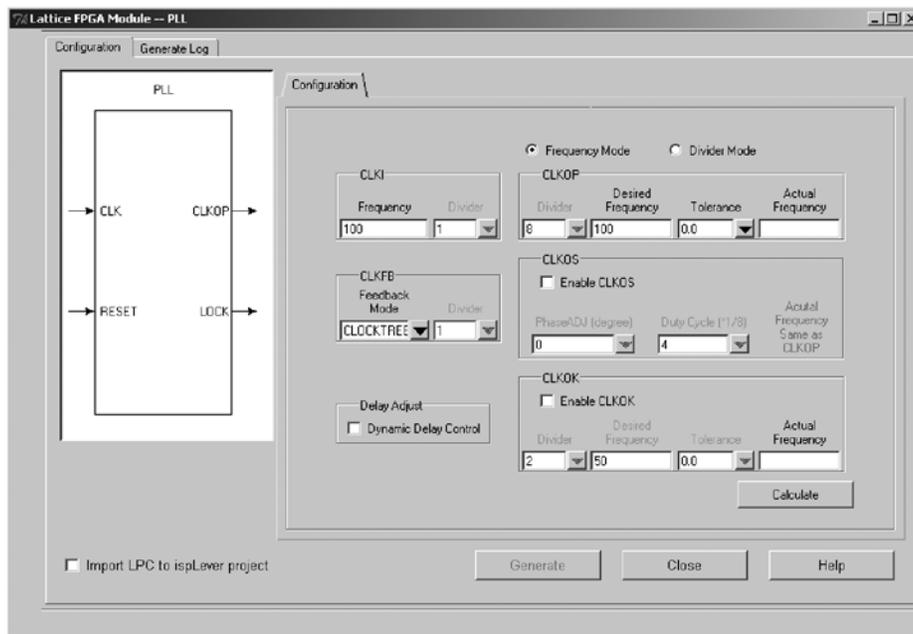
Divider Mode: In this mode, the user sets the input frequency and divider settings. It is assumed the user is familiar with the PLL operation. The user must choose the CLKOP Divider value to maximize the f_{VCO} to achieve optimum PLL performance. After input frequency and divider settings are set, clicking the 'Calculate' button will display the output frequencies. If the divider settings are out of the PLL specification, the software will generate an error.

EHXPLLB Example Projects

ispLEVER provides example PLL projects for first time PLL users.

In the ispLEVER Project Navigator, go to the **File** menu and select **Open Examples...**.

Select the **FPGA** folder. The LatticeEC and LatticeXP folders include PLL example projects in both Verilog and VHDL.

Equations for Generating Input and Output Frequency Ranges

The values of f_{IN}, f_{OUT} and f_{VCO} are the absolute frequency ranges for the PLL. The values of f_{INMIN}, f_{INMAX}, f_{OUTMIN} and f_{OUTMAX} are the calculated frequency ranges based on the divider settings. These calculated frequency ranges become the limits for the specific divider settings used in the design.

Table 11-4. Frequency Limits

Parameter	LatticeECP/EC	LatticeXP
f_{IN}	Note 1	
f_{OUT}	Note 1	
f_{OUTK}	Note 1	
f_{VCO} (Hz)	Note 1	
CLKI Divider	1 to 16	1 to 15
CLKFB Divider	1 to 16	1 to 15
CLKOP Divider	See Table 11-2	
CLKOK Divider	2, 4, 6, 8,.. ,126, 128	
Maximum (N*V)	32	30
f_{PFD} (f_{IN}/M) (Hz)	Note 1	

Note: Refer to data sheet for the latest data.

The divider names are abbreviated with legacy names as:

- CLKI DIVIDER:M
- CLKFB DIVIDER:N
- CLKOP DIVIDER:V
- CLKOK DIVIDER:K

for use in the equations below.

f_{VCO} Constraint

From the loop:
$$f_{OUT} = f_{IN} * (N/M) \tag{1}$$

From the loop:
$$f_{VCO} = f_{OUT} * V \tag{2}$$

Substitute (1) in (2) yields:
$$f_{VCO} = f_{IN} * (N/M) * V \tag{3}$$

Arrange (3):
$$f_{IN} = (f_{VCO} / (V*N))*M \tag{4}$$

From equation (4):
$$f_{INMIN} = ((f_{VCOMIN} /(V*N))*M \tag{5}$$
$$f_{INMAX} = (f_{VCOMAX}/(V*N))*M \tag{6}$$

f_{PFD} Constraint

From the loop:
$$f_{PFD} = f_{IN} / M \tag{7}$$
$$f_{IN} = f_{PFD} * M$$
$$f_{INMIN} = f_{PFDMIN} * M = 25 * M \text{ (assume } f_{PFDMIN} = 25) \tag{8}$$

Equation (5) becomes:

$\mathbf{f_{INMIN} = ((f_{VCOMIN} / (V*N))*M, \text{ if below } 25 * M \text{ round up to } 25 * M}$ (9)

From the loop:
$\mathbf{f_{INMAX} = f_{PFDMAX} * M = 420 * M}$ (10)

Assume $f_{INMAX} = 420$

Equation (6) becomes:
$\mathbf{f_{INMAX} = (f_{VCOMAX} / (V*N))*M, \text{ if above } 420 \text{ round down to } 420}$ (11)

From equation (1):
$\mathbf{f_{OUTMIN} = f_{INMIN} * (N/M), \text{ if below } 25 * N \text{ round up to } 25 * N}$ (12)
$\mathbf{f_{OUTMAX} = f_{INMAX} * (N/M), \text{ if above } 420 \text{ round down to } 420}$ (13)

$f_{OUTKMIN} = f_{OUTMIN} / K$
$f_{OUTKMAX} = f_{OUTMAX} / K$

Clock Distribution in LatticeECP/EC and LatticeXP

The clock inputs are selected from external I/Os, the sysCLOCK PLLs or general routing. These clock inputs are fed through the chip via a clock distribution system.

LatticeECP/EC and LatticeXP devices provide a quadrant-based primary and secondary clock structure.

Primary Clock Sources and Distribution

Each quadrant has four primary clock nets: CLK0, CLK1, CLK2 and CLK3. CLK2 and CLK3 provide dynamic clock selection (DCS) capability. Figure 11-8 illustrates the block diagram of the primary clock distribution.

Figure 11-8. Primary Clocks and Center Switch Boxes
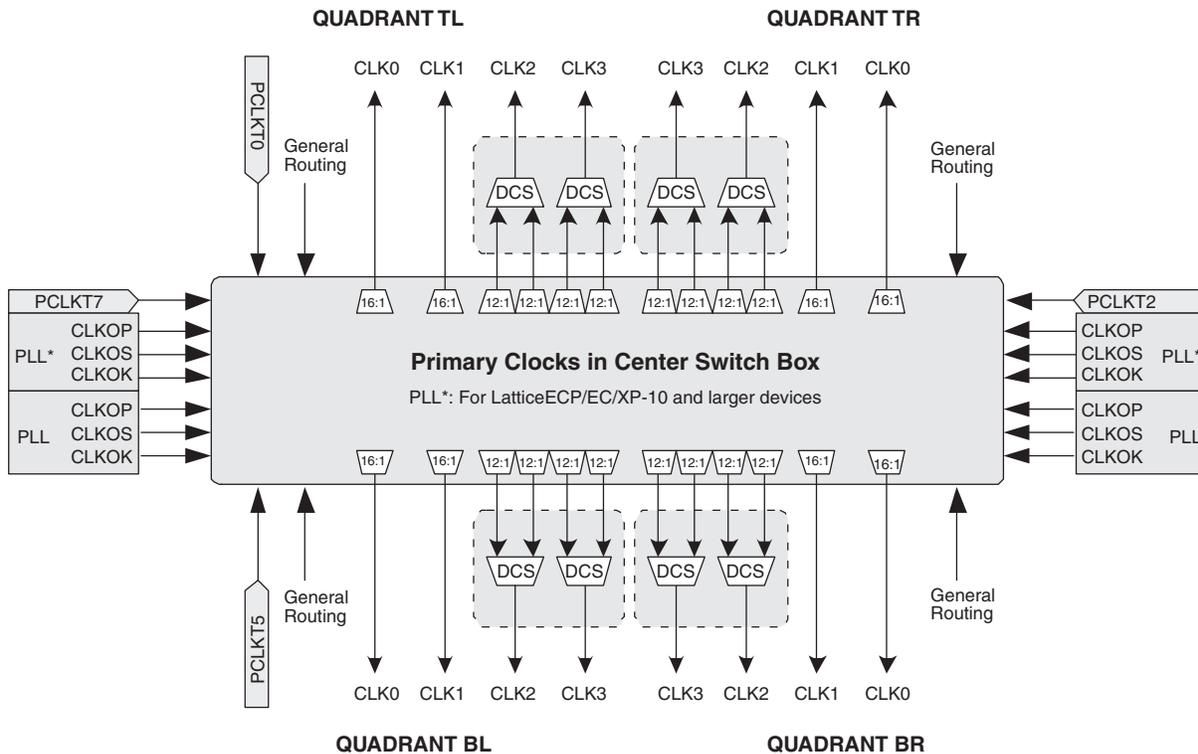
Note: Two PLLs are available in LatticeECP/EC/XP-6 or smaller devices.

Note on the Primary Clock
The CLKOP must be used as the feedback source to optimize the PLL performance.

Most designers use the PLL for the clock tree injection removal mode and the CLKOP should be assigned as the primary clock. This is done automatically by the software unless the user specifies otherwise.

CLKOP can route to CLK0 and CLK1 only. CLKOS/CLKOK can route to all primary clocks (CLK0 to CLK3).

When CLK2 or CLK3 is used as a primary clock and there is only one clock input to the DCS, the DCS is assigned as a buffer mode by the software. Please see the DCS section of this document for further information.

Clock Net Preferences

There are two clock nets, primary clock and secondary clock.

As illustrated in Figure 11-9, users can set each clock to the desired clock net in the Pre-map Preference Editor or write in the Preference File as shown in the examples below.

Primary-Pure and Primary-DCS

Primary Clock Net can be assigned to either Primary-Pure (CLK0 and CLK1) or Primary-DCS (CLK2 and CLK3).

Syntax Example
```
USE PRIMARY DCS NET "bf_clk";
```

Global Primary Clock and Quadrant Primary Clock

Global Primary Clock
If a primary clock is not assigned as a quadrant clock, the software assumes it is a Global Clock.

There are two Global Primary/Pure Clocks and two Global Primary/DCS Clocks available.

Quadrant Primary Clock
Any Primary Clock may be assigned to a Quadrant Clock. The Clock may be assigned to a single quadrant or to two adjacent quadrants (not diagonally adjacent).

When the quadrant clock net is used, users must ensure that the registers each clock drives can be assigned in that quadrant without any routing issues.

With the Quadrant Primary Clocking scheme, the maximum number of Primary Clocks is 16 as long as all the Primary Clock Sources are avaialble.

Syntax Example
```
USE PRIMARY PURE NET "bf_clk" QUADRANT_TL;
```

Figure 11-9. Clock Preferences in the Pre-map Preference Editor
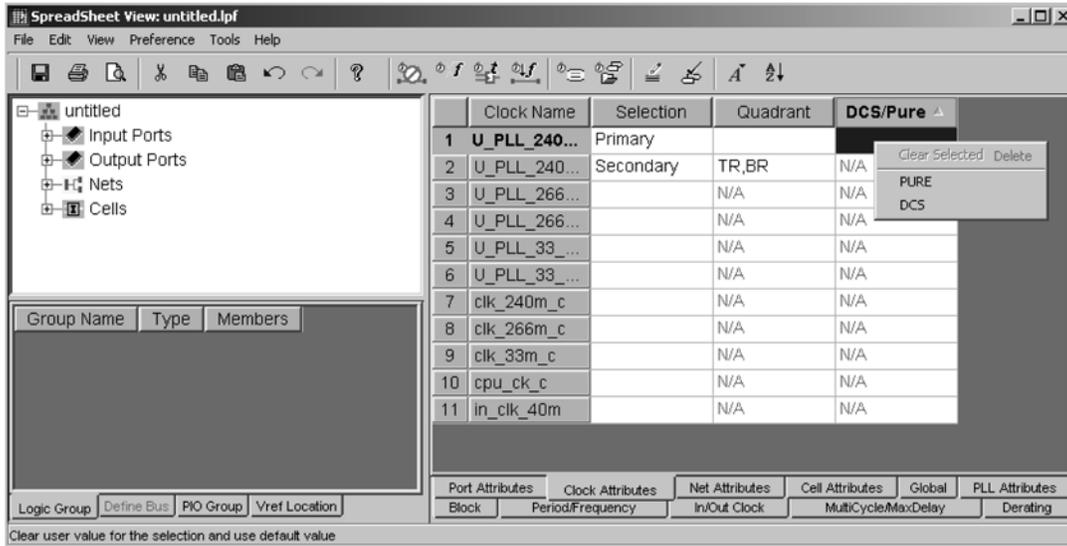
Secondary Clock Sources and Distribution

LatticeECP/EC and LatticeXP devices support quadrant base Secondary Clocks. Figure 11-10 describes the Secondary Clock arrangement.

Figure 11-10. Secondary Clock Center Switch Box
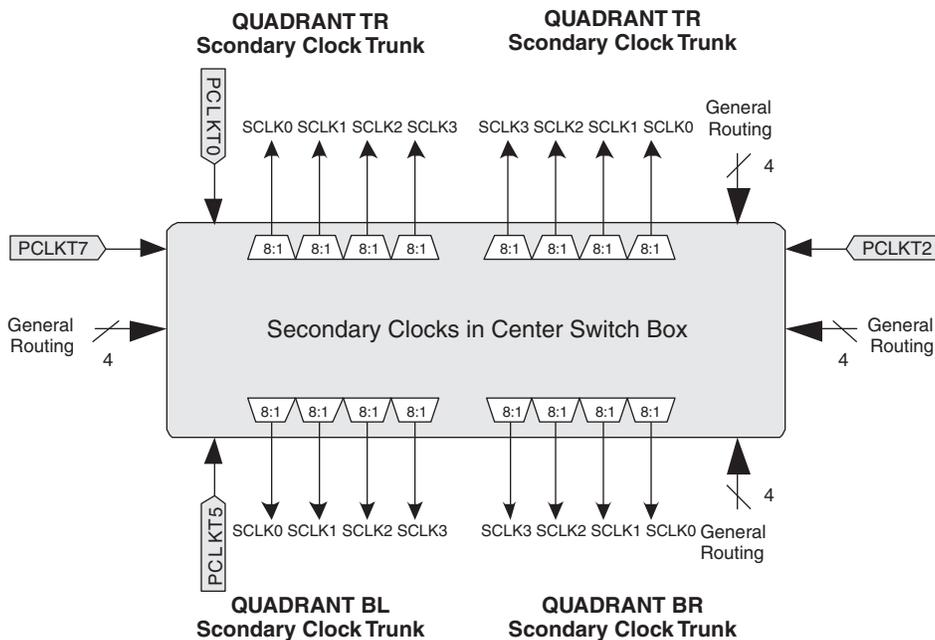
Limitations on Secondary Clock Availability

As illustrated in Figure 11-11, three secondary clocks are shared with CLK, CE and LSR.

This routing scheme limits the secondary clocks available per quadrant base to three, which results in a maximum of 12 available secondary clocks per device. Figure 11-11 illustrates the primary and secondary clock distribution structure of the PFUs.

LFEC6/LFXP6 and smaller devices have limited routing resources and can implement a maximum of nine secondary clocks per device.

Figure 11-11. Primary Clock and Secondary Clock/CE/LSR Distribution
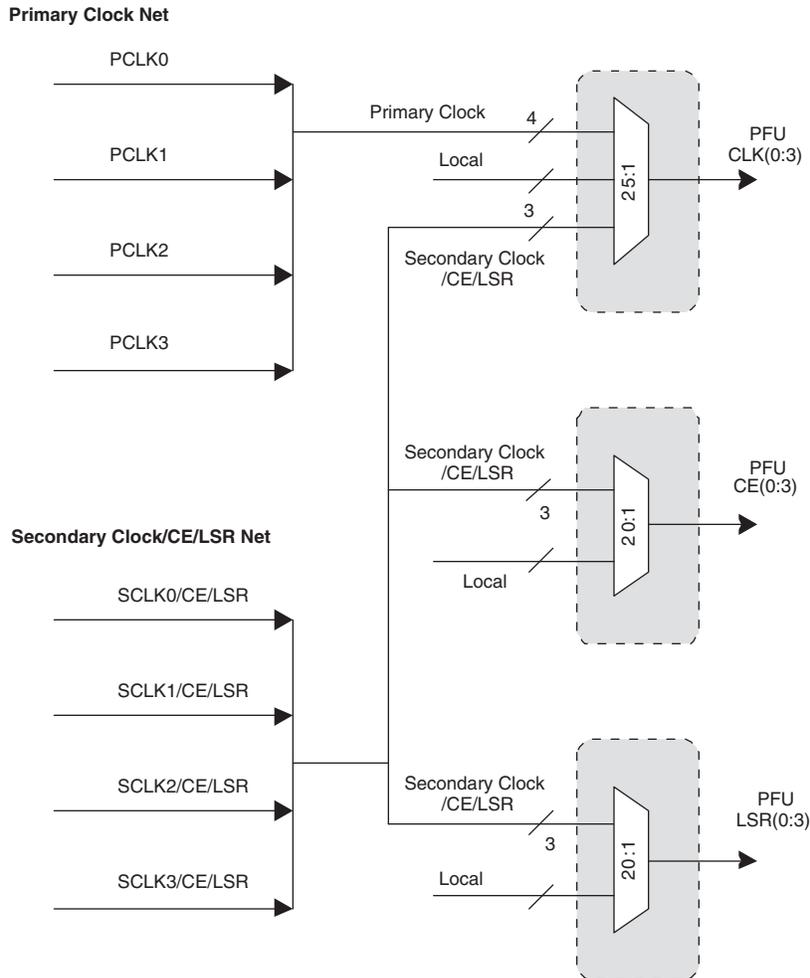
Dynamic Clock Selection (DCS)

DCS is a global clock buffer incorporating a smart multiplexer function that takes two independent input clock sources and avoids glitches or runt pulses on the output clock, regardless of when the enable signal is toggled. The DCS blocks are located in pairs at the center of each side of the device. Thus, there are eight of them in every device.

Table 11-5. DCS I/O

I/O	Name	Description
Input	SEL	Input Clock Select
	CLK0	Primary Clock Input 0
	CLK1	Primary Clock Input 1
Output	DCSOUT	To Primary Clock

Table 11-6. DCS Attributes

Attribute Name	Description	Output		Value
		SEL=0	**SEL=1**	
DCS MODE	Rising edge triggered, latched state is high	CLK0	CLK1	POS (Default)
	Falling edge triggered, latched state is low	CLK0	CLK1	NEG
	Sel is active high, Disabled output is low	0	CLK1	HIGH_LOW
	Sel is active high, Disabled output is high	1	CLK1	HIGH_HIGH
	Sel is active low, Disabled output is low	CLK0	0	LOW_LOW
	Sel is active low, Disabled output is high	CLK0	1	LOW_HIGH
	Buffer for CLK0	CLK0	CLK0	CLK0
	Buffer for CLK1	CLK1	CLK1	CLK1

Figure 11-12. DCS Primitive Symbol
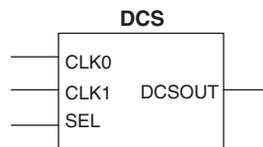
DCS Waveforms

The DCSOUT waveform timing is described in Figure 11-13 for each mode. The 'POS' and 'NEG' modes describe DCSOUT timing at both the falling and rising edges of SEL.

Figure 11-13. DCS Waveforms

DCS MODE = POS
At the rising edge (POS) of SEL, the DCSOUT changes from CLK0 to CLK1. This mode is the default mode.
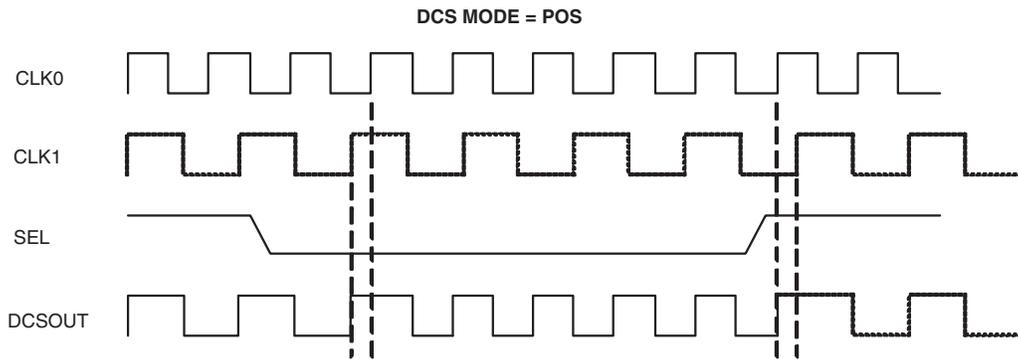
SEL Falling edge:
- Wait for CLK1 rising edge,
  latch output & remain high
- Switch output at CLK0 rising edge

SEL Rising edge:
- Wait for CLK0 rising edge,
  latch output & remain high
- Switch output at CLK1 rising edge

DCS MODE = NEG

At the falling edge (NEG) of SEL, the DCSOUT changes from CLK0 to CLK1.

DCS MODE = NEG

CLK0

CLK1

SEL

DCSOUT
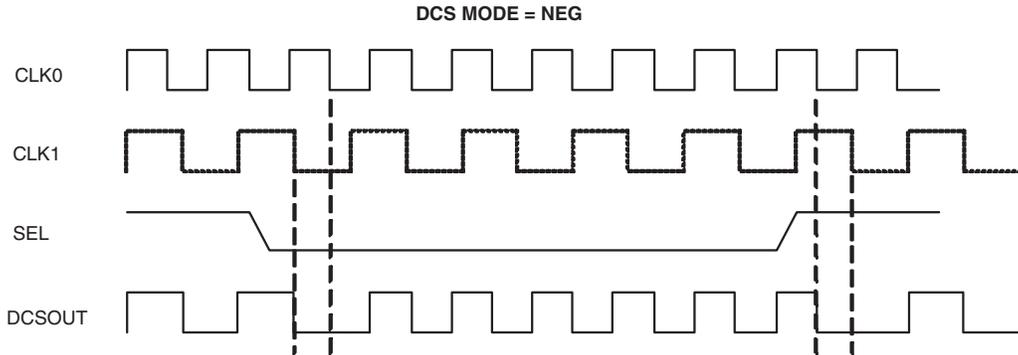
SEL Falling edge:
- Wait for CLK1 falling edge,
  latch output & remain low
- Switch output at CLK0 falling edge

SEL Rising edge:
- Wait for CLK0 falling edge,
  latch output & remain low
- Switch output at CLK1 falling edge

DCS MODE = HIGH_LOW

SEL is active high (HIGH) to select CLK1, and the disabled output is LOW.

DCS MODE = LOW_LOW

SEL is active low (LOW) to select CLK0, and the disabled output is LOW.

DCS MODE = HIGH_LOW

CLK1

SEL

DCSOUT
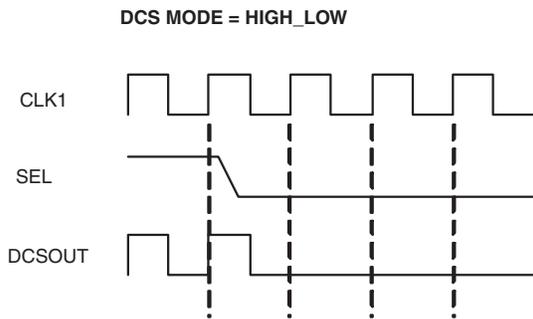
- Switch low at CLK1 falling edge.
- If SEL is low, output stays low at on
  CLK1 rising edge. SEL must not
  change during setup prior to rising clock.

DCS MODE = LOW_LOW

CLK0

SEL

DCSOUT
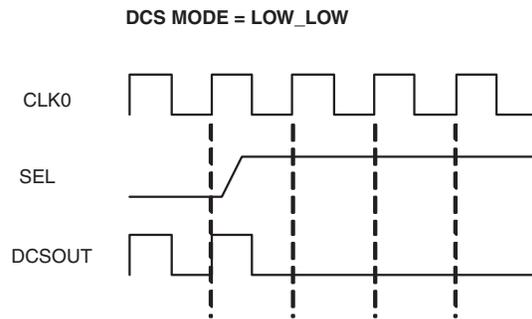
- Switch low at CLK0 falling edge.
- If SEL is high, output stays low at
  on CLK0 rising edge.

DCS MODE = HIGH_HIGH

SEL is active high (HIGH) to select CLK1, and the disabled output is HIGH.

DCS MODE = LOW_HIGH

SEL is active low (LOW) to select CLK0, and the disabled output is HIGH.

DCS MODE = HIGH_HIGH   DCS MODE = LOW_HIGH

CLK1   CLK0

SEL   SEL

DCSOUT   DCSOUT

- Switch high at CLK1 rising edge.   - Switch high at CLK0 rising edge.
- If SEL is low, output stays low high   - If SEL is high, output stays high on
  on CLK1 falling edge.   CLK0 falling edge.
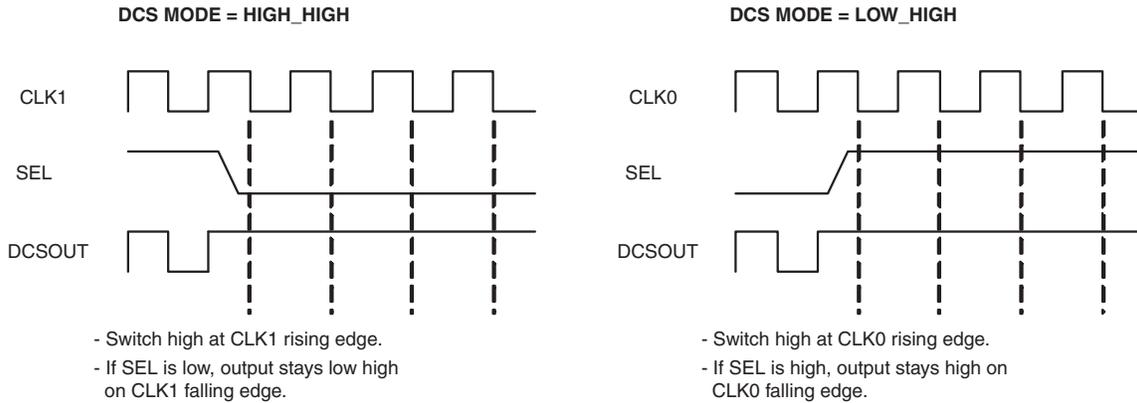
Use of DCS with PLL

The four PLL CLKOP sources reach CLK0 and CLK1 of the quadrant clock. When using the DCS, the PLL needs a free-running feedback path to keep the PLL in lock. The user should use CLKOP as this feedback path, and CLKOS as the input into the DCS. CLKOP does not reach CLK2 or CLK3 to prevent the user from using the PLL improperly with DCS. See Figure 11-14.

Figure 11-14. Implementation of Dynamic Clock Select for a PLL Clock (Must Use Both CLKOP and CLKOS)
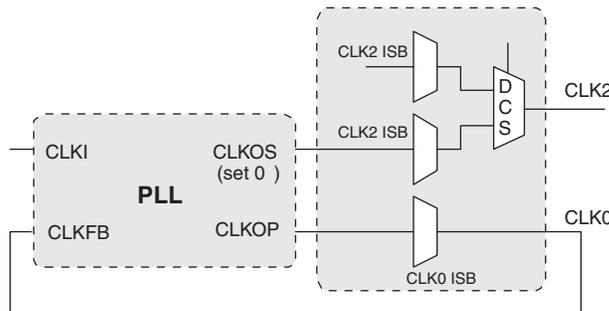
Other Design Considerations

Jitter Considerations

The Clock Output jitter specifications assume that the reference clock is free of jitter. Even if the clock source is clean, there are a number of sources that place noise in the PLL clock input. While intrinsic jitter is not avoidable, there are ways to minimize the input jitter and output jitter.

Signal inputs that share the same I/O bank with PLL clock inputs are preferably less noisy inputs and slower switching signals. Try to avoid placing any high speed and noisy signals in the same I/O bank with clock signals if possible. Use differential signaling if possible.

When external feedback is used, the PCB path must be well designed to avoid reflection as well as noise coupling from adjacent signal sources. A shorter PCB feedback path length does not necessarily reduce feedback input jitter.

Simulation Limitations

- Simulation does not compensate for external delays and dividers in the feedback loop.
- The LOCK signal is not simulated according to the t_{LOCK} specification. The LOCK signal will appear active shortly after the simulation begins, but will remain active throughout the simulation.
- The jitter specifications are not included.

PCB Layout Recommendations for VCCPLL and GNDPLL if Separate Pins are Available

It is best to connect VCCPLL to VCC at a single point using a filter and to create a separate GNDPLL plane directly under it (tied via a single point to GND).

Separate islands for both VCCPLL and GNDPLL are recommended if applicable.

DCS Usage with Verilog

```
module dcs(clk0,clk1,sel,dcsout);

input clk0, clk1, sel;
output dcsout;

DCS DCSInst0 (.SEL(sel),.CLK0(clk0),.CLK1(clk1),.DCSOUT(dcsout));
defparam DCSInst0.DCSMODE = "CLK0";

endmodule
```

DCS Usage with VHDL

```
COMPONENT DCS
-- synthesis translate_off
        GENERIC     (
                    DCSMODE : string :=  "POS"
                    );
-- synthesis translate_on

        PORT        (
                    CLK0            :IN    std_logic;
                    CLK1            :IN    std_logic;
                    SEL             :IN    std_logic;
                    DCSOUT          :OUT   std_logic
                    );
END COMPONENT;

    attribute DCSMODE : string;
    attribute DCSMODE of DCSinst0 : label is "POS";

begin

DCSInst0: DCS
-- synthesis translate_off

        GENERIC MAP(
                    DCSMODE  => "POS"
                    )
-- synthesis translate_on

        PORT MAP    (
                    SEL             =>  clksel,
                    CLK0            =>  dcsclk0,
                    CLK1            =>  sysclk1,
                    DCSOUT          =>  dcsclk
);
```

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

             +1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
June 2004	01.0	Initial release.
October 2004	01.1	RST description with timing diagram added.
		Primitive 'Wake_On_Lock' removed.
		Added RST input in EPLLB.
		Fin min 33 replaced with 25.
		FB_MODE default = CLOCKTREE
		CLKOP_DIV values for EHXPLLB 2, 4, 8,16, 32.
December 2004	02.0	Appendices C and D integrated to body of the document.
		DCS source code example moved to Appendix A.
January 2005	03.0	LatticeXP information added.
		Figures 6 and 7 updated.
		CLKOP_FREQ, CLKOK_FREQ user attributes added.
		FB_MODE added.
October 2005	04.0	Clock Distribution section added.
		Example code section removed and referred to help file.
		GUI screen shots updated.
		MM/IP Manager renamed as IPexpress.
		Epllb definition section removed.
		Clkos/clkok select attributes added.
September 2006	04.1	Detailed clock distribution information added.
April 2007	04.2	Updated footnote 3 in the LatticeECP/EC and LatticeXP PLL Attributes table.
June 2007	04.3	Updated Lock Output section of PLL Inputs and Outputs..
		LatticeECP/EC and LatticeXP PLL I/O Definitions table - corrected signal name DDAIDEL to read DDAIDEL[2:0] in the Signals column.

Appendix A. Clock Preferences

A few key clock preferences are introduced below. Refer to the 'Help' file for other preferences and detailed information.

ASIC

The following preference command assigns a phase of 90° to the CIMDLLA CLKOP.

```
ASIC "my_pll" TYPE "EXHXPLLB" CLKOS_PHASE=90;
```

FREQUENCY

The following physical preference command assigns a frequency of 100 MHz to a net named clk1.

```
FREQUENCY NET "clk1" 100 MHz;
```

The following preference specifies a hold margin value for each clock domain.

```
FREQUENCY NET "RX_CLKA_CMOS_c" 100.000 MHz HOLD_MARGIN 1 ns;
```

MAXSKEW

The following command assigns a maximum skew of 5 ns to a net named NetB.

```
MAXSKEW NET "NetB" 5 NS;
```

MULTICYCLE

The following command will relax the period to 50 ns for the path starting at COMPA to COMPB (NET1).

```
MULTICYCLE "PATH1" START COMP "COMPA" END COMP "COMPB" NET "NET1" 50 NS ;
```

PERIOD

The following command assigns a clock period of 30 ns to the port named Clk1.

```
PERIOD PORT "Clk1" 30 NS;
```

PROHIBIT

This command prohibits the use of a primary clock to route a clock net named bf_clk.

```
PROHIBIT PRIMARY NET "bf_clk";
```

CLOCK_TO_OUT

Specifies a maximum allowable output delay relative to a clock.

Below are two preferences using both the CLKPORT and CLKNET keywords showing the corresponding scope of TRACE reporting.

The CLKNET will stop tracing the path before the PLL, so you will not get PLL compensation timing numbers.

```
CLOCK_TO_OUT PORT "RxAddr_0" 6.000000 ns CLKNET "pll_rxclk" ;
```

The above preference will yield the following clock path:

Physical Path Details:

Clock path pll_inst/pll_utp_0_0 to PFU_33:

Name	Fanout	Delay (ns)	Site	Resource
ROUTE	49	2.892	ULPPLL.MCLK to	R3C14.CLK0 pll_rxclk

2.892   (0.0% logic, 100.0% route), 0 logic levels.

If CLKPORT is used, the trace is complete back to the clock port resource and provides PLL compensation timing numbers.

CLOCK_TO_OUT PORT "RxAddr_0" 6.000000 ns CLKPORT "RxClk" ;

The above preference will yield the following clock path:

Clock path RxClk to PFU_33:

Name	Fanout	Delay (ns)	Site	Resource
IN_DEL	---	1.431	D5.PAD to	D5.INCK RxClk
ROUTE	1	0.843	D5.INCK to	ULPPLL.CLKIN RxClk_c
MCLK_DEL	---	3.605	ULPPLL.CLKIN to	ULPPLL.MCLK pll_inst/pll_utp_0_0
ROUTE	49	2.892	ULPPLL.MCLK to	R3C14.CLK0 pll_rxclk

8.771   (57.4% logic, 42.6% route), 2 logic levels.

INPUT_SETUP

Specifies an setup time requirement for input ports relative to a clock net.

INPUT_SETUP PORT "datain" 2.000000 ns HOLD 1.000000 ns CLKPORT "clk" PLL_PHASE_BACK ;

PLL_PHASE_BACK

This preference is used with INPUT_SETUP when the user wants a trace calculation based on the previous clock edge.

This preference is useful when setting the PLL output phase adjustment. Since there is no negative phase adjustment provided, the PLL_PHASE_BACK preference works as if negative phase adjustment is available.

For example:

If phase adjustment of -90° of CLKOS is desired, the user can set the phase to 270° and set the INPUT_SETUP preference with PLL_PHASE_BACK.

Introduction

One of the requirements when using FPGA devices is the ability to calculate power dissipation for a particular device used on a board. Lattice's ispLEVER® design tools include a Power Calculator tool which allows designers to calculate the power dissipation for a given device. This technical note explains how to use Power Calculator to calculate the power consumption of Lattice devices. General guidelines to reduce power consumption are also included.

Power Supply Sequencing and Hot Socketing

LatticeECP™, LatticeEC™ and LatticeXP devices have eight sysIO™ buffer banks in addition to the V_{CC}, V_{CCAUX} and V_{CCJ} power supplies; each is capable of supporting multiple I/O standards. Each sysIO bank has its own I/O supply voltage (V_{CCIO}), and two voltage references V_{REF1} and V_{REF2} resources allowing each bank to be completely independent from each other.

The LatticeECP/EC and LatticeXP devices are designed to ensure predictable behavior during power-up and power-down. Power supplies can be sequenced in any order. The I/Os remain in tristate until the power supply voltage is high enough to ensure reliable operation during power up and power-down sequences and the leakage into I/O pins is controlled to within specified limits. Refer to the Typical I/O Behavior During Power-up and Hot Socketing sections of the device data sheet for more details.

Power Calculator Hardware Assumptions

The power consumption for a device can be coarsely broken down into the DC portion and the AC portion.

The power calculator reports the power dissipation in terms of:

1. DC portion of the power consumption.
2. AC portion of the power consumption.

The DC power (or the static power consumption) is the total power consumption of the used and unused resources. These components are fixed for each resource used and depend upon the number of resource units utilized. The DC component also includes the static power dissipation for the unused resources of the device.

The AC portion of power consumption is associated with the used resources and it is the dynamic part of the power consumption. Its power dissipation is directly proportional to the frequency at which the resource is running and the number of resource units used.

Power Calculator

Power Calculator is a powerful tool which allows users to make an estimate of the power consumption at two different levels:

1. Estimate of the utilized resources before completing place and route
2. Post place and route design

For first level estimation, the user provides estimates of device usage in the Power Calculator Wizard and the tool provides a rough estimate of the power consumption.

The second level is a more accurate approach where the user imports the actual device utilization by importing the post Place and Route netlist (NCD) file.

Power Calculator Equations

The power equations used in the Power Calculator have the following general form:

Total DC Power (Resource)

$\qquad$ = Total DC Power of Used Portion + Total DC Power of Unused Portion

$\qquad$ = [DC Leakage per resource when Used * $N_{RESOURCE}$]

$\qquad\quad$ + [DC Leakage per resource when Unused * ($N_{TOTAL\ RESOURCE}$ - $N_{RESOURCE}$)]

Where:

$\quad$ $N_{TOTAL\ RESOURCE}$ $\qquad$ is the total number of resources in a device.

$\quad$ $N_{RESOURCE}$ $\qquad$ is the number of resources used in the design.

The total DC power consumption for all the resources as per the design data is the sum of Quiescent Power and the individual DC power of the resources in the Power Calculator.

Total DC Power (I_{CCAUX})

$\qquad$ = $K_{RESOURCE}$ * 500 µA + Typical Standby I_{CCAUX}

Where:

$\quad$ KRESOURCE $\qquad$ is the number of reference input I/O such as HSTL/SSTL. For LVDS KRE-SOURCE is number of inputs divided by two.

$\quad$ I_{CCAUX} $\qquad$ is a DC current that does not change with I/O toggle rate or temperature.

Typical Standby I_{CCAUX} is found in the data sheet.

The AC power, on the other hand, is governed by the equation:

Total AC Power (Resource)

$\qquad$ = $K_{RESOURCE}$ * f_{MAX} * $AF_{RESOURCE}$ * $N_{RESOURCE}$

Where:

$\quad$ $N_{TOTAL\ RESOURCE}$ $\qquad$ is the total number of resources in a device.

$\quad$ $N_{RESOURCE}$ $\qquad$ is the number of resources used in the design.

$\quad$ $K_{RESOURCE}$ $\qquad$ is the power constant for the resource, measured in mW/MHz.

$\quad$ f_{MAX} $\qquad$ is the maximum frequency at which the resource is running, measured in MHz.

$\quad$ $AF_{RESOURCE}$ $\qquad$ is the activity factor for the resource group, as a percentage (%) of switching frequency.

Based on the above equations, if we wish to calculate the power consumption of the Slice portion, it will be as follows:

Total DC Power (Slice)

$\qquad$ = Total DC Power of Used Portion + Total DC Power of Unused Portion

$\qquad$ = [DC Leakage per Slice when Used * N_{SLICE}]

$\qquad\quad$ + [DC Leakage per Slice when Unused * ($N_{TOTAL\ SLICE}$ - N_{SLICE})]

Total AC Power (Slice)

$\qquad$ = K_{SLICE} * f_{MAX} * AF_{SLICE} * N_{SLICE}

The DC and AC power, for a dedicated block, like DSP in LatticeECP devices, is governed by the following equations.

Total DC Power (Resource)

$\qquad$ = DC Leakage per Resource * $N_{RESOURCE}$

Total AC Power (Resource)

$$= K_{RESOURCE} * f_{MAX} * N_{RESOURCE}$$

Where:

$N_{RESOURCE}$	is the total number of resources in a device.
$K_{RESOURCE}$	is the power constant for the resource, measured in mW/MHz.
f_{MAX}	is the maximum frequency at which the resource is running, measured in MHz.

Starting the Power Calculator

The user can launch the Power Calculator by one of the two methods. The first method is by clicking the Power Calculator button in the toolbar as shown in Figure 12-1.

Figure 12-1. Starting Power Calculator from Toolbar
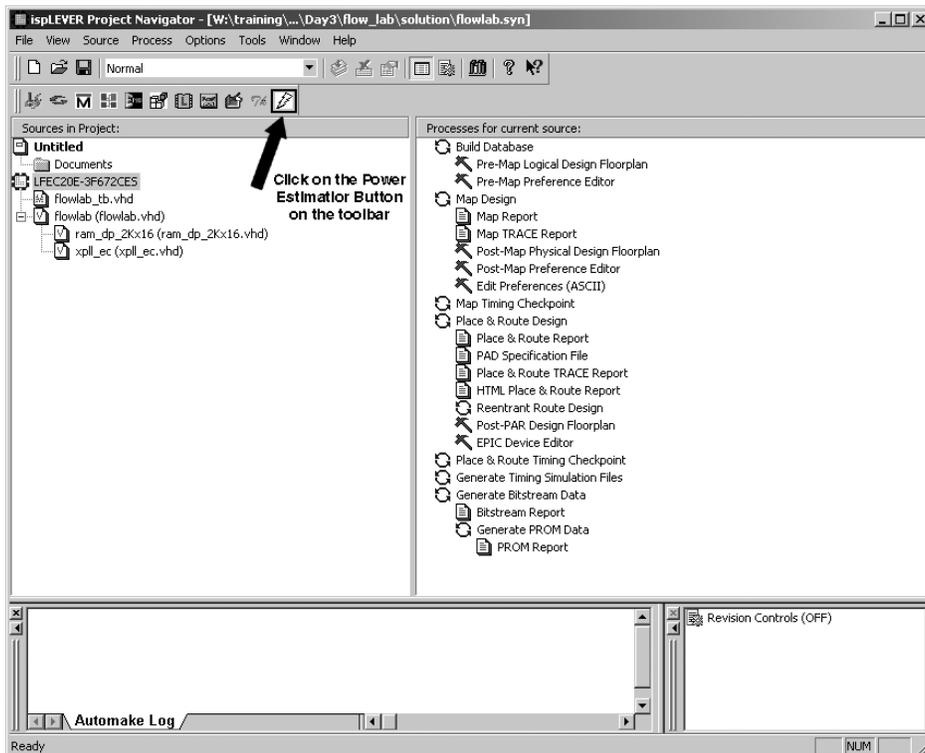
Alternatively, users can launch the Power Calculator by going to the Tools menu and selecting the option Power Calculator as shown in Figure 12-2.

Figure 12-2. Starting Power Calculator from Tools Menu
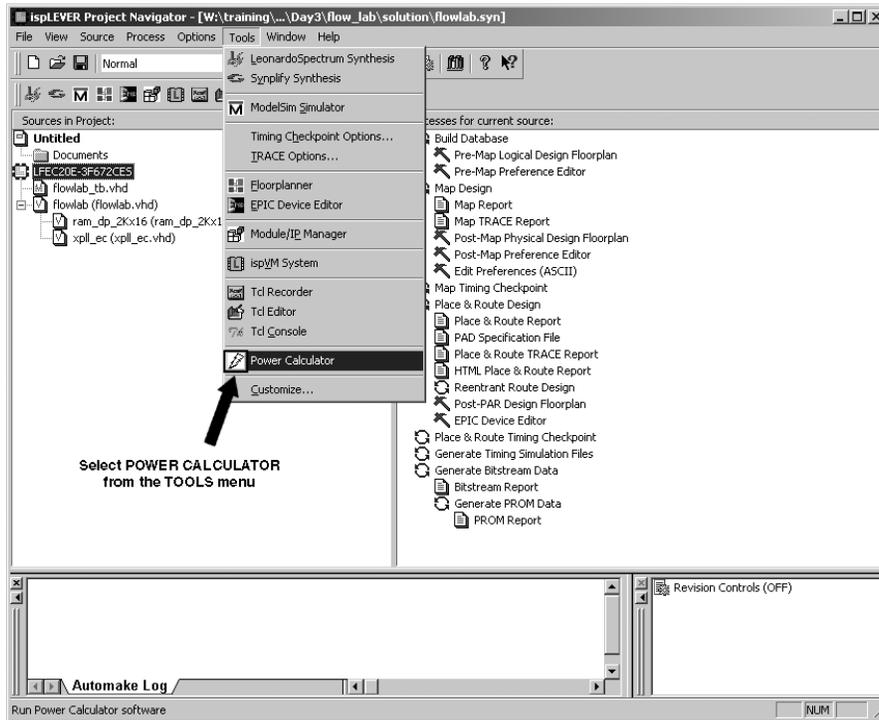
The Power Calculator does not support some of Lattice's older devices. The toolbar button and menu item is only present when supported devices are selected.

Starting a Power Calculator Project

Once the Power Calculator has been started, the Power Calculator window appears. Click on **File ->Menu**, and select **New** to get to the Start Project window, as shown in Figure 12-3.

Figure 12-3. Power Calculator Start Project Window (Create New Project)
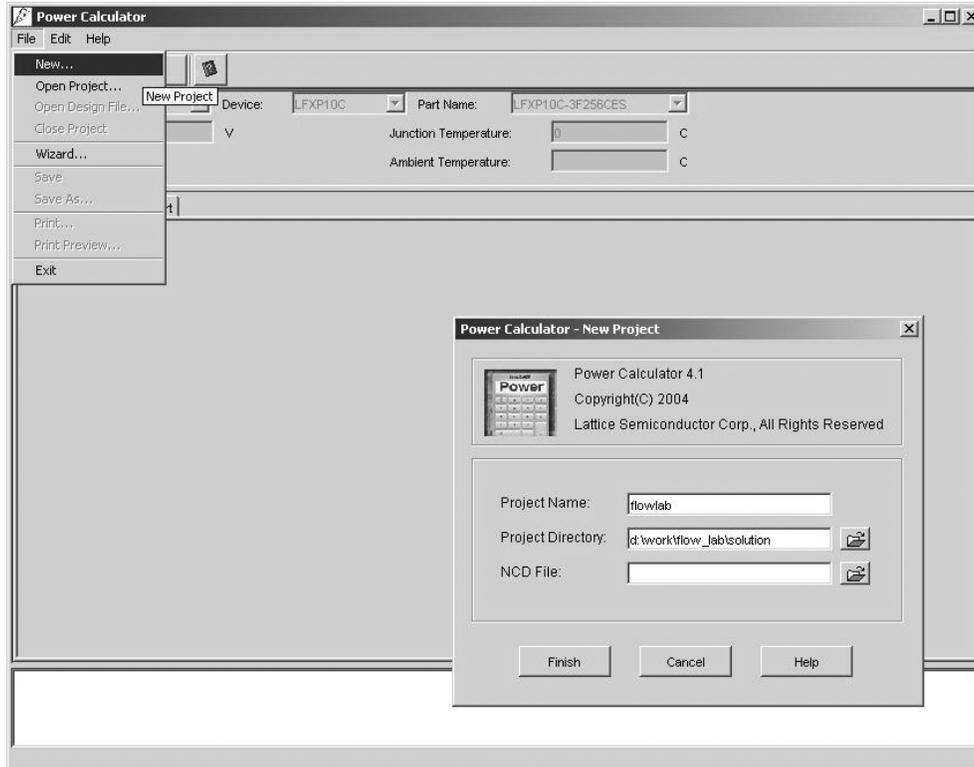
The Start Project window is used to create a new Power Calculator Project (*.pep project). Three pieces of data are input in the Start Project window.

1. The Power Calculator project name by default is same as the Project Navigator project name. The name can be changed, if desired.
2. Project Directory is where the Power Calculator project (*.pep) file will be stored. By default, the file is stored in the main project folder.
3. Input an NCD file (if available) or browse to the NCD file in a different location.

Power Calculator Main Window

The main power calculator window is shown in Figure 12-4.

Figure 12-4. Power Calculator Main Window (Type View)
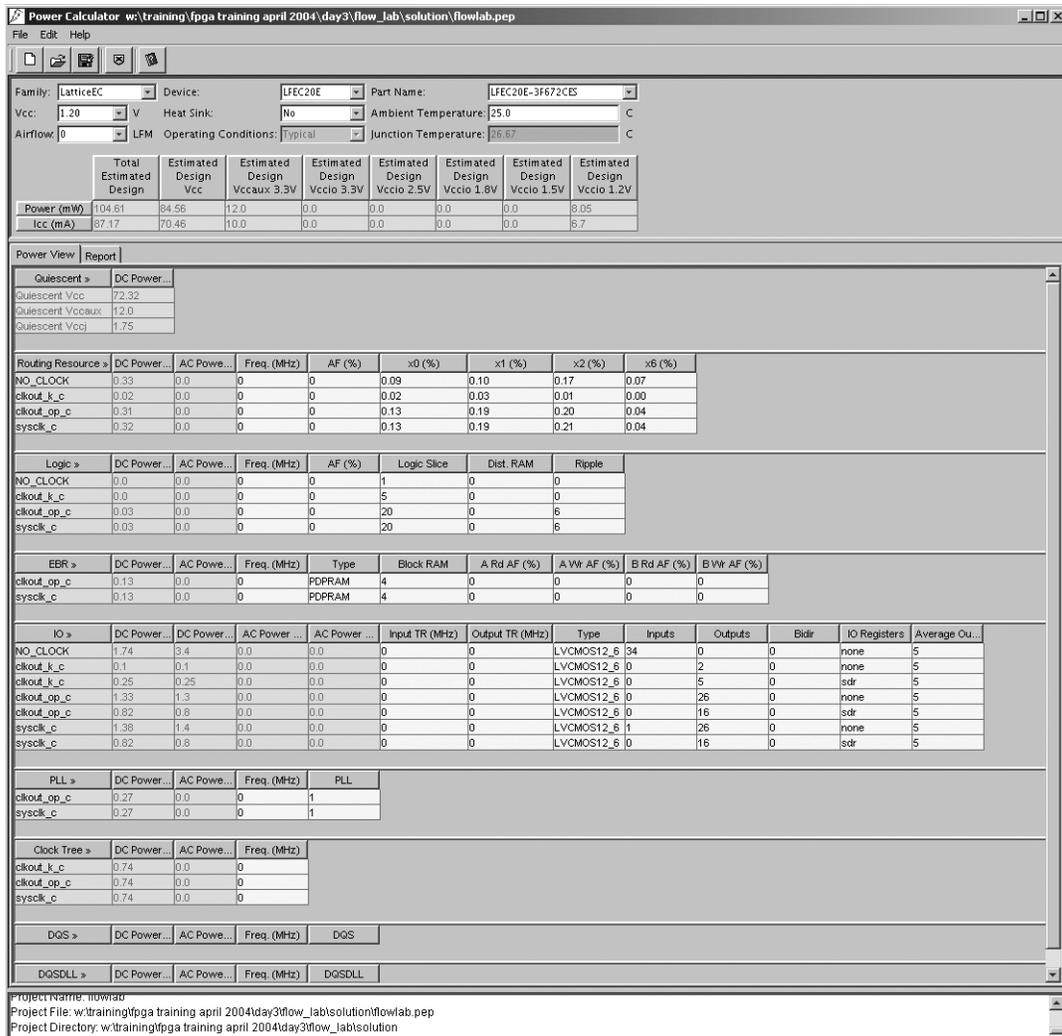
The top pane of the window shows information about the device family, device and the part number as it appears in the Project Navigator. The V_{CC} used for the Power Calculation is also listed. Users have an option to provide the ambient temperature, and the junction temperature is calculated based on that.

Users can also enter values of airflow in Linear Feet per Minute (LFM) along with heat sink to get the junction temperature. A table in the top part of the Power Calculator summarizes the currents and power consumption associated with each type of power supply for the device. This also takes into consideration the I/O power supplies.

In the middle pane of the window, there are two tabs:

1. Power View
2. Report

The first tab is the Power view. Under this tab, the Power Calculator tool has an interactive spreadsheet type interface.

The second and third columns, which are shaded blue in the tool, provide the DC (or static) and AC (or dynamic) power consumption, respectively.

In case of the I/O, there are four columns that are shaded blue. These provide the DC and AC power for I/Os for the core voltage, V_{CC} and the I/O voltage supply, V_{CCIO}.

The first three rows show the Quiescent Power for V_{CC}, V_{CCAUX} and V_{CCJ}. These are DC power numbers for a blank device or device with no resource utilization.

Some of the cells are shaded yellow in the tool. These cells are editable cells and users can type in values such as frequency, activity factors and resource utilization.

The second tab or the Report tab is the summary of the Power View. This report is in text format that provides the details of the power consumption.

The final pane or the lower pane of the window is the log pane where users can see the log of the various operations in the Power Calculator.

Figure 12-5. Power Calculator Main Window (Power Report View)
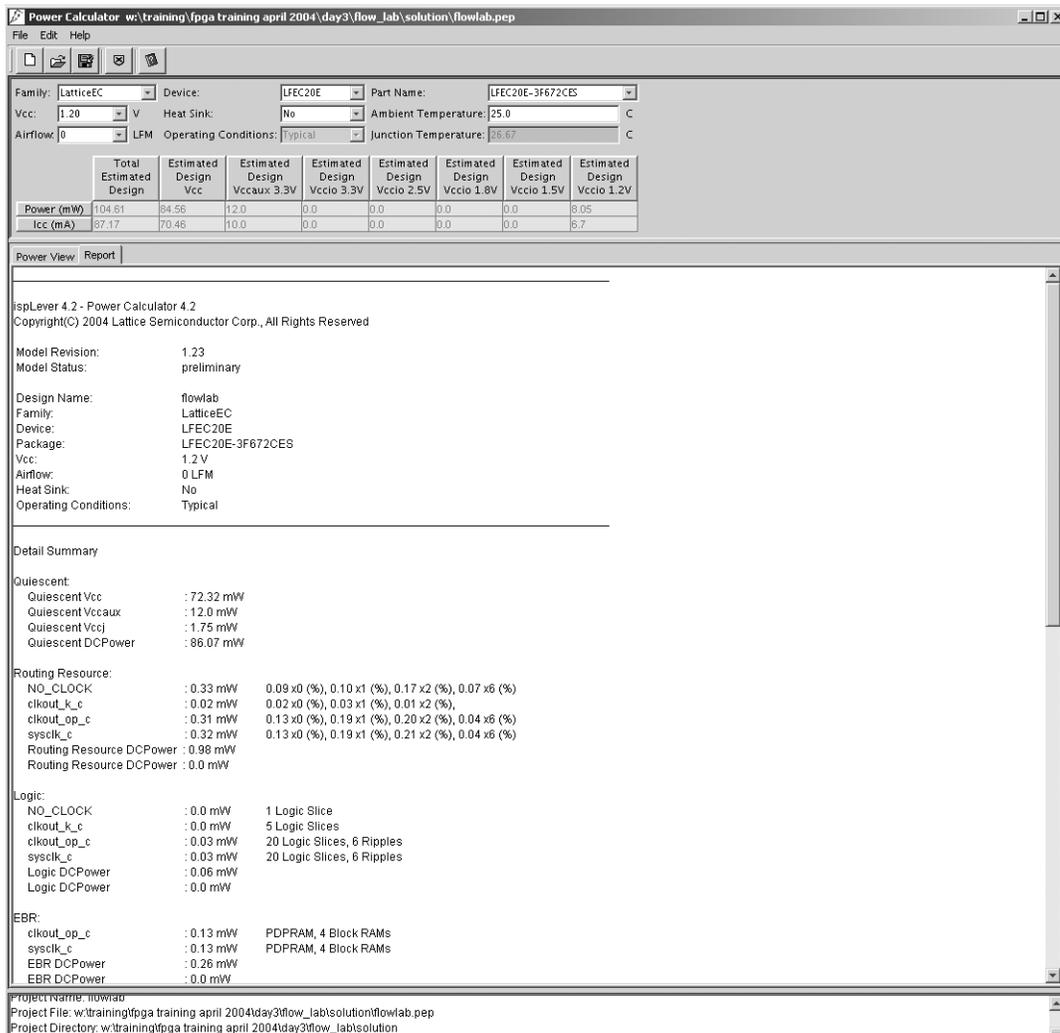
Power Calculator Wizard

The Power Calculator Wizard allows users to estimate the power consumption of a design. This estimation is done before actually creating the design. The user must understand the logic requirements of the design. The wizard asks the user to provide these parameters and then estimates the power consumption of the device.

To start the Power Calculator in the wizard mode, go to the **File** menu and select **Wizard**. Alternatively, click on the **Wizard** button and get the **Power Calculator - Wizard** window, as shown in Figure 12-6. Select the option **Create a New Project** and check the **Wizard** check box in the Power Calculator Start Project window. Users provide the project name and the project folder and click **Continue**. Since power is being estimated before the actual design, no NCD file is required.

Figure 12-6. Power Calculator Start Project Window (Using the New Project Window Wizard)
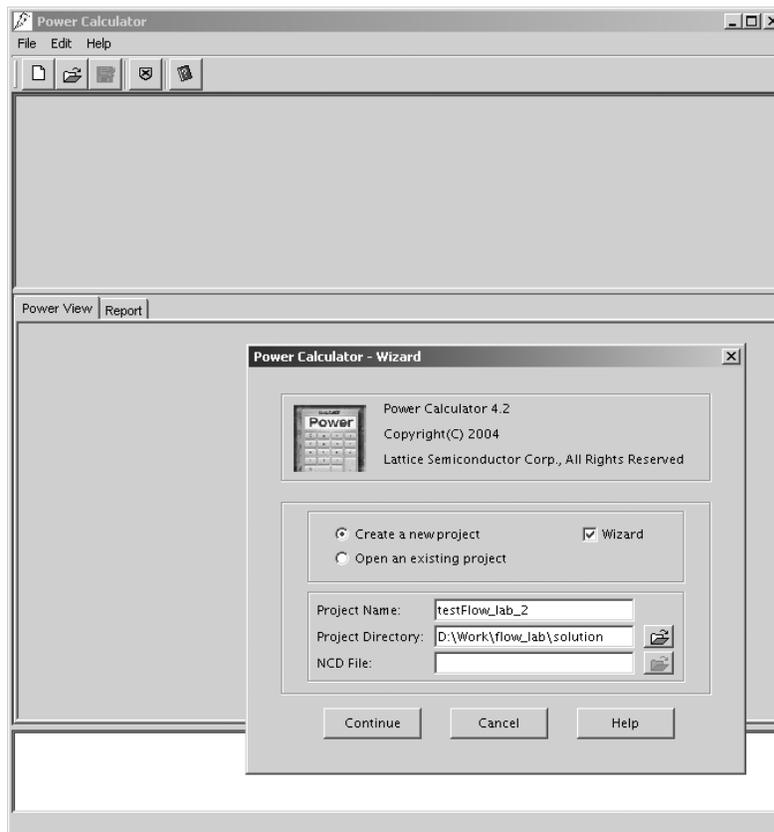
The next screen, as shown in Figure 12-7, allows users to select the device family, device and appropriate part number. After making proper the selections, click **Continue**. This is shown in Figure 12-7.

Figure 12-7. Power Calculator Wizard Mode Window - Device Selection
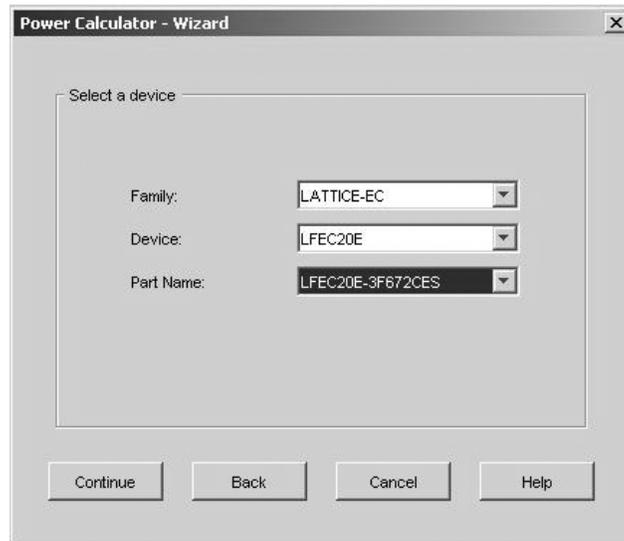
In the following screens, as shown in Figures 12-8-12-12, users can select further resources such as I/O types and provide a clock name, frequency at which the clock in running and other parameters, by selecting the appropriate resource using the pull-down Type menu:

1. Routing Resources
2. Logic
3. EBR
4. I/O
5. PLL
6. Clock Tree

The numbers in these windows refers to the number of clocks and the index corresponds to each of the clocks. By default, the clock names are clk_1, clk_2, and so on. The name of each clock can be changed by typing in the Clock Name text box. For each clock domain and resource users can specify parameters such as frequency, activity factor, etc. Users can click the Create button for each clock-driven resource using the pull-down Type menu.

These parameters are then used in the Power Type View window (see Figure 12-13) which can be seen by clicking Finish.

Figure 12-8. Power Calculator Wizard Mode Window - Resource Specification - Logic
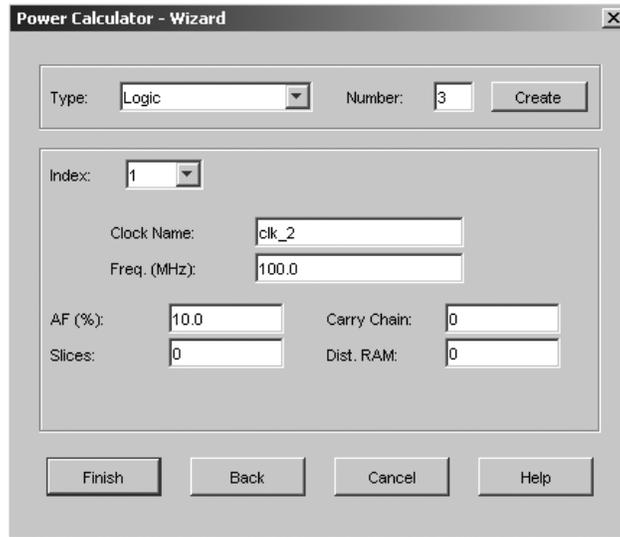
Figure 12-9. Power Calculator Wizard Mode Window - Resource Specification - EBR
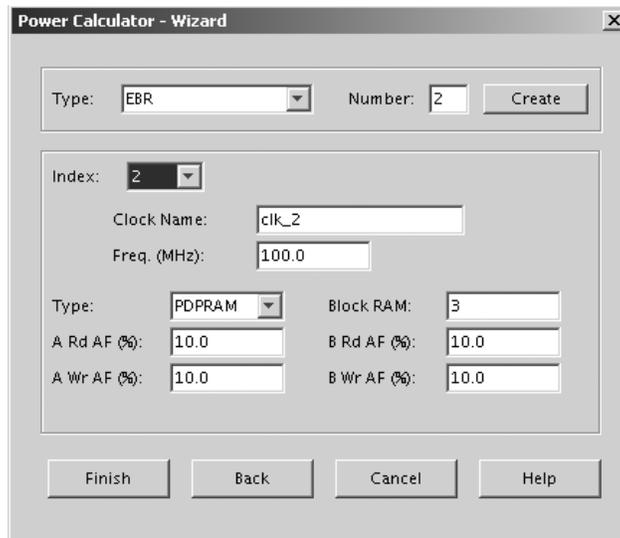
Figure 12-10. Power Calculator Wizard Mode Window - Resource Specification - PLL
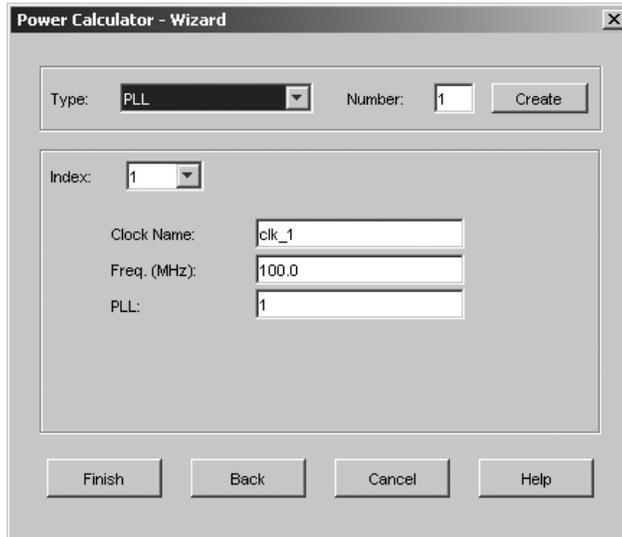
Figure 12-11. Power Calculator Wizard Mode Window - Resource Specification - Routing Resources
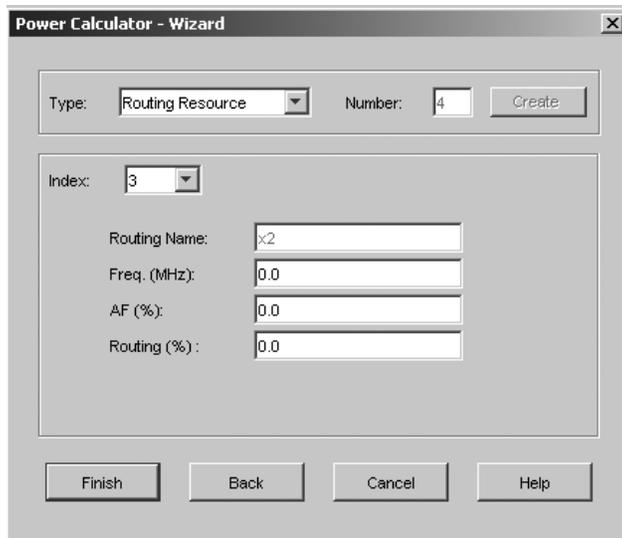
Figure 12-12. Power Calculator Wizard Mode Window - Resource Specification - I/Os
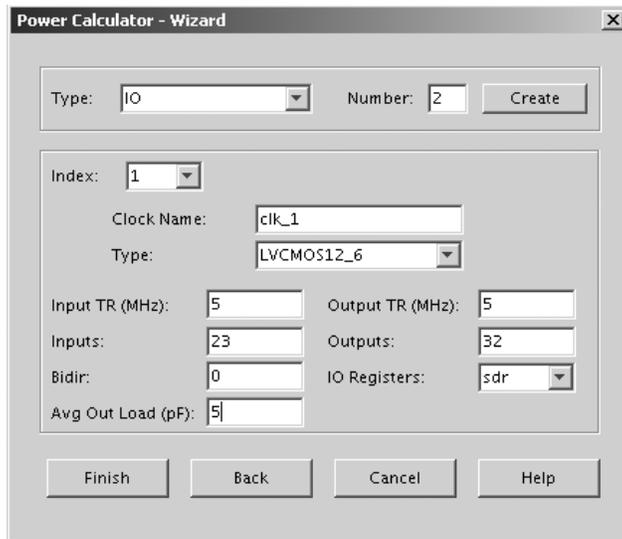
Figure 12-13. Power Calculator Wizard Mode - Main Window

Power Calculator – Creating a New Project Without the NCD File

A new project can be started without the NCD file by either using the Wizard (as discussed above) or by selecting the **Create a New Project** option in the **Power Calculator – Start Project**. A project name and project directory must be provided. After clicking **Continue**, the Power Calculator main window will be displayed.

However, in this case there are no resources added. The power estimation row for the Routing resources is always available in the Power Calculator. Users are then asked to add more information like the slice, EBR, I/O, PLL and clock tree utilization to calculate the power consumption.

For example, to add logic resources (as shown in Figure 12-14), right-click on **Logic >>** and then select **Add** in the menu that pops up.

Figure 12-14. Power Calculator Main Window – Adding Resources

This adds a new row for the logic resource utilization with clock domain as clk_1.

Similarly, other resources like EBR, I/Os, PLLs and routing can be added. Each of these resources is for AC power estimation and categorized by clock domains.

Power Calculator – Creating a New Project With the NCD File

If the post place and routed NCD file is available, the Power Calculator can use it to import the accurate information about the design data and resource utilization and calculate the power. When the Power Calculator is started, the NCD file is automatically placed in the NCD File option, if available in the project directory. Otherwise, the user can browse to the NCD file in the Power Calculator.

Figure 12-15. Power Calculator Start Project Window – With Post Place and Route NCD File

The information from the NCD file is automatically inserted into the correct rows and the Power Calculator uses the Clock names from the design, as shown in Figure 12-16.

Figure 12-16. Power Calculator Main Window – Resource Utilization Picked Up From the NCD File

Power Calculator – Open Existing Project

The Power Calculator – Start Project window also allows users to open an existing project. Select the option **Open Existing Project** and browse to the *.pep project file and click **Continue**. This opens the existing project in similar windows as discussed above. This is shown in Figure 12-17.

Figure 12-17. Opening Existing Project in Power Calculator
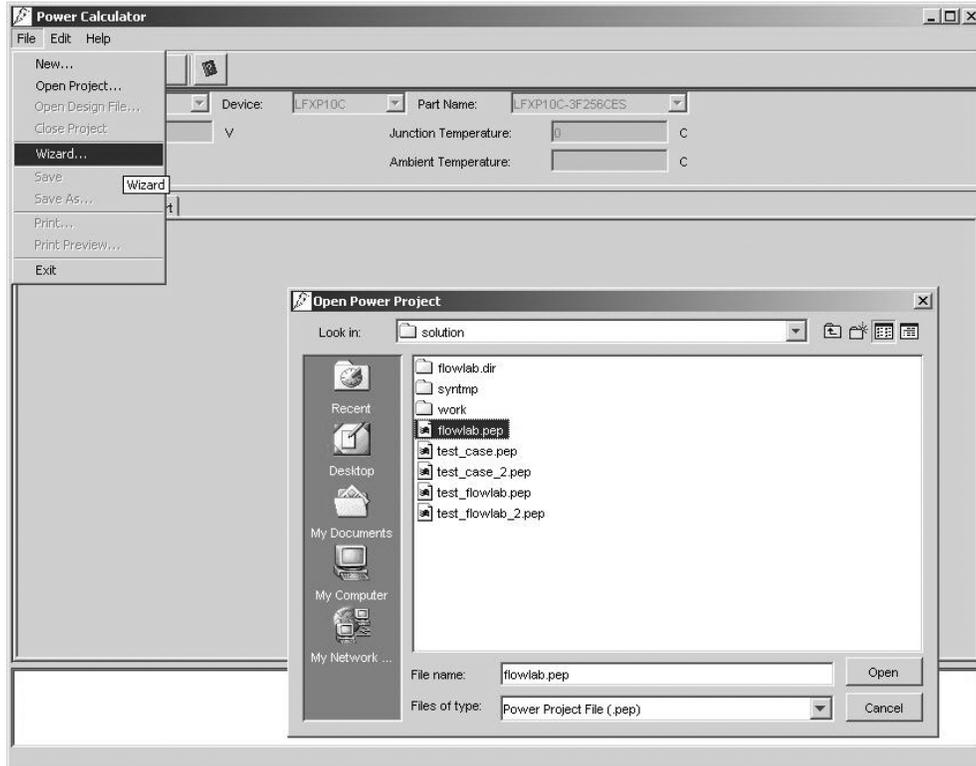
Power Calculator – Total Power

The Power Calculator project created or opened using any of the methods discussed here would allow a user to calculate the power consumption for the device running with their design.

The estimated power is indicated in the Total section at the bottom of the table as shown in Figure 12-18.

Figure 12-18. Calculated Power in the Power Calculator Main Window

Power Calculator W:\training\FPGA Training April 2004\Day3\flow_lab\solution\flowlab.pep

File   Edit   Help

Family: LatticeEC   Device: LFEC20E   Part Name: LFEC20E-3F672CES
Vcc: 1.20 V   Heat Sink: No   Ambient Temperature: 25.0 C
Airflow: 0 LFM   Operating Conditions: Typical   Junction Temperature: 29.68 C

	Total Estimated Design	Estimated Design Vcc	Estimated Design Vccaux 3.3V	Estimated Design Vccio 3.3V	Estimated Design Vccio 2.5V	Estimated Design Vccio 1.8V	Estimated Design Vccio 1.5V	Estimated Design Vccio 1.2V
Power (mW)	292.71	267.01	12.0	0.0	0.0	0.0	0.0	13.7
Icc (mA)	243.92	222.5	10.0	0.0	0.0	0.0	0.0	11.41

Power View | Report

Quiescent »	DC Power...
Quiescent Vcc	72.32
Quiescent Vccaux	12.0
Quiescent Vccj	1.75

Routing Resource »	DC Power...	AC Powe...	Freq. (MHz)	AF (%)	x0 (%)	x1 (%)	x2 (%)	x6 (%)
NO_CLOCK	0.33	1.74	100	10	0.09	0.10	0.17	0.07
clkout_k_c	0.02	0.06	100	10	0.02	0.03	0.01	0.00
clkout_op_c	0.31	1.38	100	10	0.13	0.19	0.20	0.04
sysclk_c	0.32	1.4	100	10	0.13	0.19	0.21	0.04

Logic »	DC Power...	AC Powe...	Freq. (MHz)	AF (%)	Logic Slice	Dist. RAM	Ripple
NO_CLOCK	0.0	0.02	100	10	1	0	0
clkout_k_c	0.0	0.11	100	10	5	0	0
clkout_op_c	0.03	0.58	100	10	20	0	6
sysclk_c	0.03	0.58	100	10	20	0	6

EBR »	DC Power...	AC Powe...	Freq. (MHz)	Type	Block RAM	A Rd AF (%)	A Wr AF (%)	B Rd AF (%)	B Wr AF (%)
clkout_op_c	0.13	1.28	100	PDPRAM	4	10	10	10	10
sysclk_c	0.13	1.28	100	PDPRAM	4	10	10	10	10

IO »	DC Power...	DC Power...	AC Power...	AC Power...	Input TR (M...	Output TR (...	Type	Inputs	Outputs	Bidir	IO Registers	Average Ou...
NO_CLOCK	1.74	3.4	0.01	3.96	5	5	LVCMOS12_6	34	0	0	none	5
clkout_k_c	0.1	0.1	8.96	0.03	5	5	LVCMOS12_6	0	2	0	none	5
clkout_k_c	0.25	0.25	0.0	0.08	5	5	LVCMOS12_6	0	5	0	sdr	5
clkout_op_c	1.33	1.3	0.01	0.45	5	5	LVCMOS12_6	0	26	0	none	5
clkout_op_c	0.82	0.8	0.0	0.28	5	5	LVCMOS12_6	0	16	0	sdr	5
sysclk_c	1.38	1.4	0.01	0.57	5	5	LVCMOS12_6	1	26	0	none	5
sysclk_c	0.82	0.8	0.0	0.28	5	5	LVCMOS12_6	0	16	0	sdr	5

PLL »	DC Power...	AC Powe...	Freq. (MHz)	PLL
clkout_op_c	0.27	4.92	100	1
sysclk_c	0.27	4.92	100	1

Clock Tree »	DC Power...	AC Powe...	Freq. (MHz)
clkout_k_c	0.74	51.73	100
clkout_op_c	0.74	51.73	100
sysclk_c	0.74	51.73	100

DQS »	DC Power...	AC Powe...	Freq. (MHz)	DQS

DQSDLL »	DC Power...	AC Powe...	Freq. (MHz)	DQSDLL

Project File: W:\training\FPGA Training April 2004\Day3\flow_lab\solution\flowlab.pep
Project Directory: W:\training\FPGA Training April 2004\Day3\flow_lab\solution

The second and third columns from the left indicate the DC (or static) and AC (or dynamic) power consumption. The total power consumption for the design can be seen in the same table. Scroll down to the row labeled **Total**.

Activity Factor

Activity Factor % (or AF%) is defined as the percentage of frequency (or time) that a signal is active or toggling of the output.

Most of the resources associated with a clock domain are running or toggling at some percentage of the frequency at which the clock is running. Users are required to provide this value as a percentage under the AF% column in the Power Calculator tool.

Another term used for I/Os is the I/O Toggle Rate or the I/O Toggle Frequency. 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). So, the rates at which I/Os toggle define their activity. The Toggle Rate (or TR) in MHz of the output is defined as:

Toggle Rate (MHz) = 1/2 * f_{MAX} * AF%

Users are required to provide the TR (MHz) value for the I/O instead of providing the Frequency and AF% in case of other resources.

The AF can be calculated for each routing resource, output or PFU, however it involves long calculations. The general recommendation of a design occupying roughly 30% to 70% of the device is that the AF% used can be between 15% to 25%. This is an average value that can be seen most of the design. The accurate value of an AF depends upon clock frequency, stimulus to the design and the final output.

Ambient and Junction Temperature and Airflow

A common method of characterizing a packaged device's thermal performance is with thermal resistance, Θ. 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 Θ_{JA}, thermal resistance junction-to-ambient (in °C/W) and Θ_{JC}, thermal resistance junction-to-case (also in °C/W). Another factor is Θ_{JB}, thermal resistance junction-to-board (in °C/W).

Knowing the reference (i.e. ambient, case or board) temperature, the power, and the relevant Θ value, the junction temperature can be calculated as per following equations.

$$T_J = T_A + \Theta_{JA} * P \qquad (1)$$
$$T_J = T_C + \Theta_{JC} * P \qquad (2)$$
$$T_J = T_B + \Theta_{JB} * P \qquad (3)$$

Where T_J, T_A, T_C and T_B are the junction, ambient, case (or package) and board temperatures (in °C) respectively. P is the total power dissipation of the device.

Θ_{JA} is commonly used with natural and forced convection air-cooled systems. Θ_{JC} is useful when the package has a high conductivity case mounted directly to a PCB or heatsink. And Θ_{JB} applies when the board temperature adjacent to the package is known.

The Power Calculator utilizes the 25°C junction temperature as its basis to calculate power, per Equation 1 above. Users can also provide the airflow values (in LFM) and ambient temperature to get a calculated value of the junction temperature based on the power estimate.

Managing Power Consumption

One of the most critical design factors today is reducing system power consumption, especially for modern hand-held devices and electronics. There are several design techniques that designers can use to significantly reduce overall system power consumption. Some of these include:

1. Reducing operating voltage.

2. Operating within the specified package temperature limitations.

3. Using optimum clock frequency reduces power consumption, as the dynamic power is directly proportional to the frequency of operation. Designers must determine if a portion of their design can be clocked at a lower rate that will reduce power.

4. Reducing the span of the design across the device. A more closely placed design utilizes fewer routing resources for less power consumption.

5. Reducing the voltage swing of the I/Os where possible.

6. Using optimum encoding where possible. For example, a 16-bit binary counter has, on average, only 12% Activity Factor and a 7-bit binary counter has an average of 28% Activity Factor. On the other hand, a 7-bit Linear Feedback Shift Register could toggle as much as 50% Activity Factor, which causes higher power consumption. A gray code counter, where only one bit changes at each clock edge, will use the least amount of power, as the Activity Factor would be less than 10%.

7. Minimize the operating temperature, by the following methods:
   a. Use packages that can better dissipate heat. For example, packages with lower thermal impedance.
   b. Place heat sinks and thermal planes around the device on the PCB.
   c. Better airflow techniques using mechanical airflow guides and fans (both system fans and device mounted fans).

Power Calculator Assumptions

Following are the assumptions made in the Power Calculator:

1. The Power Calculator tool is based on equations with constants based on room temperature of 25°C.

2. The user can define the Ambient Temperature (Ta) for device Junction Temperature (Tj) calculation based on the power estimation. Tj is calculated from user-entered Ta and power calculation of typical room temperature.

3. The I/O power consumption is based on output loading of 5pF. Users have ability to change this capacitive loading.

4. The current version of the Power Calculator allows users to get an estimate of the power dissipation and the current for each type of power supplies, that are V_{CC}, V_{CCIO}, V_{CCJ} and V_{CCAUX}. For V_{CCAUX}, only static I_{CCAUX} values are provided in the Calculator.

   Additional V_{CCAUX} contributions due to differential output buffers, differential input buffers and reference input buffers must be added per pair for differential buffers or per pin for reference input buffers according to the user's design. See the equation given in this technical note for Total DC Power (I_{CCAUX}).

5. The nominal V_{CC} is used by default to calculate the power consumption. Users can choose a lower or higher V_{CC} from a list of available values. For example, the nominal V_{CC} of 1.2V is used by default for the LatticeECP/EC and LatticeXP families of devices.

6. The current versions also allows users to enter an airflow in Linear Feet per Minute (LFM) along with the Heat Sink option to calculate the Junction Temperature.

7. The default value of the I/O types for the LatticeEC and LatticeXP devices is LVCMOS12, 6mA.

8. The Activity Factor (AF) is defined as the toggle rate of the registered output. For example, assuming that the input of a flip-flop is changing at every clock cycle, 100% AF of a flip-flop running at 100MHz is 50MHz.

Revision History

Date	Version	Change Summary
June 2004	01.0	Initial release.
July 2004	01.1	Provided additional description of the assumptions used in the power model.
October 2004	01.2	Updated screen shots for ispLEVER 4.1.
February 2005	02.0	Added support for LatticeXP family throughout.
		Added DC and AC power for a dedicated block like DSP for LatticeECP devices.
May 2005	02.1	Updated the Power Supply Sequencing and Hot Socketing section.
		Updated the total DC Power consumption to be sum of Quiescent and the DC power of resources.
November 2006	02.2	Added calculation of I_{CCAUX} in Power Calculator Equations section.
September 2007	02.3	Document title changed to "Power Estimation and Management for LatticeECP/EC and LatticeXP Devices".

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

   +1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Appendix A. Power Calculator Project Example

This example assumes that you have post Place and Route NCD netlist in the design folder. Click on File > New or click on the New Project button. The New Project Window will open as shown below.
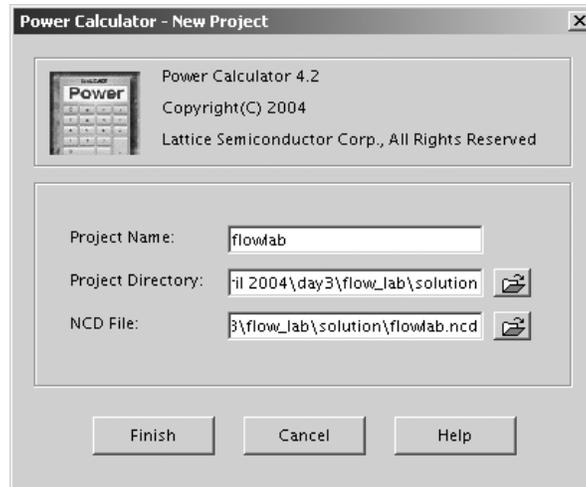
The various fields are filled in automatically with the project name the same as the ispLEVER project name and the directory also the same as the design folder. If the Post Place and Route NCD file is available, the NCD File field is also automatically filled. Users can also browse to the particular location to change the folder where they wish to create the Power Calculator project. Users can also browse to the NCD file in case it is not available at the root design folder.

Click Finish. This opens the Main Power Calculator project window, as shown below.

Note that the project window above imports all the resource utilization information from the NCD file. It does not, however, include information such as the frequency at which the design is operating or the activity factors at which the various components are toggling. This information is to be filled in by the user.

The top portion of the Power Calculator window shows information such as the device family and device being considered for power calculation, the V_{CC}, which is by default the nominal V_{CC} for the device, and operating conditions. Operating conditions users can enter include the ambient temperature, and heat sink available. Users can also select the air flow values.

There is a grayed box for junction temperature that shows Tj based on the given conditions and the calculated power.

If we assume the design is running at 100 MHz with a 10% Activity Factor, the final Power Calculator will be as shown below.

Power Calculator W:\training\FPGA Training April 2004\Day3\flow_lab\solution\flowlab.pep

File   Edit   Help

Family:	LatticeEC	Device:	LFEC20E	Part Name:	LFEC20E-3F672CES
Vcc:	1.20 V	Heat Sink:	No	Ambient Temperature:	25.0 C
Airflow:	0 LFM	Operating Conditions:	Typical	Junction Temperature:	29.68 C

	Total Estimated Design	Estimated Design Vcc	Estimated Design Vccaux 3.3V	Estimated Design Vccio 3.3V	Estimated Design Vccio 2.5V	Estimated Design Vccio 1.8V	Estimated Design Vccio 1.5V	Estimated Design Vccio 1.2V
Power (mW)	292.71	267.01	12.0	0.0	0.0	0.0	0.0	13.7
Icc (mA)	243.92	222.5	10.0	0.0	0.0	0.0	0.0	11.41

Power View | Report

Quiescent »	DC Power...
Quiescent Vcc	72.32
Quiescent Vccaux	12.0
Quiescent Vccj	1.75

Routing Resource »	DC Power...	AC Powe...	Freq. (MHz)	AF (%)	x0 (%)	x1 (%)	x2 (%)	x6 (%)
NO_CLOCK	0.33	1.74	100	10	0.09	0.10	0.17	0.07
clkout_k_c	0.02	0.06	100	10	0.02	0.03	0.01	0.00
clkout_op_c	0.31	1.38	100	10	0.13	0.19	0.20	0.04
sysclk_c	0.32	1.4	100	10	0.13	0.19	0.21	0.04

Logic »	DC Power...	AC Powe...	Freq. (MHz)	AF (%)	Logic Slice	Dist. RAM	Ripple
NO_CLOCK	0.0	0.02	100	10	1	0	0
clkout_k_c	0.0	0.11	100	10	5	0	0
clkout_op_c	0.03	0.58	100	10	20	0	6
sysclk_c	0.03	0.58	100	10	20	0	6

EBR »	DC Power...	AC Powe...	Freq. (MHz)	Type	Block RAM	A Rd AF (%)	A Wr AF (%)	B Rd AF (%)	B Wr AF (%)
clkout_op_c	0.13	1.28	100	PDPRAM	4	10	10	10	10
sysclk_c	0.13	1.28	100	PDPRAM	4	10	10	10	10

IO »	DC Power...	DC Power...	AC Power ...	AC Power ...	Input TR (M...	Output TR (...	Type	Inputs	Outputs	Bidir	IO Registers	Average Ou...
NO_CLOCK	1.74	3.4	0.01	3.96	5	5	LVCMOS12_6	34	0	0	none	5
clkout_k_c	0.1	0.1	8.96	0.03	5	5	LVCMOS12_6	0	2	0	none	5
clkout_k_c	0.25	0.25	0.0	0.08	5	5	LVCMOS12_6	0	5	0	sdr	5
clkout_op_c	1.33	1.3	0.01	0.45	5	5	LVCMOS12_6	0	26	0	none	5
clkout_op_c	0.82	0.8	0.0	0.28	5	5	LVCMOS12_6	0	16	0	sdr	5
sysclk_c	1.38	1.4	0.01	0.57	5	5	LVCMOS12_6	1	26	0	none	5
sysclk_c	0.82	0.8	0.0	0.28	5	5	LVCMOS12_6	0	16	0	sdr	5

PLL »	DC Power...	AC Powe...	Freq. (MHz)	PLL
clkout_op_c	0.27	4.92	100	1
sysclk_c	0.27	4.92	100	1

Clock Tree »	DC Power...	AC Powe...	Freq. (MHz)
clkout_k_c	0.74	51.73	100
clkout_op_c	0.74	51.73	100
sysclk_c	0.74	51.73	100

DQS »	DC Power...	AC Powe...	Freq. (MHz)	DQS

DQSDLL »	DC Power...	AC Powe...	Freq. (MHz)	DQSDLL

Project File: W:\training\FPGA Training April 2004\Day3\flow_lab\solution\flowlab.pep
Project Directory: W:\training\FPGA Training April 2004\Day3\flow_lab\solution

Introduction

The memory in LatticeECP™ and LatticeEC™ FPGAs is built using volatile SRAM. When the power is removed, the SRAM cells lose their contents. A supporting non-volatile memory is required to configure the device on power-up and at any time the device needs to be updated. The LatticeECP/EC devices support a sysCONFIG™ interface that provides multiple configuration modes as well as the dedicated ispJTAG™ port and boundary scan. The different programming modes are listed below.

- SPI

- SPIX

- Master Serial

- Slave Serial

- Master Parallel

- Slave Parallel

- ispJTAG (1149.1 Interface)

This technical note covers all the configuration options available for LatticeECP/EC devices.

Configuration Pins

The LatticeECP/EC devices support two types of sysCONFIG pins, dedicated and dual-purpose. The dedicated pins are used exclusively for configuration; the dual-purpose pins are available as extra I/O pins. If a dual-purpose pin is to be used both for configuration and as a general purpose I/O the user must adhere to the following:

- The general purpose I/O (GPIO) must maintain the same direction as it has during configuration, in other words, if the pin is an input during configuration it must remain an input as a GPIO, if an output during configuration it must remain an output as a GPIO, if a bi-directional it must remain a bi-directional as a GPIO.

- The I/O type must remain the same, in other words if the pin is a 3.3V CMOS pin (LVCMOS33) during configuration it must remain a 3.3V CMOS pin as a GPIO.

- The Persistent option must be set to OFF. The Persistent option can be accessed by using the Preference Editor in ispLEVER®.

- The user is responsible for insuring that no internal or external logic will interfere with device configuration.

Also note, if a parallel configuration mode is not being used then one or both of the parallel port chip selects (CSN, CS1N) must be high or tri-state during configuration.

Programmable options control the dual-purpose configuration pins. These options are controlled via a preference in Lattice ispLEVER software, or as an HDL source file attribute. The LatticeECP/EC devices also support the ispJTAG port for configuration, including transparent read back and JTAG testing. The following sections describe the functionality of the sysCONFIG and JTAG pins. Table 12-1 is provided for reference.

Table 12-1. Configuration Pins for LatticeECP/EC Devices

Pin(s)	Description	Default Pin Function	Mode Used
CFG[0:2]	Input	Dedicated	All
PROGRAMN	Input	Dedicated	All
INITN	Bi-directional open drain	Dedicated	All
DONE[1]	Bi-directional	Dedicated	All
CCLK	Output or input	Dedicated	MASTER = output, SLAVE = input
DI/CSSPIN	Input/output with weak pull-up	—	SERIAL/SPI
DOUT/CSON	Output	—	SERIAL/PARALLEL
CSN	Input	—	PARALLEL
CS1N	Input	—	PARALLEL
WRITEN	Input	—	PARALLEL
BUSY/SISPI	Output	—	PARALLEL/SPI
D[0:7]/SPID[7:0]	Input or output	—	PARALLEL/SPI
TDI	Input with pull-up	Dedicated	JTAG
TDO	Output	Dedicated	JTAG
TCK	Input with hysteresis, no pull-up	Dedicated	JTAG
TMS	Input with pull-up	Dedicated	JTAG

1. Defaults to open drain with an internal pull-up.

Dedicated Control Pins

The following is a description of the LatticeECP/EC's dedicated sysCONFIG pins used for controlling configuration.

CFG[0:2]
The Configuration Mode pins CFG[0:2] are input pins. They are used to select the configuration mode. Depending on the configuration mode selected, different groups of dual-purpose configuration pins will be activated on Power-On-Reset or when the PROGRAMN pin is driven low.

PROGRAMN
The PROGRAMN pin is an input to the device used to initiate a Programming sequence. A high to low signal applied to the pin sets the device into configuration mode. The PROGRAMN pin can be used to trigger programming other than at powering up. If the device is using JTAG, the device will ignore the PROGRAMN pin until the device is released from the JTAG mode.

INITN
The INITN pin is a bidirectional open drain control pin. It is capable of driving a low pulse out as well as detecting a low pulse driven in. When the PROGRAMN Pin is driven low, or after the internal Power-On-Reset signal is released during Power-up, the INITN pin will be driven low to reset the configuration circuitry and any External PROM. The configuration memory will be cleared and the INITN pin will remain low as long as the PROGRAMN pin is low. To delay configuration the INITN pin can be held low externally. The device will not enter configuration mode as long as the INITN pin is held low. Toggling the PROGRAMN pin in Serial and Parallel programming modes will initiate the configuration sequence and reset the INITN pin. For SPI mode, power cycling the device will initiate the reconfiguration sequence.

During configuration, the INITN pin becomes an error detection pin. It will be driven low whenever a configuration error occurs.

DONE
The DONE pin is a bidirectional control pin. It can be configured as an open drain or active drive control pin. The DONE pin will be driven low when the device is in configuration mode and the internal DONE bit is not programmed. When the INITN and PROGRAMN pins are high and the DONE bit is programmed, the DONE pin will be

released. An open drain DONE pin can be held low externally and, depending on the wake-up sequence selected, the device will not become functional until the DONE pin is released.

CCLK
The CCLK pin is a bi-directional pin. The direction depends on whether a Master Mode or Slave Mode is selected. If a Master Mode is selected when the CFG pins are sampled, the CCLK pin will become an output pin; otherwise CCLK will become an input pin. If the CCLK pin becomes an output pin, the internal programmable oscillator is connected to the CCLK and is driven out to slave devices. CCLK will stop 120 clocks cycles after the DONE pin is brought high and the device wake-up sequence completed. The extra clock cycles are provided to ensure that enough clock cycles are provided to wake up other devices in the chain. When stopped, CCLK will become tri-stated as an input. The CCLK will restart on the next configuration initialization sequence, such as the PRO-GRAMN pin being toggled. The MCCLK_FREQ Parameter controls the CCLK Master frequency. See the Master Clock Selection section of this document for more information. For Serial and Parallel Slave modes, it is recommended that CCLK is continuously active during configuration and error recovery sequence.

Dual-Purpose sysCONFIG Pins

The following is a list of dual-purpose sysCONFIG pins. If any of these pins are used for configuration and user I/O, the user must adhere to the requirements listed above in the section entitled Configuration Pins.

DI/CSSPIN
The DI/CSSPIN dual-purpose pin is designated as DI (Data Input) for all of the serial bit stream configurations, such as Slave Serial. DI has an internal weak pull up.

In either SPI or SPIX mode, the DI/CSSPIN becomes the dedicated Chip Select output to drive the SPI Flash chip select. CSSPIN will drive high when the LatticeECP/EC device is not in the process of configuration through the SPI Port.

D[0:7]/SPID[7:0]
The D[0:7] pins support both the SPI mode and Parallel configuration modes. In the Parallel configuration modes, the D[0:7] pins are tri-stated bi-directional I/O pins used for parallel data write and read. A byte of data is driven into or read from these pins. When the WRITEN signal is low and the CSN and CS1N pins are low, the D[0:7] pins will become an input. When the WRITEN signal is driven high and the CSN and CS1N pins are low, the pins become output pins for reading. The PERSISTENT preference must be set to support read back to preserve the D[0:7] pins so the device can monitor for the read back instruction. The CSN and CS1N pins will enable the Data D[0:7] pins.

In SPI mode, the D[0:7]/SPID[7:0] pins become individual inputs for one or more SPI memory outputs. If more than one SPI memory is used, SPI memory zero output will be wired to D7/SPID0, SPI memory one output will be wired to D6/SPID1, the data fed to these pins will be interleaved and then sent to the internal configuration engine. For SPIX Mode, the D[0:7]/SPID[7:0] pins will also support sampling of external resistors for determining the Read Op Code.

DOUT/CSON
The DOUT/CSON pin is an output pin and has two purposes. For serial and parallel configuration modes, when the BYPASS mode is selected, this pin will become DOUT. When the device in BYPASS becomes fully configured, a BYPASS instruction will be executed and the data on DI or D[0:7] will then be presented to the DOUT pin through a bypass register to serially pass the data to the next device. In a parallel configuration mode D0 will be shifted out first followed by D1, D2, and so on.

For parallel configuration modes, when the FLOW_THROUGH mode is selected, this pin will become the Chip Select OUT (CSON). In the FLOW_THROUGH mode, when the device is fully configured, the Flow Through instruction will be executed and the CSON pin will be driven low to enable the next device chip select pin.

The DOUT/CSON bypass register will drive out a HIGH upon power up and continue to do so till the execution of the Bypass/Flow Through instruction within the bit stream.

CSN and CS1N

Both CSN and CS1N are active low control input pins. When CSN OR CS1N are high, D[0:7] and BUSY pins are tri-stated. When the CSN and CS1N pins are both high, they will reset the flow-through/bypass register. CSN and CS1N are interchangeable when controlling the D[0:7], INITN and BUSY pins.

WRITEN

The WRITEN pin is an active low control input pin. The WRITEN pin is used to determine the direction of the data pins D[0:7]. The WRITEN pin is driven low when a byte of data is to be shifted into the device during programming. The WRITEN pin will be driven high when data is to be read from the device through a parallel configuration mode. The WRITEN pin is not used for serial configuration modes.

BUSY/SISPI

The BUSY/SISPI pin is a dual function pin. In the parallel configuration mode, the BUSY pin is a tri-stated output. The BUSY pin will be driven low by the device only when it is ready to receive a byte of data at D[0:7] pins or a byte of data is ready for reading. The BUSY pin can be used to support asynchronous peripheral mode. This is to acknowledge that the device might need extra time to execute a command.

In the SPI configuration modes, the BUSY/SISPI pin becomes an output pin that drives read control data back to the SPI memory.

ispJTAG Pins

The ispJTAG pins are the standard IEEE 1149.1 TAP pins. The ispJTAG pins are dedicated pins and are always accessible when the LatticeECP/EC device is powered up. In addition, the dedicated sysCONFIG pins such as the DONE pin as described in the Dual-Purpose Control Pins section of this document are also available when using LatticeECP/EC ispJTAG pins. The dedicated sysCONFIG pins are not required for JTAG operation, but may be useful at times.

TDO

The Test Data Output pin is used to shift out serial test instructions and data. When TDO is not being driven by the internal circuitry, the pin will be in a high impedance state.

TDI

The Test Data Input pin is used to shift in serial test instruction and data. An internal pull-up resistor on the TDI pin is provided. The internal resistor is pulled up to V_{CCJ}.

TMS

The Test Mode Select pin controls test operations on the TAP controller. On the falling edge of TCK, depending on if TMS is high or low, a transition will be made in the TAP controller state machine. An internal pull-up resistor on the TMS pin is provided. The internal resistor is pulled up to V_{CCJ}.

TCK

The test clock pin TCK provides the clock to run the TAP controller, loading and reloading the data and instruction registers. TCK can be stopped in either the high or low state and can be clocked at frequencies up to the frequency indicated in the device data sheet. The TCK pin supports hysteresis, with the value shown in the DC parameter table of the LatticeECP/EC Family Data Sheet.

Optional TRST

The JTAG Test Reset pin TRST in not supported in the LatticeECP/EC devices.

V_{CCJ}

JTAG V_{CC} supplies independent power to the JTAG port to allow chaining with other JTAG devices at a common voltage.

Configuration and JTAG Pin Physical Description

All of the control pins and programming bus default to LVCMOS. The bank V_{CCO} pin determines the voltage level of the sysCONFIG pins. The JTAG pin voltage levels are determined by the V_{CCJ} pin voltage level. Controlling the

JTAG pin by V_{CCJ} allows the device to support different JTAG chain voltages. For further JTAG chain questions, see *In-System Programming Design Guidelines for ispJTAG Devices,* available on the Lattice web site at www.lattices-emi.com.

Configuration Modes

The LatticeECP/EC devices support many different types of configuration modes utilizing either serial or parallel data inputs. On power-up or upon driving the PROGRAMN pin low, the CFG[2:0] pins are sampled to determine the mode the devices will be configured in. Table 12-2 lists the Mode, CFG[0:2] state and the software CONFIG_MODE parameters. The following subsections break down each configuration mode individually.

Table 12-2. Configuration Modes for the LatticeECP/EC Devices

Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
SPI Master	0	0	0	SPI
SPIX Master	0	0	1	SPIX
Master Serial (Bypass OFF)	1	0	0	MASTER_SERIAL
Master Serial (Bypass ON)	1	0	0	MASTER_SERIAL_BYPASS
Slave Serial (Bypass OFF)	1	0	1	SLAVE_SERIAL
Slave Serial (Bypass On)	1	0	1	SLAVE_SERIAL_BYPASS
Master Parallel (Flow Through OFF)	1	1	0	MASTER_PARLLEL
Master Parallel (Flow Through ON)	1	1	0	MASTER_PARLLEL_FLOWTHR
Slave Parallel	1	1	1	SLAVE_PARALLEL
Slave Parallel (Bypass ON)	1	1	1	SLAVE_PARALLEL_BYPASS
Slave Parallel (Flow Through ON)	1	1	1	SLAVE_PARALLEL_FLOWTHR
ispJTAG (1149.1 interface)	X	X	X	Any CONFIG_MODE or NONE1

Configuration Options

Several configuration options are available for each configuration mode. When daisy chaining multiple FPGA devices, an overflow option is provided for serial and parallel configuration modes. By setting the proper parameter in the Lattice design software, the selected configuration options are set in the generated bit stream. As the bit stream is loaded into the device, the selected configuration options will take effect. These options are described in the following sections and are software selectable by the Lattice design software.

Bypass Option
The Bypass option is used in parallel and serial device daisy chains. When the device has completed configuration and the Bypass option preference is selected, data coming into the device configuration port will overflow serially out of DOUT to the DI of the next slave serial device. The Bypass configuration selection is supported in the CONFIG_MODE selections as shown in Table 12-2.

In serial configuration mode, the Bypass option connects the DI to DOUT, via a bypass register upon completion of configuration. The bypass register is initialized with a '1' at the beginning of configuration. In parallel configuration mode, the Bypass option causes the data incoming from D[0:7] to be serially shifted to DOUT after completion of configuration. The serialized byte wide register will be shifted to DOUT through the bypass register. D0 of the byte wide data will be shifted out first and followed by D1, D2, and so on.

Once the Bypass option starts, the device will remain in Bypass until the Wake-up sequence completes. One option to get out of the Bypass option is to toggle CSN and CS1N, which will act as a reset signal. Refer to the Master Parallel Mode section of this document for more details.

Flow Though Option
The Flow Through option pulls the CSON pin low when the device has completed its configuration. The Flow Through option can be implemented with either Master or Slave Parallel configuration modes as referenced in

Table 12-2. The Flow Through option will drive out a static low signal on the CSON pin. The Flow Through option will also tri-state the device D[0:7] and BUSY pins when configuration is completed on the device in order to not interfere with the next daisy chained device to be configured.

Once the Flow Through option starts, the device will remain in Flow Through until the Wake-up sequence completes. One option to get out of the Flow Through option is to toggle CSN and CS1N, which will act as a reset signal. Refer to the Master Parallel Mode section of this document for more details.

Master Clock
When the user has determined that a device will be a Master, the CCLK will become an output clock with the frequency set by the user. Until early in the configuration, the device is configured with a default Master Clock Frequency of 2.5MHz. One of the first configuration bits set will be the Master Clock. See the device-specific section of the CFG[0:2] descriptions.

The user can select which Master Clock frequency to use by setting the MCCLK_FREQ preference in the Lattice design software. The MCCLK_FREQ preference will set the frequency of the Master Clock if selected by the CONFIG_MODE and the CFG[0:2] pins. Default is the lowest frequency supported by the device. The user can select a different clock speed, which will take effect just after configuration starts or if the device is reconfigured prior to power down. Configuration time is computed by dividing the maximum configuration bits to be loaded, as given in Figure 12-7, by the Master Clock frequency. See the *LatticeECP/EC FPGA Family Data Sheet* for MCLK_FREQ selections.

SPI Mode

Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
SPI Master	0	0	0	SPI

The LatticeECP/EC devices offer a direct connection for memories that support the SPI standard. By setting the configuration pins CFG[0:2] = b'000, the LatticeECP/EC devices will configure using the SPI interface. The SPI interface offers several combinations of memory to FPGA.

1. One FPGA, one SPI Flash

2. Multiple FPGA, one SPI Flash

3. One FPGA, two SPI Flash

4. Multiple FPGA, multiple SPI Flash is not allowed because the circuitry to support serialization of multiple SPI Flash through DOUT is not available.

For a more detailed discussion the EC/ECP to SPI interface requirements please refer to TN1078.

One FPGA, One SPI Flash
The simple SPI application is one SPI Flash serial connected to the SPID0 of the LatticeECP/EC devices in SPI mode, as shown in Figure 12-1.

Figure 12-1. Simple Interface for FPGA Bootup in SPI Mode
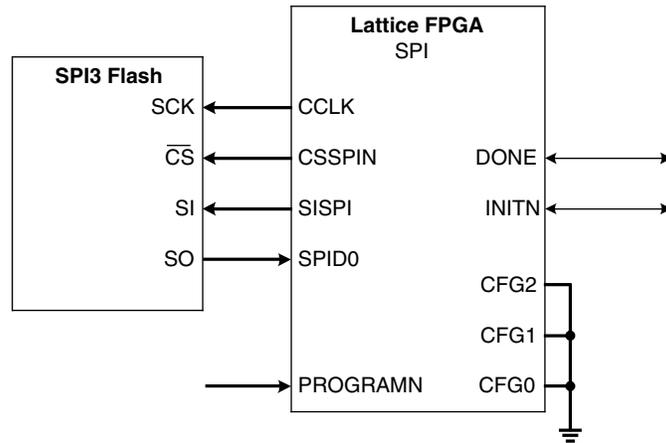
Multiple FPGA, One SPI Flash

With a sufficiently large SPI Flash, multiple FPGAs can be configured as shown in Figure 12-2. The first FPGA is configured in SPI Mode, the following FPGAs are configured in Slave Serial Mode.

Figure 12-2. Multiple FPGAs Configured by One SPI Flash
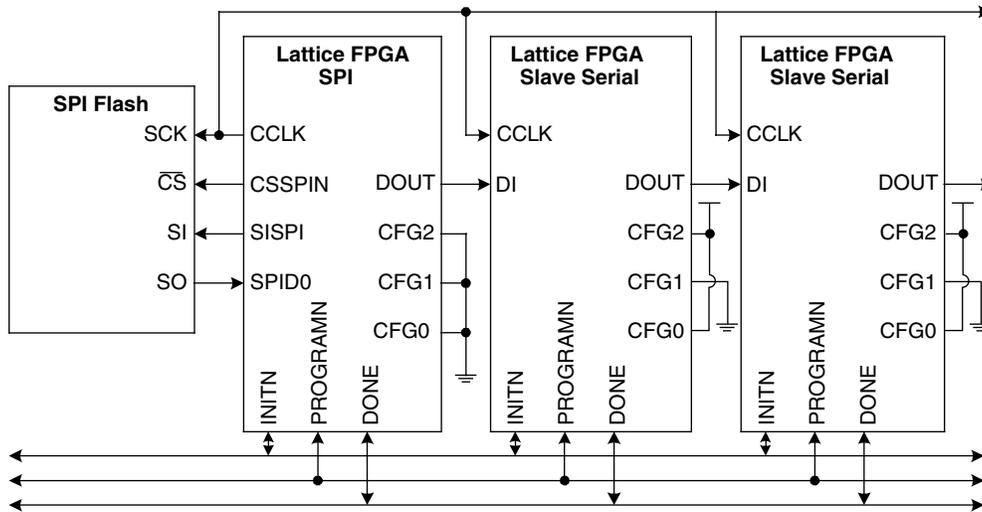
One FPGA, Two SPI Flash

The LatticeECP/EC devices support two Flash to configure a single device as shown in Figure 12-3. The two Flash option is supported to allow use of smaller SPI Flash devices to configure a larger FPGA. Lattice's ispVM® System software divides the configuration bit stream evenly among each selected SPI memory. As the LatticeECP/EC device starts to download from the two SPI memories, the data streams feed into SPID0 and SPID1 in a parallel fashion and are reassembled internally.

Figure 12-3. Two SPI Flash
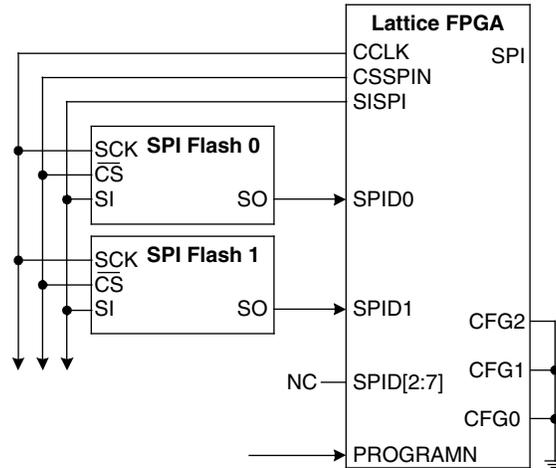
SPIX Mode

Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
SPIX Master	0	0	1	SPIX

Not all SPI memories are the same. A read operation code is required to be fed to the SPI Flash at the time config-uration starts. For many, that op code is 03 Hex. For other memories that require a different read operation code other than 03 Hex, the SPIX format is supported. In SPIX mode the read operation code is coded into the SPID[7:0] pins through the use of pull-ups and pull-downs as shown in Figure 12-4. When configuration begins in the SPIX mode the SPID[7:0] pins are sampled and the corresponding sampled read operation code will be fed to the SPI device so the FPGA can begin read back.

All combinations of SPI Flash and LatticeECP/EC FPGAs are valid in the SPIX mode as well. The only addition is the pull-up and pull-down resistors placed on SPID[7:0] as shown in Figure 12-4.

Figure 12-4. Simple SPIX Example with OP CODE Resistors
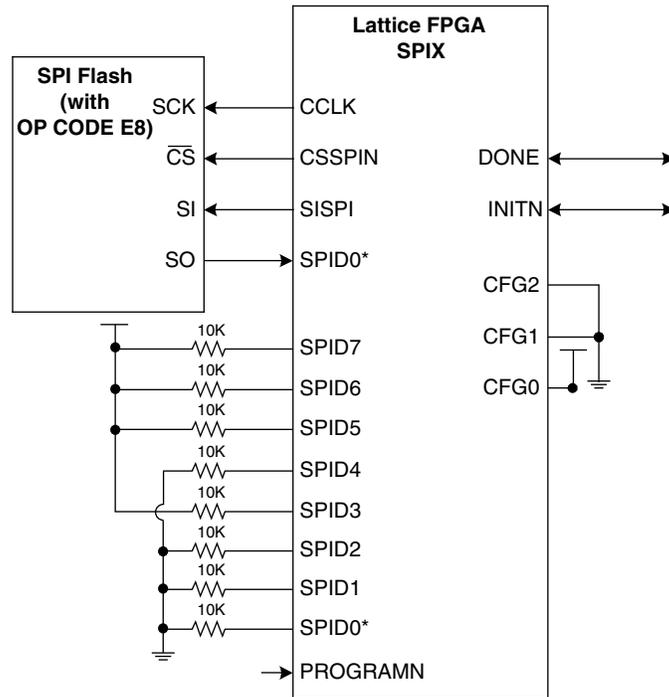
*SPID0 connects to SO and resistor.

Master Serial Mode

Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
Master Serial (no overflow option)	1	0	0	MASTER_SERIAL
Master Serial (Bypass ON)	1	0	0	MASTER_SERIAL_BYPASS

Configuration of the LatticeECP/EC device in Master Serial mode will drive the CCLK signal out to the Slave Serial devices in the chain and the SPROM that will provide the serial bit stream. The device accepts the data at DI on the rising edge of CCLK. The Master Serial device starts driving CCLK after INITN transitions from low to high and continues to drive the CCLK until the external DONE pin is driven high and one hundred plus clock cycles have been generated. The CCLK frequency on power-up defaults to 2.5MHz. The master clock frequency default remains until the new clock frequency is loaded from the bit stream into the device.

If a Master Serial device is daisy chained with other serial devices, once the master device is fully configured, the bypass option will take effect. As additional data is presented to the Master DI pin, the data will be bypassed to the next device on the DOUT pin.

Figure 12-5 shows a master serial daisy chain. The daisy chain method allows multiple Lattice FPGA devices to be configured together. The first device in the daisy chain operates in Master Serial Mode with the Bypass option, while the other Lattice FPGA devices in the daisy chain operate in Slave Serial Mode.

Figure 12-5. Master and Slave Serial Daisy Chained
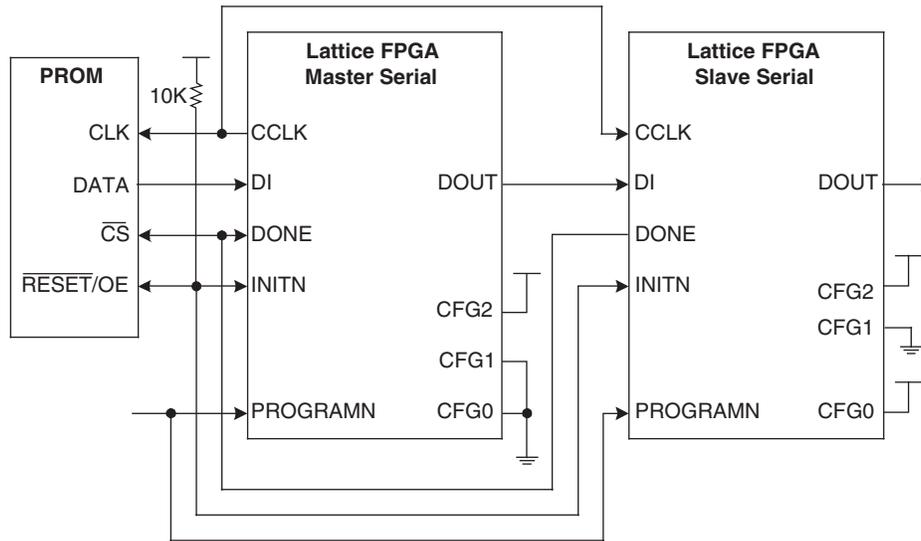
Slave Serial Mode

Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
Slave Serial (no overflow option)	1	0	1	SLAVE_SERIAL (Default)
Slave Serial (Bypass On)	1	0	1	SLAVE_SERIAL_BYPASS

Slave Serial Mode is the default mode for configuration in the Lattice design software. In Slave Serial mode the CCLK pin becomes an input and will receive the incoming clock. The device accepts the data at DI on the rising edge of CCLK. After the device is fully configured, if the Bypass option has been set, data sent to DI will be presented to the next device on the DOUT pin as shown in Figure 12-5.

Master Parallel Mode

Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
Master Parallel (no overflow option)	1	1	0	MASTER_PARLLEL
Master Parallel (Bypass ON)	1	1	0	MASTER_PARLLEL_BYPASS
Master Parallel (Flow Through ON)	1	1	0	MASTER_PARLLEL_FLOWTHR

Configuration using Master Parallel Mode is used to work together with a parallel port PROM without additional external logic. When Master Parallel Mode is chosen, the device will generate CCLK as specified by the MCLK_FREQ preference. The CCLK signal is used to provide a programming clock to the PROM and slave devices. Data is transferred byte wide to the D[0:7] pins. The WRITEN pin must be held low to write to the device. If an overflow option is not selected, the CSN and CS1N pins must be driven low to enable configuration and read back.

The Master Parallel Mode can support two types of overflow, Bypass and Flow Through. If the Bypass option is set, the data presented to the D[0:7] pins will be serialized and bypassed to the DOUT pin when the configuration is complete. If the Flow Through option is set, upon completion of the configuration, the CSOUT signal will drive the following Parallel Mode device chip select as shown in Figure 12-6.

If either overflow option is selected, the CSN or CS1N pins can be toggled to reset the Master Parallel device out of the Overflow option, otherwise both chip select pins should be held low to keep the device active for configuration.

Figure 12-6. Master and Slave Parallel Daisy Chain
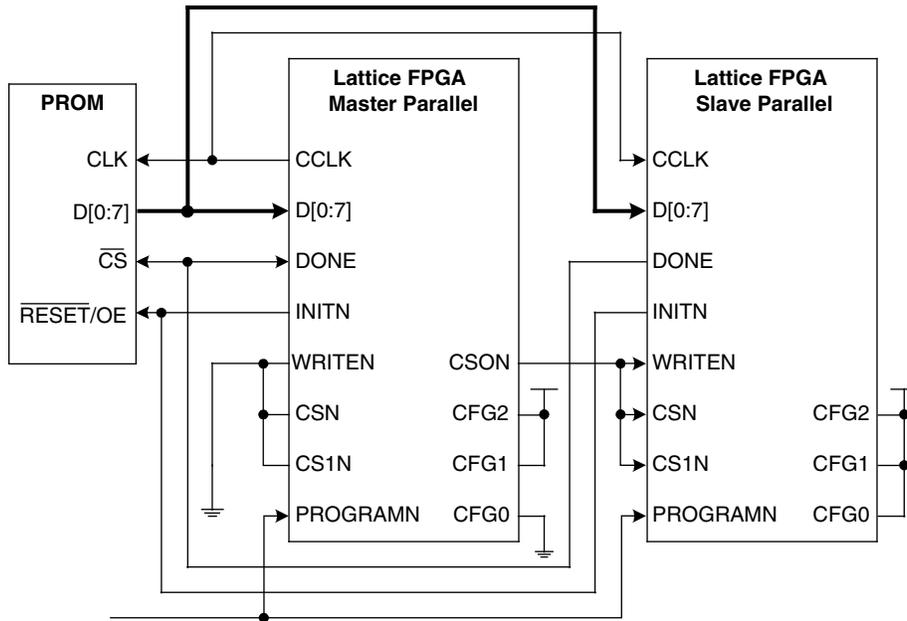
Slave Parallel Mode

Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
Slave Parallel (no overflow option)	1	1	1	SLAVE_PARALLEL
Slave Parallel (Bypass ON)	1	1	1	SLAVE_PARALLEL_BYPASS
Slave Parallel (Flow Through ON)	1	1	1	SLAVE_PARALLEL_FLOWTHR

In Slave Parallel Mode, a host system sends the configuration data in a byte wide stream to the device. The CCLK, CSN, CS1N and the WRITEN signal are provided by the host system such as a Master Parallel mode device as shown in Figure 12-6.

The Slave Parallel configuration mode allows multiple devices to be chained in parallel.

To support asynchronous configuration, where the host may provide data faster than the FPGA can handle it, the Slave Parallel mode can use the BUSY signal. By driving the BUSY signal high, the Slave Parallel device tells the host to pause sending data.

Figure 12-7. Asynchronous Usage of Slave Parallel Configuration Mode
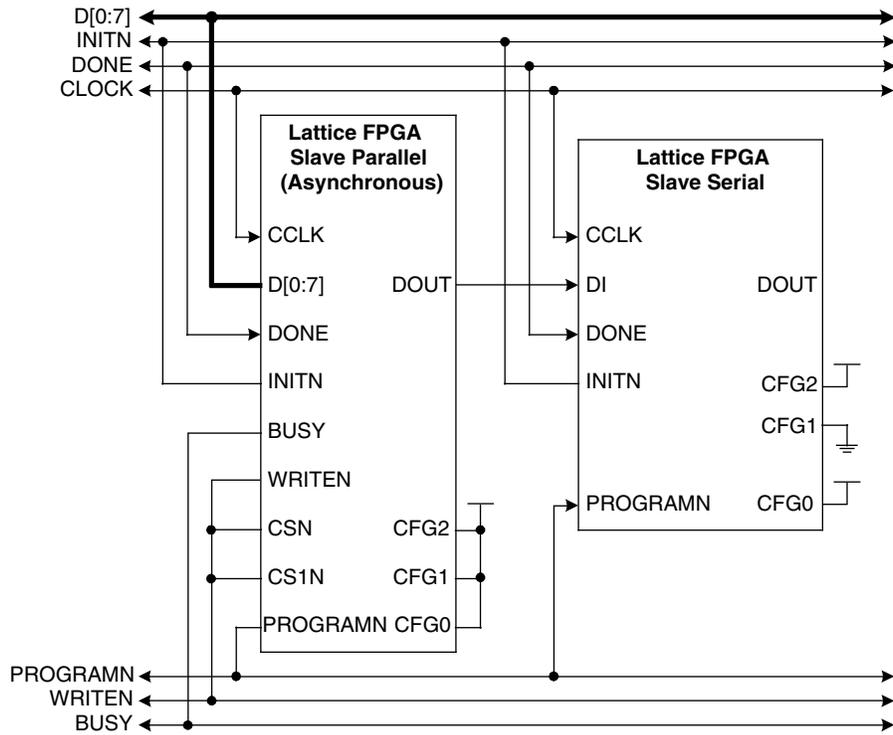
Figure 12-7 shows the Asynchronous peripheral write sequence using the Bypass option. To send configuration data to a device, the WRITEN signal has to be asserted. During the write cycle, the BUSY signal provides handshaking between the host system and the LatticeECP/EC device. When the BUSY signal is 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 or CS1N signals can be used to temporarily stop the write process by setting either to a high state if the host system is busy. The LatticeECP/EC device will resume the configuration when the both CSN and CS1N signals are set low again.

ispJTAG Mode

Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
ispJTAG (1149.1 interface)	X	X	X	Any CONFIG_MODE or NONE1

The LatticeECP/EC device can be configured through the ispJTAG port. The JTAG port is always on and available, regardless of the configuration mode selected. The NONE mode (1) can be selected in the Lattice design software to say that the JTAG port will be used exclusively, but is not required.

ISC 1532
Configuration through the JTAG port conforms to the IEEE 1532 Standard. The Boundary Scan cells take control of the I/Os during any 1532 mode instruction. The Boundary Scan cells can be set to a pre-determined values whenever using the JTAG 1532 mode. Once configuration is complete, an internal Done bit is set, which will release the DONE pin.

Transparent Read Back
The ispJTAG transparent read back mode allows the user to read the content of the device while the device remains in a functional state. The I/O and non-JTAG configuration pins remain active during a Transparent Read Back. The device will enter the Transparent Read Back mode through a JTAG instruction. The user must ensure

that User Mode read/write operations from/to the EBR and distributed RAM do not interfere with the transparent read back.

Boundary Scan and BSDL Files
The LatticeECP/EC BSDL files can be found on the Lattice Semiconductor web site. The boundary scan ring will cover all the I/O pins, dedicated and dual-purpose sysCONFIG pins. The sysCONFIG pins can be observed using the Boundary Scan.

Configuration Flow

The writing to the configuration SRAM memory can generally be split into three phases.

1. **Clear the configuration memory.**
   After power-up or toggling the PROGRAMN pin low, the configuration memory is cleared automatically. The INITN pin is driven high by the EC/ECP device when the device has finished clearing the configuration memory and Done bit. The INITN pin can also be driven externally by the user to delay the configuration process.

2. **Load configuration data into the memory.**
   Loading the bit stream from DI or D[0:7], depending on the selected configuration mode. The INITN pin is set to low on any error and BUSY can be used to delay configuration

3. **Wake up the device.**

The Wake-up sequence puts the device into functional mode after full configuration. Choosing a proper Wake-up sequence is important, to prevent contention.

The following sections describe the three steps to configure LatticeECP/EC devices.

Clearing the Configuration Memory

Two possible methods can clear the internal configuration memory of the LatticeECP/EC device. The first is when the device powers up, the second is by toggling the PROGRAMN pin.

Power-up Sequence
On power-up the device tri-states all the I/Os, sets them to LVCMOS type, and sets the INITN and DONE pin to low. The device prepares for configuration by resetting the configuration circuitry, clearing the DONE bit and CRC registers. The device clears the configuration memory, including I/O options such as PCI clamp, and gets ready to start configuration. The JTAG port is ready to be used as soon as the device clears the configuration memory.

After the device clears the POR, the device samples the Configuration Mode pins CFG[0:2] and recovers the relevant configuration pins according to the Configuration Mode pin settings. The device will then release the INITN pin if the PROGRAMN pin is high. If a Master Mode is selected, the device starts driving the master clock out of the CCLK pin. The INITN pin can be driven low externally to delay device configuration. Once the INITN pin goes high, the device is ready for configuration to start.

Toggling the PROGRAMN Pin
After a device is powered up, toggling the PROGRAMN pin will initiate a sequence to prepare the LatticeECP/EC device for re-configuration from an external memory source. Upon driving the PROGRAMN pin low, the INITN and DONE pins will drive low, the memory will start clearing, and the I/O pins will become tri-stated and pulled up to V_{CCIO}.

Upon driving the PROGRAMN pin high, the CFG[0:2] are sampled to determine the configuration mode to implement as well as which configuration pins will be used for configuration. If a master mode is selected, CCLK will be driven. The INITN pin will be released once the configuration memory is cleared and the PROGRAMN pin is driven high. Holding the INITN pin low will delay configuration. Configuration will begin as soon as the INITN pin is released and pulled high.

Loading the Configuration Memory

Once the PROGRAMN and INITN pins are high, configuration can begin. Depending on the configuration mode selected, data will be accepted on either the DI or D[0:7] pins on the rising edge of CCLK. If an error occurs at any time during transfer of the data, the INITN pin will be driven low by the LatticeECP/EC device. For handshaking configurations, the CSN or CS1N pin can be driven high to pause configuration and stop the Master clock. The BUSY pin can be used by the LatticeECP/EC device to pause the configuration host EC/ECP. Once the full data stream has been shifted in a CRC calculation done during configuration will be compared to the bit stream CRC. If they match, the device will either proceed to the Wake-up sequence or overflow the next data to the next device. If the CRC does not match, then the INITN pin will be driven low and the device will remain in configuration mode.

Wake Up the Device

When configuration is complete, the device will wake up in a predictable fashion. The following selections determine how the device will wake up. Two synchronous wake-up processes are available. One automatically wakes the device up when the internal Done Bit is set even if the DONE pin is held low externally. The other waits for the DONE pin to be driven high externally before starting the wake-up process. The DONE_EX preference determines if the synchronous wake up will be controlled by the external driving of the DONE pin or ignores the external driving of the DONE pin. Table 12-3 provides a list of the wake-up sequences supported by the devices.

Table 12-3. Wake-up Sequences supported by LatticeEC

Sequence	Phase T0	Phase T1	Phase T2	Phase T3
Default		GOE	GSR, GWDIS	DONE
1	DONE	GOE, GWDIS, GSR		
2	DONE		GOE, GWDIS, GSR	
3	DONE			GOE, GWDIS, GSR
4	DONE	GOE	GWDIS, GSR	
5	DONE	GOE		GWDIS, GSR
6	DONE	GOE	GWDIS	GSR
7	DONE	GOE	GSR	GWDIS
8		DONE	GOE, GWDIS, GSR	
9		DONE		GOE, GWDIS, GSR
10		DONE	GWDIS, GSR	GOE
11		DONE	GOE	GWDIS, GSR
12			DONE	GOE, GWDIS, GSR
13		GOE, GWDIS, GSR	DONE	
14		GOE	DONE	GWDIS, GSR
15		GOE, GWDIS	DONE	GSR
16		GWDIS	DONE	GOE, GSR
17		GWDIS, GSR	DONE	GOE
18		GOE, GSR	DONE	GWDIS
19			GOE, GWDIS, GSR	DONE
20		GOE, GWDIS, GSR		DONE
21 (Default)		GOE	GWDIS, GSR	DONE
22		GOE, GWDIS	GSR	DONE
23		GWDIS	GOE, GSR	DONE
24		GWDIS, GSR	GOE	DONE
25		GOE, GSR	GWDIS	DONE

Synchronous to Internal Done Bit

If the LatticeECP/EC 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 the configuration. Once the configuration is complete the internal Done Bit will be set and the wake-up process will begin.

Synchronous to External DONE Signal
The DONE Pin can be selected to delay wake up. If DONE_EX is true the wake-up sequence will be delayed until the DONE pin is driven high externally, then the device will follow the selected wake-up sequence.

Wake-up Clock Selection
The wake-up sequence is synchronized to a clock source, the user can select the clock source for wake up. The clock sources are CCLK, TCK and User Clock. The default clock is TCK if using ispJTAG, and CCLK if using sys-CONFIG. The User Clock is chosen at the time of design. The user can select any of the CLK pins of the device or a net (routing node) as the User Clock source. The WAKEUP_CLK shall default to CCLK or TCK.

Read Back

Read Sequence

To read the configuration memory data or register contents back, WRITEN is first set to low to send the read instruction into the device. The device will read in the command from the host and execute the command once read in. If the LatticeECP/EC device cannot have the data ready by the next clock cycle, it will drive the BUSY pin high. When BUSY is high, the device will continue to execute the command regardless of the state of the CSN or CS1N pins. The device will drive the BUSY pin low when the data is ready but will not drive the D[0:7] until the CSN and CS1N pins are pulled low by the host. The WRITEN pin should be pulled high after sending in the command. The CSN, CS1N and WRITEN signals are latched and the device will switch to read mode on the rising edge of CCLK. If the LatticeECP/EC device needs more than one clock cycle to switch the bus around, BUSY will be kept high until the D[0:7] is ready.

As in the Write sequence, CSN and CS1N signals can be used to temporarily pause the read sequence in case the host system is busy. The data is read at the next rising CCLK edge, after CSN and CS1N pins are set to low and the BUSY pin is low.

Transparent Read Back
Using the Slave Parallel Mode for read back, the user I/Os will remain functional. The Slave Parallel port pins must be retained in order to allow read back by setting the PERSISTENT preference to ON. CCLK becomes input only. The user must ensure that User Mode read/write operations from/to the EBR and distributed RAM do not interfere with the transparent read back.

Configuration Mode Read Back
Read back can also be done with the LatticeECP/EC device in configuration mode. Only the Slave Parallel Mode is supported for configuration read back. By driving the WRITEN pin high, the Slave Parallel port will watch for the read back request from the host device.

Software Control

In order to control the configuration of the LatticeECP/EC device beyond the default settings, software preferences can be used. Table 12-4 is a list of the preference, the default settings and the section more information about the preference can be found.

Table 12-4. LatticeECP/EC Device Preference List

Preference (Preference)	Default Setting (All Settings)
PERSISTENT	ON [off, on]
CONFIG_MODE	SLAVE_SERIAL (see Table 12-2)
DONE_OD	ON [on, off]
DONE_EX	OFF [off, on]
MCCLK_FREQ	Lowest Frequency (see device tables)
CONFIG_SECURE	OFF [off, on]
WAKE_UP	21 (DONE_EX = Off) [1:25] 4 (DONE_EX = On) [1:7]
COMPRESS_CONFIG	OFF (off, on)

Persistent

When using the sysCONFIG port to perform transparent readback the PERSISTENT preference must be set to ON, this reserves the dual-purpose pins for configuration. PERSISTENT = ON prevents the user and the software fitter from using any of the dual-purpose pins as general purpose I/O.

Configuration Mode

The device knows which physical port will be used by reading the highs and lows wired to the CFG[0:2] pins, however sometimes software needs to know the setting of the CFG pins as well. The Configuration Mode serves this purpose. The fitter will be prevented from using the I/O pins associated with the selected Configuration Mode, however the user may assign these pins (a warning will be generated by the software). If the user requires these pins as general purpose I/O they must adhere to the requirements listed above in the Configuration Pins section.

Normally, proper selection of the Configuration Mode is not strictly required. For pin reservation, for instance, the user can place prohibits in the source code to insure that the fitter doesn't use the configuration pins as general purpose I/O. However, if multiple devices are being configured from one configuration device then the Configuration Mode must match the CFG pins. This is required because the overflow option (Flow Through or Bypass) is determined by the Configuration Mode (in software) and the CFG pins (in hardware). Note that if either overflow option is selected, then the DONE_EX and WAKE_UP selections will be changed to correspond. See Table 12-5 for details on the Overflow Option defaults. For more information on the overflow options, see the Configuration Options section of this document.

Table 12-5. Overflow Option Defaults

Overflow Option (Bypass, Flow Through)	DONE_EX Preference	WAKE_UP Preference
Off	Off (default)	Default 21 (user selectable 1 through 25)
Off	On	Default 21 (user selectable 1 through 25)
On	ON (automatically set by software)	Default 4 (user selectable 1 through 7)

DONE Open Drain

The "DONE_OD" preference allows the user to configure the DONE pin as an open drain pin. The "DONE_OD" preference is only used for the DONE pin. When the DONE pin is driven low, internally or externally, this indicates that programming is not complete and the device is not ready for wake up. Once configuration is complete, with no errors, and the device is ready for wake-up, the DONE pin must be driven high. For other devices to be used to control the wake-up process an open drain configuration is needed to avoid contention on the DONE pin. The "DONE_OD" preference for the DONE pin defaults to ON. The DONE_OD preference will be automatically set to the default if the DONE_EX preference is set to on. See Table 12-6 for more information on the relationship between DONE_OD and DONE_EX.

DONE External

The LatticeECP/EC device can wake up on its own after the Done Bit is set or wait for the DONE pin to be driven externally. The DONE_EX preference will determine if the wake-up sequence is triggered by an external DONE signal. The DONE_EX preference shall take a user entered ON or OFF. ON if the user wants to delay wake-up until the DONE pin is driven high by an external signal and synchronous to the clock. The user will select OFF to synchronously wake up when the internal Done bit is set and ignore any external driving of the DONE Pin. The default for DONE_EX preference is OFF. If DONE_EX is set to ON, DONE_OD should be set to the default value of ON. If and external signal is driving the DONE pin, it should be an open drain pin. See Table 12-6 for more information on the relationship between DONE_OD and DONE_EX.

Table 12-6. Summary of DONE Pin Preferences (Preferences)

DONE_EX	Wake-up Process	DONE_OD
OFF	External DONE ignored	User selected
ON	External DONE Low delays	Set to Default (ON)

Master Clock Selection

When the user has determined that the LatticeECP/EC device will be a Master Configuration device and will provide the clocking source for configuration, the CCLK will become an output clock with the frequency set by the user. At the start of configuration the device operates with the default Master Clock Frequency of 2.5MHz. One of the first configuration bits set will be the Master Clock. Once the Master Clock configuration bits are set, the clock will start operating at the user-defined frequency.

In order to control the Master Clock frequency, the MCCLK_FREQ preference can be set. The MCCLK_FREQ preference shall set the frequency of the MASTER clock if selected by the CONFIG_MODE and the CFG[0:2] pins. See the LatticeECP/EC data sheet for the Master Clock frequencies supported by the MCLK_FREQ preference.

Security

When CONFIG_SECURE is set to ON, NO read back operation will be supported through the sysCONFIG port or ispJTAG port of the configuration SRAM. The USERCODE register is readable and not considered securable. Default is OFF. OFF indicates read back of the configuration memory is enabled through all ports.

Wake-up Sequence

The wake-up sequence controls three internal signals and the DONE pin will be driven post configuration and prior to user mode. See the Wake-up Sequence section of this document for an example of the phase controls and the device-specific section for specific info on the wake-up selections. The default setting for the WAKE_UP preference will be determined by the DONE_EX setting.

Wake-up with DONE_EX = Off (Default setting)
The WAKE_UP preference will support the user-selectable options (1-25) as shown in Table 12-3. If the user does not select a wake-up sequence, the default will be wake-up sequence 21 for DONE_EX preference set to OFF (Default).

Wake-up with DONE_EX = On
The WAKE_UP preference will take the user selectable options (1-7) as shown in Table 12-3. If the user does not select a wake-up sequence, the default will be wake-up sequence 4 for the DONE_EX preference set to ON.

Start_Up Clock Selection

Once the FPGA is configured, it enters the start-up state, which is the transition between the configuration and operational states. This sequence is synchronized to a clock source, which defaults to CCLK when sysCONFIG is used, or TCK when JTAG is used.

If desired, a user-defined clock source can be used instead of CCLK/TCK. You need to specify this clock signal, and instantiate the STRTUP library element in your design. The example shown below shows the proper syntax of instantiating the STRTUP library element.

Verilog

```
STRTUP u1 (.UCLK(<clock_name>)) /* synthesis syn_noprune=1 */;
```

VHDL

```
component STRTUP
    port(STRTUP: in STD_ULOGIC );
end component;

attribute syn_noprune: boolean ;
attribute syn_noprune of STRTUP: component is true;
begin
u1: STRTUP port map (UCLK =><clock name>);
```

Bit Stream Compression

The LatticeECP/EC devices support bit stream compression. When the Compression preference is set to ON, the Lattice design software will generate a compressed version of the bit stream file internally along with an uncompressed bit stream file. The LatticeECP/EC devices will route the compressed bit stream through the decompression engine when the COMPRESS_CONFIG preference is set to ON. The COMPRESS_CONFIG preference defaults to OFF. It is possible for the compressed bit stream to be larger than the uncompressed bit stream.

Table 12-7. LatticeECP/EC Configuration Memory Requirements

Family	Device	Max. Config. Bits (Mb)	Required Boot Memory (Mb)	
			Without Compression	Typical Compression
LatticeECP/EC	EC1	0.6	1	25%
	EC3	1.1	2	25%
	ECP/EC6	1.8	2	25%
	ECP/EC10	3.1	4	25%
	ECP/EC15	4.3	8	25%
	ECP/EC20	5.3	8	25%
	ECP/EC33	7.9	8	25%

SPI Compatible SPI Flash Vendors

• ST Microelectronics - M25Pxx

• Winbond - W25Pxx

• Silicon Storage Technology - SST25VFxx

• Spansion - S25FLxx

• PMC pFLASH - Pm25LVxx

• Atmel - AT25Fxx

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

 +1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
June 2004	01.0	Initial release.
July 2004	1.1	Changed DO/CSON to DOUT/CSON.
		Clarified signal names associated with data pins during SPI and other modes.
		Indicated value for INITN pull up resistor in "Master and Slave Serial Daisy Chained" diagram
		Added section detailing compatible SPI flash vendors.
October 2004	1.2	Update table 6-7 for formatting and compression %.
May 2005	2.0	Changed SPI3 to SPI
		Updated INITN description to include the power cycling instruction for SPI mode
		Updated CCLK to include toggling the CCLK continuously throughout configuration and error recovery
		Updated dual-purpose use pin
		Updated Master Parallel Mode
October 2005	2.1	Added information on how to use the sysCONFIG dual-purpose pins as GPIO
April 2006	2.2	Updated Wake-Up Options
July 2008	2.3	Updated LatticeECP/EC Device Preference List table.
		Replaced Wake-up Clock Selection text section with new Start_Up Clock Selection text section.
September 2008	02.4	Updated CCLK text section.

February 2006   **Technical Note TN1054**

Introduction

This document describes the functionality and usage of ispTRACY™, Lattice's integrated logic analyzer for the ispXPGA®, LatticeSC™, LatticeECP2™, LatticeECP™, LatticeEC™ and LatticeXP™ FPGA families. The isp-TRACY tool consists of an Intellectual Property (IP) hardware block and three software tools – Core Generator, Core Linker and ispLA. ispTRACY allows for fast debugging and functional verification inside Lattice FPGA devices without the need for expensive test and measurement equipment. Debugging is accomplished through the hardware IP compiled in the design, on device block RAM and the device JTAG port.

ispTRACY IP Core Features

The ispTRACY IP core is highly configurable. These configurable features include width and depth of data capture lines, multiple edge and level sensitive trigger signals, complex comparison for trigger events, delayed trigger events and more. ispTRACY allows multiple ispTRACY IP cores to be included in a single design. The following table summarizes the features of the ispTRACY IP core.

Table 14-1. ispTRACY IP Core Features

Feature	Description
Depth of Memory Capture	256 to 4096 samples
Data Capture Width	8 to 256 bits
Triggering Schemes	Rising/falling edges, level logic, comparison, trigger after combination of events
Number of Triggers	4 to 128 bits, can be a combination of edge and level sensitive signals
Number of Core	Up to 16 ispTRACY cores

ispTRACY IP Module Generator

To include ispTRACY cores in a design, the first step is to run IPexpress™ from the ispLEVER® Project Navigator. Figure 14-1 shows the launch button for the IP Manager program.

Figure 14-1. IPexpress Launch Button in ispLEVER Project Navigator
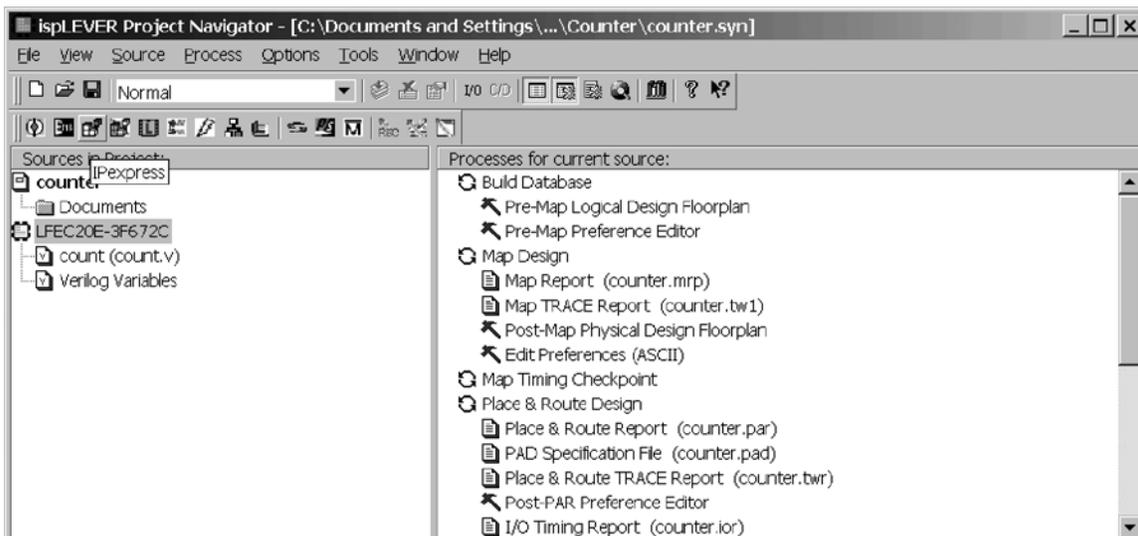
Once IPexpress is launched, you will be presented with the option of generating the ispTRACY IP module by selecting the JTAG module uner Architecture and clicking the Customize button. Figure 14-2 shows the IPexpress window. Selections for the project path and module name are made through this window

Figure 14-2. ispTRACY IP Manager Program Window
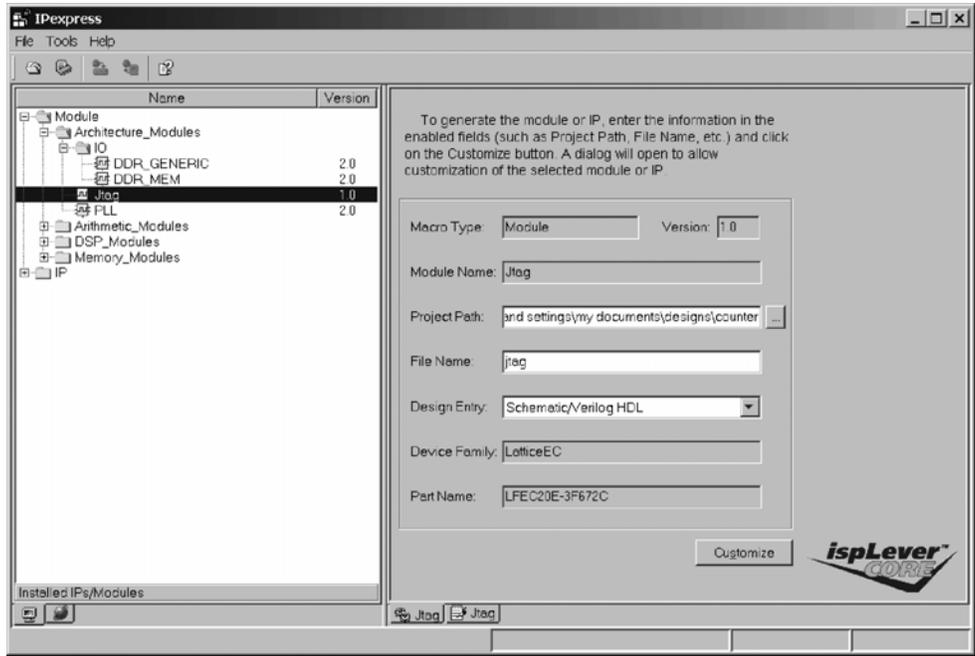
ispTRACY Core Generator

The ispTRACY Core Generator under JTAG Module provides all the controls for customizing the ispTRACY core(s). Selections on this page influence the final size of the core(s) inside the FPGA and features available in terms of triggers, size of data bus and depth of memory capture. Figure 14-3 shows the Core Generator window and Table 14-3 contains descriptions of each of the figures available in the IP core. Once the core features are selected, clicking on the Generate button will create the necessary files for the Core Linker Program.

Figure 14-3. ispTRACY Core Generator Window
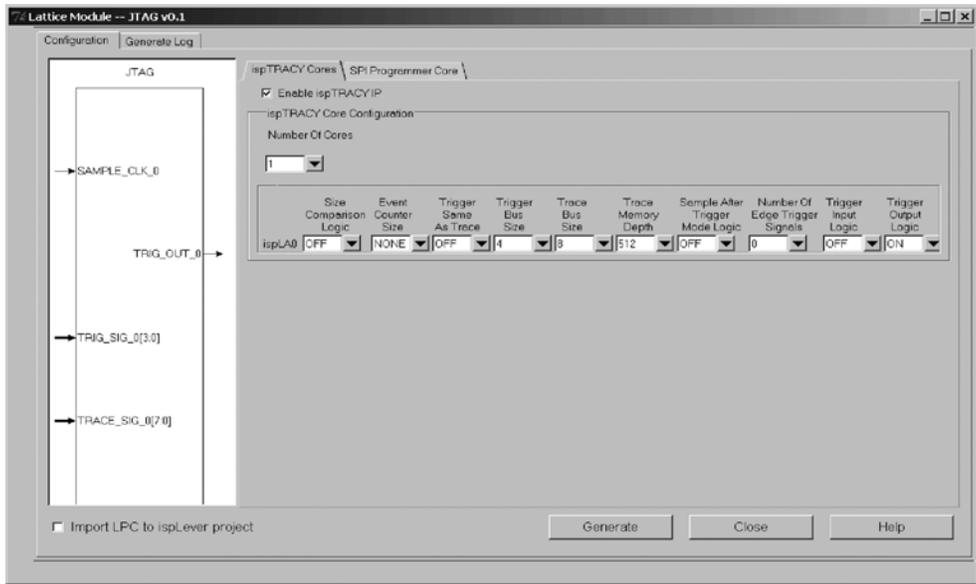
Table 14-2. ispTRACY Core Generator Features and Descriptions

Feature	Description
Number of Core	The number of ispLA(s) in an XPGA can be either 1, 2, 3, ..., or 16. The ispLA needs to be configured before use. After [Generate] button is clicked, the ispTRACY software will generate the required logic for each ispLA based on its own configuration.
ispTRACY Core	Lists the ispLA.
Size Comparison Logic	The comparison logic can compare the trigger bus with the patterns setting by the user. This field needs to be ON for ">", ">=", "<", "<=" comparison. If it's OFF, only = (equal) and <> (not equal) comparison can be preformed.
Event Counter Size	This field configures the size of the event counter. If this field is 8, the counter can be set to the value from 1 to 255. If this field is 16, the counter can be set to the value from 1 to 65535. If this field is "None", the counter logic is removed (i.e. counter value N always equal 1). If the counter value is set to N, then when the pattern occurs N times, the corresponding event will be TRUE.
Trigger Same as Trace	If this is ON, the trace bus and trigger bus are the same bus. If this is OFF, the trace bus and trigger bus are different and they can have different bus sizes.
Trigger Bus Size	This specifies the trigger bus size. It can be 4, 5, 6, ..., up to 128.
Trace Bus Size	This specifies the trace bus size. It can be a multiple of 8 up to 256 (i.e. 8, 16, 24, 32, 40,..., up to 256).
Trace Memory Depth	This is the depth of the trace memory. It defines the number of trace bus samples that ispLA can capture. This field can be set be set to 512, 1024, 2048, or 4096. It can also be set to 256 if the trace bus size is a multiple of 16 (i.e. 16, 32, 48, etc.).
Sample_After_Trigger Mode Logic	The field causes the Sample_After_Trigger mode logic to be removed or not. If this is ON, the trace mode can be set to "One Shot" mode or "Sample After Trigger" mode. If this is OFF, the trace can only be running at "One Shot" mode. When this logic is turned off, the ispLA will use less logic.
Number of Edge Trigger Signals	The trigger bus signals can be either edge sensitive signals or level sensitive signals. The level sensitive trigger signals can only be set to 0, 1 or X (don't care). The edge sensitive trigger signals can be set to 0, 1, X, R (rising edge), F (falling edge) or B (both edges). This field specifies the number of edge sensitive trigger signals.
Number of Level Trigger Signals	This field specifies the number of level sensitive trigger signals. Note that (Number of Edge Trigger Signals) + (Number of Level Trigger Signals) = (Trigger Bus Size).
Trigger Input Logic	This specifies if the "Trigger Input" logic exists. If this field is "None", the logic will be removed and the trigger condition can only be set using EV0 and EV1. If this field is set to "Pin" the trigger input logic exists and the trigger input should come from an ispXPGA device I/O pin. The trigger input can be set to either active low or active high.
Trigger Output Logic	This specifies if the "Trigger Output" logic exists. If this field is "None", the logic will be removed. If this field is set to "Pin" the trigger output logic exists and the trigger output should go out through an XPGA device I/O pin. If this is set to "ispLA", the trigger output will be connected to the trigger input of other ispLAs. You must choose "Trigger Input Logic" of the other ispLAs to be this ispLA. Same as the trigger input, the trigger output can be set to either active low or active high. At least one ispLA should have this option set to "Pin."
Generate	Generates the core.
Cancel	Cancels the action and closes the dialog box without saving any changes.
Help	Displays online Help topics for this dialog box.

ispTRACY Core Linker

Once the ispTRACY core is created, it must be linked into the target design. This is accomplished through the Core Linker program. The Core Linker program allows the user access to internal and external signals of the target design. The internal signals can be named signals or component ports. This window also displays the available ispTRACY ports. To connect ispTRACY signals, the desired signal(s) are selected in the left-hand signal window. Signals chosen from this window are reflected in the Selected signals window. Signals must be highlighted in this window, the ispTRACY port window and then click on the connect button to connect the signals in the RTL code. Multiple instances (for example, a data bus) can be connected at once by highlighting the first signal, holding down the SHIFT key and clicking on the last desired signal. Figure 14-4 shows the ispTRACY Core Linker window.

Figure 14-4. ispTRACY Core Linker Program Window
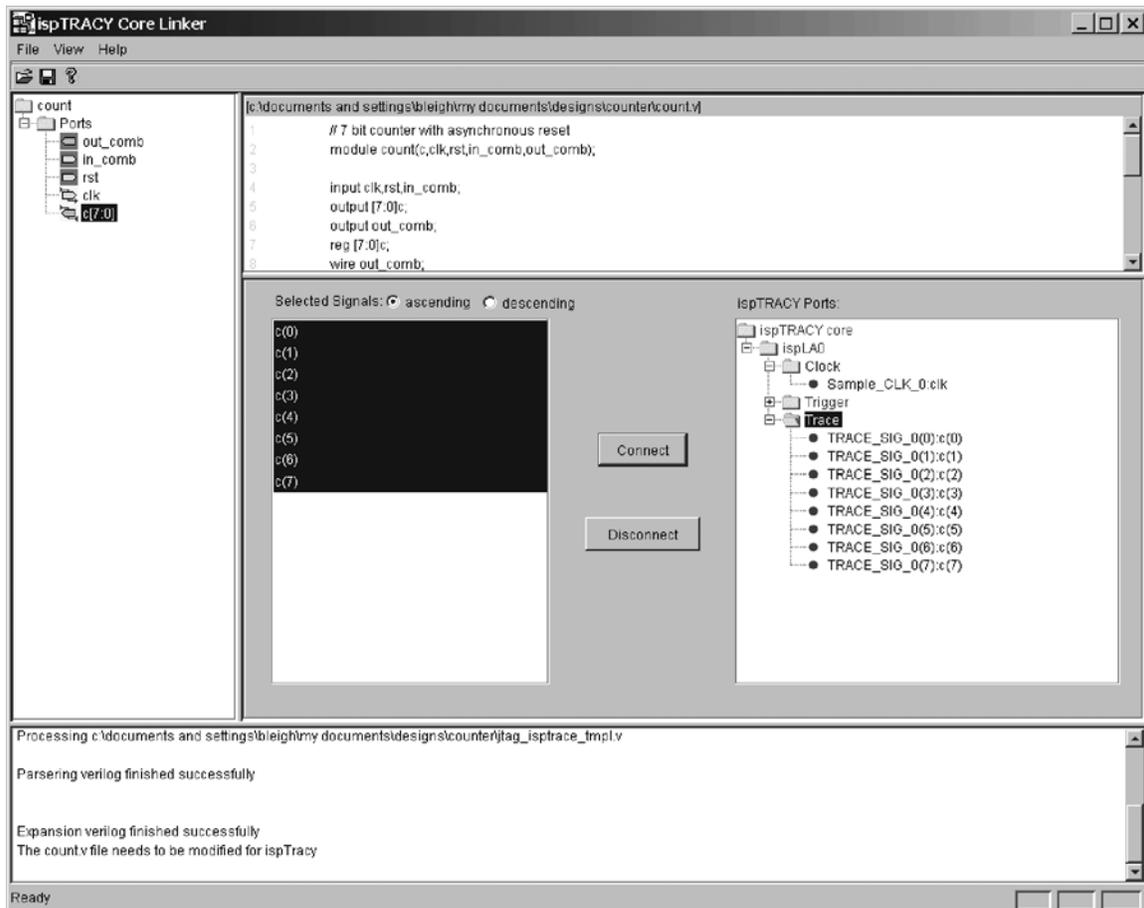
When you click the Save button (or File -> Save menu selection), the Core Linker will create modified versions of your source file, with the ispTRACY core linked into these modified files. Only design files that are directly connected to the core will be modified. A dialogue box will indicate which files have been changed and will need to be replaced in the design project for ispTRACY to function. The design files names will be the original files names with the module name for the ispTRACY core (from the IPexpress ispTRACY core generation) appended. Figure 14-5 shows the changed files dialogue box.

Figure 14-5. ispTRACY Core Linker Output Window
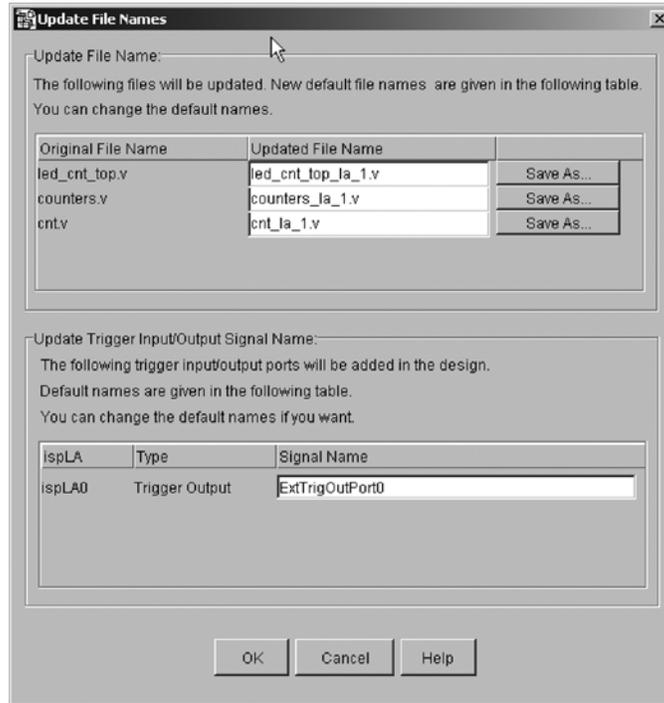
After clicking OK, you will be back in the ispTRACY Core Linker. You may now close this and return to the isp-LEVER Project Navigator. At this point, it is necessary to replace the original design files with the ispTRACY Core Linker modified files. Figure 6 shows this file replacement process on a design.

Figure 14-6. Original Project Navigator (left) and ispTRACY Project Navigator (right)
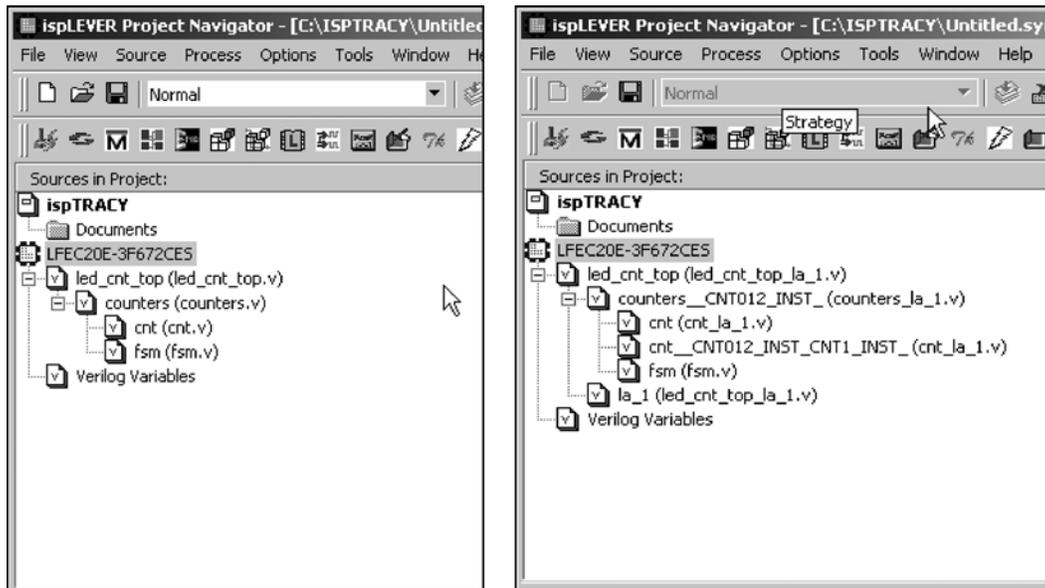
ispTRACY ispLA Program

After replacing the original design files with the ispTRACY Core Linker modified files, it is necessary to re-compile the design and then program the device using the ispVM® software (see online documentation for running the ispVM program). Once the device is successfully programmed, the ispLA program needs to be started. Figure 14-7 shows the ispLA launch button from the Project Navigator.

Figure 14-7. ispLA Launch Button in ispLEVER Project Navigator
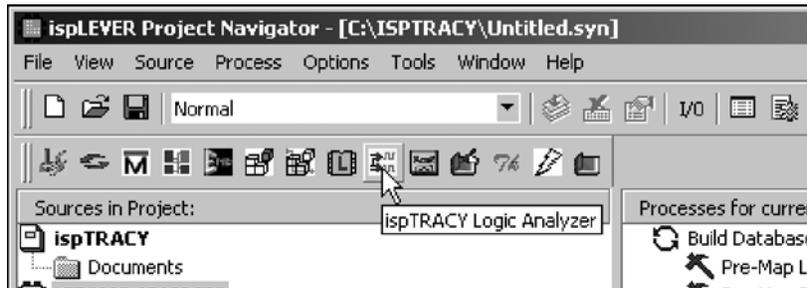
ispLA provides for capture and display of data from the ispTRACY core. The program is used to setup the trigger events counters, trigger location in data buffer, event patterns, event comparisons and data display. Figure 14-8 shows the initial startup windows required to run ispLA.

Figure 14-8. ispLA Program Window – New Project Setup
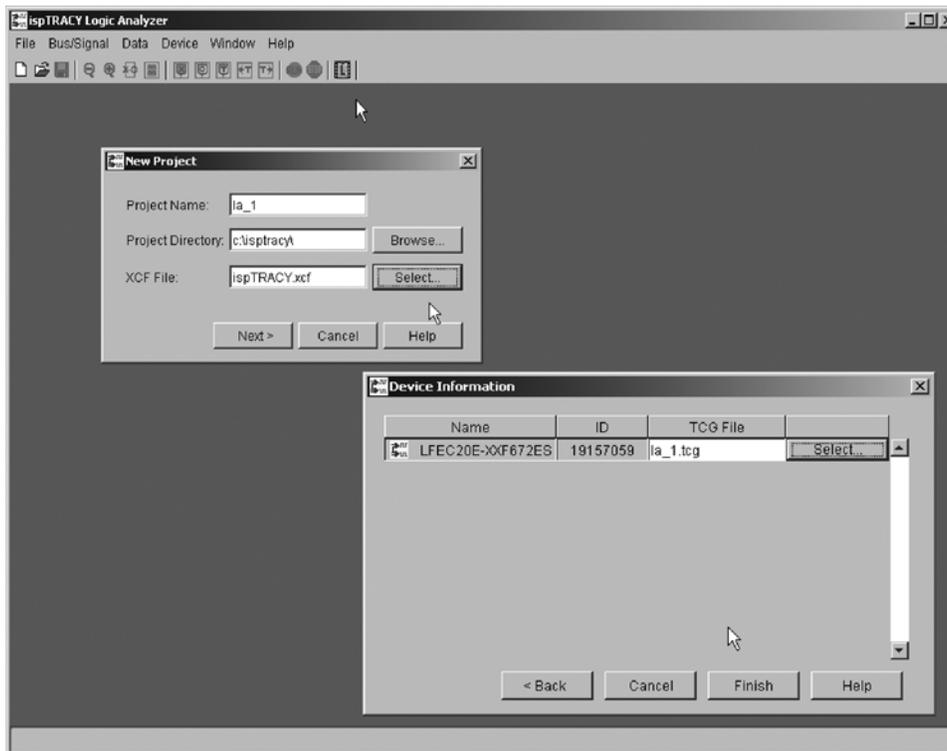
Once the project is set up, you can open up a trigger setup and viewing window for the project by clicking on Window -> Show ispLA Window -> Device 0 -> Device 0 LA0. Figure 9 shows the ispLA window. There are three tabs available under this window, trigger setup, event pattern and signal analysis (data view).

Figure 14-9. ispLA Project Setup Window - Trigger Setup options
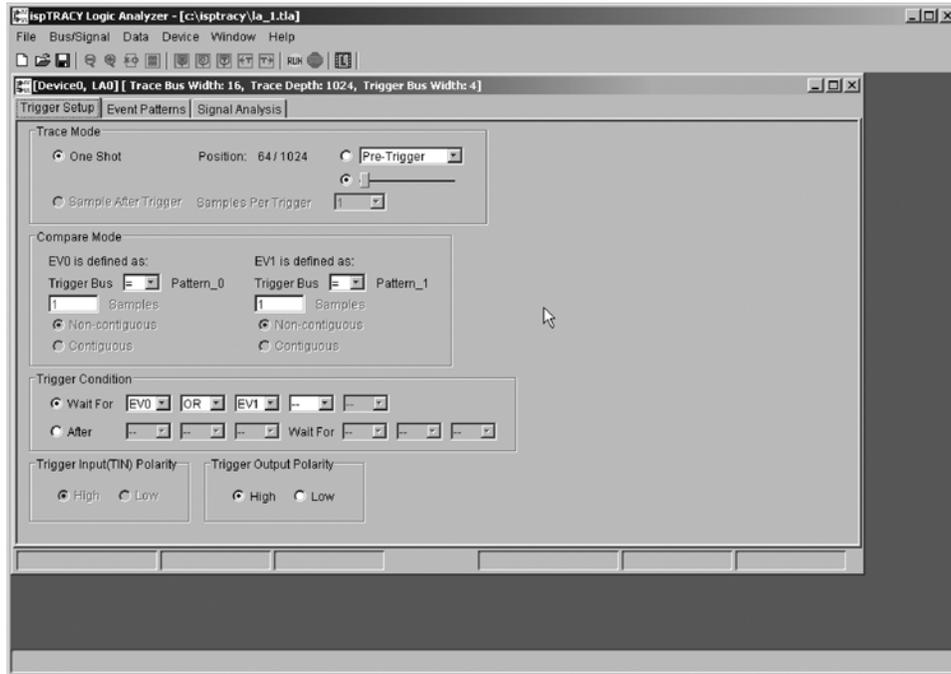
The options available in the Trigger setup window are based on selected ispTRACY core options. In the Trace Mode box, One Shot mode will always be available, but Sample After Trigger availability is dependent on selecting Sample_After_Trigger Mode Logic => ON. The position slider can be used to select the trigger point anywhere within the data memory depth. There are three preconfigured trigger positions in the drop-down menu box. They are Pre-Trigger (5% before and 95% after trigger), Center (50% before and 50% after trigger) and Post Trigger (95% before and 5% after trigger). In the Compare Mode box the options are also dependant on core configuration. EV0 and EV1 are always available. The comparisons available and number of samples will be determined by the Size Comparison Logic and Event Counter Size. The equal to comparison is always available. Additions comparisons include >, <. !=, <=, >=. The Trigger Condition box configures which event or combination of events will cause the ispLA program to trigger and upload captured data from the device. In a simple case, this would be set to Wait for EV0. More complex cases could possible be wait for EV0 and EV1, or after EV1 wait until EV0. The final two boxed on this screen control the signal polarity of Trigger In (if available) and Trigger Out.

Figure 14-10. ispLA Project Setup Window - Event Pattern Setup
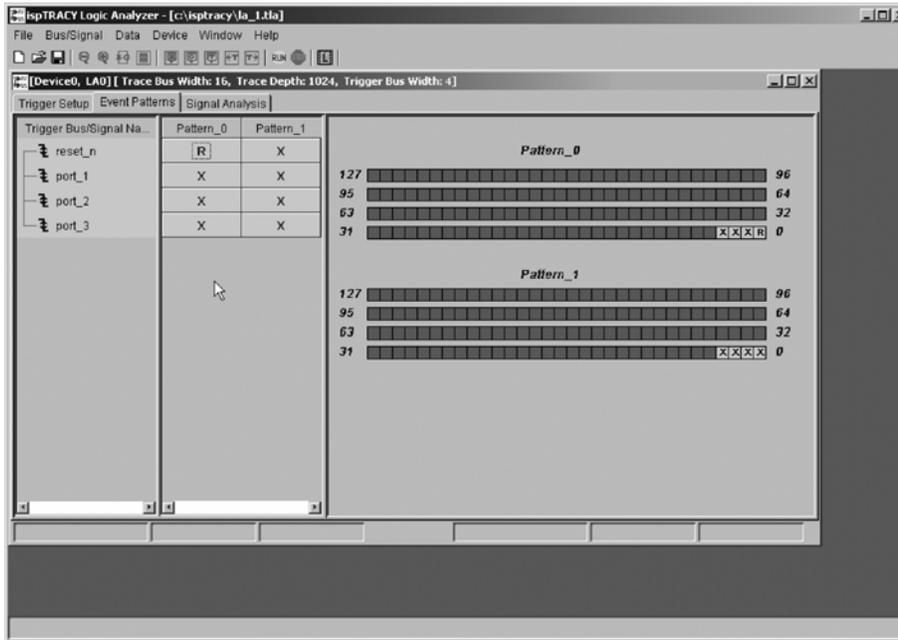
The Event Pattern window configures the patterns for EV0 and EV1. Pattern 0 corresponds to EV0 and Pattern 1 corresponds to EV1. The sample types for trigger signals depends on whether the signal is edge or level sensitive. If the signal is level sensitive, in the middle pattern window, only logic 0 (0) ,logic (1) or don't care (X) are available. For edge sensitive signals, the options are logic 0 (0), logic 1 (1), rising (R), falling (F), both/either edge (B) or don't care (X). In this window, changes to the pattern are made in the center section (Pattern 1 or Pattern 0) and the changes are reflected in the right section.

To begin sampling , click on the green run button. Sampling will complete when the trigger conditions are met. The data will be uploaded through the JTAG cable and results displayed I the Signal Analysis window. Figure 14-11 shows the Signal Analysis window after a data capture cycle.

Figure 14-11. ispLA Signal Analysis Window
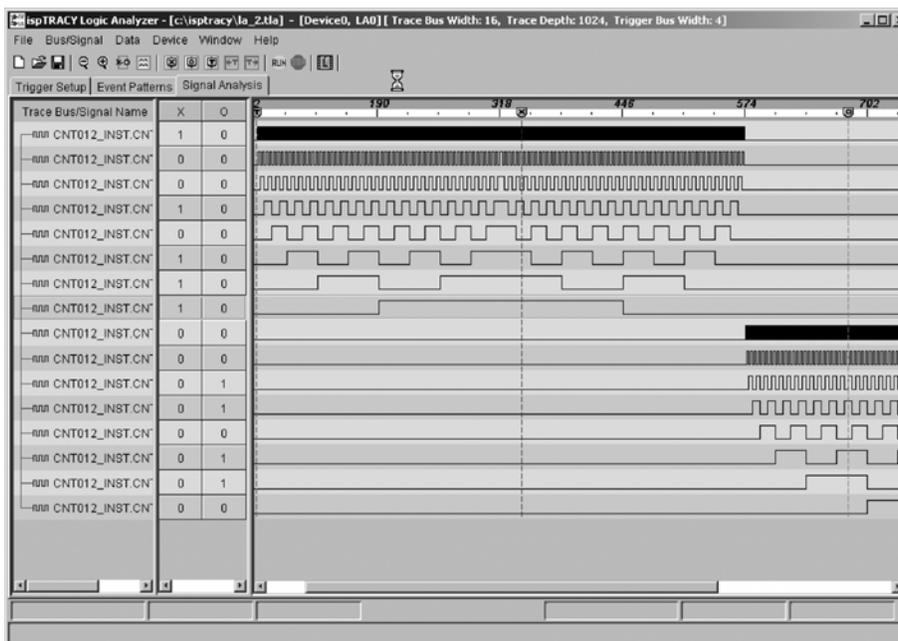
Conclusion

ispTRACY is a full-featured logic analysis tool for use in Lattice FPGA products including ispXPGA, LatticeECP/EC and LatticeXP families. Using internal device resources, including PFF/PFU, Embedded Block Ram and the device JTAG port, the user can quickly verify functionality and assist in device debugging. ispTRACY reduces the need for external test and measurement equipment to debug FPGA projects, while providing full access to a wide range of internal signals, components and design elements.

References

- *ispTRACY Usage Guide,* 2/07/04 Rev. 0.1, page 14 of 14.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

+1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

LatticeECP-DSP sysDSP Usage Guide

Introduction

This technical note discusses how to access the features of the LatticeECP™-DSP sysDSP™ (Digital Signal Processing) Block described in the LatticeECP/EC Family data sheet. Designs targeting the sysDSP Block offer significant improvement over traditional LUT-based implementations. Table 15-1 provides an example of the performance and area benefits of this approach.

Table 15-1. sysDSP Block vs. LUT-based Multipliers

Multiplier Width	Register Pipelining	LatticeECP LFECP20E-5 Uses One DSP Block		LatticeEC LFEC20E-5 Uses LUTs	
		f_{MAX}	LUTs	f_{MAX}	LUTs
9x9	Input, Multiplier, Output	235	0	76	174
18x18	Input, Multiplier, Output	211	0	50	608
36x36	Input, Multiplier, Output	177	0	35	2225

sysDSP Block Hardware

Overview

The sysDSP Blocks are located in a row at the center of the LatticeECP-DSP device. A sysDSP Block block diagram is shown in Figure 15-1.

Figure 15-1. LatticeECP sysDSP Block Diagram
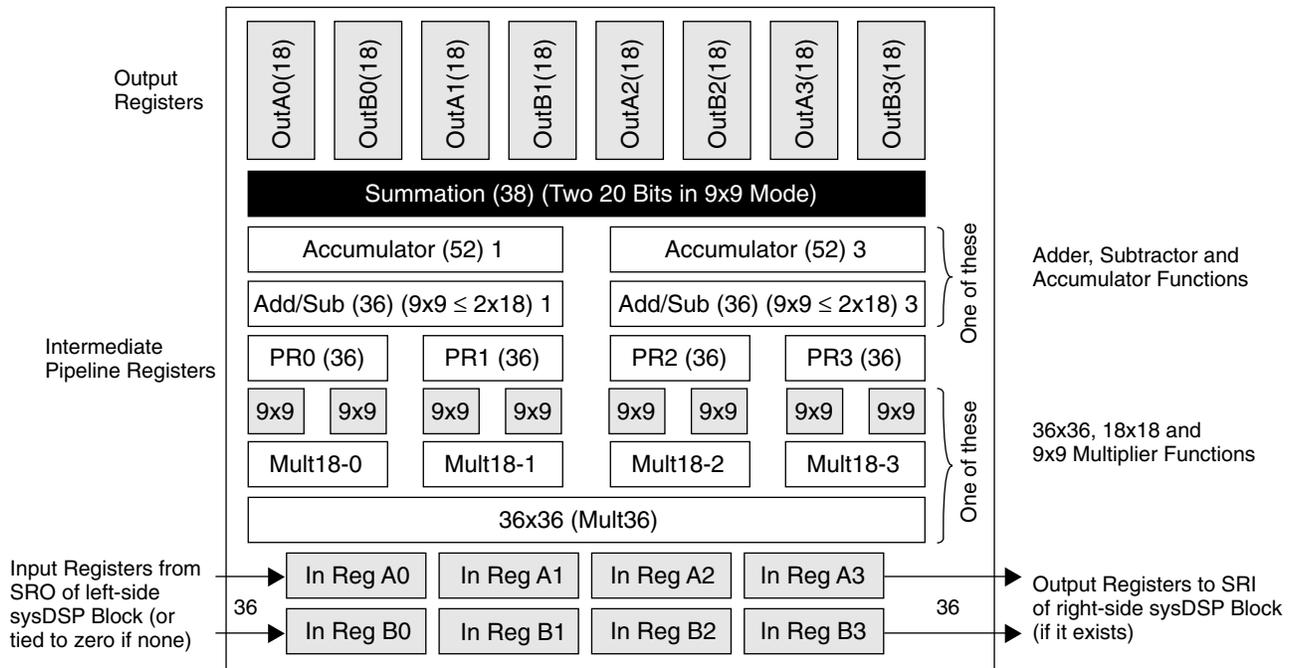
Note: Each sysDSP Block spans eight columns of PFUs.

sysDSP Blocks have three operating modes:

36x36 Mode
- One 36x36 multiplier

18x18 Mode
- Four Multipliers
- Two 52-bit MACs
- Two sums of two 18x18 multipliers each
- One sum of four 18x18 multipliers

9x9 Mode
- Eight Multipliers
- Two 34-bit MACs
- Four sums of two 9x9 multipliers each
- Two sums of four 9x9 multipliers each

sysDSP Block Software

Overview

The sysDSP Block of the LatticeECP-DSP device can be targeted in a number of ways.

- IPexpress™ in the Lattice ispLEVER® design tools allows the rapid creation of modules implementing sysDSP elements. These modules can then be used in HDL designs as appropriate.
- The coding of certain functions into a design's HDL allows the synthesis tools to inference the use of a sysDSP Block.
- The implementation of designs in Mathwork's Simulink tool using a Lattice Block set. The ispLEVER sysDSP portion of the ispLEVER tools then converts these blocks into HDL as appropriate.
- Instantiation of sysDSP primitives directly in the source code.

Targeting the sysDSP Block Using IPexpress

IPexpress allows you to graphically specify sysDSP elements. Once the element is specified, an HDL file is generated which can be instantiated in a design. IPexpress allows users to configure all ports and set all available parameters. The following show modules which target the sysDSP Block. For design examples using IPexpress, refer to EXAMPLES in your ispLEVER software. From the Project Navigator pull-down menu, go to File -> Open Example). The following four element types can be specified via IPexpress:

- MULT (Multiplier)
- MAC (Multiplier Accumulate)
- MULTADDSUB (Multiplier Add/Subtract)
- MULTADDSUBSUM (Multiply Add/Subtract and SUM)

MULT Module

The MULT Module configures elements to be packed into the sysDSP primitives. The Basic mode screen illustrated in Figure 15-2 consists of one clock (optional), one clock enable and one reset tied to all registers. Using the multiplier you can span multiple sysDSP Blocks. The SRI and SRO ports can only be enabled if inputs are less than 18 bits. The Advanced mode screen, illustrated in Figure 15-3, allows finer control over the register. In the Advanced mode you can control each register with independent clocks, clock enables and resets.

Figure 15-2. MULT Mode Basic Set-up
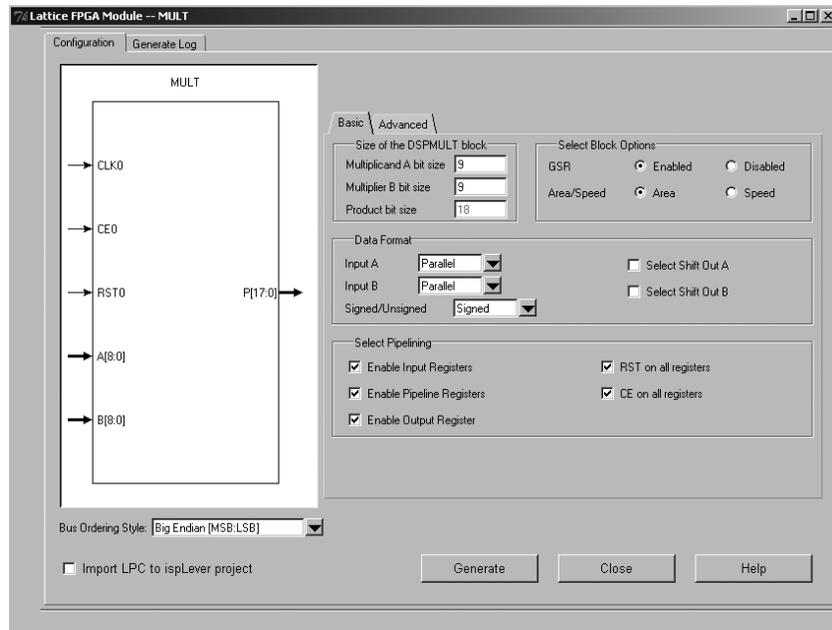
Figure 15-3. MULT Mode Advanced Set-up
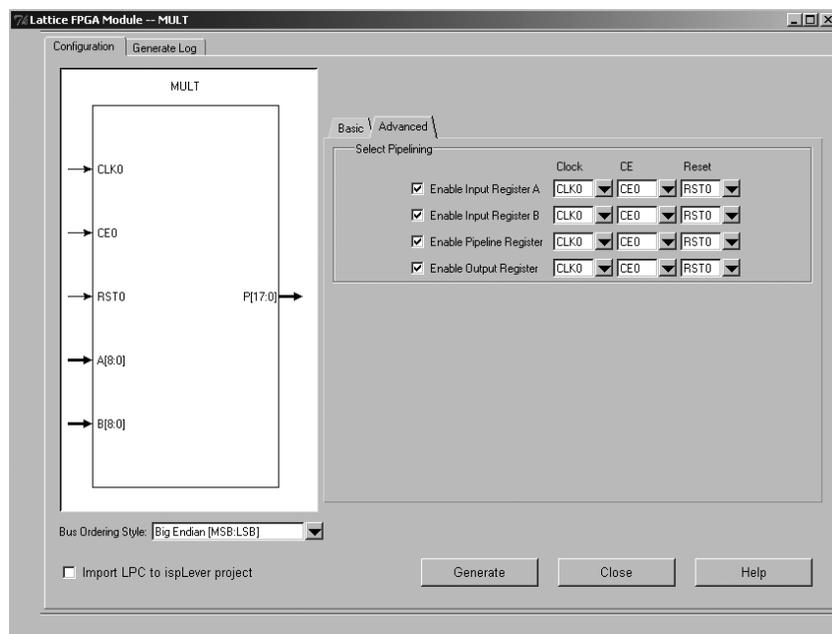
MAC Module

The MAC Module configures multiply accumulate elements to be packed into the primitives. The Basic mode, shown in Figure 15-4, consists of one clock (optional), one clock enable and one reset tied to all registers. Because of the Accumulator, the output register is automatically enabled. The SRI and SRO ports can only be enabled if inputs are less than 18 bits. The Accumsload loads the accumulator with the value from the multiplier. This is required to initialize and load the first value of the accumulation. The Advanced mode, shown in Figure 15-5, allows finer control over the registers. In the Advanced mode you can control each register with independent clocks, clock enables and resets.

Figure 15-4. MAC Mode Basic Set-up
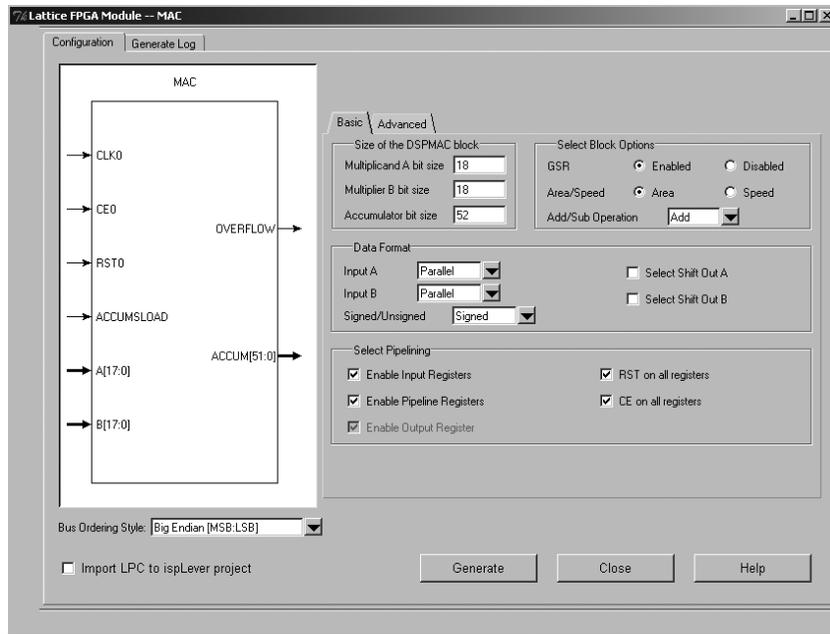
Figure 15-5. MAC Mode Advanced Set-up
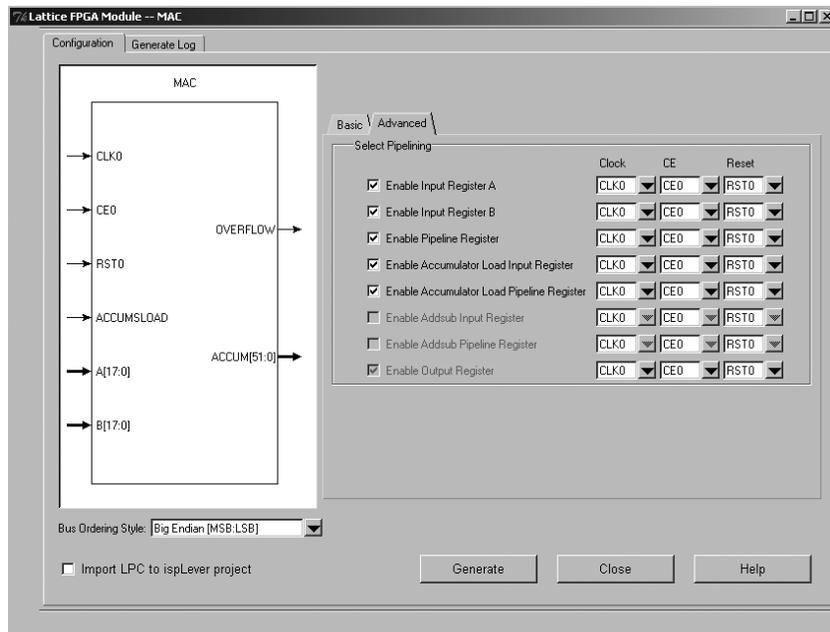
MULTADDSUB Module

The MULTADDSUB GUI configures Multiplier Addition/Subtraction elements to be packed into the primitives MULT18X18ADDSUB or MULT9X9ADDSUB. The Basic mode, shown in Figure 15-6, consists of one clock (optional), one clock enable and one reset tied to all registers. Using the MULTADDSUB you can span multiple sys-DSP Blocks. The SRI and SRO ports can only be enabled if inputs are less than 18 bits. The Advanced mode, shown in Figure 15-7, provides finer control over the registers. In the Advanced mode you can control each register with independent clocks, clock enables and resets.

Figure 15-6. MULTADDSUB Mode Basic Set-up
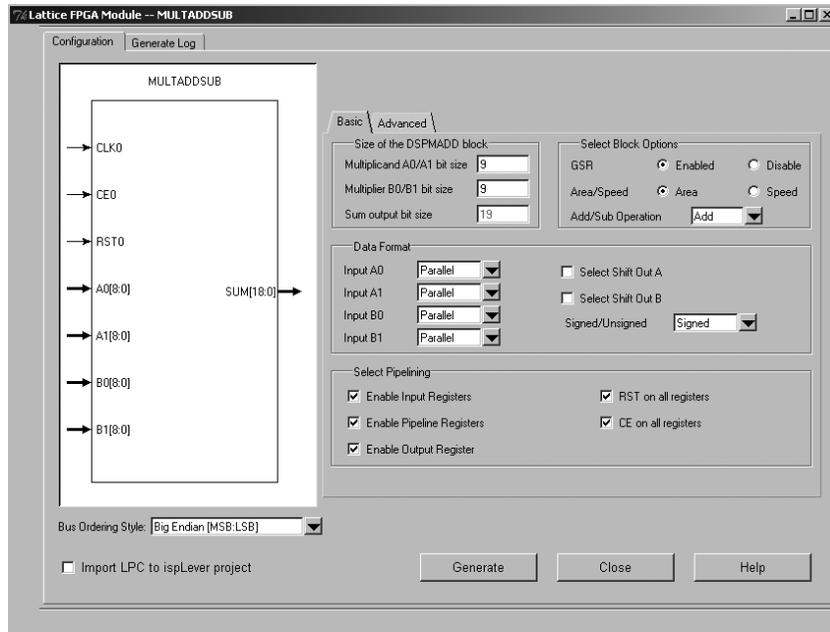
Figure 15-7. MULTADDSUB Mode Advanced Set-up
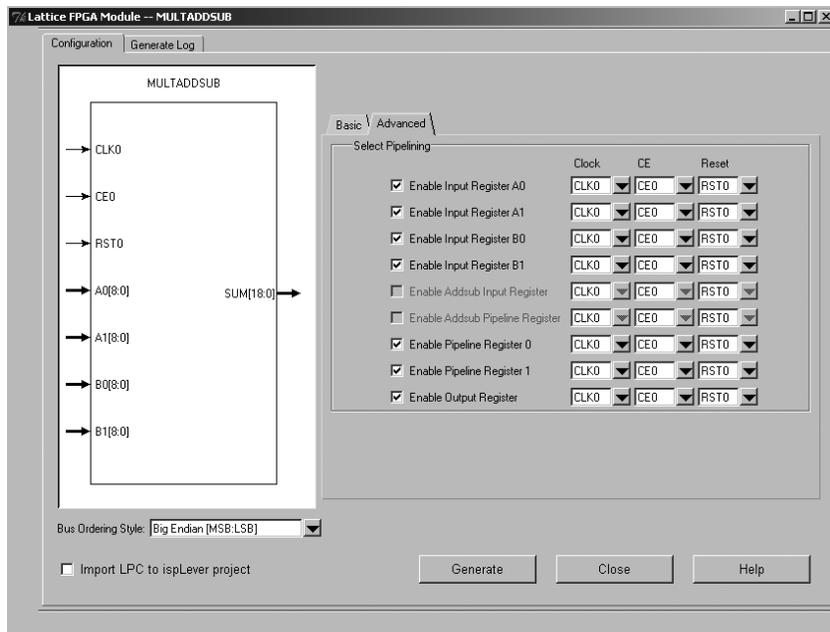
MULTADDSUBSUM Module

The MULTADDSUBSUM GUI configures Multiplier Addition/Subtraction Addition elements to be packed into the primitives MULT18X18ADDSUBSUM or MULT9X9ADDSUBSUM. The Basic mode, shown in Figure 15-8, consists of one clock (optional), one clock enable and one reset tied to all registers. Using the MULTADDSUBSUM you can span multiple sysDSP Blocks. The SRI and SRO ports can only be enabled if inputs are less than 18 bits. The Advanced mode, shown in Figure 15-9, provides finer control over the registers. In the Advanced mode you can control each register with independent clocks, clock enables and resets.

Figure 15-8. MULTADDSUBSUM Mode Basic Set-up
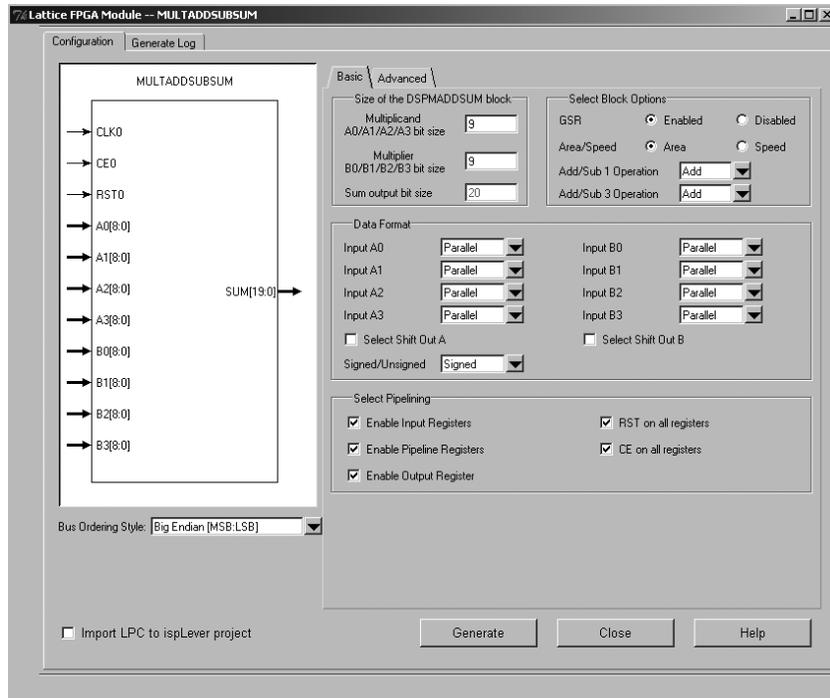
Figure 15-9. MULTADDSUBSUM Mode Advance 1 Set-up
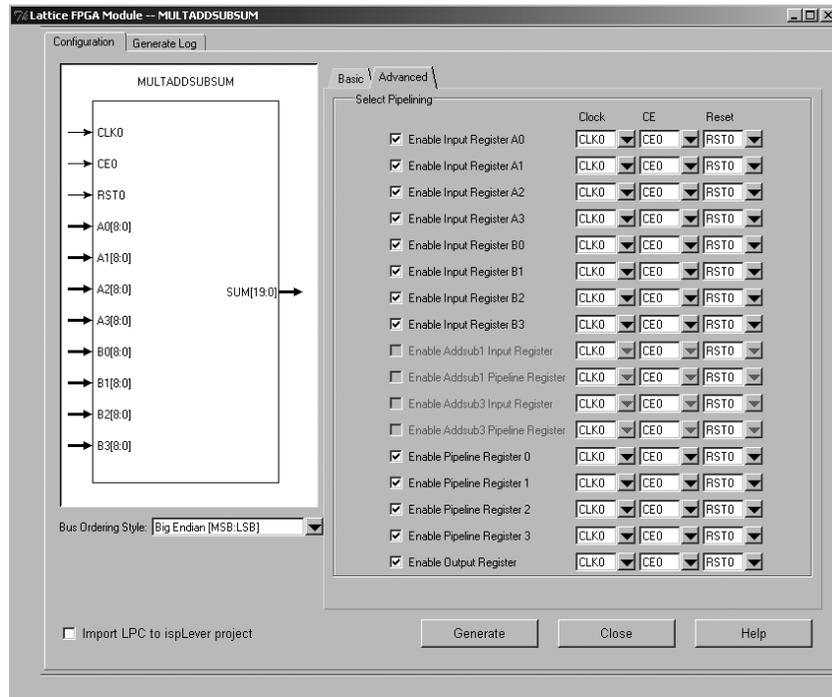
Targeting the sysDSP Block by Inference

The Inferencing flow enables the design tools to infer sysDSP Blocks from a HDL design. It is important to note that when using the Inferencing flow; unless the code style matches the sysDSP Block results will not be optimal results. Consider the following Veriflog and VHDL examples:

```verilog
// This Verilog example will be mapped into single MULT9X9MAC with the output register enabled
// This will be mapped into single MULT9X9MAC with the output register enabled
module mult_acc (dataout, dataax, dataay, clk);
    output [16:0] dataout;
    input [7:0] dataax, dataay;
    input clk;
    reg  [16:0] dataout;

    wire [15:0] multa = dataax * dataay; // 9x9 Multiplier
    wire [16:0] adder_out;
    assign adder_out = multa + dataout; // Accumulator
    always @(posedge clk)
    begin
      dataout <= adder_out; // Output Register of the Accumulator
    end
endmodule
```

```
-- This VHDL example will be mapped into single MULT9X9MAC with the output register enabled
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;

entity mult_acc is
port (
  clk  : in std_logic;
  dataax : in std_logic_vector(7 downto 0);
  dataay : in std_logic_vector(7 downto 0);
  dataout : out std_logic_vector(16 downto 0));
end;

architecture arch of mult_acc is
  signal multout : std_logic_vector(15 downto 0);
  signal adder_out : std_logic_vector(16 downto 0);
  signal dataout_reg : std_logic_vector(16 downto 0);
begin
  multout <= dataax * dataay; -- 9x9 Multiplier

  adder_out <= multout + dataout_reg;
  process (clk)
  begin
    if (clk'event and clk='1') then
      dataout_reg <= adder_out; -- Output Register of the Accumulator
    end if;
end process;
    dataout <= dataout_reg;
end arch;
```

The above RTL will infer the following block diagram.

Figure 15-10. MULT9X9MAC Block Diagram
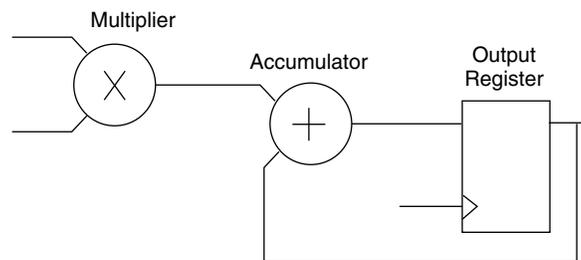
As you can see, this block diagram can be mapped directly into the sysDSP primitives If we were to add a test point between the multiplier and the accumulator, or two output registers, the code could not be mapped into a MULT9X9MAC of a sysDSP Block. So options you could include in the design would be input registers, pipeline registers, etc. For more inferring design examples refer to EXAMPLES in your ispLEVER software.

Figure 15-11. MAC9X9MAC Packed into a sysDSP Block
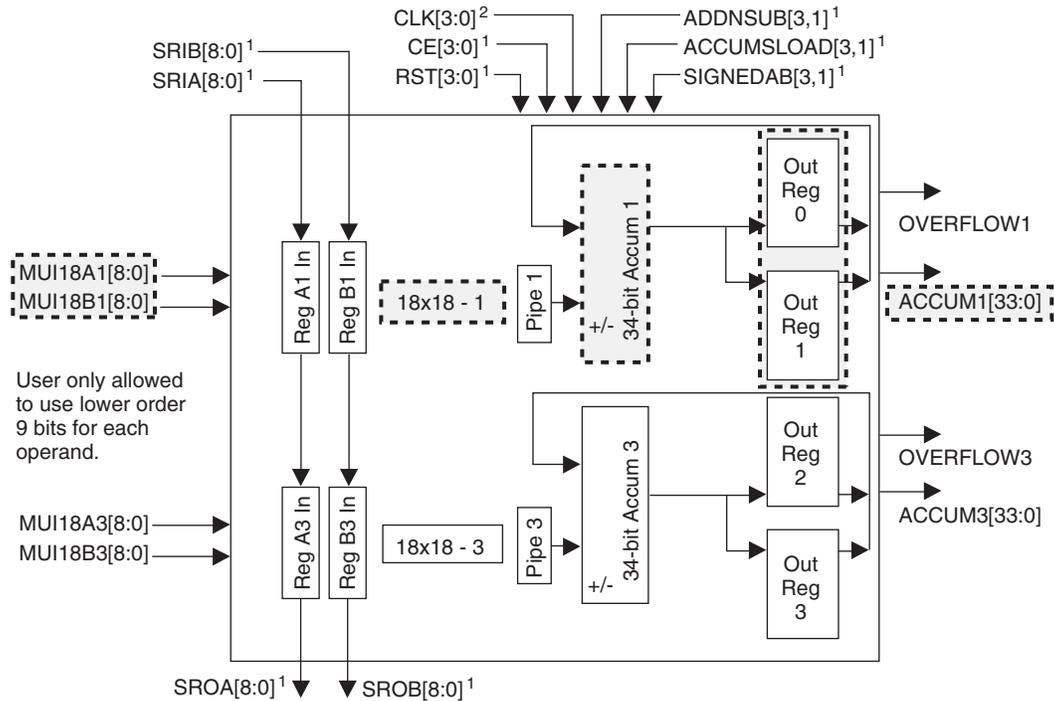
Notes:
1. These signals are optional.
2. At least one clock is required.

Targeting the sysDSP Block using Simulink

Simulink Overview
Simulink is a graphical add-on (similar to schematic entry) for Matlab, which is produced by Mathworks. For more information refer to the Simulink web site at www.mathworks.com/products/simulink/.

Why is Simulink Used?
- Simulink allows users to create algorithms using floating point numbers
- It helps users convert floating point algorithms into fixed point algorithms

How Does Simulink Fit into the Normal ispLEVER Design Flow?
Once you have converted have your algorithm working in fixed point you can use the Lattice ispDSP Block to create HDL files, which can be instantiated in your HDL design. Currently there is only support for VHDL.

What Does Lattice Provide?
Lattice provides a library of blocks for the Simulink tool, which include multipliers, adders, registers and other standard building blocks. Besides the basic building blocks there are a couple of unique Lattice blocks.

Gateways In and Out
Everything between Gateways In and Out represents HDL code. Everything before a Gateway In is the stimulus your test bench. Everything after a Gateway Out are the signals you will be monitoring in your test bench. Figure 15-12 is an example.

ispDSP
The ispDSP block is used to convert fixed point Simulink design into HDL files which can be instantiated in your HDL design. The ispDSP Block is identified by the Lattice logo and can be seen in Figure 21.

Figure 15-12. Simulink Design
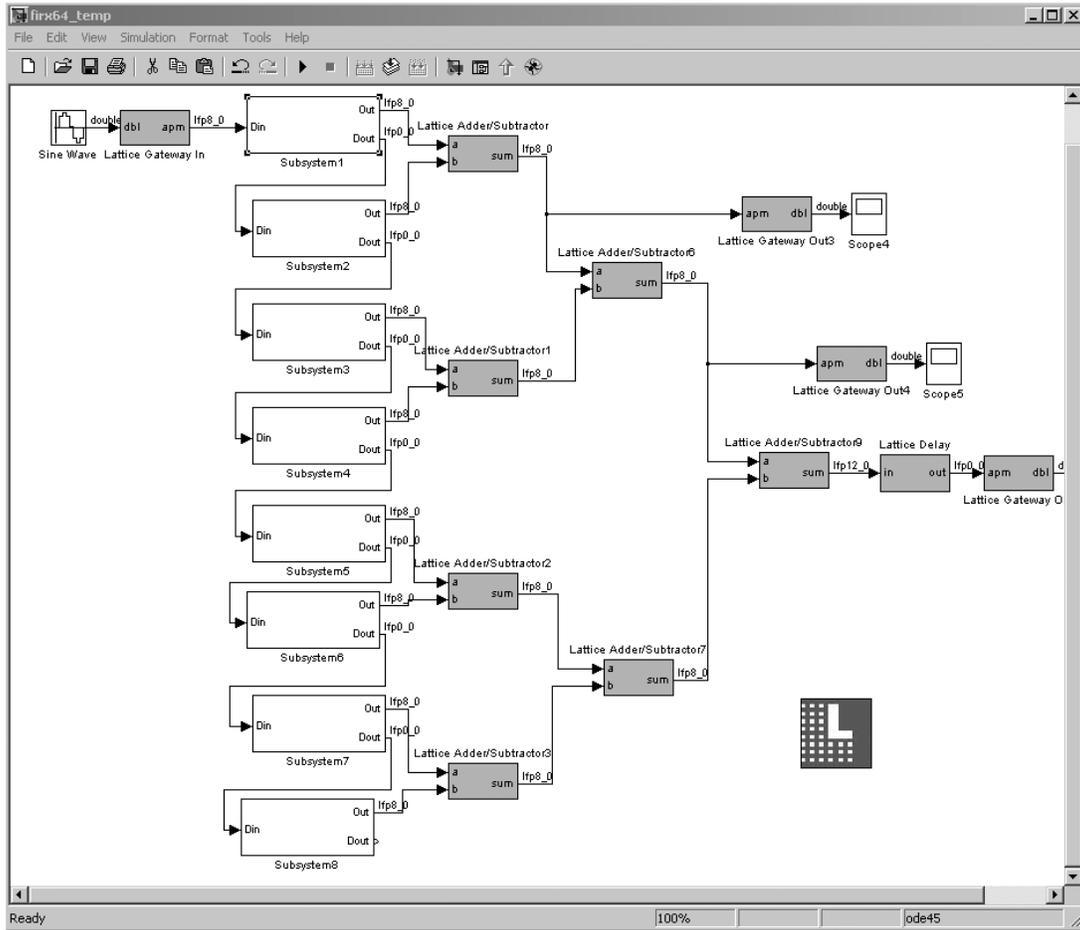
Targeting the sysDSP Block by Instantiating Primitives

You can target the sysDSP Block by instantiating the sysDSP Block primitives into your design. The advantage of instantiating primitives is that it gives you access to all ports and set all available parameters. The disadvantage of this flow is that all this customization comes at a cost of extra coding for the user. Appendix A details the syntax for the sysDSP Block primitives.

sysDSP Blocks in the Report File

To check configuration of the sysDSP Blocks in your design, look at the MAP and Post PAR report files. The MAP Report File shows the mapped sysDSP components/primitives in your design. The POST PAR Report File shows the number of components in each sysDSP Block. The following gives an example of the sysDSP section from each of the Report Files.

MAP Report File

```
Number Of Mapped DSP Components:
--------------------------------
MULT36X36                 0
MULT18X18                 1
MULT18X18MAC              0
MULT18X18ADDSUB           0
MULT18X18ADDSUBSUM        0
MULT9X9                   0
MULT9X9MAC                0
MULT9X9ADDSUB             0
MULT9X9ADDSUBSUM          0
```

Post PAR Report File

```
DSP Utilization Summary:
------------------------------------
DSP Block #:            1    2    3    4    5    6    7
# of MULT36X36
# of MULT18X18          1
# of MULT18X18MAC
# of MULT18X18ADDSUB
# of MULT18X18ADDSUBSUM
# of MULT9X9
# of MULT9X9MAC
# of MULT9X9ADDSUB
# of MULT9X9ADDSUBSUM
```

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

          +1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Appendix A. DSP Block Primitives

MULT36X36 Primitive

```
MULT36X36(CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDAB,
A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A17,
A18,A19,A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A30,A31,A32,A33,A34,A35,
B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,B13,B14,B15,B16,B17,
B18,B19,B20,B21,B22,B23,B24,B25,B26,B27,B28,B29,B30,B31,B32,B33,B34,B35,
P0,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15,P16,P17,
P18,P19,P20,P21,P22,P23,P24,P25,P26,P27,P28,P29,P30,P31,P32,P33,P34,P35,
P36,P37,P38,P39,P40,P41,P42,P43,P44,P45,P46,P47,P48,P49,P50,P51,P52,P53,
P54,P55,P56,P57,P58,P59,P60,P61,P62,P63,P64,P65,P66,P67,P68,P69,P70,P71) is black-box
{
property REG_INPUTA_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE_RST {"RST0","RST1","RST2","RST3"};
property REG_OUTPUT_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_OUTPUT_CE {"CE0","CE1","CE2","CE3"};
property REG_OUTPUT_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_1_RST {"RST0","RST1","RST2","RST3"};
property GSR {"ENABLED","DISABLED"};
```

MULT18X18 Primitive

```
MULT18X18(CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDAB,
A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A17,
B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,B13,B14,B15,B16,B17,
SRIA0,SRIA1,SRIA2,SRIA3,SRIA4,SRIA5,SRIA6,SRIA7,SRIA8,SRIA9,SRIA10,SRIA11,SRIA12,SRIA13,SRIA1
4,SRIA15,SRIA16,SRIA17,
SRIB0,SRIB1,SRIB2,SRIB3,SRIB4,SRIB5,SRIB6,SRIB7,SRIB8,SRIB9,SRIB10,SRIB11,SRIB12,SRIB13,SRIB1
4,SRIB15,SRIB16,SRIB17,
SROA0,SROA1,SROA2,SROA3,SROA4,SROA5,SROA6,SROA7,SROA8,SROA9,SROA10,SROA11,SROA12,SROA13,SROA1
4,SROA15,SROA16,SROA17,
SROB0,SROB1,SROB2,SROB3,SROB4,SROB5,SROB6,SROB7,SROB8,SROB9,SROB10,SROB11,SROB12,SROB13,SROB1
4,SROB15,SROB16,SROB17,
P0,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15,P16,P17,
P18,P19,P20,P21,P22,P23,P24,P25,P26,P27,P28,P29,P30,P31,P32,P33,P34,P35) is black-box
{
property REG_INPUTA_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE_RST {"RST0","RST1","RST2","RST3"};
property REG_OUTPUT_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
```

```
property REG_OUTPUT_CE {"CE0","CE1","CE2","CE3"};
property REG_OUTPUT_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_1_RST {"RST0","RST1","RST2","RST3"};
property SHIFT_IN_A {"FALSE", "TRUE"};
property SHIFT_IN_B {"FALSE", "TRUE"};
property GSR {"ENABLED","DISABLED"};
```

MULT18X18MAC Primitive

```
MULT18X18MAC(CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDAB,ADDNSUB,ACCUMS-
LOAD,
A0,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14,A15,A16,A17,
B0,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,B13,B14,B15,B16,B17,
SRIA0,SRIA1,SRIA2,SRIA3,SRIA4,SRIA5,SRIA6,SRIA7,SRIA8,
SRIA9,SRIA10,SRIA11,SRIA12,SRIA13,SRIA14,SRIA15,SRIA16,SRIA17,
SRIB0,SRIB1,SRIB2,SRIB3,SRIB4,SRIB5,SRIB6,SRIB7,SRIB8,
SRIB9,SRIB10,SRIB11,SRIB12,SRIB13,SRIB14,SRIB15,SRIB16,SRIB17,
SROA0,SROA1,SROA2,SROA3,SROA4,SROA5,SROA6,SROA7,SROA8,
SROA9,SROA10,SROA11,SROA12,SROA13,SROA14,SROA15,SROA16,SROA17,
SROB0,SROB1,SROB2,SROB3,SROB4,SROB5,SROB6,SROB7,SROB8,
SROB9,SROB10,SROB11,SROB12,SROB13,SROB14,SROB15,SROB16,SROB17,
ACCUM0,ACCUM1,ACCUM2,ACCUM3,ACCUM4,ACCUM5,ACCUM6,ACCUM7,ACCUM8,
ACCUM9,ACCUM10,ACCUM11,ACCUM12,ACCUM13,ACCUM14,ACCUM15,ACCUM16,ACCUM17,
ACCUM18,ACCUM19,ACCUM20,ACCUM21,ACCUM22,ACCUM23,ACCUM24,ACCUM25,ACCUM26,
ACCUM27,ACCUM28,ACCUM29,ACCUM30,ACCUM31,ACCUM32,ACCUM33,ACCUM34,ACCUM35,
ACCUM36,ACCUM37,ACCUM38,ACCUM39,ACCUM40,ACCUM41,ACCUM42,ACCUM43,ACCUM44,
ACCUM45,ACCUM46,ACCUM47,ACCUM48,ACCUM49,ACCUM50,ACCUM51,OVERFLOW) is black-box
{
property REG_INPUTA_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE_RST {"RST0","RST1","RST2","RST3"};
property REG_OUTPUT_CLK {"CLK0","CLK1","CLK2","CLK3"};
property REG_OUTPUT_CE {"CE0","CE1","CE2","CE3"};
property REG_OUTPUT_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_1_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB_1_RST {"RST0","RST1","RST2","RST3"};
property REG_ACCUMSLOAD_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ACCUMSLOAD_0_CE {"CE0","CE1","CE2","CE3"};
```

```
property REG_ACCUMSLOAD_0_RST {"RST0","RST1","RST2","RST3"};
property REG_ACCUMSLOAD_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ACCUMSLOAD_1_CE {"CE0","CE1","CE2","CE3"};
property REG_ACCUMSLOAD_1_RST {"RST0","RST1","RST2","RST3"};
property SHIFT_IN_A {"FALSE", "TRUE"};
property SHIFT_IN_B {"FALSE", "TRUE"};
property GSR {"ENABLED","DISABLED"};
```

MULT18X18ADDSUB Primitive

```
MULT18X18ADDSUB(CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDAB,ADDNSUB,
A00,A01,A02,A03,A04,A05,A06,A07,A08,A09,A010,A011,A012,A013,A014,A015,A016,A017,
A10,A11,A12,A13,A14,A15,A16,A17,A18,A19,A110,A111,A112,A113,A114,A115,A116,A117,
B00,B01,B02,B03,B04,B05,B06,B07,B08,B09,B010,B011,B012,B013,B014,B015,B016,B017,
B10,B11,B12,B13,B14,B15,B16,B17,B18,B19,B110,B111,B112,B113,B114,B115,B116,B117,
SRIA0,SRIA1,SRIA2,SRIA3,SRIA4,SRIA5,SRIA6,SRIA7,SRIA8,
SRIA9,SRIA10,SRIA11,SRIA12,SRIA13,SRIA14,SRIA15,SRIA16,SRIA17,
SRIB0,SRIB1,SRIB2,SRIB3,SRIB4,SRIB5,SRIB6,SRIB7,SRIB8,
SRIB9,SRIB10,SRIB11,SRIB12,SRIB13,SRIB14,SRIB15,SRIB16,SRIB17,
SROA0,SROA1,SROA2,SROA3,SROA4,SROA5,SROA6,SROA7,SROA8,
SROA9,SROA10,SROA11,SROA12,SROA13,SROA14,SROA15,SROA16,SROA17,
SROB0,SROB1,SROB2,SROB3,SROB4,SROB5,SROB6,SROB7,SROB8,
SROB9,SROB10,SROB11,SROB12,SROB13,SROB14,SROB15,SROB16,SROB17,
SUM0,SUM1,SUM2,SUM3,SUM4,SUM5,SUM6,SUM7,SUM8,
SUM9,SUM10,SUM11,SUM12,SUM13,SUM14,SUM15,SUM16,SUM17,
SUM18,SUM19,SUM20,SUM21,SUM22,SUM23,SUM24,SUM25,SUM26,
SUM27,SUM28,SUM29,SUM30,SUM31,SUM32,SUM33,SUM34,SUM35,
SUM36) is black-box
{
property REG_INPUTA0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA0_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA0_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTA1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA1_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA1_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB0_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB0_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB1_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB1_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE0_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE0_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE1_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE1_RST {"RST0","RST1","RST2","RST3"};
property REG_OUTPUT_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_OUTPUT_CE {"CE0","CE1","CE2","CE3"};
property REG_OUTPUT_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_1_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB_0_RST {"RST0","RST1","RST2","RST3"};
```

```
property REG_ADDNSUB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB_1_RST {"RST0","RST1","RST2","RST3"};
property SHIFT_IN_A0 {"FALSE", "TRUE"};
property SHIFT_IN_B0 {"FALSE", "TRUE"};
property SHIFT_IN_A1 {"FALSE", "TRUE"};
property SHIFT_IN_B1 {"FALSE", "TRUE"};
property GSR {"ENABLED","DISABLED"};
```

MULT18X18ADDSUBSUM Primitive

```
MULT18X18ADDSUBSUM(CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDAB,ADDNSUB1,
ADDNSUB3,
A00,A01,A02,A03,A04,A05,A06,A07,A08,A09,A010,A011,A012,A013,A014,A015,A016,A017,
A10,A11,A12,A13,A14,A15,A16,A17,A18,A19,A110,A111,A112,A113,A114,A115,A116,A117,
A20,A21,A22,A23,A24,A25,A26,A27,A28,A29,A210,A211,A212,A213,A214,A215,A216,A217,
A30,A31,A32,A33,A34,A35,A36,A37,A38,A39,A310,A311,A312,A313,A314,A315,A316,A317,
B00,B01,B02,B03,B04,B05,B06,B07,B08,B09,B010,B011,B012,B013,B014,B015,B016,B017,
B10,B11,B12,B13,B14,B15,B16,B17,B18,B19,B110,B111,B112,B113,B114,B115,B116,B117,
B20,B21,B22,B23,B24,B25,B26,B27,B28,B29,B210,B211,B212,B213,B214,B215,B216,B217,
B30,B31,B32,B33,B34,B35,B36,B37,B38,B39,B310,B311,B312,B313,B314,B315,B316,B317,
SRIA0,SRIA1,SRIA2,SRIA3,SRIA4,SRIA5,SRIA6,SRIA7,SRIA8,
SRIA9,SRIA10,SRIA11,SRIA12,SRIA13,SRIA14,SRIA15,SRIA16,SRIA17,
SRIB0,SRIB1,SRIB2,SRIB3,SRIB4,SRIB5,SRIB6,SRIB7,SRIB8,
SRIB9,SRIB10,SRIB11,SRIB12,SRIB13,SRIB14,SRIB15,SRIB16,SRIB17,
SROA0,SROA1,SROA2,SROA3,SROA4,SROA5,SROA6,SROA7,SROA8,
SROA9,SROA10,SROA11,SROA12,SROA13,SROA14,SROA15,SROA16,SROA17,
SROB0,SROB1,SROB2,SROB3,SROB4,SROB5,SROB6,SROB7,SROB8,
SROB9,SROB10,SROB11,SROB12,SROB13,SROB14,SROB15,SROB16,SROB17,
SUM0,SUM1,SUM2,SUM3,SUM4,SUM5,SUM6,SUM7,SUM8,
SUM9,SUM10,SUM11,SUM12,SUM13,SUM14,SUM15,SUM16,SUM17,
SUM18,SUM19,SUM20,SUM21,SUM22,SUM23,SUM24,SUM25,SUM26,
SUM27,SUM28,SUM29,SUM30,SUM31,SUM32,SUM33,SUM34,SUM35,
SUM36,SUM37) is black-box
{
property REG_INPUTA0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA0_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA0_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTA1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA1_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA1_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTA2_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA2_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA2_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTA3_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA3_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA3_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB0_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB0_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB1_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB1_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB2_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB2_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB2_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB3_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB3_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB3_RST {"RST0","RST1","RST2","RST3"};
```

```
property REG_PIPELINE0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE0_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE0_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE1_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE1_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE2_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE2_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE2_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE3_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE3_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE3_RST {"RST0","RST1","RST2","RST3"};
property REG_OUTPUT_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_OUTPUT_CE {"CE0","CE1","CE2","CE3"};
property REG_OUTPUT_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_1_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB1_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB1_0_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB1_0_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB1_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB1_1_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB1_1_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB3_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB3_0_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB3_0_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB3_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB3_1_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB3_1_RST {"RST0","RST1","RST2","RST3"};
property SHIFT_IN_A0 {"FALSE", "TRUE"};
property SHIFT_IN_B0 {"FALSE", "TRUE"};
property SHIFT_IN_A1 {"FALSE", "TRUE"};
property SHIFT_IN_B1 {"FALSE", "TRUE"};
property SHIFT_IN_A2 {"FALSE", "TRUE"};
property SHIFT_IN_B2 {"FALSE", "TRUE"};
property SHIFT_IN_A3 {"FALSE", "TRUE"};
property SHIFT_IN_B3 {"FALSE", "TRUE"};
property GSR {"ENABLED","DISABLED"};
```

MULT9X9 Primitive

```
MULT9X9(CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDAB,
A0,A1,A2,A3,A4,A5,A6,A7,A8,B0,B1,B2,B3,B4,B5,B6,B7,B8,
SRIA0,SRIA1,SRIA2,SRIA3,SRIA4,SRIA5,SRIA6,SRIA7,SRIA8,
SRIB0,SRIB1,SRIB2,SRIB3,SRIB4,SRIB5,SRIB6,SRIB7,SRIB8,
SROA0,SROA1,SROA2,SROA3,SROA4,SROA5,SROA6,SROA7,SROA8,
SROB0,SROB1,SROB2,SROB3,SROB4,SROB5,SROB6,SROB7,SROB8,
P0,P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14,P15,P16,P17) is black-box
{
property REG_INPUTA_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB_RST {"RST0","RST1","RST2","RST3"};
```

```
property REG_PIPELINE_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE_RST {"RST0","RST1","RST2","RST3"};
property REG_OUTPUT_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_OUTPUT_CE {"CE0","CE1","CE2","CE3"};
property REG_OUTPUT_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_1_RST {"RST0","RST1","RST2","RST3"};
property SHIFT_IN_A {"FALSE", "TRUE"};
property SHIFT_IN_B {"FALSE", "TRUE"};
property GSR {"ENABLED","DISABLED"};
```

MULT9X9MAC Primitive

```
MULT9X9MAC(CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDAB,ADDNSUB,ACCUMS-
LOAD,
A0,A1,A2,A3,A4,A5,A6,A7,A8,
B0,B1,B2,B3,B4,B5,B6,B7,B8,
SRIA0,SRIA1,SRIA2,SRIA3,SRIA4,SRIA5,SRIA6,SRIA7,SRIA8,
SRIB0,SRIB1,SRIB2,SRIB3,SRIB4,SRIB5,SRIB6,SRIB7,SRIB8,
SROA0,SROA1,SROA2,SROA3,SROA4,SROA5,SROA6,SROA7,SROA8,
SROB0,SROB1,SROB2,SROB3,SROB4,SROB5,SROB6,SROB7,SROB8,
ACCUM0,ACCUM1,ACCUM2,ACCUM3,ACCUM4,ACCUM5,ACCUM6,ACCUM7,ACCUM8,
ACCUM9,ACCUM10,ACCUM11,ACCUM12,ACCUM13,ACCUM14,ACCUM15,ACCUM16,ACCUM17,
ACCUM18,ACCUM19,ACCUM20,ACCUM21,ACCUM22,ACCUM23,ACCUM24,ACCUM25,ACCUM26,
ACCUM27,ACCUM28,ACCUM29,ACCUM30,ACCUM31,ACCUM32,ACCUM33,OVERFLOW) is black-box
{
property REG_INPUTA_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE_RST {"RST0","RST1","RST2","RST3"};
property REG_OUTPUT_CLK {"CLK0","CLK1","CLK2","CLK3"};
property REG_OUTPUT_CE {"CE0","CE1","CE2","CE3"};
property REG_OUTPUT_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_1_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB_1_RST {"RST0","RST1","RST2","RST3"};
property REG_ACCUMSLOAD_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ACCUMSLOAD_0_CE {"CE0","CE1","CE2","CE3"};
property REG_ACCUMSLOAD_0_RST {"RST0","RST1","RST2","RST3"};
property REG_ACCUMSLOAD_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
```

```
property REG_ACCUMSLOAD_1_CE {"CE0","CE1","CE2","CE3"};
property REG_ACCUMSLOAD_1_RST {"RST0","RST1","RST2","RST3"};
property SHIFT_IN_A {"FALSE", "TRUE"};
property SHIFT_IN_B {"FALSE", "TRUE"};
property GSR {"ENABLED","DISABLED"};
```

MULT9X9ADDSUB Primitive

```
MULT9X9ADDSUB(CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDAB,ADDNSUB,
A00,A01,A02,A03,A04,A05,A06,A07,A08,
A10,A11,A12,A13,A14,A15,A16,A17,A18,
B00,B01,B02,B03,B04,B05,B06,B07,B08,
B10,B11,B12,B13,B14,B15,B16,B17,B18,
SRIA0,SRIA1,SRIA2,SRIA3,SRIA4,SRIA5,SRIA6,SRIA7,SRIA8,
SRIB0,SRIB1,SRIB2,SRIB3,SRIB4,SRIB5,SRIB6,SRIB7,SRIB8,
SROA0,SROA1,SROA2,SROA3,SROA4,SROA5,SROA6,SROA7,SROA8,
SROB0,SROB1,SROB2,SROB3,SROB4,SROB5,SROB6,SROB7,SROB8,
SUM0,SUM1,SUM2,SUM3,SUM4,SUM5,SUM6,SUM7,SUM8,
SUM9,SUM10,SUM11,SUM12,SUM13,SUM14,SUM15,SUM16,SUM17,SUM18) is black-box
{
property REG_INPUTA0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA0_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA0_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTA1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA1_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA1_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB0_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB0_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB1_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB1_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE0_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE0_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE1_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE1_RST {"RST0","RST1","RST2","RST3"};
property REG_OUTPUT_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_OUTPUT_CE {"CE0","CE1","CE2","CE3"};
property REG_OUTPUT_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB_1_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_1_RST {"RST0","RST1","RST2","RST3"};
property SHIFT_IN_A0 {"FALSE", "TRUE"};
property SHIFT_IN_B0 {"FALSE", "TRUE"};
property SHIFT_IN_A1 {"FALSE", "TRUE"};
property SHIFT_IN_B1 {"FALSE", "TRUE"};
property GSR {"ENABLED","DISABLED"};
```

MULT9X9ADDSUBSUM Primitive

```
MULT9X9ADDSUBSUM(CE0,CE1,CE2,CE3,CLK0,CLK1,CLK2,CLK3,RST0,RST1,RST2,RST3,SIGNEDAB,ADDNSUB1,AD
DNSUB3,
A00,A01,A02,A03,A04,A05,A06,A07,A08,
A10,A11,A12,A13,A14,A15,A16,A17,A18,
A20,A21,A22,A23,A24,A25,A26,A27,A28,
A30,A31,A32,A33,A34,A35,A36,A37,A38,
B00,B01,B02,B03,B04,B05,B06,B07,B08,
B10,B11,B12,B13,B14,B15,B16,B17,B18,
B20,B21,B22,B23,B24,B25,B26,B27,B28,
B30,B31,B32,B33,B34,B35,B36,B37,B38,
SRIA0,SRIA1,SRIA2,SRIA3,SRIA4,SRIA5,SRIA6,SRIA7,SRIA8,
SRIB0,SRIB1,SRIB2,SRIB3,SRIB4,SRIB5,SRIB6,SRIB7,SRIB8,
SROA0,SROA1,SROA2,SROA3,SROA4,SROA5,SROA6,SROA7,SROA8,
SROB0,SROB1,SROB2,SROB3,SROB4,SROB5,SROB6,SROB7,SROB8,
SUM0,SUM1,SUM2,SUM3,SUM4,SUM5,SUM6,SUM7,SUM8,
SUM9,SUM10,SUM11,SUM12,SUM13,SUM14,SUM15,SUM16,SUM17,
SUM18,SUM19) is black-box
{
property REG_INPUTA0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA0_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA0_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTA1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA1_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA1_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTA2_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA2_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA2_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTA3_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTA3_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTA3_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB0_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB0_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB1_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB1_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB2_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB2_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB2_RST {"RST0","RST1","RST2","RST3"};
property REG_INPUTB3_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_INPUTB3_CE {"CE0","CE1","CE2","CE3"};
property REG_INPUTB3_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE0_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE0_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE1_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE1_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE2_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE2_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE2_RST {"RST0","RST1","RST2","RST3"};
property REG_PIPELINE3_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_PIPELINE3_CE {"CE0","CE1","CE2","CE3"};
property REG_PIPELINE3_RST {"RST0","RST1","RST2","RST3"};
property REG_OUTPUT_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_OUTPUT_CE {"CE0","CE1","CE2","CE3"};
property REG_OUTPUT_RST {"RST0","RST1","RST2","RST3"};
```

```
property REG_SIGNEDAB_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_0_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_0_RST {"RST0","RST1","RST2","RST3"};
property REG_SIGNEDAB_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_SIGNEDAB_1_CE {"CE0","CE1","CE2","CE3"};
property REG_SIGNEDAB_1_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB1_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB1_0_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB1_0_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB1_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB1_1_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB1_1_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB3_0_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB3_0_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB3_0_RST {"RST0","RST1","RST2","RST3"};
property REG_ADDNSUB3_1_CLK {"NONE","CLK0","CLK1","CLK2","CLK3"};
property REG_ADDNSUB3_1_CE {"CE0","CE1","CE2","CE3"};
property REG_ADDNSUB3_1_RST {"RST0","RST1","RST2","RST3"};
property SHIFT_IN_A0 {"FALSE", "TRUE"};
property SHIFT_IN_B0 {"FALSE", "TRUE"};
property SHIFT_IN_A1 {"FALSE", "TRUE"};
property SHIFT_IN_B1 {"FALSE", "TRUE"};
property SHIFT_IN_A2 {"FALSE", "TRUE"};
property SHIFT_IN_B2 {"FALSE", "TRUE"};
property SHIFT_IN_A3 {"FALSE", "TRUE"};
property SHIFT_IN_B3 {"FALSE", "TRUE"};
property GSR {"ENABLED","DISABLED"};
```

HDL Synthesis Coding Guidelines for Lattice Semiconductor FPGAs

Introduction

Coding style plays an important role in utilizing FPGA resources. Although many popular synthesis tools have significantly improved optimization algorithms for FPGAs, it still is the responsibility of the user to generate meaningful and efficient HDL code to guide their synthesis tools to achieve the best result for a specific architecture. This application note is intended to help designers establish useful HDL coding styles for Lattice Semiconductor FPGA devices. It includes VHDL and Verilog design guidelines for both novice and experienced users.

The application note is divided into two sections. The general coding styles for FPGAs section provides an overview for effective FPGA designs. The following topics are discussed in detail:

- Hierarchical Coding
- Design Partitioning
- Encoding Methodologies for State Machines
- Coding Styles for Finite State Machines (FSM)
- Using Pipelines
- Comparing IF Statements and CASE Statements
- Avoiding Non-intentional Latches

The HDL Design with Lattice Semiconductor FPGA Devices section covers specific coding techniques and examples:

- Using the Lattice Semiconductor FPGA Synthesis Library
- Implementation of Multiplexers
- Creating Clock Dividers
- Register Control Signals (CE, LSR, GSR)
- Using PIC Features
- Implementation of Memories
- Preventing Logic Replication and Fanout
- Comparing Synthesis Results and Place and Route Results

General Coding Styles for FPGA

The following recommendations for common HDL coding styles will help users generate robust and reliable FPGA designs.

Hierarchical Coding

HDL designs can either be synthesized as a flat module or as many small hierarchical modules. Each methodology has its advantages and disadvantages. Since designs in smaller blocks are easier to keep track of, it is preferred to apply hierarchical structure to large and complex FPGA designs. Hierarchical coding methodology allows a group of engineers to work on one design at the same time. It speeds up design compilation, makes changing the implementation of key blocks easier, and reduces the design period by allowing the re-use of design modules for current and future designs. In addition, it produces designs that are easier to understand. However, if the design mapping into the FPGA is not optimal across hierarchical boundaries, it will lead to lower device utilization and design performance. This disadvantage can be overcome with careful design considerations when choosing the design hierarchy. Here are some tips for building hierarchical structures:

- The top level should only contain instantiation statements to call all major blocks
- Any I/O instantiations should be at the top level
- Any signals going into or out of the devices should be declared as input, output or bi-directional pins at the top level

• Memory blocks should be kept separate from other code

Design Partitioning

By effectively partitioning the design, a designer can reduce overall run time and improve synthesis results. Here are some recommendations for design partitioning.

Maintain Synchronous Sub-blocks by Registering All Outputs

It is suggested to arrange the design boundary such that the outputs in each block are registered. Registering outputs helps the synthesis tool to consider the implementation of the combinatorial logic and registers into the same logic block. Registering outputs also makes the application of timing constraints easier since it eliminates possible problems with logic optimization across design boundaries. Single clock is recommended for each synchronous block because it significantly reduces the timing consideration in the block. It leaves the adjustment of the clock relationships of the whole design at the top level of the hierarchy. Figure 15-1 shows an example of synchronous blocks with registered outputs.

Figure 15-1. Synchronous Blocks with Registered Outputs
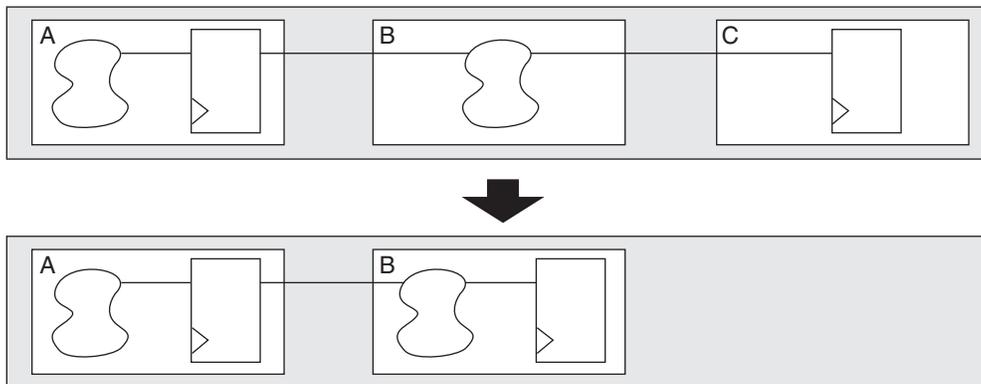
Keep Related Logic Together in the Same Block

Keeping related logic and sharable resources in the same block allows the sharing of common combinatorial terms and arithmetic functions within the block. It also allows the synthesis tools to optimize the entire critical path in a single operation. Since synthesis tools can only effectively handle optimization of certain amounts of logic, optimization of critical paths pending across the boundaries may not be optimal. Figure 15-2 shows an example of merging sharable resource in the same block.

Figure 15-2. Merge Sharable Resource in the Same Block
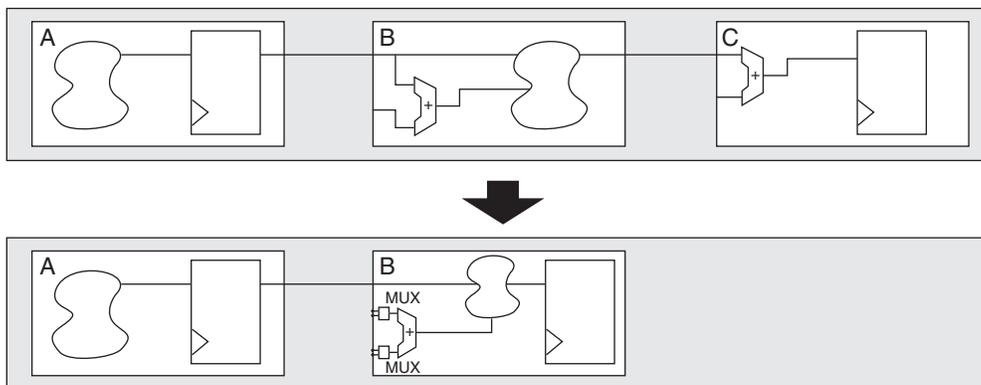
Separate Logic with Different Optimization Goals

Separating critical paths from non-critical paths may achieve efficient synthesis results. At the beginning of the project, one should consider the design in terms of performance requirements and resource requirements. If there are

two portions of a block, one that needs to be optimized for area and a second that needs to be optimized for speed, they should be separated into two blocks. By doing this, different optimization strategies for each module can be applied without being limited by one another.

Keep Logic with the Same Relaxation Constraints in the Same Block

When a portion of the design does not require high performance, this portion can be applied with relaxed timing constraints such as "multicycle" to achieve high utilization of device area. Relaxation constraints help to reduce overall run time. They can also help to efficiently save resources, which can be used on critical paths. Figure 15-3 shows an example of grouping logic with the same relaxation constraint in one block.

Figure 15-3. Logic with the Same Relaxation Constraint
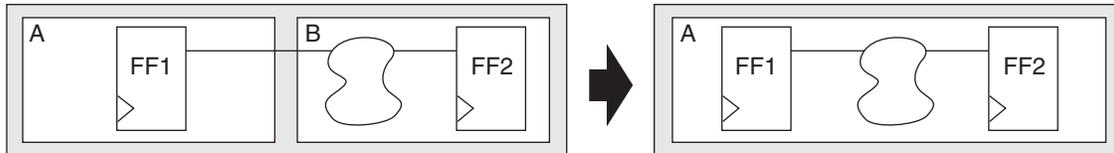
Keep Instantiated Code in Separate Blocks

It is recommended that the RAM block in the hierarchy be left in a separate block (Figure 15-4). This allows for easy swapping between the RAM behavioral code for simulation, and the code for technology instantiation. In addition, this coding style facilitates the integration of the ispLEVER® IPexpress™ tool into the synthesis process.

Figure 15-4. Separate RAM Block
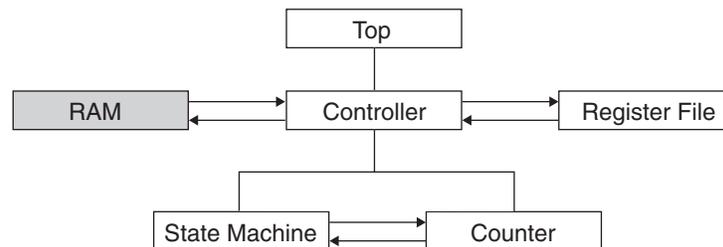
Keep the Number FPGA Gates at 30 to 80 PFUs Per Block

This range varies based on the computer configuration, time required to complete each optimization run, and the targeted FPGA routing resources. Although a smaller block methodology allows more control, it may not produce the most efficient design since it does not provide the synthesis tool enough logic to apply "Resource Sharing" algorithms. On the other hand, having a large number of gates per block gives the synthesis tool too much to work on and causes changes that affect more logic than necessary in an incremental or multi-block design flow.

State Encoding Methodologies for State Machines

There are several ways to encode a state machine, including binary encoding, gray-code encoding and one-hot encoding. State machines with binary or gray-code encoded states have minimal numbers of flip-flops and wide combinatorial functions, which are typically favored for CPLD architectures. However, most FPGAs have many flip-flops and relatively narrow combinatorial function generators. Binary or gray-code encoding schemes can result in inefficient implementation in terms of speed and density for FPGAs. On the other hand, one-hot encoded state machine represents each state with one flip-flop. As a result, it decreases the width of combinatorial logic, which matches well with FPGA architectures. For large and complex state machines, one-hot encoding usually is the preferable method for FPGA architectures. For small state machines, binary encoding or gray-code encoding may be more efficient.

There are many ways to ensure the state machine encoding scheme for a design. One can hard code the states in the source code by specifying a numerical value for each state. This approach ensures the correct encoding of the state machine but is more restrictive in the coding style. The enumerated coding style leaves the flexibility of state machine encoding to the synthesis tools. Most synthesis tools allow users to define encoding styles either through

attributes in the source code or through the tool's Graphical User Interface (GUI). Each synthesis tool has its own synthesis attributes and syntax for choosing the encoding styles. Refer to the synthesis tools documentation for details about attributes syntax and values.

The following syntax defines an enumeration type in VHDL:

```
type type_name is (state1_name,state2_name,.......,stateN_name)
```

Here is a VHDL example of enumeration states:

```
type STATE_TYPE is (S0,S1,S2,S3,S4);
signal CURRENT_STATE, NEXT_STATE : STATE_TYPE;
```

The following are examples of Synplify® and LeonardoSpectrum® VHDL synthesis attributes.

Synplify:

```
attribute syn_encoding : string;
attribute syn_encoding of <signal_name> : type is "value ";
-- The syn_encoding attribute has 4 values : sequential, onehot, gray and safe.
```

LeonardoSpectrum:

```
-- Declare TYPE_ENCODING_STYLE attribute
-- Not needed if the exemplar_1164 package is used
type encoding_style is (BINARY, ONEHOT, GRAY, RANDOM, AUTO);
attribute TYPE_ENCODING_STYLE : encoding_style;
...
attribute TYPE_ENCODING_STYLE of <typename> : type is ONEHOT;
```

In Verilog, one must provide explicit state values for states. This can be done by using the bit pattern (e.g., 3'b001), or by defining a parameter and using it as the case item. The latter method is preferable. The following is an example using parameter for state values.

```
Parameter state1 = 2'h1, state2 = 2'h2;
...
current_state = state2; // setting current state to 2'h2
```

The attributes in the source code override the default encoding style assigned during synthesis. Since Verilog does not have predefined attributes for synthesis, attributes are usually attached to the appropriate objects in the source code as comments. The attributes and their values are case sensitive and usually appear in lower case. The following examples use attributes in Verilog source code to specify state machine encoding style.

Synplify:

```
Reg[2:0] state; /* synthesis syn_encoding = "value" */;
// The syn_encoding attribute has 4 values : sequential, onehot, gray and safe.
```

In LeonardoSpectrum, it is recommended to set the state machine variable to an enumeration type with enum pragma. Once this is set in the source code, encoding schemes can be selected in the LeonardoSpectrum GUI.

LeonardoSpectrum:

```
Parameter /* exemplar enum <type_name> */ s0 = 0, s1 = 1, s2 = 2, s3 = 3, S4 = 4;
Reg [2:0] /* exemplar enum <type_name> */ present_state, next_state ;
```

In general, synthesis tools will select the optimal encoding style that takes into account the target device architecture and size of the decode logic. One can always apply synthesis attributes to override the default encoding style if necessary.

Coding Styles for FSM

A finite state machine (FSM) is a hardware component that advances from the current state to the next state at the clock edge. As mentioned in the Encoding Methodologies for State Machines section, the preferable scheme for FPGA architectures is one-hot encoding. This section discusses some common issues encountered when constructing state machines, such as initialization and state coverage, and special case statements in Verilog.

General State Machine Description

Generally, there are two approaches to describe a state machine. One is to use one process/block to handle both state transitions and state outputs. The other is to separate the state transition and the state outputs into two different process/blocks. The latter approach is more straightforward because it separates the synchronous state registers from the decoding logic used in the computation of the next state and the outputs. This will make the code easier to read and modify, and makes the documentation more efficient. If the outputs of the state machine are combinatorial signals, the second approach is almost always necessary because it will prevent the accidental registering of the state machine outputs.

The following examples describe a simple state machine in VHDL and Verilog. In the VHDL example, a sequential process is separated from the combinatorial process. In Verilog code, two *always* blocks are used to describe the state machine in a similar way.

<table>
<tr><td>

VHDL Example for State Machine

```
. . .
architecture lattice_fpga of dram_refresh is
   type state_typ is (s0, s1, s2, s3, s4);
   signal present_state, next_state : state_typ;
begin
   -- process to update the present state
   registers: process (clk, reset)
   begin
      if (reset='1') then
         present_state <= s0;
      elsif clk'event and clk='1' then
         present_state <= next_state;
      end if;
   end process registers;

   -- process to calculate the next state & output
   transitions: process (present_state, refresh, cs)
   begin
      ras <= '0'; cas <= '0'; ready <= '0';
      case present_state is
         when s0 =>
            ras <= '1'; cas <= '1'; ready <= '1';
            if (refresh = '1') then next_state <= s3;
            elsif (cs = '1') then next_state <= s1;
            else next_state <= s0;
            end if;
         when s1 =>
            ras <= '0'; cas <= '1'; ready <= '0';
            next_state <= s2;
         when s2 =>
            ras <= '0'; cas <= '0'; ready <= '0';
            if (cs = '0') then next_state <= s0;
            else next_state <= s2;
            end if;
         when s3 =>
            ras <= '1'; cas <= '0'; ready <= '0';
            next_state <= s4;
         when s4 =>
            ras <= '0'; cas <= '0'; ready <= '0';
            next_state <= s0;
         when others =>
            ras <= '0'; cas <= '0'; ready <= '0';
            next_state <= s0;
      end case;
   end process transitions;
. . .
```

</td><td>

Verilog Example for State Machine

```
. . .
parameter s0 = 0, s1 = 1, s2 = 2, s3 = 3, s4 = 4;

reg [2:0] present_state, next_state;
reg ras, cas, ready;

// always block to update the present state
always @ (posedge clk or posedge reset)
begin
   if (reset) present_state = s0;
   else present_state = next_state;
end

// always block to calculate the next state & outputs
always @ (present_state or refresh or cs)
begin
   next_state = s0;
   ras = 1'bX; cas = 1'bX; ready = 1'bX;
   case (present_state)
      s0 : if (refresh) begin
              next_state = s3;
              ras = 1'b1; cas = 1'b0; ready = 1'b0;
           end
           else if (cs) begin
              next_state = s1; ras = 1'b0; cas = 1'b1; ready = 1'b0;
           end
           else begin
              next_state = s0; ras = 1'b1; cas = 1'b1; ready = 1'b1;
           end
      s1 : begin
              next_state = s2; ras = 1'b0; cas = 1'b0; ready = 1'b0;
           end
      s2 : if (~cs) begin
              next_state = s0; ras = 1'b1; cas = 1'b1; ready = 1'b1;
           end
           else begin
              next_state = s2; ras = 1'b0; cas = 1'b0; ready = 1'b0;
           end
      s3 : begin
              next_state = s4; ras = 1'b1; cas = 1'b0; ready = 1'b0;
           end
      s4 : begin
              next_state = s0; ras = 1'b0; cas = 1'b0; ready = 1'b0;
           end
   endcase
end
. . .
```

</td></tr>
</table>

Initialization and Default State

A state machine must be initialized to a valid state after power-up. This can be done at the device level during power up or by including a reset operation to bring it to a known state. For all Lattice Semiconductor FPGA devices, the Global Set/Reset (GSR) is pulsed at power-up, regardless of the function defined in the design source code. In the above example, an asynchronous reset can be used to bring the state machine to a valid initialization state. In the same manner, a state machine should have a default state to ensure the state machine will not go into an invalid state if not all the possible combinations are clearly defined in the design source code. VHDL and Verilog have different syntax for default state declaration. In VHDL, if a CASE statement is used to construct a state machine, "When Others" should be used as the last statement before the end of the statement, If an IF-THEN-ELSE statement is used, "Else" should be the last assignment for the state machine. In Verilog, use "default" as the last assignment for a CASE statement, and use "Else" for the IF-THEN-ELSE statement.

```
                  When Others in VHDL
...
architecture lattice_fpga of FSM1 is
   type state_typ is (deflt, idle, read, write);
   signal next_state : state_typ;
begin
process(clk, rst)
begin
   if (rst='1') then
      next_state <= idle; dout <= '0';
   elsif (clk'event and clk='1') then
      case next_state is
         when idle =>
            next_state <= read; dout <= din(0);
         when read =>
            next_state <= write; dout <= din(1);
         when write =>
            next_state <= idle; dout <= din(2);
         when others =>
            next_state <= deflt; dout <= '0';
      end case;
   end if;
end process;
...
```

```
                  Default Clause in Verilog
...
// Define state labels explicitly
parameter deflt=2'bxx;
parameter idle =2'b00;
parameter read =2'b01;
parameter write=2'b10;

reg [1:0] next_state;
reg dout;

always @(posedge clk or posedge rst)
   if (rst) begin
      next_state <= idle;
      dout <= 1'b0;
   end
   else begin
      case(next_state)
         idle: begin
            dout <= din[0]; next_state <= read;
            end
         read: begin
            dout <= din[1]; next_state <= write;
            end
         write: begin
            dout <= din[2]; next_state <= idle;
            end
         default: begin
            dout <= 1'b0; next_state <= deflt;
            end
```

Full Case and Parallel Case Specification in Verilog

Verilog has additional attributes to define the default states without writing it specifically in the code. One can use "full_case" to achieve the same performance as "default". The following examples show the equivalent representations of the same code in Synplify. LeonardoSpectrum allows users to apply Verilog-specific options in the GUI settings.

```
…
case (current_state)  // synthesis full_case
   2'b00 : next_state <= 2'b01;
   2'b01 : next_state <= 2'b11;
   2'b11 : next_state <= 2'b00;
…
```

```
…
case (current_state)
   2'b00 : next_state <= 2'b01;
   2'b01 : next_state <= 2'b11;
   2'b11 : next_state <= 2'b00;
   default : next_state <= 2bx;
```

"Parallel_case" makes sure that all the statements in a case statement are mutually exclusive. It is used to inform the synthesis tools that only one case can be true at a time. The syntax for this attribute in Synplify is as follows:

```
// synthesis parallel_case
```

Using Pipelines in the Designs

Pipelining can improve design performance by restructuring a long data path with several levels of logic and breaking it up over multiple clock cycles. This method allows a faster clock cycle by relaxing the clock-to-output and setup time requirements between the registers. It is usually an advantageous structure for creating faster data paths in register-rich FPGA devices. Knowledge of each FPGA architecture helps in planning pipelines at the

beginning of the design cycle. When the pipelining technique is applied, special care must be taken for the rest of the design to account for the additional data path latency. The following illustrates the same data path before (Figure 15-5) and after pipelining (Figure 15-6).

Figure 15-5. Before Pipelining

Slow Clock
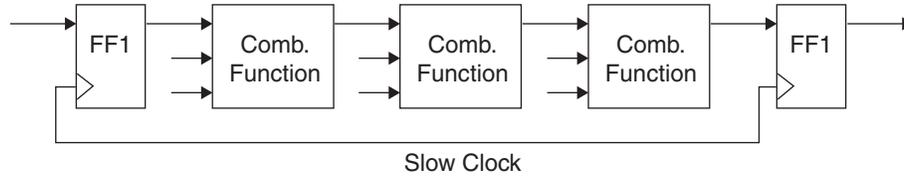
Figure 15-6. After Pipelining

Fast Clock
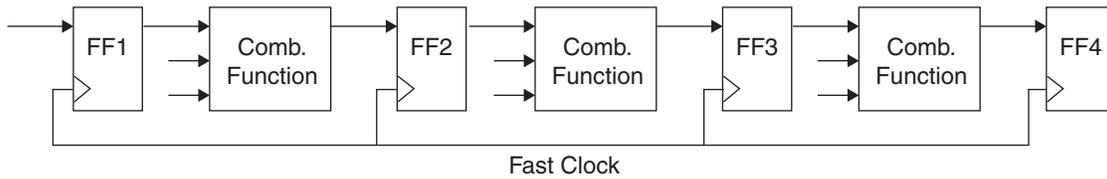
Before pipelining, the clock speed is determined by the clock-to-out time of the source register, the logic delay through four levels of combinatorial logic, the associated routing delays, and the setup time of the destination register. After pipelining is applied, the clock speed is significantly improved by reducing the delay of four logic levels to one logic level and the associated routing delays, even though the rest of the timing requirements remain the same. It is recommended to check the Place and Route timing report to ensure that the pipelined design gives the desired performance.

Comparing IF statement and CASE statement

CASE and IF-THEN-ELSE statements are common for sequential logic in HDL designs. The IF-THEN-ELSE statement generally generates priority-encoded logic, whereas the CASE statement implements balanced logic. An IF-THEN-ELSE statement can contain a set of different expressions while a Case statement is evaluated against a common controlling expression. Both statements will give the same functional implementation if the decode conditions are mutually exclusive, as shown in the following VHDL codes.

```
-- Case Statement — mutually exclusive conditions
process (s, x, y, z)
begin
   O1 <= '0';
   O2 <= '0';
   O3 <= '0';
   case (s) is
      when "00" => O1 <= x;
      when "01" => O2 <= y;
      when "10" => O3 <= z;
   end case;
end process;
```

```
-- If-Then-Else — mutually exclusive conditions
process (s, x, y, z)
begin
   O1 <= '0';
   O2 <= '0';
   O3 <= '0';
   if s = "00" then O1 <= x;
   elsif s = "01" then O2 <= y;
   elsif s = "10" then O3 <= z;
   end if;
end process;
```

However, the use of If-Then-Else construct could be a key pitfall to make the design more complex than necessary, because extra logic are needed to build a priority tree. Consider the following examples:

```
--A: If-Then-Elese Statement: Complex O3 Equations
process(s1, s2, s3, x, y, z)
begin
   O1 <= '0';
   O2 <= '0';
   O3 <= '0';
   if s1 = '1' then
      O1 <= x;
   elsif s2 = '1' then
      O2 <= y;
   elsif s3 = '1' then
      O3 <= z;
   end if;
end process;
```

```
--B: If-Then-Else Statement: Simplified O3 Equation
process (s1, s2, s3, x, y, z)
begin
   O1 <= '0';
   O2 <= '0';
   O3 <= '0';
   if s1 = '1' then
      O1 <= x;
   end if;
   if s2 = '1' then
      O2 <= y;
   end if;
   if s3 <= '1' then
      O3 <= z;
   end if;
 end process;
```

If the decode conditions are not mutually exclusive, IF-THEN-ELSE construct will cause the last output to be dependent on all the control signals. The equation for O3 output in example A is:

```
O3 <= z and (s3) and (not (s1 and s2));
```

If the same code can be written as in example B, most of the synthesis tools will remove the priority tree and decode the output as:

```
O3 <= z and s3;
```

This reduces the logic requirement for the state machine decoder. If each output is indeed dependent of all of the inputs, it is better to use a CASE statement since CASE statements provide equal branches for each output.

Avoiding Non-intentional Latches

Synthesis tools infer latches from incomplete conditional expressions, such as an IF-THEN-ELSE statements without an Else clause. To avoid non-intentional latches, one should specify all conditions explicitly or specify a default assignment. Otherwise, latches will be inserted into the resulting RTL code, requiring additional resources in the device or introducing combinatorial feedback loops that create asynchronous timing problems. Non-intentional latches can be avoided by using clocked registers or by employing any of the following coding techniques:

- Assigning a default value at the beginning of a process
- Assigning outputs for all input conditions
- Using else, (when others) as the final clause

Another way to avoid non-intentional latches is to check the synthesis tool outputs. Most of the synthesis tools give warnings whenever there are latches in the design. Checking the warning list after synthesis will save a tremendous amount of effort in trying to determine why a design is so large later in the Place and Route stage.

HDL Design with Lattice Semiconductor FPGA Devices

The following section discusses the HDL coding techniques utilizing specific Lattice Semiconductor FPGA system features. This kind of architecture-specific coding style will further improve resource utilization and enhance the performance of designs.

Lattice Semiconductor FPGA Synthesis Library

The Lattice Semiconductor FPGA Synthesis Library includes a number of library elements to perform specific logic functions. These library elements are optimized for Lattice Semiconductor FPGAs and have high performance and utilization. The following are the classifications of the library elements in the Lattice Semiconductor FPGA Synthe-

sis Library. The definitions of these library elements can be found in the *Reference Manuals* section of the isp-LEVER on-line help system.

- Logic gates and LUTs
- Comparators, adders, subtractors
- Counters
- Flip-flops and latches
- Memory, 4E-specific memory (block RAM function)
- Multiplexors
- Multipliers
- All I/O cells, including I/O flip-flops
- PIC cells
- Special cells, including PLL, GSR, boundary scan, etc.
- FPSC elements

IPepxress, a parameterized module complier optimized for Lattice FPGA devices, is available for more complex logic functions. IPexpress supports generation of library elements with a number of different options such as PLLs and creates parameterized logic functions such as PFU and EBR memory, multipliers, adders, subtractors, and counters. IPexpress accepts options that specify parameters for parameterized modules such as data path modules and memory modules, and produces a circuit description with Lattice Semiconductor FPGA library elements. Output from IPexpress can be written in EDIF, VHDL, or Verilog. In order to use synthesis tools to utilize the Lattice FPGA architectural features, it is strongly recommended to use IPexpress to generate modules for source code instantiation. The following are examples of Lattice Semiconductor FPGA modules supported by IPexpress:

- PLL
- Memory implemented in PFU:
    – Synchronous single-port RAM, synchronous dual-port RAM, synchronous ROM, synchronous FIFO
- Memory implemented with EBR:
    – Quad-port Block RAM, Dual-Port Block RAM, Single-Port Block RAM, ROM, FIFO
- Other EBR based Functions
    – Multiplier, CAM
- PFU based functions
    – Multiplier, adder, subtractor, adder/subtractor, linear feedback shifter, counter
- MPI/System Bus

IPexpress is especially efficient when generating high pin count modules as it saves time in manually cascading small library elements from the synthesis library. Detailed information about IPexpress and its user guide can be found in the ispLEVER help system.

Implementing Multiplexers

The flexible configurations of LUTs can realize any 4-, 5-, or 6-input logic function like 2-to-1, 3-to-1 or 4-to-1 multi-plexers. Larger multiplexers can be efficiently created by programming multiple 4-input LUTs. Synthesis tools camn automatically infer Lattice FPGA optimized multiplexer library elements based on the behavioral description in the HDL source code. This provides the flexibility to the Mapper and Place and Route tools to configure the LUT mode and connections in the most optimum fashion.

```
16:1 MUX
…
process(sel, din)
   begin
      if    (sel="0000") then muxout <= din(0);
      elsif (sel="0001") then muxout <= din(1);
      elsif (sel="0010") then muxout <= din(2);
      elsif (sel="0011") then muxout <= din(3);
      elsif (sel="0100") then muxout <= din(4);
      elsif (sel="0101") then muxout <= din(5);
      elsif (sel="0110") then muxout <= din(6);
      elsif (sel="0111") then muxout <= din(7);
      elsif (sel="1000") then muxout <= din(8);
      elsif (sel="1001") then muxout <= din(9);
      elsif (sel="1010") then muxout <= din(10);
      elsif (sel="1011") then muxout <= din(11);
      elsif (sel="1100") then muxout <= din(12);
      elsif (sel="1101") then muxout <= din(13);
      elsif (sel="1110") then muxout <= din(14);
      elsif (sel="1111") then muxout <= din(15);
      else muxout <= '0';
      end if;
   end process;
…
```

Clock Dividers

There are two ways to implement clock dividers in Lattice Semiconductor FPGA devices. The first is to cascade the registers with asynchronous clocks. The register output feeds the clock pin of the next register (Figure 15-7). Since the clock number in each PFU is limited to two, any clock divider with more than two bits will require multiple PFU implementations. As a result, the asynchronous daisy chaining implementation of clock divider will be slower due to the inter-PFU routing delays. This kind of delays is usually ambiguous and inconsistent because of the nature of FPGA routing structures.

Figure 15-7. Daisy Chaining of Flip-flops
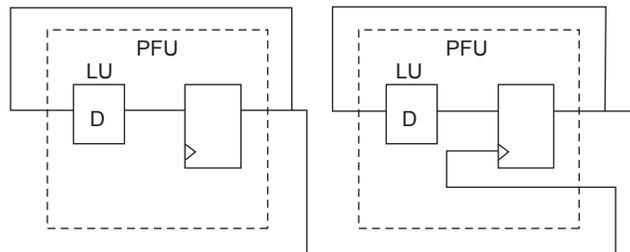
The following are the HDL representations of the design in Figure 15-7.

```
-- VHDL Example of Daisy Chaining FF
...
-- 1st FF to divide Clock in half
CLK_DIV1: process(CLK, RST)
begin
    if (RST='1') then
        clk1 <= '0';
    elsif (CLK'event and CLK='1') then
        clk1 <= not clk1;
    end if;
end process CLK_DIV1;

-- 2nd FF to divide clock in half
CLK_DIV2: process(clk1, RST)
begin
    if (RST='1') then
        clk2 <= '0';
    elsif (clk1'event and clk1='1') then
        clk2 <= not clk2;
    end if;
end process CLK_DIV2;
```

```
//Verilog Example of Daisy Chaining FF
...
always @(posedge CLK or posedge RST)
begin
    if (RST)
        clk1 = 1'b0;
    else
        clk1 = !clk1;
end

always @(posedge clk1 or posedge RST)
begin
    if (RST)
        clk2 = 1'b0;
    else
        clk2 = !clk2;
end

...
```

The preferable way is to fully employ the PFU's natural "Ripple-mode". A single PFU can support up to 8-bit ripple functions with fast carry logic. Figure 15-8 is an example of 4-bit counter in PFU "Ripple Mode". In Lattice Semiconductor FPGA architectures, an internal generated clock can get on the clock spine for small skew clock distribution, further enhancing the performance of the clock divider.

Figure 15-8. Use PFU "Ripple Mode"
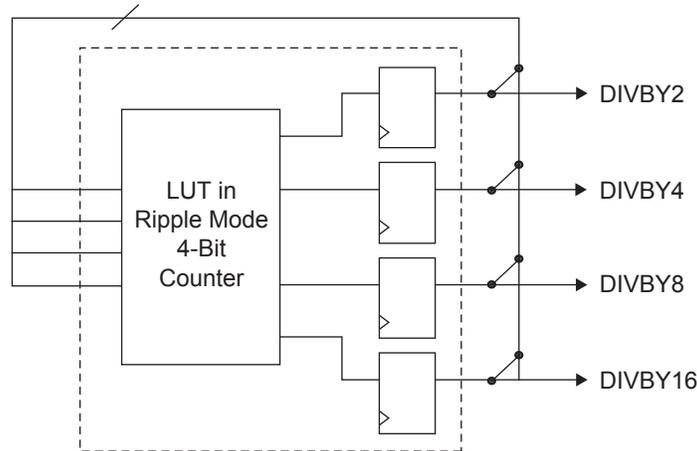
Here are the HDL representations of the design in Figure 15-8.

```
-- VHDL : "RippleMode" Clock Divider
...

COUNT4: process(CLK, RST)
begin
    if (RST='1') then
        cnt <= (others=>'0');
    elsif (CLK'event and CLK='1') then
        cnt <= cnt + 1;
    end if;
end process COUNT4;

DIVBY4  <= cnt(1);
DIVBY16 <= cnt(3);
```

```
//Verilog : "RippleMode" Clock Divider
...
always @(posedge CLK or posedge RST)
begin
    if (RST)
        cnt = 4'b0;
    else
        cnt = cnt + 1'b1;
end

assign DIVBY4  = cnt[1];
assign DIVBY16 = cnt[3];

...
```

Register Control Signals

The general-purpose latches/FFs in the PFU are used in a variety of configurations depending on device family. For example, the Lattice EC, ECP, SC and XP family of devices clock, clock enable and LSR control can be applied to the registers on a slice basis. Each slice contains two LUT4 lookup tables feeding two registers (programmed asto be in FF or Latch mode), and some associated 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 (prgorammable as synchronous/asynchronous), clock select, chip-select and wider RAM/ROM functions. The ORCA Series 4 family of devices clock, clock enable and LSR control can be applied to the registers on a nibble-wide basis. When writing design codes in HDL, keep the architecture in mind to avoid wasting resources in the device. Here are several points for consideration:

- If the register number is not a multiple of 2 or 4 (dependent on device family), try to code the registers in a way that all registers share the same clock, and in a way that all registers share the same control signals.
- Lattice Semiconductor FPGA devices have multiple dedicated Clock Enable signals per PFU. Try to code the asynchronous clocks as clock enables, so that PFU clock signals can be released to use global low-skew clocks.
- Try to code the registers with Local synchronous Set/Reset and Global asynchronous Set/Reset

For more detailed architecture information, refer to the Lattice Semiconductor FPGA data sheets.

Clock Enable

Figure 15-9 shows an example of gated clocking. Gating clock is not encouraged in digital designs because it may cause timing issues such as unexpected clock skews. The structure of the PFU makes the gating clock even more undesirable since it will use up all the clock resources in one PFU and sometimes waste the FF/ Latches resources in the PFU. By using the clock enable in the PFU, the same functionality can be achieved without worrying about timing issues as only one signal is controlling the clock. Since only one clock is used in the PFU, all related logic can be implemented in one block to achieve better performance. Figure 15-10 shows the design with clock enable signal being used.

Figure 15-9. Asynchronous: Gated Clocking
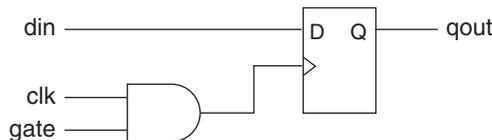
Figure 15-10. Synchronous: Clock Enabling
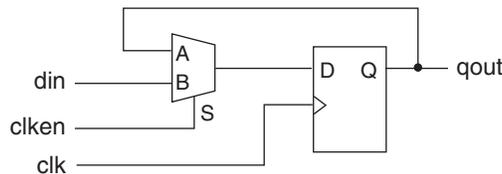
The VHDL and Verilog coding for Clock Enable are as shown in Figure 15-10.

```
-- VHDL example for Clock Enable

...
Clock_Enable: process(clk)
begin
   if (clk'event or clk='1') then
      if (clken='1') then
         qout <= din;
      end if;
   end if;
end process Clock_Enable;
```

```
// Verilog example for Clock Enable

...
always @(posedge clk)
   qout <= clken ? din : qout;

...
```

The following are guidelines for coding the Clock Enable in Lattice Semiconductor FPGAs:

- Clock Enable is only supported by FFs, not latches.
- Nibble wide FFs and slices inside a PFU share the same Clock Enable
- All flip-flops in the Lattice Semiconductor FPGA library have a positive clock enable signal
- In the ORCA Series 4 architecture, the Clock Enable signal has the higher priority over synchronous set/reset by default. However, it can be programmed to have the priority of synchronous LSR over the priority of Clock Enable. This can be achieved by instantiating the library element in the source code. For example, the library element FD1P3IX is a flip-flop that allows synchronous Clear to override Clock Enable. Users can also specify the priority of generic coding by setting the priority of the control signals differently. The following examples demonstrate coding methodologies to help the synthesis tools to set the higher priority of Clock Enable or synchronous LSR.

```
-- VHDL Example of CE over Sync. LSR
...
COUNT8: process(CLK, GRST)
begin
   if (GRST = '1') then
      cnt <= (others => '0');
   elsif (CLK'event and CLK='1') then
   -- CE Over LSR: Clock Enable has higher priority
      if (CKEN = '1') then
         cnt <= cnt + 1;
      elsif (LRST = '1') then
         cnt <= (others =>'0');
      end if;
   end if;
end process COUNT8;
```

```
// Verilog Example of CE over Sync. LSR
...
always @(posedge CLK or posedge GRST)
begin
   if (GRST)
      cnt = 4'b0;
   else
      if (CKEN)
         cnt = cnt + 1'b1;
      else if (LRST)
         cnt = 4'b0;
end...
```

```
-- VHDL Example of Sync. LSR Over CE
...
COUNT8: process(CLK, GRST)
begin
   if (GRST = '1') then
      cnt <= (others => '0');
   elsif (CLK'event and CLK='1') then
   -- LSR over CE: Sync. Set/Reset has higher priority
      if (LRST = '1') then
         cnt <= (others => '0');
      elsif (CKEN = '1') then
         cnt <= cnt + 1;
      end if;
```

```
// Verilog Example of Sync. LSR Over CE
...
always @(posedge CLK or posedge GRST)
begin
   if (GRST)
      cnt = 4'b0;
   else if (LRST)
      cnt = 4'b0;
   else if (CKEN)
      cnt = cnt + 1'b1;
end
...
```

SET / Reset

There are two types of set/reset functions in Lattice Semiconductor FPGAs: Global (GSR) and Local (LSR). The GSR signal is asynchronous and is used to initialize all registers during configuration. It can be activated either by an external dedicated pin or from internal logic after configuration. The local SET/Reset signal may be synchronous or asynchronous. GSR is pulsed at power up to either set or reset the registers depending on the configuration of the device. Since the GSR signal has dedicated routing resources that connect to the set and reset pin of the flip-flops, it saves general-purpose routing and buffering resources and improves overall performance. If asynchronous reset is used in the design, it is recommended to use the GSR for this function, if possible. The reset signal can be forced to be GSR by the instantiation library element. Synthesis tools will automatically infer GSR if all

registers in the design are asynchronously set or reset by the same wire. The following examples show the correct syntax for instantiating GSR in the VHDL and Verilog codes.

```
-- VHDL Example of GSR Instantiation
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity gsr_test is
        port (rst, clk: in std_logic;
        cntout : out std_logic_vector(1 downto 0));
end gsr_test;

architecture behave of gsr_test is
        signal cnt : std_logic_vector(1 downto 0);
begin

        u1: GSR port map (gsr=>rst);

        process(clk, rst)
        begin
                if rst = '1' then
                        cnt <= "00";
                elsif rising_edge (clk) then
                        cnt <= cnt + 1;
                end if;
        end process;
        cntout <= cnt;
end behave;
```

```
// Verilog Example of GSR Instantiation

module gsr_test(clk, rst, cntout);

input clk, rst;
output[1:0] cntout;

reg[1:0] cnt;

GSR u1 (.GSR(rst));

always @(posedge clk or negedge rst)
begin
        if (!rst)
                cnt = 2'b0;
        else
                cnt = cnt + 1;
end

assign cntout = cnt;

endmodule
```

Use PIC Features

Using I/O Registers/Latches in PIC

Moving registers or latches into Input/Output cells (PIC) may reduce the number of PFUs used and decrease routing congestion. In addition, it reduces setup time requirements for incoming data and clock-to-output delay for output data, as shown in Figure 15-11. Most synthesis tools will infer input registers or output registers in PIC if possible. Users can set synthesis attributes in the specific tools to turn off the auto-infer capability. Users can also instantiate library elements to control the implementation of PIC resource usage.

Figure 15-11. Moving FF into PIC Input Register

Before Using Input Register   **After Using Input Register**
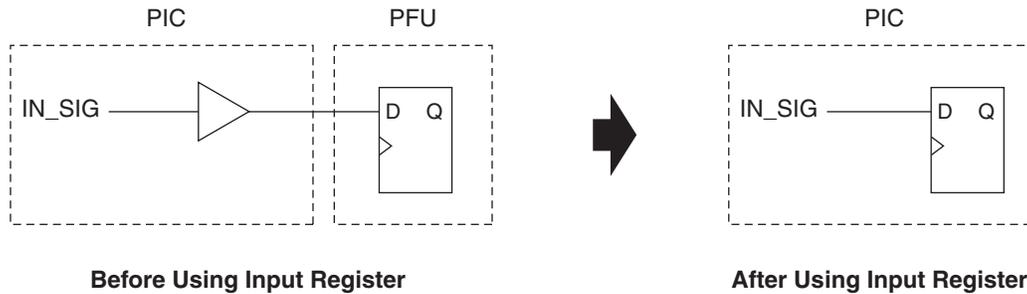
Figure 15-12. Moving FF into PIC Output Register

Before Using Output Register   **After Using Output Register**
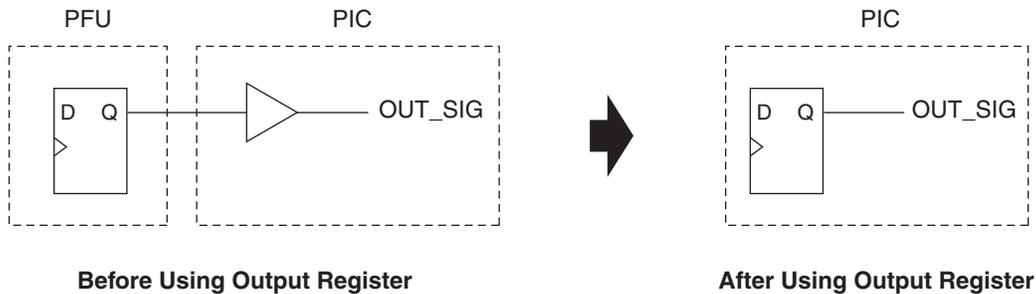
Inferring Bi-directional I/O

Users can either structurally instantiate the bi-directional I/O library elements, or behaviorally describe the I/O paths to infer bi-directional buffers. The following VHDL and Verilog examples show how to infer bi-directional I/O buffers.

```
-- Inferring Bi-directional I/O in VHDL

library ieee;
use ieee.std_logic_1164.all;

entity bidir_infer is
   port(A, B   : inout std_logic;
         dir    : in std_logic);
end bidir_infer;

architecture lattice_fpga of bidir_infer is
begin
   B <= A when (dir='1') else 'Z';
   A <= B when (dir='0') else 'Z';
end lattice_fpga
```

```
// Inferring Bi-directional I/O in Verilog

module bidir_infer (A, B, DIR);
inout A, B;
input DIR;

assign B = (DIR)  ? A : 1'bz;
assign A = (~DIR) ? B : 1'bz;

endmodule
```

Specifying I/O Types and Locations

Users can either assign I/O types and unique I/O locations in the Preference Editor or specify them as attributes in the VHDL or Verilog source code. The following examples show how to add attributes in the Synplify and Leonardo-Spectrum synthesis tool sets. For a complete list of supported attributes, refer to the HDL Attributes section of the ispLEVER on-line help system.

```
-- VHDL example of specifying I/O type and location attributes for Synplify & Leonardo

entity cnt is
    port(clk: in std_logic;
          res: out std_logic);
    attribute LEVELMODE: string:
    attribute LEVELMODE of clk : signal is "SSTL2";
    attribute LOC of clk : signal is "V2";
    attribute LEVELMODE of res : signal is "SSTL2";
    attribute LOC of res : signal is "V3";
end entity cnt;
```

```
-- Verilog example of specifying I/O type and location attributes for Synplify & Leonardo

module cnt(clk,res);

    input clk /* synthesis LEVELMODE="SSTL2" LOC="V2"*/;
    output res /* synthesis LEVELMODE="SSTL2" LOC="V3" */;

    ...

    // exemplar begin
        // exemplar attribute clk LEVELMODE SSTL2
        // exemplar attribute clk LOC V2
        // exemplar attribute res LEVELMODE SSTL2
        // exemplar attribute res LOC V3
    // exemplar end

endmodule
```

Implementation of Memories

Although an RTL description of RAM is portable and the coding is straightforward, it is not recommended because the structure of RAM blocks in every architecture is unique. Synthesis tools are not optimized to handle RAM implementation and thus generate inefficient netlists for device fitting. For Lattice Semiconductor FPGA devices, RAM blocks should be generated through IPexpress as shown in the following screen shot.
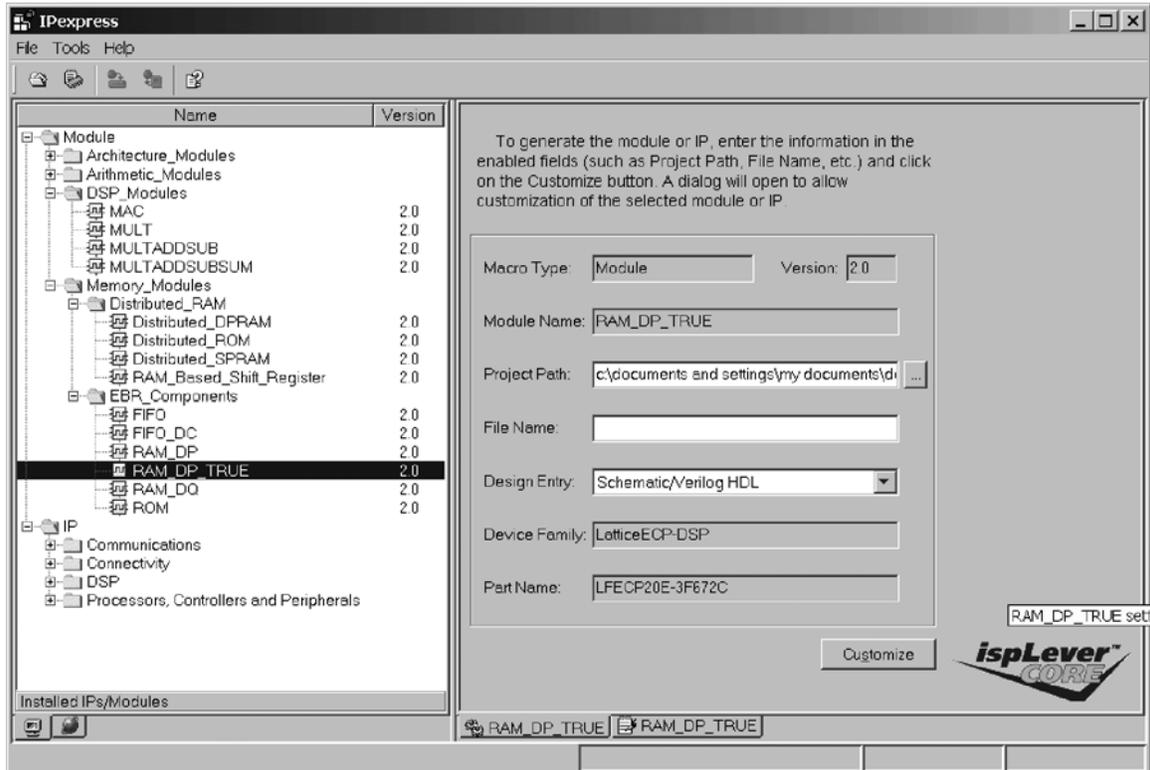
When implementing large memories in the design, it is recommended to construct the memory from the Enhanced Block RAM (EBR) components found in every Lattice Semiconductor FPGA device. When implementing small memories in the design, it is recommended to construct the memory from the resources in the PFU. The memory utilizing resources in the PFU can also be generated by IPexpress.

Lattice Semiconductor FPGAs support many different memory types including synchronous dual-port RAM, synchronous single-port RAM, synchronous FIFO and synchronous ROM. For more information on supported memory types per FPGA architecture, please consult the Lattice Semiconductor FPGA data sheets.

Preventing Logic Replication and Limited Fanout

Lattice Semiconductor FPGA device architectures are designed to handle high signal fanouts. When users make use of clock resources, there will be no hindrance on fanout problems. However, synthesis tools tend to replicate logic to reduce fanout during logic synthesis. For example, if the code implies Clock Enable and is synthesized with speed constraints, the synthesis tool may replicate the Clock Enable logic. This kind of logic replication occupies more resources in the devices and makes performance checking more difficult. It is recommended to control the logic replication in synthesis process by using attributes for high fanout limit.

In the Synplicity® project GUI, under the Implementation Options => Devices tab, users can set the Fanout Guide value to 1000 instead of using the default value of 100. This will guide the tool to allow high fanout signals without replicating the logic. In the LeonardoSpectrum tool project GUI, under Technology => Advanced Settings, users can set the Max Fanout to be any number instead of the default value "0".
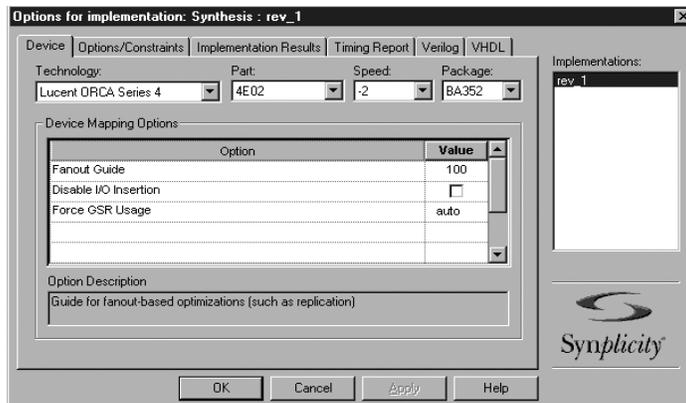
Use ispLEVER Project Navigator Results for Device Utilization and Performance

Many synthesis tools give usage reports at the end of a successful synthesis. These reports show the name and the number of library elements used in the design. The data in these reports do not represent the actual implementation of the design in the final Place and Route tool because the EDIF netlist will be further optimized during Mapping and Place and Route to achieve the best results. It is strongly recommended to use the MAP report and the PAR report in the ispLEVER Project Navigator tool to understand the actual resource utilization in the device. Although the synthesis report also provides a performance summary, the timing information is based on estimated logic delays only. The Place & Route TRACE Report in the ispLEVER Project Navigator gives accurate performance analysis of the design by including actual logic and routing delays in the paths.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

             +1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Introduction

Lattice Semiconductor's ispLEVER® software, together with Lattice Semiconductor's catalog of programmable devices, provides options to help meet design timing and logic utilization requirements. Additionally, for those instances where objectives push the capabilities of the device architecture, ispLEVER provides the tools for meeting the most challenging requirements.

For the most aggressive design requirements, the designer should become familiar with a variety of timing constraints (called preferences) and Place And Route (PAR) techniques for providing the optimal PAR results as discussed in technical note number TN1018, Lattice Semiconductor Successful Place and Route.

If performance goals cannot be met with FPGA timing preferences and additional levels of the Place & Route design process, improved performance can be achieved by directing the physical layout of the circuit in the FPGA. This step, often referred to as floorplanning, is done by specifying FPGA location preferences.

This application note explains what floorplanning is, when it should be used, and how it is done with respect to Lattice Semiconductor FPGA designs. This document is divided into four major sections:

- Floorplanning definition, logical and physical
- When to floorplan, general versus specific reasons
- How to floorplan: grouping constraints, examples, and Floorplanner GUI presentation
- Special considerations, large and special groupings

Supported Architectures

Floorplanning can be done on all Lattice Semiconductor FPGA architectures.

Related Documentation

Designers are encouraged to reference the ispLEVER on-line documentation.

Floorplanning Definition

Floorplanning a digital logic design for implementation in a programmable device involves the physical or logical partitioning of the design elements which results in a change in the physical placement or implementation of the design elements. In other words, floorplanning is the grouping of design elements in a certain way to improve the performance of a design.

With Lattice Semiconductor FPGAs, floorplanning is an optional methodology to help designers improve the performance and density of a fully, automatically placed and routed design. Floorplanning is particularly useful in structured designs and data path logic. Design floorplanning is very powerful and provides a combination of automation and user control for design reuse and modular, hierarchical, and incremental design flows.

Complex FPGA Design Management

Lattice Semiconductor FPGAs can implement large system designs that contain millions of gates, hundreds of thousands of embedded memory bits, and intellectual property (IP) components. Design teams often work on large designs. The design complexity requires EDA tools to manage and optimize the design. Large design management is difficult, but performance optimization is even more difficult. Optimization requires many design iterations when adding or modifying components. Complex, large system designs require the following:

- The use of modular, hierarchical, or incremental design methods
- Software that makes management and optimization easier

- Use of IP blocks
- Reuse of previously optimized design elements

By controlling the placement of specified logic elements, design floorplanning methodologies help designers meet the requirements of large system design.

Floorplanning Design Flow

In both traditional and floorplanning FPGA design flows, the designer divides the system into modules. The modules can be individual circuits, parts of circuits, or parts of the design hierarchy. After module design and optimization, the designer integrates the modules into the system. Finally, the designer tests and optimizes the system.

In the traditional flow, the system may not meet performance requirements even if each module meets the requirements before integration. Even when timing requirements have been satisfied, changes to one module can affect the performance of others. Re-optimizing modules to meet system performance results in many design iterations.

Floorplanning methodologies assist in the design, testing, and optimization of each individual module while retaining the optimized characteristics of the individual modules. Module integration into the system requires only system optimization between modules. The floorplanning methodologies provide additional flexibility by allowing the ispLEVER software to automatically place defined modules, or allowing the user to control the placement of specific modules, which provide performance preservation and optimization.

When to Floorplan

Floorplanning methodologies are intended to assist users who require some degree of handcrafting for their designs. The designer must understand both the details of the device architectures and the ways floorplanning can be used to refine a design. Successful floorplanning is very much an iterative process and it can take time to develop a floorplan that outperforms an automatic, software processed design. Because of the nature of floorplanning and its interaction with the automatic MAP and PAR software tools, several prerequisites are necessary in order to floorplan a design successfully.

- Detailed knowledge of the specifics of the target architecture and device
- Detailed knowledge of the specifics of the design being implemented
- A design that lends itself to floorplanning
- A willingness to iterate a floorplan to achieve the desired results
- Realistic performance and density goals

For Lattice Semiconductor FPGAs, the general rule-of-thumb is that floorplanning should be considered when the performance needed cannot be met and routing delays account for over 60% of the critical path delays. That is, interacting components are too far apart in the FPGA array to achieve short routing delays. This has shown to be a problem especially with large designs in high density FPGAs because of the possibilities of long distance routes. As programmable logic design densities continue to escalate beyond 100,000 gates, traditional design flow — design entry to synthesis to place and route — will sometimes not yield predictable, timely, and optimized results.

Note that the guidelines discussed above only apply to designs that have been routed by the software for several routing iterations. The default number of routing iterations via the ispLEVER Project Navigator is variable depending on the Lattice Semiconductor FPGA device family chosen.

A note on delays: Path delays in programmable devices are made up of two parts: logical delays and routing delays. Logical delays in this context are delays through components, such as a a programmable function unit (PFU), a programmable input/output (PIO), a slice, or an embedded function (i.e. a block RAM, PLL, or embedded FPSC ASIC). The routing delay is the delay of the interconnect between components. Figures 1 and 2 show delay examples from timing wizard report files (.twr).

Figure 16-1. Floorplanning May be Able to Help Bring These Registers Closer

```
Logical Details:  Cell type  Pin type   Cell name  (clock net +/-)


 Source:          FF         Q             ibuf/reg_init_start (from clk_ib+)
 Destination:     FF         Data in    ibuf/sd/reg_new_state (to clk_ib +)


 Delay:                 8.062ns   (18.2% logic, 81.8% route), 2 logic levels.
```

Figure 16-2. Floorplanning is Not Needed Here Because the Routing is Efficient

```
Logical Details:  Cell type  Pin type   Cell name  (clock net +/-)


 Source:          FF         Q             mem_if_tx_address_8 (from clk_c +)
 Destination:     FF         Data in    mem_if_tx_address_17 (to clk_c +)


 Delay:                 7.358ns   (61.2% logic, 38.8% route), 4 logic levels.
```

Floorplan to Improve Design Performance

Properly applied, design floorplanning not only preserves, but improves design performance. Floorplanning methodologies can be used to place modules, entities, or any group of logic to regions in a device's floorplan. Because floorplanning assignments can be hierarchical, designers can have more control over the placement and performance of modules and groups of modules.

In addition to hierarchical blocks, like grouping an entire VHDL entity or Verilog module, designers can floorplan individual nodes (e.g., instantiate a library element for a function in the critical path and then group the library element). This technique is useful if the critical path spans multiple design blocks.

Note: Although floorplanning can increase performance, it may also degrade performance if it is not applied correctly within software limitations.

Floorplan to Preserve Module Performance

Floorplanning with design constraints maintains design performance by grouping the placement of nodes in a device, i.e., the relative placement of logic within a grouped region remains constant. The ispLEVER software then places the grouped region into the top-level design with these constraints. When placing logic in a region, the ispLEVER software does not preserve the routing information. This approach provides more flexibility when the software imports the region into the top-level design and helps fitting.

Floorplan for Design Reuse

Floorplanning facilitates design reuse by its ability to reproduce the performance of a module designed in a different project. For frequently used modules, designers can create a library of verified designs that can be incorporated into other larger designs. The library only has to contain the VHDL or Verilog source code along with grouping attributes and some comments detailing useful information to the user, such as performance and size. With a parameterized module, in-code assignments can specify the module's size and grouping assignments.

Targeting the same device used in the original design likely achieves the best results, although other devices in the same family will likely work well. When using a different device in the same family, the exact placement of the region may not be possible. Similar performance, however, may be possible by moving or floating regions. A floating region groups the logic together and guides the ispLEVER software toward achieving a placement that meets the performance requirements of the module. A similar approach can also be taken if exact placement of a module is not applicable because of multiple instantiations of a module in a top-level design.

How to Floorplan a Design

Design Performance Enhancement Strategies

Floorplanning methodologies improve the performance of designs that do not necessarily consist of individually optimized modules. The ability of specifying regions to group nodes together and provide relative placement enhances the usability of the ispLEVER place-and-route software tools. The design strategies for performance enhancement depend on the structure of a particular circuit. Strategies include:

- Defining regions based on design hierarchy if the hierarchy closely resembles the structure of the circuit. These designs typically consist of tightly integrated modules, where the logic for each module is self-contained and modules communicate through well-defined interfaces.

- Defining regions based on the critical path, if the critical path is long and spans multiple modules. Keeping the nodes in the critical path or the modules containing the critical path together may lead to improved performance.

- Defining regions based on connections by grouping nodes with high fan-outs and high fan-ins together to reduce delays in connections and wiring congestion in the device.

Note that designers may need to change existing design hierarchy and structure to make the design more amiable to floorplanning. This is especially applicable if modular hierarchy and structure was not considered at the beginning of design conception.

With the floorplaning methodologies, the user can choose to optimize modules either individually or after they have been integrated with the top-level design. The user can exercise varying amounts of control over the placement by using different types of regions. By using bounding boxes and location anchors selectively, the ispLEVER software can easily determine the best size and location for a region. Another approach is to optimize the top-level design without first optimizing the individual modules. This approach allows the ispLEVER software to place nodes within regions and move regions across the device. The user assigns modules to regions and then compiles the entire design. With this approach the user can place elements from different modules in a region.

Design Floorplanning Methodologies

There are several methods available to aid in the floorplanning of a logical design in a Lattice Semiconductor FPGA. This section illustrates these methods with examples of how to use the ispLEVER software tools to achieve performance goals. The three main floorplanning tools available include:

- The **PGROUP Physical Constraint** can be used as an attribute in VHDL and Verilog HDL source code or as a physical Preference in the .prf design constraint file. This can be used directly by the ispLEVER Placer software to bound and locate sections of a design for grouping in the FPGA array.

- The **UGROUP Logical Constraint** can be used as an attribute in VHDL and Verilog HDL source code to gather logical sections of a design for grouping in the FPGA array.

- The **Floorplanner Graphical User Interface (GUI)** can be used to interactively specify placement parameters in either the logical or physical domain for some of a design's modules (e.g. logic gates, registers, arithmetic functions) from one graphical user interface.

When to use PGROUP vs. UGROUP

UGROUPing differs from PGROUPing as follows:

- A PGROUP logical identifier, in EDIF, is prepended with text that describes the identifier's hierarchy.

- A UGROUP logical identifier, in EDIF, is not changed by prepending the hierarchy on the block instance identifier.

In other words, PGROUPing enforces strict hierarchical control while UGROUPing allows for a grouping of blocks in different hierarchies or a grouping of blocks with no hierarchy at all.

Also note that the PGROUP attribute can be placed on multiple instantiations of modules (e.g. VHDL generate statements) and each instantiation will have its own PGROUP. Using a UGROUP will not work in this case.

In Figures 3 and 4, the arrows represent control and data paths where there is interaction between different levels of hierarchy. The thick lined arrow represents the critical path where the design is failing to make performance.

Figure 16-3. PGROUP Same Hierarchy Example, PGROUP CONTROLLER
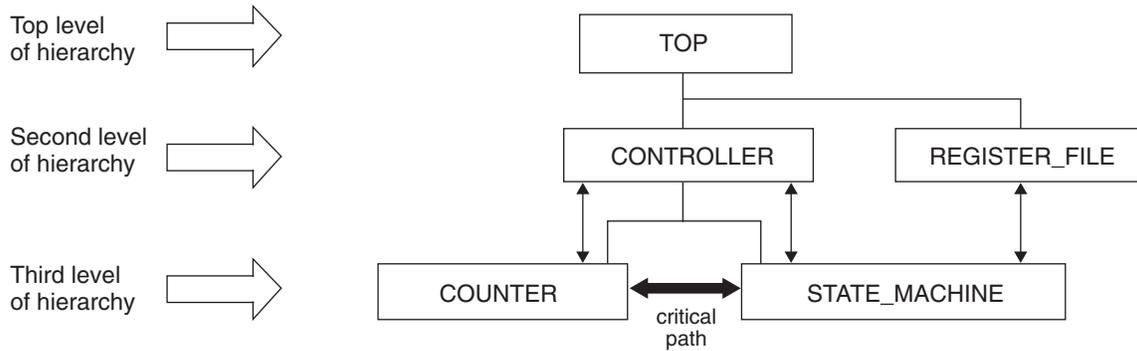
Figure 16-3 illustrates a design hierarchy where the failing paths are the connections between COUNTER and STATE_MACHINE design blocks. The easiest implementation for this example is to PGROUP the CONTROLLER, which is the module in which the COUNTER and STATE_MACHINE are instantiated within.

For example, if the following Synplify attribute is in the Verilog HDL file:

```
module CONTROLLER (<port_list>)/* synthesis pgroup="CONTROL_GROUP" */;
```

then the COUNTER and STATE_MACHINE will be grouped in the FPGA inside a boundary box. Now assume that the COUNTER is mapped into PFU_0 and PFU_1 and the STATE_MACHINE is mapped into PFU_2. The resulting preference generated by map in the `.prf` file will be:

```
PGROUP "TOP/CONTROLLER/CONTROL_GROUP"
   COMP "PFU_0"
   COMP "PFU_1"
   COMP "PFU_2";
```

Notice the `TOP/` hierarchy is prepended to the CONTROLLER PGROUP identifier.

Figure 16-4. UGROUP Different Hierarchy Example, UGROUP REGISTER_FILE and STATE MACHINE
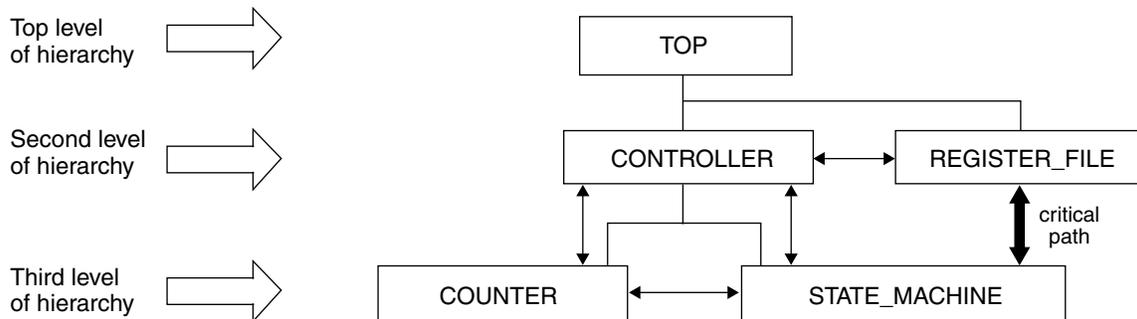
Figure 16-4 shows an example design hierarchy where the failing paths are the connections between REGISTER_FILE and STATE_MACHINE modules. The simplest thing to do here is to UGROUP the REGISTER_FILE and STATE_MACHINE together.

For example, if the following Synplify attributes are in the Verilog HDL file:

```
module REGISTER_FILE (<port_list>) /*synthesis ugroup="CRITICAL_GROUP" */;
```

and

```
module STATE_MACHINE (<port_list>) /*synthesis ugroup="CRITICAL_GROUP" */;
```

then the REGISTER_FILE and STATE_MACHINE will be grouped in the FPGA inside a default boundary box. Now assume that the REGISTER_FILE is mapped into PFU_4 and PFU_5 and the STATE_MACHINE is mapped into PFU_3. The resulting preference generated by map in the `.prf` file will be:

```
PGROUP "CRITICAL_GROUP"
    COMP "PFU_3"
    COMP "PFU_4"
    COMP "PFU_5";
```

Notice the `TOP/` hierarchy is not appended to the PGROUP identifier CRITICAL_GROUP. Also notice that UGROUP attributes result in PGROUP preferences. There is no UGROUP preference.

If PGROUP attributes instead of UGROUP attributes had been used for Figure 16-4:

```
module REGISTER_FILE (<port_list>) /*synthesis pgroup="CRITICAL_GROUP" */;
```

and

```
module STATE_MACHINE (<port_list>) /*synthesis pgroup="CRITICAL_GROUP" */;
```

then the resulting preference generated by map in the `.prf` file would be:

```
PGROUP "TOP/CONTROLLER/STATE_MACHINE/CRITICAL_GROUP"
    COMP "PFU_3"

PGROUP "TOP/REGISTER_FILE/CRITICAL_GROUP"
    COMP "PFU_4"
    COMP "PFU_5";
```

So, with PGROUP attributes, the STATE_MACHINE module would be grouped together in one bounding box and REGISTER_FILE module would be grouped together separately in another bounding box and the critical path shown in Figure 16-4 will not be optimized.

These examples do not utilize all the possible tools available for floorplanning. Please refer to ispLEVER On-line Help PGROUP section for many small syntax examples.

Floorplanner GUI Usage

Generally, the PGROUPs and UGROUPs are preferable to the Floorplanner GUI since they are easier to implement. For example, it is easier to type in a PGROUP attribute in the HDL code then to load the GUI with large netlists and find the desired block and perform add the PGROUP via mouse clicks. More importantly, the GUI does not allow the retention of floorplanning the way PGROUPing and UGROUPing does. Since the GUI does not back annotate the grouping attributes into the HDL, the GUI operations have to be redone every time there is a new design iteration.

The Floorplanner GUI can be useful for

- Viewing elements in a graphical environment to see a design's logical hierarchy.
- Viewing existing PGROUPs and UGROUPs.
- Resizing regions and boundary boxes (BBOXes).
- Graphically placing regions, PGROUPs, and UGROUPs and then running map, place, route, and trace to see the effects. This is usually an iterative process before finding an optimal solution.

If the Floorplanner GUI is used, it is strongly recommended that after finding the optimal grouping with the GUI, grouping attributes (PGROUP and/or UGROUPS) should be inserted into the HDL source code to preserve module performance over design revisions.

Special Floorplanning Considerations

Embedded Block RAM Placement

Block RAM placement can be done with simple LOCATE preferences. It is not always necessary to locate block RAMs. Do not use the PGROUPs, UGROUPs, or the Floorplanner GUI to group Block RAMs.

I/O Grouping

There is a complete set of physical constraints on PGROUPing I/O components. Please refer to ispLEVER On-line Help, Defining PIO Component Groups, for keyword explanations and syntax examples.

Large Module Grouping

It is strongly recommended that larger PGROUPs/UGROUPs (with many logical elements) be anchored and bounded by LOCATE and BBOX keywords. From the STATE_MACHINE example, we see that without anchoring the groups, the performance worsened compared to no floorplanning at all.

The BBOX should be strategically shaped and sized according to the module to be placed inside the BBOX. If the BBOX shape and size is not specified, the default BBOX size will be a square that is as small as possible. This is not the optimal BBOX for typical modules. The designer should shape the design with the datapath in mind and size the BBOX to be larger then needed so that the ispLEVER placer program can have more flexibility in placing logic elements inside the BBOX.

Carry Chains and Bus Grouping

Carry chains (used by ripple arithmetic functions like adders, counters, and multipliers) and logic modules connected by busses can easily be floorplanned inappropriately by a designer that is not aware of the internal routing resources available to optimize these carry chain and bus routes. As we saw from the multiplier example, certain groupings can reduce the performance of a design compared to no floorplanning at all. Great care should be used when floorplanning designs that use carry chains or busses so that these routes fall in optimal locations for optimal performance.

Broken Carry Chain Example
A 9-bit adder that is PGROUPed with no relative placement on the adder. Logic elements such as PFUs may give worse performance because the adder carry-chain is broken.

SLICs in Groups

For Lattice Semiconductor FPGA device families that contain Supplemental Logic and Interconnect Cells (SLICs), the SLICs are automatically removed from PGROUPs and UGROUPs by the ispLEVER software if they are not relatively placed. This is because SLICs are used by the tools for interconnects that are not foreseeable by designers. If SLIC placement has to be controlled for a design, the designer will need to instantiate and locate the SLICS in their preference or HDL files. It is recommended to allow the ispLEVER software to automatically place SLICs.

Summary

This application note defined floorplanning, discussed when it should be used, and detailed how floorplanning is done with respect to Lattice Semiconductor FPGA designs. Examples were used to illustrate and compare the different tools available to the designer for floorplanning.

Important items discussed:

- Floorplanning can improve timing for targeted critical paths.
- Improper floorplanning can make timing worse.

- Correct floorplanning can increase logic element count slightly.

- Completed floorplanning should be annotated into the HDL.

- Floorplanning enhances the reusability of designs by keeping placement directives in the HDL.

- Use PGROUPs for physical grouping in the same hierarchy.

- Use UGROUPs for logical grouping across hierarchies.

With Lattice Semiconductor FPGAs, floorplanning is an *optional* methodology to help designers improve performance and density of a fully, automatically placed and routed design. Floorplanning is particularly useful on structured designs and data path logic.

Design floorplanning is very powerful and provides a combination of automation and user control for design reuse and modular, hierarchical, and incremental design flows. It advances the design of large systems on FPGAs. It provides the designer with capabilities in the management and optimization of large systems. Its multifaceted capabilities result in shortened design cycles and faster time to market.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

   +1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Lattice Semiconductor FPGA Successful Place and Route

Introduction

Lattice Semiconductor's ispLEVER® software, together with Lattice Semiconductor's catalog of programmable devices, provides options to help meet design timing and logic utilization requirements. Additionally, for those instances where objectives push the capabilities of the device architecture, ispLEVER provides the tools for meeting the most challenging requirements.

For the most aggressive design requirements, the designer should become familiar with a variety of timing constraints (called preferences) and Place And Route (PAR) techniques for providing the optimal PAR results. This document describes these tips and techniques. Advanced techniques in floorplanning will not be discussed in this document. Instead they are covered in technical note number TN1010, *Lattice Semiconductor Design Floorplanning*.

ispLEVER Place and Route Software (PAR)

In the ispLEVER design flow, after a design has undergone the necessary translation to bring it into the mapped physical design (.ncd file) format, it is ready for placement and routing. This phase is handled by the timing-driven PAR software program. Designers can invoke PAR from the ispLEVER Project Navigator or from the command line.

PAR performs the following:

- Takes a mapped physical design (.ncd file) and a preference file (.prf) as input files.
- Places and routes the design, attempting to meet the timing preferences in the input .prf file.
- Creates a file which can then be processed by the ispLEVER design implementation tools.

Placement

The PAR process places the mapped physical design (.ncd file) in two stages: a constructive placement and an optimizing placement. PAR writes the physical design after each of these stages is complete.

During constructive placement, PAR places components into sites based on factors such as:

- Constraints specified in the input file (for example, certain components must be in certain locations).
- The length of connections.
- The available routing resources.
- Cost tables which assign random weighted values to each of the relevant factors. There are 100 possible cost tables.

Constructive placement continues until all components are placed. Optimizing placement is a fine-tuning of the results of the constructive placement.

Routing

Routing is also done in two stages: iterative routing and delay reduction routing (also called cleanup). PAR writes the physical design (.ncd file) only after iterations where the routing score has improved.

During iterative routing, the router performs an iterative procedure to converge on a solution that routes the design to completion or minimizes the number of unrouted nets.

During cleanup routing (also called delay reduction), the router takes the results of iterative routing and reroutes some connections to minimize the signal delays within the device. There are two types of cleanup routing that can be performed:

- A faster cost-based cleanup routing, which makes routing decisions by assigning weighted values to the factors (for example, the type of routing resources used) affecting delay times between sources and loads.
- A more CPU-intensive, delay-based cleanup routing, which makes routing decisions based on computed delay times between sources and loads on the routed nets.

Note that if PAR finds timing preferences in the preference file, timing-driven placement and routing is automatically invoked.

Timing Driven PAR Process

The ispLEVER software offers timing driven placement and routing through an integrated static timing analysis utility (i.e., it does not depend on input stimulus to the circuit). This means that placement and routing is executed according to timing constraints (preferences) that the designer specifies up front in the design process. PAR attempts to meet timing constraints in the preference file without exceeding the specified timing constraints.

To use timing-driven PAR, the designer simply writes timing preferences into a preference (.prf) file, which serves as input to the integrated static timing analysis utility. See the *Process Flows* section of the ispLEVER on-line help system for more information about the PAR software and ispLEVER design flow.

General Strategy Guidelines

Preferences should be inserted at the front end of a design flow. This prevents designers from having to change PAR physical preferences as net names may change with every synthesis run.

The tips below are general recommendations.

- Analyze Trace results in the integrated static timing analysis utility report (.twr) file carefully.

- Look at mapped frequency before you PAR a design to check for errors and warnings in the preference file and to check for logic depth. Logic depth is reported in .twr files as logic levels (components).

- Determine if design changes are required. A typical example design change is pipelining, or registering, the datapath. This technique may be the only way to achieve high internal frequencies if the designs logic levels are too deep.

- It is recommended to perform place and route early in the design phase with a preliminary preference file to gather information about the design.

- Tune up your preference file to include all I/O and internal timing paths as appropriate. The Translating Board Requirements into FPGA Preferences section of this document goes over an appropriate preference file example.

- Establish the pin-out in the preference file. Locating I/O can also be done in the HDL, as well as in synthesis constraint files.

- Push PAR when necessary by running multiple routing iterations and multiple placement iterations.

- Revise the preference file as appropriate, especially utilizing multicycle opportunities when possible.

- Floorplan the design if necessary (see technical note number TN1010, *Lattice Semiconductor Design Floorplanning*).

- For Lattice Semiconductor ORCA Series devices, use clock boosting as a last resort, remembering to run trace hold timing checks on the clock boosted design. Refer to the Clock Boosting section of this document for more information on clock boosting.

Typical Design Preferences

The full preference language includes many different design constraints from very global preferences to very specific preferences. To a new user this is a very large list to digest and utilize effectively. Listed here are the recommended preferences that should be applied to all designs. Refer to the *Constraints & Preferences* section of the ispLEVER on-line help system for more information on preferences.

- **Block Asynchronous Paths:** Prevents the timing tools from analyzing any paths from input pads to registers or from input pads to output pads.

- **Block RAM Reads during Write:** If using PFU based RAM, this will prevent timing analysis on a RAM read during a write on the same address in a single clock period.

- **Frequency/Period <net>:** Each clock net in the design should contain a frequency or period preference.

- **Input Setup:** Each synchronous input should have an input_setup preference.

- **Clock-to-Out:** Each synchronous output should have a clock_to_out preference.

- **Block <net>:** All asynchronous reset nets in the design should be blocked.

- **Multicycle:** The multicycle preference allows the designer to relax a frequency/period constraint on selected paths.

Proper Preferences

Providing proper preferences is key to a successful design. If the constraints of a preference file are tighter than the system requirements, the design will end up being over-constrained. As a consequence, PAR run times will be considerably longer. In addition, over-constraining non-critical paths will force PAR to waste unnecessary processing power trying to meet these constraints, hence creating possible conflicts with real critical paths that ought to be optimized first.

On the other hand, if a preference file is under-constrained compared to real system requirements, real timing issues not previously seen during dynamic timing simulations and static timing analysis could be observed on a test board, or during production.

Common causes of over-constrained timing preferences include:

- Multicycle paths not specified.
- Multiple paths to/from I/Os with different specifications.
- Attempt to fool the PAR tool with tighter than necessary specifications.

Note that over-constrained designs will also need a significantly larger amount of processing power and computing resources. As a result, it might be necessary to increase some of the allocated system resources (as in increasing your PC virtual memory paging size).

Common causes of under-constrained timing preferences include:

- I/O specifications not defined.
- Asynchronous logic without MAXDELAY preferences.
- Internally generated or unintentional clocks not specified in preference file.
- Blocking critical paths.

In general, to make sure that no critical paths were left out due to under-constraining, it is recommended to check for path coverage at the end of a Trace report file (.twr).

An example of such an output is shown in Figure 17-1.

Figure 17-1. Trace Report (.twr) Timing Summary Example

```
Timing summary:
---------------

Timing errors: 4096   Score: 25326584

Constraints cover 36575 paths, 6 nets, and 8635 connections (99.0% coverage)
```

This particular example shows a 99.0% coverage. The way to find unconstrained paths is to run Trace with the "Check Unconstrained Paths" checkbox selected. This will give a list of all of the signals that are not covered under timing analysis. In some designs, many of these signals are a common ground net that indeed does not need to be constrained. Designers should understand this point and use Trace (the ispLEVER static timing analysis tool) to check unconstrained paths to make sure they are not missing any design paths that are timing critical.

Also, note the timing score shown in Figure 17-1. The timing score shows the total amount of error (in picoseconds) for all timing preferences constraining the design. PAR attempts to minimize the timing score, PAR does not attempt to maximize frequency.

The above discussion can be summarized by the following single equality:

Quality of Preference File = Quality of PAR Results

Translating Board Requirements into FPGA Preferences

Understanding the system board level timing and design constraints is the primary requirement for producing a complete preference file. As a result, the major requirements such as clock frequency, I/O timing and loads can be translated into the appropriate preference statements in a constraint file.

The following exercise will provide an example on how to extract preferences from system conditions.

Figure 17-2 shows an example system involving the interface between a port controller and a Lattice Semiconductor FPGA.

Figure 17-2. Interface Timing Example
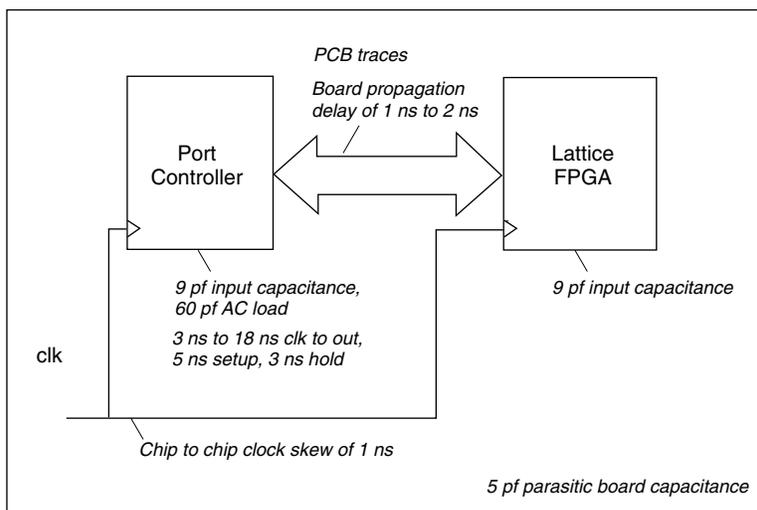
In the system above, several parameters have already been provided:

- System clock frequency: period (P): 30 ns.

- Port controller maximum output propagation delay (PDMAXp): 18ns.

- Port controller minimum output propagation delay (PDMINp): 3 ns.

- Port controller input setup specification (TSp): 5 ns.

- Port controller input hold specification (THp): 3 ns.

- Max board propagation delay (PDMAXb): 6 ns.

- Min board propagation delay (PDMINb): 1 ns.

- Port controller to FPGA device clock skew and vice versa (Tskew): 1 ns.

- Board trace AC loading (Cbac): 60 pf.

- Board trace parasitic capacitance (Cb): 5 pf.

- Port controller input capacitance (Cp) :9 pf.

- FPGA device input capacitance (Co): 9 pf.

The above information was specified under the following environmental conditions:

- Maximum ambient temperature (Ta): 70 (C.
- Estimated Power Consumption (Q): 2 W.
- 680 PBGAM Package Thermal resistance (Φj) at 0 feet per minute (fpm) airflow: 13.4 °C/W.

The goal of this exercise is to compute the following device I/O constraints:

1. Input setup specification.
2. Input hold specification.
3. Maximum output propagation delay.
4. Minimum output propagation delay.
5. Output loading.
6. Temperature.

The only parameter which can be obtained from the above is the device junction temperature:

$$
\begin{aligned}
Tj\ &= \Phi j * Q - Ta \\
&= 13.4 * 2 + 70 \\
&= 96.8\ °C
\end{aligned}
$$

The required constraints can be computed as follows:

1. Input setup specification
$$
\begin{aligned}
&= P - PDMAXp - PDMAXb\ - Tskew \\
&= 30 - 18 - 2\ - 1 \\
&= 9\ ns
\end{aligned}
$$

2. Input hold specification
$$
\begin{aligned}
&= PDMINp + PDMINb\ - Tskew \\
&= 3 + 1\ - 1 \\
&= 3\ ns
\end{aligned}
$$

3. Output maximum propagation delay requirement
$$
\begin{aligned}
&= P -\ TSp -\ PDMAXb\ - Tskew \\
&= 30 - 5 - 6 - 1 \\
&= 18\ ns
\end{aligned}
$$

4. Output minimum propagation delay requirement
$$
\begin{aligned}
&= Thp -\ PDMINb\ + Tskew \\
&= 3 - 1 + 1 \\
&= 3\ ns
\end{aligned}
$$

5. Output loading
$$
\begin{aligned}
&= Cbac + Cb + Cp \\
&= 60 + 5 + 9 \\
&= 74\ pf
\end{aligned}
$$

The preference file to use for this example is shown in Figure 17-3. For more preference language syntax and examples, refer to the Constraints & Preferences section of the ispLEVER on-line help system.

Figure 17-3. Interface Timing Preference File Example

```
PERIOD PORT "clk" 30 NS ;
INPUT_SETUP "port_controller*" 9 NS HOLD 3 NS CLKNET "clk";
CLOCK_TO_OUT "port_controller*" 18 NS MIN 3 NS CLKNET "clk";
OUTPUT PORT "port_controller*" LOAD 74 PF ;
TEMPERATURE 96.8 C ;
```

Analyzing Timing Reports

This section describes two examples of actual Trace reports (.twr report file from Trace). The purpose is to analyze both examples and understand each section of the reports given the design paths constrained.

Example 1. Multicycle Between Two Different Clocks

In this first example, CLKA and CLKB were assigned 104 MHz and 66 MHz frequencies respectively.

In addition, a multicycle constraint was specified as per the preference file:

```
FREQUENCY  NET "CLKA" 104 MHZ ;
FREQUENCY NET "CLKB" 66 MHZ ;
MULTICYCLE "M2" START CLKNET "CLKA" END CLKNET "CLKB" 2.000000 X ;
```

See Figure 17-4 for the block diagram and waveform for this example. The resulting Trace report is shown in Figure 17-5.

Figure 17-4. Multicycle Clock Domains Block Diagram and Waveform
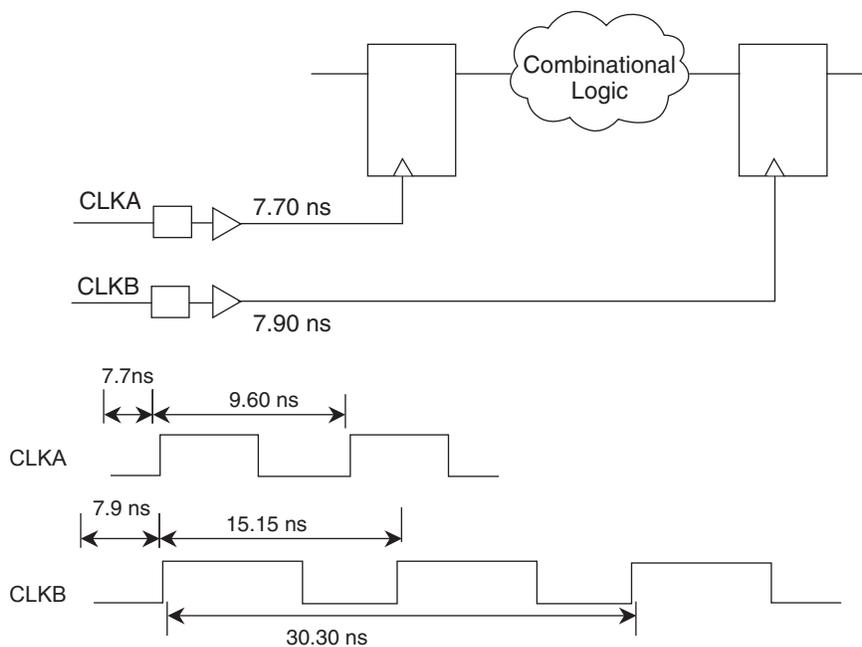
Figure 17-5. Trace Report for Multicycle Clock Domains Example

```
================================================================================
Preference: MULTICYCLE "M2" START CLKNET "CLKA"  END CLKNET "CLKB"  2.000000 X ;
            40 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------
WARNING - trce: Clock skew between net 'CLKA' and net 'CLKB' not
          computed: nets may not be related
--------------------------------------------------------------------------------


Passed:  The following path meets requirements by 27.945ns

 Logical Details: Cell type  Pin type        Cell name  (clock net +/-)

   Source:         FF         Q               v_fifo_bank_1_stfifo0_wr_count_2  (from CLKA +)
   Destination:    FF         Data in         v_fifo_bank_1_stfifo0_wr_count_r_2  (to CLKB +)

   Delay:              2.456ns  (37.3% logic, 62.7% route), 1 logic levels.

 Constraint Details:

     2.456ns physical path delay PFU_155 to PFU_156 meets
    30.302ns delay constraint less
    -0.099ns DIN_SET requirement (totaling 30.401ns) by 27.945ns

 Physical Path Details:

   Name    Fanout  Delay (ns)        Site              Resource
REG_DEL     ---     0.917    R22C16.CLK0 to     R22C16.Q2 PFU_155 (from CLKA)
ROUTE        1      1.539     R22C16.Q2 to    R23C17.DIN2 v_fifo_bank_1_stfifo0_wr_countZ0Z_2 (to CLKB)
                   --------
                    2.456   (37.3% logic, 62.7% route), 1 logic levels.

 Clock Skew Details:

 Source Clock Path:

   Name    Fanout  Delay (ns)        Site              Resource
IN_DEL      ---     1.192       AM17.PAD to       AM17.INDD ip_CLKA
ROUTE        1      2.989      AM17.INDD to    LLPPLL.CLKIN ip_CLKA_c
MCLK_DEL    ---     0.424    LLPPLL.CLKIN to     LLPPLL.MCLK v_io_ppl3_tx4_1_mtppll_rsp_rsppll_0_0
ROUTE       177     3.094     LLPPLL.MCLK to    R22C16.CLK0 CLKA
                   --------
                    7.699   (21.0% logic, 79.0% route), 2 logic levels.

 Destination Clock Path:

   Name    Fanout  Delay (ns)        Site              Resource
IN_DEL      ---     1.192        C17.PAD to        C17.INDD ip_CLKB
ROUTE        1      3.091       C17.INDD to    ULPPLL.CLKIN ip_CLKB_c
MCLK_DEL    ---     0.424    ULPPLL.CLKIN to     ULPPLL.MCLK v_io_ppl3_tx4_1_mtppll_mac_macpll_0_0
ROUTE       263     3.182     ULPPLL.MCLK to    R23C17.CLK0 CLKB
                   --------
                    7.889   (20.5% logic, 79.5% route), 2 logic levels.
```

In Figure 17-5, notice how the path is described in terms of "Logical Details."

This section shows both the source and destination registers using their unmapped names from the EDIF (Electronic Data Interchange Format) file. This is a feature that allows the user to recognize the type of logic being analyzed.

Based on the declared frequencies for both clocks, we already know the following:

- CLKA period = 9.6 ns.
- CLKB period = 15.15 ns.

- No relative phase information exists between both clocks. As a result, Trace does not factor in the skews on either clock.

As a consequence, we know that, ignoring everything else (clock skews, registers library setups, etc.), a single cycle positive edge to positive edge setup available from CLKA to CLKB is: 15.15ns (refer to waveforms in Figure 17-4). Hence, with 2X multicycle, the resulting setup would be twice that number, or:

Ts = 30.3 ns
(shows up as delay constraint under Constraint Details section of Trace report)

Having computed this, the available setup margin is known to be as follows:

M = (Ts - Td) - Ds

Where:

- Td = path delay from clock pin of source register to D pin of destination=2.456 ns. Shown in the Physical Path Details section of Trace report.

- Ds = destination cell library setup requirement= -0.099 ns. This matches DIN_SET under Constraint Details section of the .twr Trace report.

There is no phase relationship between CLKA and CLKB as indicated by the warnings in Figure 17-5. Hence, the following skews were correctly ignored:

- TSB = delay or skew on destination clock CLKB = 7.889 ns. Shown in the Clock Skews detail section of Trace report.

- TSA = delay or skew on source clock CLKA = 7.699 ns. Shown in the Clock Skews detail section of Trace report.

Hence:

- M = (30.3 - 2.46) - (-0.099) = 27.9 ns. This matches the number in the "PASSED" section at the top of the Trace report.

Example 2. CLOCK_TO_OUT with PLL Feedback

In this example, ip_macclk_c is assigned to 66 MHZ and the clock to out propagation delays are constrained in the preference file:

```
FREQUENCY NET "ip_macclk_c" 66 MHZ;
CLOCK_TO_OUT ALLPORTS 7.000000 ns CLKPORT "ip_macclk" ;
```

See Figure 17-6 for the block diagram for this example. The resulting Trace report is shown in Figure 17-7.

Figure 17-6. CLOCK_TO_OUT with PLL
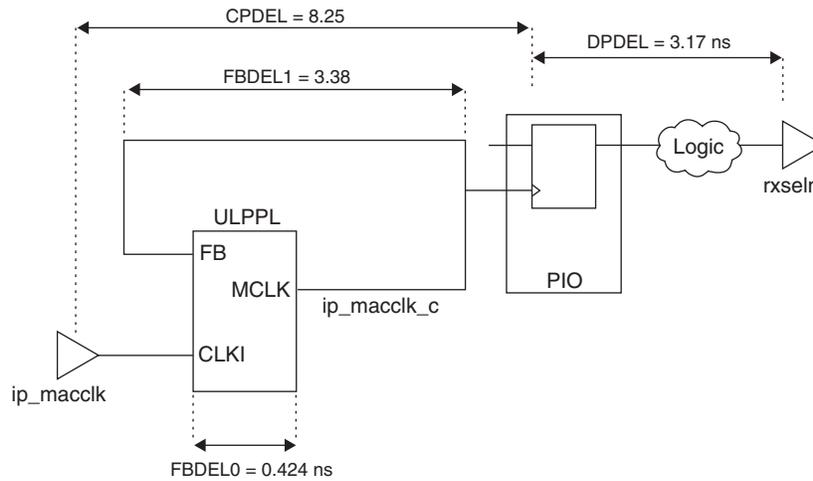
Figure 17-7. Trace Report for CLOCK_TO_OUT with PLL

```
================================================================================
Preference: CLOCK_TO_OUT ALLPORTS 7.000000 ns CLKPORT "ip_macclk" ;
            2 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------


Passed:  The following path meets requirements by 0.681ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)
   Source:         IO-FF Out  Q              ppl3_rx5_1_rxselnio  (from macclk +)
   Destination:    Port       Pad            rxseln
   Data Path Delay:    3.164ns  (100.0% logic, 0.0% route), 1 logic levels.
   Clock Path Delay:   8.249ns  (19.6% logic, 80.4% route), 2 logic levels.

 Constraint Details:
     8.249ns delay ip_macclk to rxseln less
     5.094ns feedback compensation
     3.164ns delay rxseln to rxseln (totaling 6.319ns) meets
     7.000ns offset ip_macclk to rxseln by 0.681ns

 Physical Path Details:

     Clock path ip_macclk to rxseln:
   Name      Fanout   Delay (ns)         Site              Resource
IN_DEL      ---      1.192        C17.PAD to      C17.INDD ip_macclk
ROUTE        1       3.235        C17.INDD to   ULPPLL.CLKIN ip_macclk_c
MCLK_DEL    ---      0.424    ULPPLL.CLKIN to    ULPPLL.MCLK v_io_ppl3_tx4_1/mtppll_mac/macpll_0_0
ROUTE       141      3.398    ULPPLL.MCLK to         F32.SC macclk
                     --------
                     8.249  (19.6% logic, 80.4% route), 2 logic levels.

     Data path rxseln to rxseln:
   Name      Fanout   Delay (ns)         Site              Resource
OUTREGF_DE  ---      3.164         F32.SC to      F32.PAD rxseln (from macclk)
                     --------
(100.0% logic, 0.0% route), 1 logic levels.

     Feedback path:

   Name      Fanout   Delay (ns)         Site              Resource
MCLK_DEL    ---      0.424    ULPPLL.CLKIN to    ULPPLL.MCLK v_io_ppl3_tx4_1/mtppll_mac/macpll_0_0
ROUTE       141      3.380    ULPPLL.MCLK to    ULPPLL.FB macclk
                     --------
                     3.804  (11.1% logic, 88.9% route), 1 logic levels.

Report:   6.319ns is the minimum offset for this preference.
```

The different path measurements were obtained from the Trace report shown in Figure 17-7 as follows:

- DPDEL = Data Path Delay = 3.16 ns. Shown under Physical Path Details-> Data path in the timing report.
- FBDEL0 = Feedback cell delay across PLL = 0.42 ns, which is the first entry value under Feedback Path.
- FBDEL1 = Feedback routing delay from PLL output to PLL FB pin = 3.38 ns, which is the second entry value under Feedback Path.

The full feedback delay includes both FBDEL0 and FBDEL1 (0.42 + 3.38 = 3.80) under Feedback Path, in addition to any internal PLL delay added after the FB pin. Such a delay is a programmable attribute defined as FB_PDEL. This programmable value can be set to any of one of 4 values (DEL0, DEL1, DEL2 or DEL3; DEL0 being 0 delay) in either the HDL file input to synthesis, or in the graphical Editor for Programmable Integrated Circuits (EPIC) software tool included with the ispLEVER software.

Therefore, the total feedback delay would be:

$$FBDEL = FBDEL0 + FBDEL1 + FB\_PDEL = 3.80 + FB\_PDEL$$

Under "Constraint Details" of the report file, the feedback compensation (FBDEL) is shown to be 5.09 ns. Since this value is different from 3.804, we conclude that a non-zero value of FB_PDEL was applied (5.10 - 3.80 = 1.29 ns). This value corresponds to FB_PDEL = DEL2 in an OR4E4-2 device.

Now, let's verify the available margin on this CLOCK_TO_OUT preference:

$$
\begin{aligned}
M &= CKOUT - (CPDEL + DPDEL - FBDEL) \\
&= 7.000 - (8.249 + 3.164 - 5.094) = 0.681 \text{ ns}
\end{aligned}
$$

This value matches the one at the top of the report file ("Passed" section). It also matches the final value under "Constraints Details".

ispLEVER Controlled Place and Route

Extensive benchmark experiments have been performed in order to determine the most optimum per device default settings for all PAR options. At times, improved timing results can be obtained on a design by design basis by trying different variations of the PAR options. This section describes the techniques that can be used within the ispLEVER graphical user interface (GUI) to improve timing results from Trace on placed and routed designs.

Running Multiple Routing Passes

Improved timing results can be obtained by increasing the number of routing passes during the Routing phase of PAR.

The PAR options window in Figure 17-8 can be launched by the following steps:

1. In the Project Navigator Source window, select the **target FPGA device**.
2. In the Processes window, right-click the **Place & Route Design** process and select **Properties** to open the dialog box.

In the example screen shot shown in Figure 17-8, the router will route the design for five routing iterations, or until all the timing preferences are met, whichever comes first. For example, PAR will stop after the second routing iteration if it hits a timing score of zero on the second routing iteration.

The highest selection in Placement Effort level will result in longer PAR run times but may give better design timing results. A lower Placement Effort will result in shorter PAR run times but will likely give less than optimal design timing results.

Figure 17-8. PAR Options Window
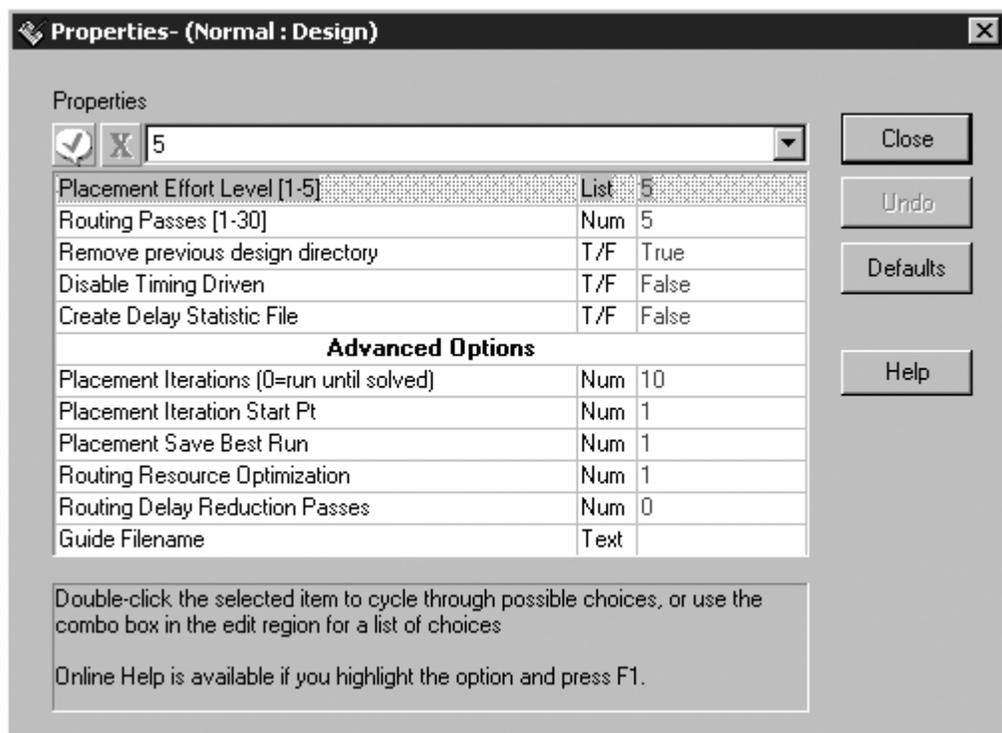
Figure 17-9. Example PAR report (.par) File, Routing Section

```
0 connections routed; 26590 unrouted.
Starting router resource preassignment
Completed router resource preassignment. Real time: 11 mins 31 secs
Starting iterative routing.
End of iteration 1
26590 successful; 0 unrouted; (151840) real time: 14 mins 29 secs
Dumping design to file
d:\ip\design.ncd.
End of iteration 2
26590 successful; 0 unrouted; (577) real time: 16 mins 23 secs
Dumping design to file
d:\ip\design.ncd.
End of iteration 3
26590 successful; 0 unrouted; (0) real time: 17 mins 39 secs
Dumping design to file
```

The place and route (.par) report file contains execution information about the PAR command run. The report also shows the steps taken as the program converges on a placement and routing solution. In the routing convergence example text in Figure 17-9, the number in parenthesis is the timing score after each iteration. In this example, timing was met after three routing iterations, as can be seen from the (0) timing score.

Using Multiple Placement Iterations (Cost Tables)

Using multiple placement iterations can be achieved by selecting the Advanced Options in Figure 17-8.

As shown in the Advanced Options of Figure 17-8, the number of iterations is set to 10 and the placement start point is set to iteration 1 (cost table 1). Only the best NCD file is to be saved as per the following line. Once PAR runs, the tool will loop back through the place and route flow until the number of iterations has reached 10. In this

example, the NCD file with the best timing score would be saved. The tool keeps track of the timing and routing performance for every iteration in a file called the multiple par report (.par). Such a file is shown in Figure 17-10.

Figure 17-10. Multiple PAR Report (.par)

```
Level/          Number      Timing      Run       NCD
Cost [ncd]      Unrouted    Score       Time      Status
----------      --------    -------     -----     --------
5_4   *         0           0           01:58     Complete
5_6             0           25          02:01     Complete
5_2             0           102         01:45     Complete
5_7             0           158         02:15     Complete
5_3             0           186         01:54     Complete
5_10            0           318         02:39     Complete
5_1             0           470         01:51     Complete
5_8             0           562         02:25     Complete
5_5             0           732         02:00     Complete
5_9             0           844         02:27     Complete

* : Design saved.
```

Figure 17-10 indicates that:

- The "5_" under the Level/Cost column means that the Placement Effort level was set to 5. The Placement Effort level can range from 1 (lowest) to 5 (highest).
- 10 different iterations ran (10 cost tables).
- Timing scores are expressed in total picoseconds (ps) by which the design is missing constraints on all preferences.
- Iteration number 4 (cost table 4) achieved a 0 timing score and hence was the design saved. More than one .ncd file can be saved. This is user-controlled via the "Placement Save Best Runs" value box shown in Figure 17-8.
- Each iteration routed completely.

Note that, in Figure 17-8, if "Placement Iterations (0=run until solved)" is set to 0, the tool will run indefinitely through multiple iterations until a 0 timing score is reached. In a design that is known to have large timing violations, a 0 timing score will never be reached. As a consequence, the user must intervene and stop the flow at a given point in time.

In general, multiple placement iterations can help placement but can also use many CPU cycles. Multiple placement iterations should be used carefully due to system limitations and the uncertainty of results. It is better to fix the root cause of timing problems in the design stage.

Clock Boosting

Clock boosting, supported in Lattice Semiconductor's ORCA Series device family, is the deliberate introduction of clock skew on a target flop to increase the setup margin. Every programmable flip-flop in the device has programmable delay elements before clock inputs for this purpose. The automated clock boosting tool will attempt to meet setup constraints by introducing delays to as many target registers as needed to meet timing, in effect, borrow register hold margins to meet register set-up timing. The following bullets summarize how clock boosting is accomplished in Lattice Semiconductor ORCA Series device family.

- A 4-tap delay cell structure in front of the clock port of every flip-flop in the device (includes I/O flip-flops)
- Ability to borrow clock cycle time from one easily-met path and give this time to a difficult-to-meet path

Clock boosting is typically most useful in designs that are only missing timing on a few paths for one or two preferences. If the design is missing timing by over a few nanoseconds on any given path, clock boosting will not be able to schedule skew in a way that will eliminate enough timing to make the critical preference. Clock boosting run times can be shortened by using a preference file with only the failing preferences in it.

Figure 17-11. Clock Boosting Example

Target Performance: 10 ns period (100 MHz)
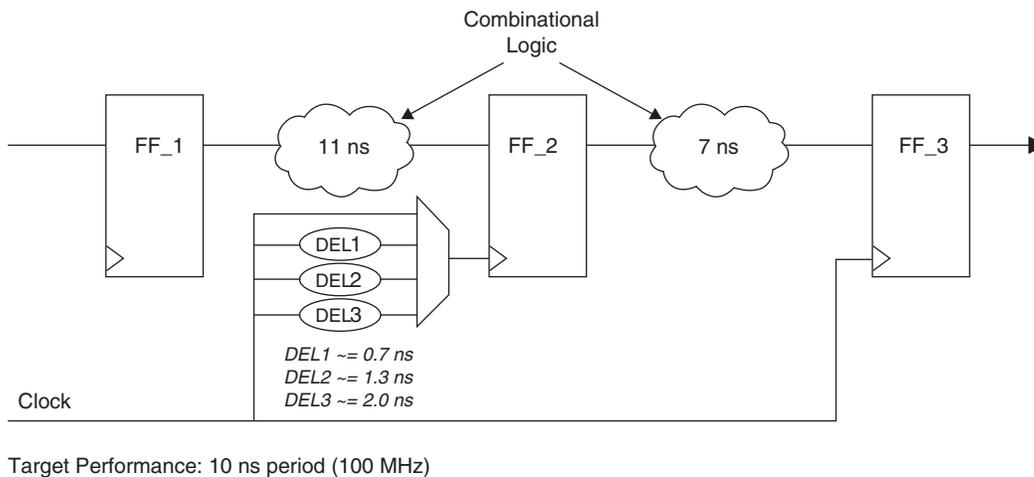
The example illustrated in Figure 17-11 shows two register-to-register transfers that both need to meet the 10 ns period constraint. By using delay cell DEL2 to delay the clock input on flip-flop FF_2, the first register transfer will make its period constraint with a new minimum period of ~9.7 ns and the second register transfer will make its period constraint by ~8.3 ns.

The D1, D2, and D3 delays shown in Figure 17-11 are variable depending on the speed grade and Lattice Semiconductor FPGA device family. For complete timing information, reference the software generated timing data sheet, included with ispLEVER, for the desired Lattice Semiconductor FPGA device family.

To Perform Clock Boosting in the Project Navigator
1. In the Project Navigator Sources window, select the **target device**.
2. In the Processes window, right-click the **Clock Boosting** under **Place & Route Design** process, and then select **Properties** to open the Properties dialog box.
3. Select the **Clock Boosting Output Filename** property from the property list and type the name of the output file name in the edit region (<file_name>.ncd).
4. Click **Close** to close the dialog box.

As shown in Figure 17-12, the original .ncd and .prf files as well as the output .ncd file are typed into the corresponding entries. Checking "Maximize Frequency" will push the tool to improve the frequency beyond the input preference requirement. This is generally only useful for bench marking.

Figure 17-12. Clock Boosting Window
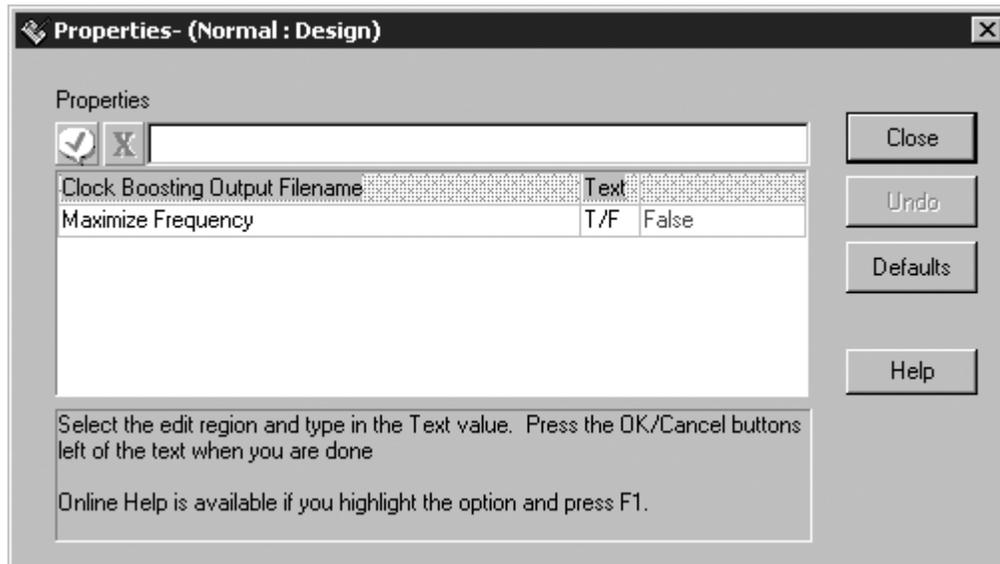
Other important considerations on the practicality of using clock boosting:

- Some circuits show big improvement, others have no gain. Clock boosting results are very design-dependent.

- Clock boosting uses minimum delay values which have not yet been validated at the system level.

- Automatic clock boosting identifies skew and hold time issues. However, after clock boosting is performed, designers are strongly recommended to run Trace twice, once with regular, maximum delay analysis, and again with minimum delays. The designer should then read over both resultant .twr timing reports to make sure there are no timing errors. The minimum delay analysis is done by checking the "Check Hold Times" checkbox in the Trace Options GUI window.

Guided Map and PAR

To decrease PAR runtimes after minor changes to a logical design, guided mapping uses a previously generated .ncd file to "guide" the mapping of the new logical design. Guided mapping can be performed from the Guide Filename property in the Project Navigator Map Design process, or specified using the command line **-g** option with the file name of the guide file. In general, guided MAP should only be used in conjunction with guided PAR.

To Perform Guided Mapping in the Project Navigator
1. In the Project Navigator Sources window, select the **target device**.
2. In the Processes window, right-click the **Map Design** process, and then select **Properties** to open the Properties dialog box.
3. Select the **Guide Filename** property from the property list and type the name of the guide file name in the edit region (<file_name>.ncd).
4. Click **Close** to close the dialog box.

The Map operation will use the guide file to generate the new design file.

To Perform Guided PAR in the Project Navigator
1. In the Project Navigator Sources window, select the **target device**.
2. In the Processes window, right-click the **Place & Route Design** process and select **Properties** to open the dialog box.
3. Under Advanced Options, select the **Guide Filename** property and type the name of the file in the text field.

4. Click **Close** to close the dialog box.

5. Double-click the **Place & Route Design** process. The ispLEVER software runs the process using the specified guide file.

Notes on Guided Mapping

All guidance criteria is based on signal name matching. Topology of combinatorial logic is considered when Soft-wire LUTs (SWLs) exist in the guided file.

Register elements are mapped in two passes. In the first pass, register control signals are matched by name exactly. In the second pass, the control signals names are not matched. This methodology provides a greater chance of matching for registers since control signal names have a tendency to change from successive synthesis runs. Other matching considerations are as follows:

- For combinatorial logic, new SWLs are matched from SWLs extracted from the guide design.

- All unmatched logic are mapped through the regular mapping process.

- The performance of the guided mapped design can be no better than the original.

- A guide report, <design_name>.gpr, gives details of the success guided map had in matching with the guide file.

Notes on Guided PAR

To decrease PAR runtimes after minor changes to the physical design file (.ncd), guided PAR uses a previously placed and/or routed .ncd file to "guide" the placement and routing of the new .ncd file. Guided PAR can be performed from the Project Navigator or specified using the command line **-g** option with the file name of the guide file.

For PAR to use a guide file for design, PAR first tries to find a guiding object (i.e., nets, components, and/or macros) in the guide file that corresponds to an object in the new .ncd file. A guiding object is an object in the guide file of the same name, type, and connectivity as an object in the new .ncd file. A guided object is an object in the new .ncd file that has a corresponding guiding object in the guide file.

After PAR compares the objects in each file, it places and routes each object of the new .ncd file based on the placement/routing of its guiding object. If PAR fails to find a guiding object for a component, for example, PAR will try to find one based on the connectivity. PAR appends the names of all objects which do not have a guiding object in the guide file to .gpr (Guided PAR Report) file. The matching factor specifies the percentage of the same connectivity that guiding and guided objects must have. It can only be specified using the **-mf** option in the command line. The matching factor option applies to nets and components only. When matching factor is 100 (the default), a guiding object must have exactly the same connectivity as the object it is guiding. When a matching factor is specified, the value specified is taken as the minimum percentage of the same connectivity that a guided object and its guiding object have. Note that the matching factor is always 100 when the guided PAR is performed from the Project Navigator.

After all guided objects are placed and routed, PAR locks down the locations of all guided components and macros and then proceeds with its normal operation. Guided PAR supports the following preferences: USE SPINE, USE PRIMARY, USE SECONDARY, USE LONGLINE, USE HALFLINE, LOCATE COMP, LOCATE MACRO, and hard-placed PGROUPs.

Conclusion

In general, different designs respond better to different strategies. The processes outlined in this application note may not be optimal for all cases. For a design's first place and route, run PAR at the low placer effort level and with a low number of routing iterations. There is no point in running 100 cost tables if the design's logic depth is too high. The techniques discussed within this document, like interpreting static timing reports and using proper preferences, will guide the user to better PAR results.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

+1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

# Board Timing Guidelines for the DDR SDRAM Controller IP Core

Introduction

This document describes how to meet board timing requirements for DDR signals. The Lattice DDR SDRAM Controller IP core, non-pipelined version (DDR-NP) is used as an example.

Figure 18-1 describes the timing diagram for the DDR signals. A total of five clocks are used in the DDR board design using the Lattice DDR IP core. The following is the clock description:

`clk:`   Input clock for PLL (max. frequency of 133MHz for DDR NP)
`ddr_clk:`   Output clock going to DDR (max. frequency of 133MHz for DDR NP)
`ddr_clk_n:`   Negated version of `ddr_clk`
`pll_mclk (clkx):`   Same as `ddr_clk`, used inside the FPGA only.
`pll_nclk (clk2x):`   A 266MHz clock for DDR NP, used inside the FPGA only.

Figure 18-1. DDR Signal Timing Diagram
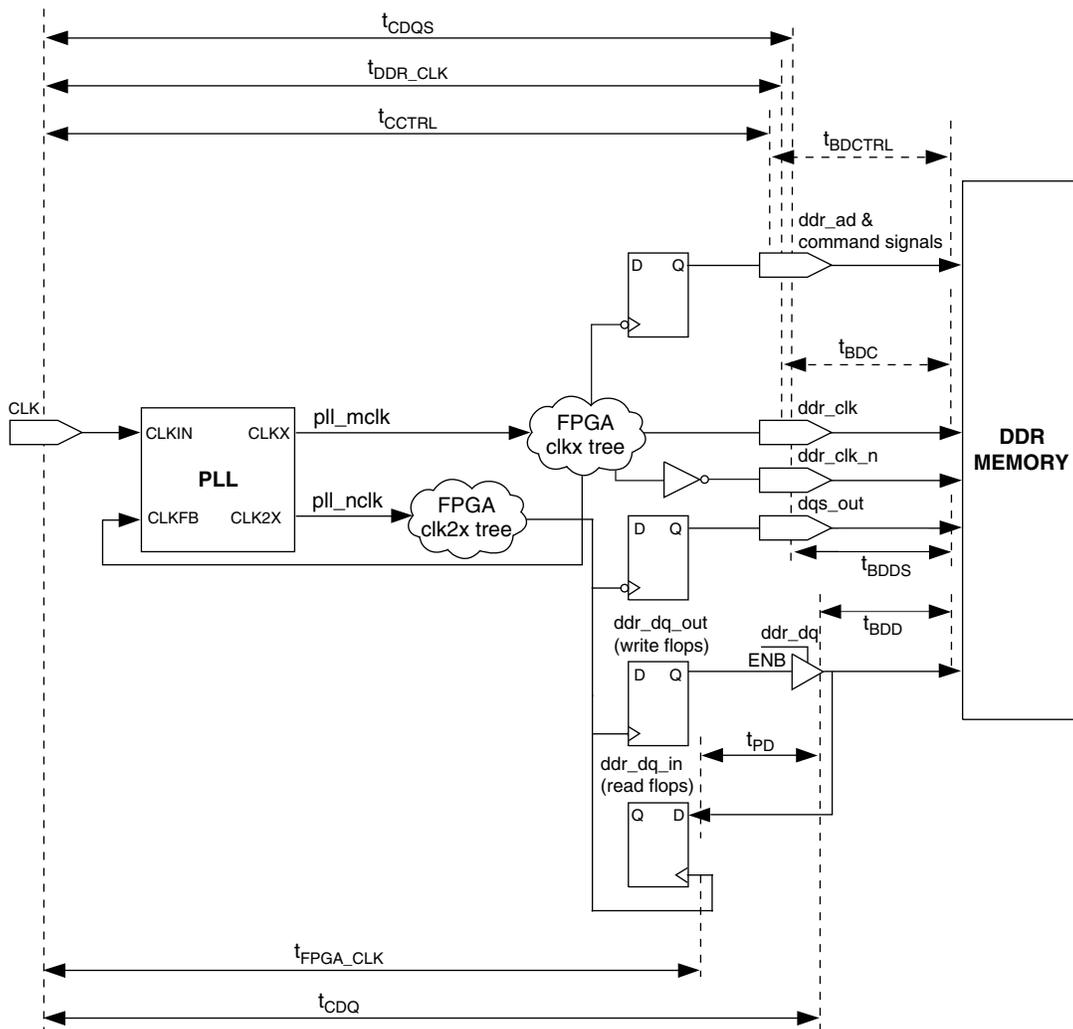
As shown in Figure 18-1, input to PLL is CLK (133MHz for DDR NP). The PLL generates `pll_mclk` (133MHz) and `pll_nclk` (266MHz). The clocks `ddr_clk` and `ddr_clk_n` go to DDR memory and are delayed by I/O pad delay with respect to `pll_mclk`. The clocks `pll_mclk` and `pll_nclk` are internal to the FPGA. Command and address signals are clocked by a negative edge of `pll_mclk`. The signal `dqs_out` acts as a clock for DDR write and is generated by negative edge of `pll_nclk`. The signal `ddr_dq_out` is the DDR write data bus and generated by positive edge of `pll_nclk`. The flops `ddr_dq_*` latch the read data and are clocked by positive edge of `pll_nclk`.

Read Operation

Figure 18-2 shows the timing of the DDR read operation. Table 18-1 describes the timing arcs of the read operation.

Figure 18-2. Read Timing Diagram
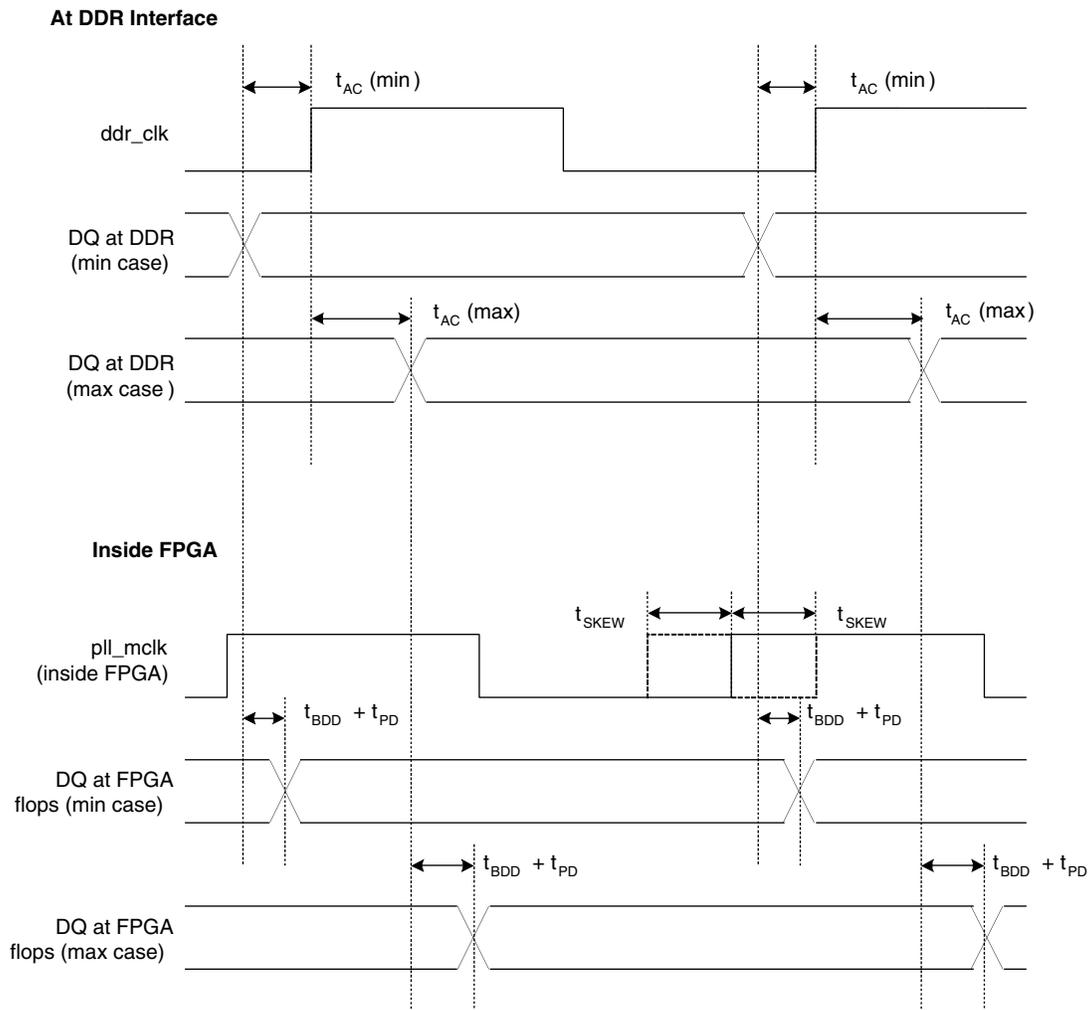
Table 18-1. Read Operation Timing Arcs

Symbol	Description	Example: DDR-NP on ORCA 4
t_{CK}	Clock period of `ddr_clk`	7.5ns
t_{DDR_CLK} (max)	Delay from the CLK input of the FPGA to the `ddr_clk` pad including Feedback compensation (Clock Path Delay - Feedback Path).	2.47[1]
t_{DDR_CLK} (min)	Delay from the CLK input of the FPGA to the `ddr_clk` pad including Feedback compensation (Clock Path Delay - Feedback Path).	1.138[1]
t_{BDC}	Board delay of `ddr_clk` from FPGA to DDR SDRAM.	—
$t_{AC(MAX)}$	Time from the rising edge of `ddr_clk` after which the data is available at DDR output pins (max.).	0.75ns
$t_{AC(MIN)}$	Time from the rising edge of `ddr_clk` after which the data is available at DDR output pins (min.).	-0.75ns
t_{BDD}	Board delay from DDR SDRAM data pad to the FPGA `ddr_dq` pad.	—
t_{PD}	Propagation delay from FPGA input pad to the `ddr_dq_in` flip-flop input pin (Data Path Delay).	0.0ns[1]
t_{FDS}	Set-up time required by the `ddr_dq_in` flip-flop (`INREG_SET`).	3.195ns[1]
t_{FDH}	Hold time required by the `ddr_dq_in` flip-flop (`INREG_HLD`).	-1.609ns[1]
t_{SKEW}	Skew of the PLL.	0.3ns
t_{FPGA_CLK} (max)	Delay from the CLK input of the FPGA to the `ddr_dq_in` flip-flop clock input including feedback compensation (Clock Out Path Delay - Feedback Path).	2.935ns[1]
t_{FPGA_CLK} (min)	Delay from the CLK input of the FPGA to the `ddr_dq_in` flip-flop clock input including feedback compensation (Clock Out Path Delay - Feedback Path).	1.239ns[1]

1. t_{FPGA_CLK}, t_{DDR_CLK}, t_{PD} and t_{FDS} can be easily obtained from the PNR time reports.

Set-up Time Calculation for the Data Input (Max. Case)

The DDR Controller IP core uses the positive edge of `pll_nclk` to latch in the data.

Table 18-1 timing arcs are used to calculate the following:

Max. delay of clock to `ddr_dq_in` flops = t_{FPGA_CLK} (max) + (t_{CK} * 1/2) - t_{SKEW} - t_{FDS}

Max. delay of DDR read data to `ddr_dq_in` flops = t_{DDR_CLK} (max) + t_{BDC} + t_{AC} (max) + t_{BDD} + t_{PD}

To meet set-up time at `ddr_dq_in` flops, Clock Delay - Data Delay > 0

Therefore:

t_{FPGA_CLK} (max) + (t_{CK} * 1/2) - t_{SKEW} - t_{FDS} - t_{DDR_CLK} (max) - t_{BDC} - t_{AC} (max) - t_{BDD} - t_{PD} > 0

Isolating the board delays, we get:

(t_{BDD} + t_{BDC}) < t_{FPGA_CLK} (max) + (t_{CK} * 1/2) - t_{SKEW} - t_{FDS} - t_{DDR_CLK} (max) - t_{AC} (max) - t_{PD}

(t_{BDD} + t_{BDC}) < 3.75 - 0.3 - 3.195 - 2.47 + 2.935 - 0.75 - 0.0

(t_{BDD} + t_{BDC}) < -0.03 ns

Hold Time Calculation for the Data Input (Min. Case)

As shown in Figure 18-2, the min data is available at DDR output pins after t_{AC} (min) time from the rising edge of `ddr_clk`. Since t_{AC} (min) is generally a negative number, data appears before the rising edge. This data will incur board delay (t_{BDD}) and propagation delay from FPGA input pad to the flip-flop input pin (t_{PD}).

Min. Delay of DDR read Data = t_{DDR_CLK} (min) + t_{BDC} + t_{AC} (min) + t_{BDD} + t_{PD}

Min. Delay of Clock to `ddr_dq_in` flops = t_{FPGA_CLK} (min) + t_{SKEW} + t_{FDH}

To meet hold time at `ddr_dq_in` flops, Data Delay - Clock Delay > 0

Therefore:

t_{DDR_CLK} (min) + t_{BDC} + t_{AC} (min) + t_{BDD} + t_{PD} - t_{FPGA_CLK} (min) - t_{SKEW} - t_{FDH} > 0

Isolating the board delays, we get:

$(t_{BDD} + t_{BDC}) > t_{FPGA\_CLK}$ (min) + t_{SKEW} + t_{FDH} - t_{DDR_CLK} (min) - t_{AC} (min) - t_{PD}

$(t_{BDD} + t_{BDC}) > (1.239)$ ns + 0.3 + (-1.609ns) - (1.138) - (-0.75) - 0

$(t_{BDD} + t_{BDC}) > -0.458$ ns

Conclusion: To meet read set-up and hold timing, board delay for `ddr_dq`, `ddr_clk` and `ddr_clk_n` should be:

-0.458ns < $(t_{BDD} + t_{BDC})$ < -0.03ns

Write Operation

For a proper write operation, data (`ddr_dq`) should meet set-up (t_{DS}) and hold (t_{DH}) time requirements of DDR SDRAM with respect to `ddr_dqs` signal. The `ddr_dqs` signal is generated with respect to negative edge of `pll_nclk` and data `ddr_dq` out is generated with respect to positive edge of `pll_nclk` as shown in Figure 18-3. As a result, 1/2 `clk2x` (3.75ns/2) is provided as set-up and hold for `ddr_dq_out` with respect to `dqs_out`.

For maximum set-up and hold margin, the `ddr_dqs` and `ddr_dq` traces on the board should be matched.

Table 18-2. Write Operation Timing Arcs

Symbol	Description	ORCA 4
t_{DS}	Set-up time required by the DQ with respect to DQS for DDR SDRAM.	0.75ns
t_{DH}	Hold time required by the DQ with respect to DQS for DDR SDRAM.	0.75 ns
t_{CDQ}	Clock-to-out timing for `ddr_dq` with respect to `pll_nclk`.	—
t_{CDQS}	Clock-to-out timing for `ddr_dqs` with respect to `pll_nclk`.	—
t_{BDDS}	Board delay of `ddr_dqs` from FPGA to DDR SDRAM pins.	—

Figure 18-3. Write Timing Diagram
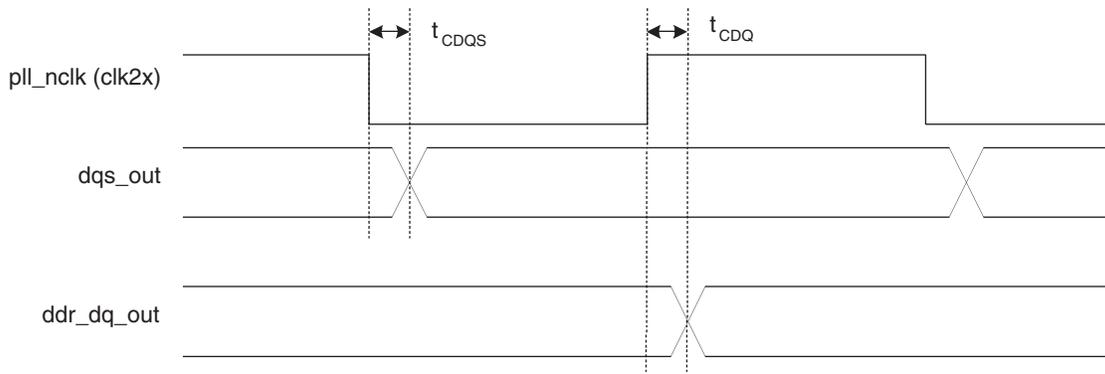
Write Set-up

Clock Delay = t_{CDQS} + 1/2 `clk2x` - t_{DS} + t_{BDDS}

Data Delay = t_{CDQ} + t_{BDD}

Clock Delay - Data Delay > 0

Therefore:

t_{CDQS} + 1/2 `clk2x` - t_{DS} + t_{BDDS} - t_{CDQ} - t_{BDD} > 0

Assumptions for write set-up and hold equations:

1. t_{BDDS} and t_{BDD} are equal (board delays are same both for `dqs_out` and `ddr_dq_out`).

2. t_{CDQ} and t_{CDQS} are equal (both are output delays from I/O flop).

Therefore:

1/2 `clk2x` - t_{DS} > 0

3.75/2 - 0.75 > 0

1.125 > 0

Write Hold

Data Delay = t_{CDQ} + t_{BDD}

Clock Delay = t_{CDQS} + 1/2 `clk2x` + t_{DH} + t_{BDDS}

Data Delay - Clock Delay > 0

Therefore:

t_{CDQS} + 1/2 `clk2x` - t_{DH} + t_{BDDS} - t_{CDQ} - t_{BDD} > 0

Assumptions for write set-up and hold equations:

1. t_{BDDS} and t_{BDD} are equal (board delays are same both for `dqs_out` and `ddr_dq_out`).

2. t_{CDQ} and t_{CDQS} are equal (both are output delays from I/O flop).

Therefore:

1/2 `clk2x` - t_{DH} > 0

3.75/2 - 0.75 > 0

1.125 > 0

Address and Command Signals

Address (`ddr_ad`) and command signals (`ddr_cas`, `ddr_ras`, `ddr_we`) should meet set-up (t_{DS}) and hold (t_{DH}) timings at DDR interface with respect to positive edge of `ddr_clk`. Address and command signals are clocked using negative edge of `pll_mclk` inside the FPGA as shown below. The `ddr_clk` signal is a delayed by pad delay and board delay at DDR interface compared to `pll_mclk` inside the FPGA. As a result, 1/2`clkx` of set-up and hold is provided by design.

Table 18-3. Timing Arcs for Address and Command Signals

Symbol	Description	ORCA4
t_{CCTRL} (max)	Is the clock-to-out time for `ddr_ad` and command signals. (Clock Path Delay - Feedback Path) + Data Path Delay	4.834 ns
t_{CCTRL} (min)	Is the clock-to-out time for `ddr_ad` and command signals. (Clock Path Delay - Feedback Path) + Data Path Delay	2.147 ns
t_{BDCTRL}	Is the board delay of `ddr_ad` and command signals from FPGA pins to DDR SDRAM pins.	—

Figure 18-4. Timing Diagram for Address and Command Signals
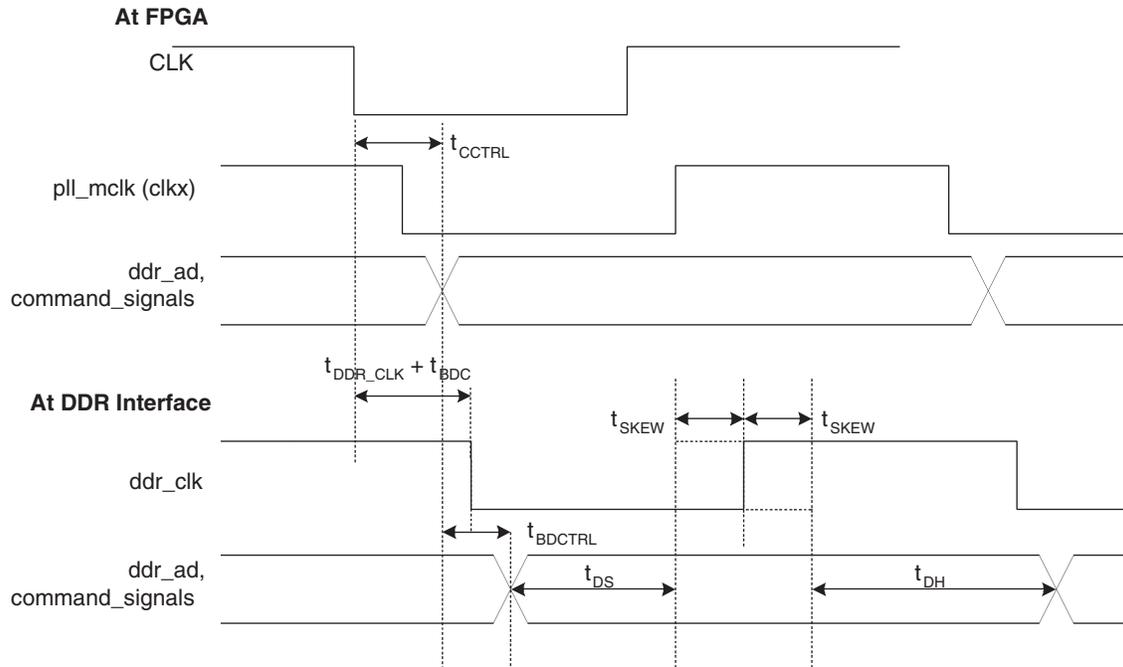
Set-up Calculation

Max Delay of Clock to DDR = t_{DDR_CLK} (max) + t_{BDC} + t_{CK} * 1/2 - t_{SKEW} - t_{DS}

Max Delays of command signals Data to DDR = t_{CCTRL} (max) + t_{BDCTRL}

To meet set up time at DDR memory, Clock Delay - Data Delay > 0

Therefore:

t_{DDR_CLK} (max) + t_{BDC} + t_{CK} * 1/2 - t_{SKEW} - t_{DS} - t_{CCTRL} (max) - t_{BDCTRL} > 0

Isolating the board delays, we get:

t_{BDCTRL} - t_{BDC} < t_{DDR_CLK} (max) + t_{CK} * 1/2 - t_{SKEW} - t_{DS} - t_{CCTRL} (max)

t_{BDCTRL} - t_{BDC} < 2.47 + 3.75 - 0.3 - 0.75 - 4.834

t_{BDCTRL} - t_{BDC} < 0.336 ns

Hold Calculation

Min Delay of command signals Data to DDR = t_{CCTRL} (min) + t_{BDCTRL} + t_{CK} * 1/2

Min Delay of Clock to DDR = t_{DDR_CLK} (min) + t_{BDC} + t_{SKEW} + t_{DH}

To meet hold time at DDR memory, Data Delay - Clock Delay > 0

Therefore:

t_{CCTRL} (min) + t_{BDCTRL} + t_{CK} * 1/2 - t_{DDR_CLK} (min) - t_{BDC} - t_{SKEW} - t_{DH} > 0

Isolating the board delays, we get:

t_{BDCTRL} - t_{BDC} > - t_{CCTRL} (min) - t_{CK} * 1/2 + t_{DDR_CLK} (min) + t_{SKEW} + t_{DH}

t_{BDCTRL} - t_{BDC} > -2.147 - 3.75 + (1.138) + 0.3 + 0.75

t_{BDCTRL} - t_{BDC} > -3.709

t_{BDCTRL} - t_{BDC} > -3.709 ns

Conclusion: To meet set-up and hold timings of command signals, board delay of command signals `ddr_clk` and `ddr_clk_n` should be:

-3.709 ns < (t_{BDCTRL} - t_{BDC}) < 0.336 ns

Board Design Guidelines

- The `ddr_clk` and `ddr_clk_n` pads should be placed adjacent to each other in the FPGA to get similar internal FPGA delays.

- The `ddr_clk` and `ddr_clk_n` trace delays on the board should be matched.

- The DQ trace delays can be calculated using the following formula, for memory reads:
t_{SKEW} + t_{FDH} - t_{AC} (min) - t_{PD} - t_{DDR_CLK} + t_{FPGA_CLK} < (t_{BDD} + t_{BDC}) < (t_{CK} * 1/2) - t_{SKEW} - t_{FDS} - t_{AC} (max) - t_{PD} - t_{DDR_CLK} + t_{FPGA_CLK}

- The DQ and DQS trace lengths should be balanced and matching to get maximum set-up/hold time during memory writes.

- The address and control signals for the DDR SDRAM are generated on the negative edge of the FPGA clock. The trace lengths for address and control lines are calculated using following equation:
-t_{CCTRL} - t_{CK} * 1/2 + t_{DDR_CLK} + t_{SKEW} + t_{DH} < (t_{BDCTRL} - t_{BDC}) < t_{DDR_CLK} + t_{CK} * 1/2 - t_{SKEW} - t_{DS} - t_{CCTRL} + t_{BDC}

- As shown in Figure 18-1, both FPGA internal clock and `ddr_clk` are generated by a single PLL. It may be difficult to meet read data Set-up and hold timing with a single PLL. As shown in Figure 18-5, a two-PLL clocking scheme is proposed to meet read data set-up and hold timing. Adjusting feedback delay of PLL2 can control delay of `pll_mclk`. Increasing delay on `pll_mclk` can increase the read set-up margin but it also decreases the hold margin. To get better timing, skew between `ddr_clk` and `pll_mclk` has to be minimized.

Figure 18-5. Two PLL Clocking Scheme
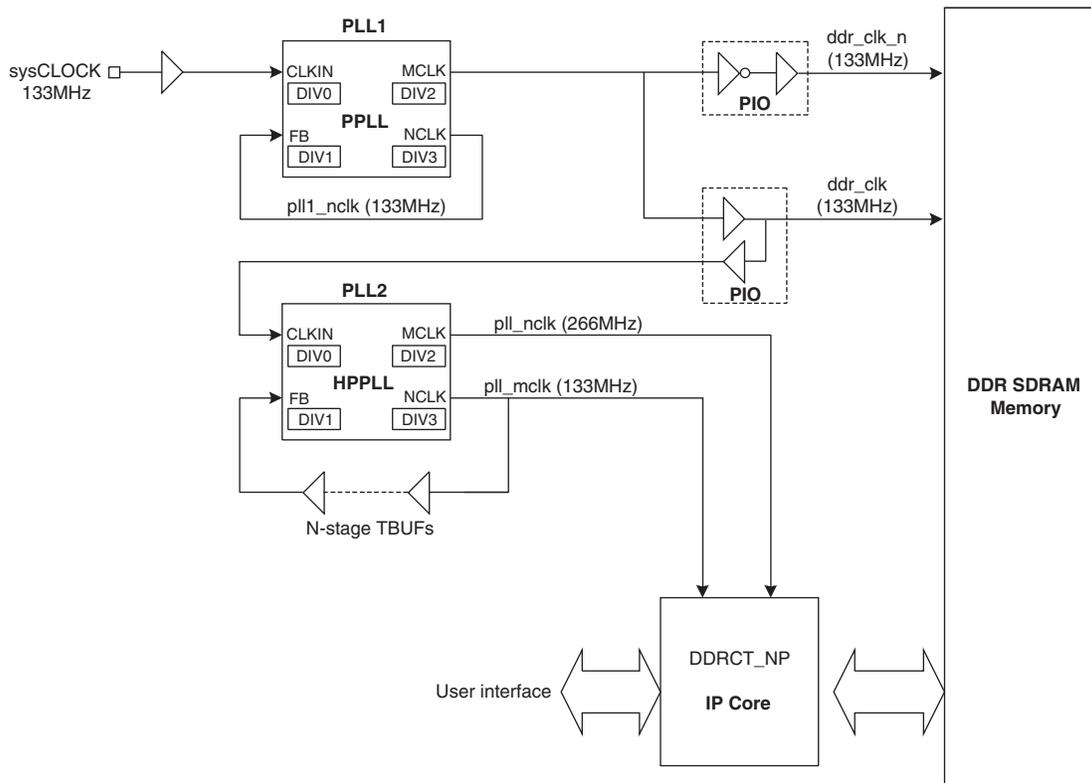
Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

              +1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Appendix A. Example Extractions of Delays from Timing Reports

From the Set-up Report below, which was run for MAX conditions:

- t_{PD} = 0.0 ns
- t_{FDS} = 3.195 ns
- t_{FPGA_CLK} (max) = 6.206 - 3.271 = 2.935 ns

```
================================================================
Preference: INPUT_SETUP PORT "ddr_dq_*" 2.000000 ns CLKNET "pll_nclk" ;
          32 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------
------------------------------------
Passed:  The following path meets requirements by 1.740ns

 Logical Details: Cell type  Pin type        Cell name  (clock net +/-)

   Source:          Port       Pad             ddr_dq_23
   Destination: O-FF In   Data in       U1_ddrct_np_o4_1_008/U3_databusif/ddr_dqoeZ0Z_23  (to
pll_nclk +)

   Data Path Delay:     0.000ns  (0.0% logic, 0.0% route), 0 logic levels.

   Clock Path Delay:    6.206ns  (29.3% logic, 70.7% route), 2 logic levels.

 Constraint Details:

     0.000ns delay ddr_dq_23 to ddr_dq_23 less
     2.000ns offset ddr_dq_23 to clk (totaling -2.000ns) meets
     6.206ns delay clk to ddr_dq_23 less
     3.271ns feedback compensation less
     3.195ns INREG_SET requirement (totaling -0.260ns) by 1.740ns

 Physical Path Details:

     Data path ddr_dq_23 to ddr_dq_23:

   Name     Fanout  Delay (ns)           Site              Resource
                    --------
                    0.000   (0.0% logic, 0.0% route), 0 logic levels.

     Clock path clk to ddr_dq_23:

   Name     Fanout  Delay (ns)           Site              Resource
IN_DEL     ---     1.431         AB4.PAD to       AB4.INCK clk
ROUTE       1      0.816         AB4.INCK to  LLHPPLL.CLKIN clk_c
NCLK_DEL   ---     0.385  LLHPPLL.CLKIN to    LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE      136     3.574   LLHPPLL.NCLK to        N24.SC pll_nclk
                    --------
                    6.206   (29.3% logic, 70.7% route), 2 logic levels.
```

```
      Feedback path:

   Name    Fanout   Delay (ns)        Site              Resource
NCLK_DEL   ---      0.385   LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE      136      2.886   LLHPPLL.NCLK to      LLHPPLL.FB pll_nclk
                            --------
                            3.271   (11.8% logic, 88.2% route), 1 logic levels.


Report:    0.260ns is the minimum offset for this preference.
```

From the Hold Report below, which was run for MIN conditions:

- t_{PD} = 0.0 ns
- t_{FDH} = -1.609 ns
- t_{FPGA_CLK} (min) = 3.144 - 1.905 = 1.239 ns

```
================================================================
Preference: INPUT_SETUP PORT "ddr_dq_*" 2.000000 ns CLKNET "pll_nclk" ;
            32 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------------
---------------


Passed:  The following path meets requirements by 0.370ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)

   Source:        Port       Pad          ddr_dq_31
   Destination:   IO-FF In   Data in        U1_ddrct_np_o4_1_008/U3_databusif/ddr_dqoeZ0Z_31
(to pll_nclk +)

   Data Path Delay:     0.000ns  (0.0% logic, 0.0% route), 0 logic levels.

   Clock Path Delay:    3.144ns  (25.7% logic, 74.3% route), 2 logic levels.

 Constraint Details:

     0.000ns delay ddr_dq_31 to ddr_dq_31 plus
     0.000ns hold offset ddr_dq_31 to clk (totaling 0.000ns) meets
     3.144ns delay clk to ddr_dq_31 plus
     1.905ns feedback compensation less
    -1.609ns INREG_HLD requirement (totaling -0.370ns) by 0.370ns

 Physical Path Details:

     Data path ddr_dq_31 to ddr_dq_31:

   Name    Fanout   Delay (ns)        Site              Resource
                    --------
                    0.000   (0.0% logic, 0.0% route), 0 logic levels.
```

```
        Clock path clk to ddr_dq_31:

    Name     Fanout   Delay (ns)          Site              Resource
IN_DEL      ---       0.576        AB4.PAD to        AB4.INCK clk
ROUTE         1       0.507        AB4.INCK to  LLHPPLL.CLKIN clk_c
NCLK_DEL    ---       0.231  LLHPPLL.CLKIN to      LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       136       1.830    LLHPPLL.NCLK to          C25.SC pll_nclk
                    --------
                      3.144    (25.7% logic, 74.3% route), 2 logic levels.


        Feedback path:

    Name     Fanout   Delay (ns)          Site              Resource
NCLK_DEL    ---       0.231  LLHPPLL.CLKIN to      LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       136       1.674    LLHPPLL.NCLK to       LLHPPLL.FB pll_nclk
                    --------
                      1.905    (12.1% logic, 87.9% route), 1 logic levels.


Report: There is no minimum offset greater than zero for this preference.
```

From the Set-up Report below, which was run for MAX conditions:

- t_{DDR_CLK} (max) = 5.741 - 3.271 = 2.47 ns

```
================================================================================
Preference: CLOCK_TO_OUT PORT "ddr_cas_n" MAX 5.500000 ns CLKPORT "clk" CLKOUT PORT "ddr_clk"
;
          1 item scored, 0 timing errors detected.
--------------------------------------------------------------------------------------
-----------------------------------


Passed:  The following path meets requirements by 3.182ns

 Logical Details: Cell type  Pin type        Cell name  (clock net +/-)

    Source:         Unknown    Q               U1_ddrct_np_o4_1_008/U1_cmdexe/ddr_cas_nZ0  (from
ddr_clk_c -)
   Destination:   Port       Pad             ddr_cas_n

   Data Path Delay:     1.713ns  (100.0% logic, 0.0% route), 1 logic levels.

   Clock Path Delay:    6.346ns  (28.6% logic, 71.4% route), 2 logic levels.

 Constraint Details:

     6.346ns delay clk to ddr_cas_n less
     3.271ns feedback compensation
     1.713ns delay ddr_cas_n to ddr_cas_n less
     2.470ns delay clk to ddr_clk (totaling 2.318ns) meets
     5.500ns offset clk to ddr_cas_n by 3.182ns

 Physical Path Details:

     Clock path clk to ddr_cas_n:
```

```
    Name    Fanout   Delay (ns)         Site              Resource
IN_DEL      ---      1.431         AB4.PAD to        AB4.INCK clk
ROUTE        1       0.816         AB4.INCK to   LLHPPLL.CLKIN clk_c
MCLK_DEL    ---      0.385   LLHPPLL.CLKIN to     LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       449      3.714     LLHPPLL.MCLK to        AE15.SC ddr_clk_c
                     --------
                     6.346   (28.6% logic, 71.4% route), 2 logic levels.


       Data path ddr_cas_n to ddr_cas_n:

    Name    Fanout   Delay (ns)         Site              Resource
OUTREG_DEL  ---      1.713         AE15.SC to        AE15.PAD ddr_cas_n (from ddr_clk_c)
                     --------
                     1.713   (100.0% logic, 0.0% route), 1 logic levels.


       Clock out path:

    Name    Fanout   Delay (ns)         Site              Resource
IN_DEL      ---      1.431         AB4.PAD to        AB4.INCK clk
ROUTE        1       0.816         AB4.INCK to   LLHPPLL.CLKIN clk_c
MCLK_DEL    ---      0.385   LLHPPLL.CLKIN to     LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       449      1.191     LLHPPLL.MCLK to       AF3.OUTDD ddr_clk_c
OUTDD_DEL   ---      1.918       AF3.OUTDD to         AF3.PAD ddr_clk
                     --------
```

5.741   (65.0% logic, 35.0% route), 3 logic levels.

```
       Feedback path:

    Name    Fanout   Delay (ns)         Site              Resource
NCLK_DEL    ---      0.385   LLHPPLL.CLKIN to     LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       136      2.886     LLHPPLL.NCLK to      LLHPPLL.FB pll_nclk
                     --------
```

3.271   (11.8% logic, 88.2% route), 1 logic levels.

```
Report:    2.318ns is the minimum offset for this preference.
```

From the Hold Report below, which was run for MIN conditions:

- t_{DDR_CLK} (min) = 3.043 - 1.905 = 1.138 ns

```
================================================================================
Preference: CLOCK_TO_OUT PORT "ddr_cas_n" MAX 5.500000 ns CLKPORT "clk" CLKOUT PORT "ddr_clk"
;
         1 item scored, 0 timing errors detected.
--------------------------------------------------------------------------------------
----------------------------------


Passed:  The following path meets requirements by 1.056ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)

   Source:         Unknown    Q              U1_ddrct_np_o4_1_008/U1_cmdexe/ddr_cas_nZ0  (from
ddr_clk_c -)
   Destination:    Port       Pad            ddr_cas_n
```

```
    Data Path Delay:      0.928ns   (100.0% logic, 0.0% route), 1 logic levels.

    Clock Path Delay:     3.171ns   (25.4% logic, 74.6% route), 2 logic levels.

  Constraint Details:

      3.171ns delay clk to ddr_cas_n less
      1.905ns feedback compensation
      0.928ns delay ddr_cas_n to ddr_cas_n less
      1.138ns delay clk to ddr_clk (totaling 1.056ns) meets
      0.000ns hold offset clk to ddr_cas_n by 1.056ns

  Physical Path Details:

      Clock path clk to ddr_cas_n:

    Name      Fanout   Delay (ns)          Site              Resource
  IN_DEL       ---      0.576         AB4.PAD to       AB4.INCK clk
  ROUTE         1       0.507         AB4.INCK to  LLHPPLL.CLKIN clk_c
  MCLK_DEL     ---      0.231   LLHPPLL.CLKIN to   LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
  ROUTE        449      1.857    LLHPPLL.MCLK to        AE15.SC ddr_clk_c
                        --------
                        3.171   (25.4% logic, 74.6% route), 2 logic levels.

      Data path ddr_cas_n to ddr_cas_n:

    Name      Fanout   Delay (ns)          Site              Resource
  OUTREG_DEL  ---      0.928         AE15.SC to       AE15.PAD ddr_cas_n (from ddr_clk_c)
                        --------
                        0.928   (100.0% logic, 0.0% route), 1 logic levels.

      Clock out path:

    Name      Fanout   Delay (ns)          Site              Resource
  IN_DEL       ---      0.576         AB4.PAD to       AB4.INCK clk
  ROUTE         1       0.507         AB4.INCK to  LLHPPLL.CLKIN clk_c
  MCLK_DEL     ---      0.231   LLHPPLL.CLKIN to   LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
  ROUTE        449      0.778    LLHPPLL.MCLK to       AF3.OUTDD ddr_clk_c
  OUTDD_DEL    ---      0.951        AF3.OUTDD to        AF3.PAD ddr_clk
                        --------
                        3.043   (57.8% logic, 42.2% route), 3 logic levels.

      Feedback path:

    Name      Fanout   Delay (ns)          Site              Resource
  NCLK_DEL     ---      0.231   LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
  ROUTE        136      1.674    LLHPPLL.NCLK to     LLHPPLL.FB pll_nclk
                        --------
                        1.905   (12.1% logic, 87.9% route), 1 logic levels.

Report:    1.056ns is the maximum offset for this preference.
========================================================================
```

From the Set-up Report below, which was run for MAX conditions. The report shown here is for ddr_ad.

- t_{CCTRL} (max) = (6.392-3.271) + 1.713 = 4.834 ns

Find delays similarly for ddr_ras_n, ddr_cas_n, ddr_we_n, ddr_ba, ddr_cs_n and ddr_cke signals. Then take the max of those delays as t_{CCTRL} (max).

```
=========================================================================================
Preference: CLOCK_TO_OUT PORT "ddr_ad_*" 5.500000 ns CLKNET "ddr_clk_c" ;
          12 items scored, 0 timing errors detected.

Passed:  The following path meets requirements by 0.666ns

 Logical Details: Cell type  Pin type       Cell name  (clock net +/-)

   Source:          Unknown   Q              U1_ddrct_np_o4_1_008/U1_cmdexe/ddr_adZ0Z_6  (from
ddr_clk_c -)
   Destination:    Port      Pad            ddr_ad_6

   Data Path Delay:     1.713ns  (100.0% logic, 0.0% route), 1 logic levels.

   Clock Path Delay:    6.392ns  (28.4% logic, 71.6% route), 2 logic levels.

 Constraint Details:

     6.392ns delay clk to ddr_ad_6 less
     3.271ns feedback compensation
     1.713ns delay ddr_ad_6 to ddr_ad_6 (totaling 4.834ns) meets
     5.500ns offset clk to ddr_ad_6 by 0.666ns

 Physical Path Details:

     Clock path clk to ddr_ad_6:

   Name      Fanout  Delay (ns)         Site                  Resource
IN_DEL      ---     1.431        AB4.PAD to       AB4.INCK clk
ROUTE        1      0.816        AB4.INCK to  LLHPPLL.CLKIN clk_c
MCLK_DEL    ---     0.385  LLHPPLL.CLKIN to    LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE      449      3.760   LLHPPLL.MCLK to       AE14.SC ddr_clk_c
                  --------
                    6.392   (28.4% logic, 71.6% route), 2 logic levels.

     Data path ddr_ad_6 to ddr_ad_6:

   Name    Fanout  Delay (ns)         Site                  Resource
OUTREG_DEL  ---    1.713         AE14.SC to       AE14.PAD ddr_ad_6 (from ddr_clk_c)
                  --------
                    1.713   (100.0% logic, 0.0% route), 1 logic levels.

     Feedback path:

   Name    Fanout  Delay (ns)         Site                  Resource
NCLK_DEL    ---    0.385  LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE      136     2.886   LLHPPLL.NCLK to       LLHPPLL.FB pll_nclk
                  --------
                    3.271   (11.8% logic, 88.2% route), 1 logic levels.

Report:   4.834ns is the minimum offset for this preference.
```

From the Hold Report below, which was run for MIN conditions. The report shown here is for ddr_ad* only.

- t_{CCTRL} (min) = (3.124-1.905) + 0.928 = 2.147 ns

Find delays similarly for ddr_ras_n, ddr_cas_n, ddr_we_n, ddr_ba, ddr_cs_n and ddr_cke signals. Then take the min of those delays as t_{CCTRL} (min).

```
============================================================================
Preference: CLOCK_TO_OUT PORT "ddr_ad_*" 5.500000 ns CLKNET "ddr_clk_c" ;
          12 items scored, 0 timing errors detected.

Passed:  The following path meets requirements by 2.147ns

 Logical Details:  Cell type  Pin type        Cell name  (clock net +/-)

    Source:         Unknown    Q              U1_ddrct_np_o4_1_008/U1_cmdexe/ddr_adZ0Z_4   (from
ddr_clk_c -)
    Destination:    Port       Pad            ddr_ad_4

    Data Path Delay:     0.928ns  (100.0% logic, 0.0% route), 1 logic levels.

    Clock Path Delay:    3.124ns  (25.8% logic, 74.2% route), 2 logic levels.

 Constraint Details:

     3.124ns delay clk to ddr_ad_4 less
     1.905ns feedback compensation
     0.928ns delay ddr_ad_4 to ddr_ad_4 (totaling 2.147ns) meets
     0.000ns hold offset clk to ddr_ad_4 by 2.147ns

 Physical Path Details:

     Clock path clk to ddr_ad_4:

   Name      Fanout  Delay (ns)         Site                 Resource
IN_DEL       ---     0.576        AB4.PAD to       AB4.INCK clk
ROUTE        1       0.507        AB4.INCK to  LLHPPLL.CLKIN clk_c
MCLK_DEL     ---     0.231  LLHPPLL.CLKIN to    LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE        449     1.810    LLHPPLL.MCLK to         T26.SC ddr_clk_c
                     --------
                     3.124   (25.8% logic, 74.2% route), 2 logic levels.

     Data path ddr_ad_4 to ddr_ad_4:

   Name      Fanout  Delay (ns)         Site                 Resource
OUTREG_DEL   ---     0.928        T26.SC to        T26.PAD ddr_ad_4 (from ddr_clk_c)
                     --------
                     0.928   (100.0% logic, 0.0% route), 1 logic levels.

     Feedback path:

   Name      Fanout  Delay (ns)         Site                 Resource
NCLK_DEL     ---     0.231  LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE        136     1.674    LLHPPLL.NCLK to      LLHPPLL.FB pll_nclk
                     --------
                     1.905   (12.1% logic, 87.9% route), 1 logic levels.

Report:   2.220ns is the maximum offset for this preference.
```

PCB Layout Recommendations for BGA Packages

Introduction

As Ball Grid Array (BGA) packages become increasingly popular and become more populated across the array with higher pin count and smaller pitch, it is important to understand how they are affected by various board layout techniques. This document provides a brief overview of PCB layout considerations when working with BGA packages. It outlines some of the most common problems and provides tips for avoiding them at the design stage. A key challenge of adopting fine-pitch (0.8 mm or less) BGA packages is the design of a route fanout pattern that maximizes I/O utilization while minimizing fabrication cost. This technical note provides an overview of PCB design examples provided by Lattice Semiconductor.

For more information and design examples see the PCB Design Support page at the Lattice Semiconductor web site (www.latticesemi.com/support/pcbdesignsupport.cfm).

BGA Board Layout Recommendations

All Lattice BGA packages have Solder Mask Defined (SMD) pads. In order to evenly balance the stress in the solder joints, Lattice recommends PCB solder pads be SMD with dimensions as similar to the applicable BGA as possible. If Non Solder Mask Defined Pads (NSMD) are used, the optimum pad dimensions will result in an equivalent surface contact area on the PCB as on the BGA component.

Table 19-1. Lattice Semiconductor SMD/NSMD Pad Recommendations

	Pitch 0.4 mm ucBGA	Pitch 0.5 mm csBGA	Pitch 0.8 mm caBGA	Pitch 1.00 mm (fpBGA, ftBGA, fpSBGA, fcBGA)				Pitch 1.27 mm PBGA, SBGA
				100 fpBGA, 256 ftBGA	All Other fpBGA, ftBGA,fpSBGA	1020 Organic fcBGA	All Other fcBGA	
Solder Mask Defined (SMD) Pad Recommendations								
Optimum Solder Land Diameter[1]	0.25	0.40	0.50	0.50	0.55	0.65	0.60	0.73
Solder Land Diameter Range	0.20 - 0.25	0.25 - 0.40	0.40 - 0.60	0.40 - 0.60	0.45 - 0.70	0.45 - 0.70	0.45 - 0.70	0.55 - 0.85
Optimum Solder Mask Opening (M)	0.20	0.25	0.40	0.40	0.45	0.55	0.50	0.63
Solder Mask Opening Range	0.18 - 0.22	0.20 - 0.30	0.35 - 0.50	0.35 - 0.50	0.40 - 0.60	0.40 - 0.60	0.40 - 0.60	0.50 - 0.75
Non Solder Mask Defined (NSMD) Pad Recommendations								
Optimum Solder Land Diameter (L)	0.16	0.23	0.35	0.35	0.40	0.52	0.45	0.58
Solder Land Diameter Range	0.14 - 0.18	0.20 - 0.30	0.35 - 0.50	0.35 - 0.50	0.40 - 0.60	0.40 - 0.60	0.40 - 0.60	0.50 - 0.78
Optimum Solder Mask Opening[1]	0.22	0.35	0.45	0.50	0.55	0.70	0.60	0.73
Solder Mask Opening Range	0.20 - 0.25	0.30 - 0.40	0.45 - 0.55	0.45 - 0.60	0.50 - 0.70	0.50 - 0.85	0.50 - 0.75	0.60 - 0.88

1. Assuming 0.050 mm solder mask tolerance.

Table 19-2. Lattice Semiconductor BGA Package Types

Package Type	Description
PBGA	**Plastic BGA** with 1.27 mm solder ball pitch. Die up configuration.
fpBGA	**Fine Pitch BGA** – Plastic BGA with 1.0 mm solder ball pitch. Die up configuration.
ftBGA	**Fine Pitch Thin BGA** – Thin plastic BGA with 1.0 mm solder ball pitch. Die up configuration.
caBGA	**Chip Array BGA** – Plastic BGA with 0.8 mm solder ball pitch. Die up configuration.
csBGA	**Chip Scale BGA** – Plastic BGA with 0.5 mm solder ball pitch. Die up configuration.
fcBGA	**Flip-Chip BGA** with 1.0 mm solder ball pitch. Die down configuration. May have a ceramic or plastic substrate.
SBGA	**Super BGA** – Similar to PBGA, but with an integrated heatsink plate. This package has 1.27 mm solder ball pitch and die down configuration. SBGA packages offer enhanced thermal dissipation capability.
fpSBGA	**Fine Pitch SBGA** – Super BGA with 1.0 mm solder ball pitch. Die down configuration.
ucBGA	**Ultra Chip BGA** – Saw singulated plastic ball grid array package with 0.4 mm ball pitch.

BGA Breakout and Routing Examples

Lattice provides several resources and different design implementations that show BGA breakout and routing of various fine-pitch BGA packages. Different stack up and layer counts are also used to show a range of design rules and fabrication costs. It is important to consult with your board fabrication and assembly houses as to the most economical and reliable process for your application.

Currently there is a wide choice of BGAs from Lattice, with many devices offered in multiple packages and pitches of BGA densities as well as non-BGA options such as TQFP, QFN and others. The BGA pitch or "center to center" ball dimensions include, 1.00 mm BGAs, space-saving 0.5 mm pitch chip scale BGA and 0.4mm pitch ultra chip scale BGA packages. Fine pitch packages offer advantages and disadvantages alike. Finer pitch means that the trace and space limits will have to be adjusted down to match the BGA. Many times a design can get away with small traces underneath the BGA then fan out with a slightly larger trace width. The PCB fabrication facility will need to be aware of your design objectives and check for the smallest trace dimensions supported. Smaller traces take more time to inspect, check and align etc. Etching needs to be closely monitored when trace and space rules reach their lower limit.

The combination of fanout traces, escape vias, and escape traces that allow routing out from under the BGA pin array to the perimeter of the device are collectively referred to as the "BGA breakout". The fanout pattern will arrange the breakout via, layer, and stack-up to maximize the number of I/Os that can be routed. Fanout patterns are an important consideration for devices over 800 pins and can be follow polar (north/south/east/west) or layer-biased directions. (Source: *BGA Breakouts and Routing*, Charles Pfeil, Mentor Graphics).

Figure 19-1. BGA Breakout Routing Terms
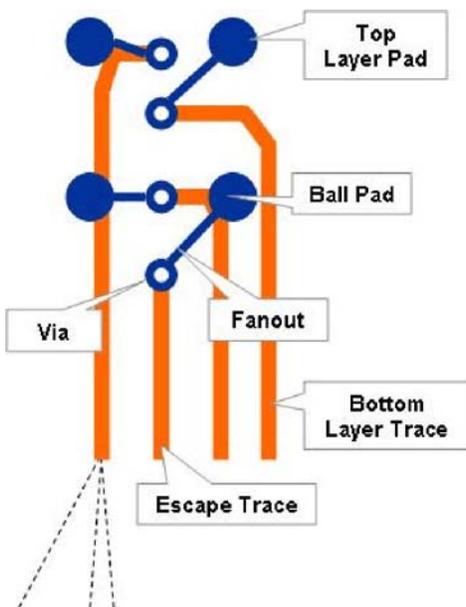
Lattice provides BGA breakout and routing examples for various fine pitch packages (www.latticesemi.com/support/pcbdesignsupport/bgabreakoutroutingexample/index.cfm). Each package example is built to comply with IPC7351 (www.ipc.org) specifications and nomenclature conventions. Some examples include different layout options depending on design and cost goals. For example, the 256-ball chip array BGA (BN256) examples demonstrate a design with fully-utilized I/Os, fine trace width and pitch, on a 6-layer PCB stack-up and a less expensive design with relaxed design rules, and fewer I/O pads routed, on a 4-layer PCB stack up.

Table 19-3. Package Layout Example Summary

Package	Example #	Pitch (mm)	Signal/ Power Layers	Trace/ Width- Space (mm)	Ball Pad (mm)	Ball Mask (mm)	Via Pad (mm)	Via Drill (mm)
MN64	1	0.5	6	.100/.100	.23	.33	.30	.125
UMN64	1	0.4	6	.100/.100	.18	.28	.25	.10
MN100	1	0.5	4	.085/.085	.23	.38	.45	.20
	2	0.5	4	.100/.100	.23	.38	.45	.20
MN132	1	0.5	4	.085/.085	.23	.38	.40	.15
	2	0.5	4	.100/.100	.23	.38	.40	.15
MN144	1	0.5	6	.100/.100	.23	.33	.30	.125
	2	0.5	4	.100/.100	.23	.38	.30	.125
BN256	1	0.8	6	.100/.100	.35	.50	.40	.125
	2	0.8	4	.100/.100	.35	.50	.40	.15

For mechanical dimension details on packages, see the Lattice Package Diagrams document.

In order to show how some of the routing challenges are solved, examples are provided for fine-pitch BGA packages from the MachXO™ and the ispMACH® 4000ZE families. Principles for these apply to other Lattice BGA packaged products.

64-ball csBGA BGA Breakout and Routing Example

This example places an ispMACH 4000ZE CPLD in a 5x5 mm, 0.5 mm pitch, 64-ball csBGA package (LC4064ZE-MN64) in an 6-layer stack up with maximum I/O utilization.

Figure 19-2. CAM Artwork Screen Shots 64-Ball csBGA

Layer 1 Primary

Layer 2 Signal

Layer 3 GND

Layer 4 Power

Layer 5 Signal

Layer 6 Secondary
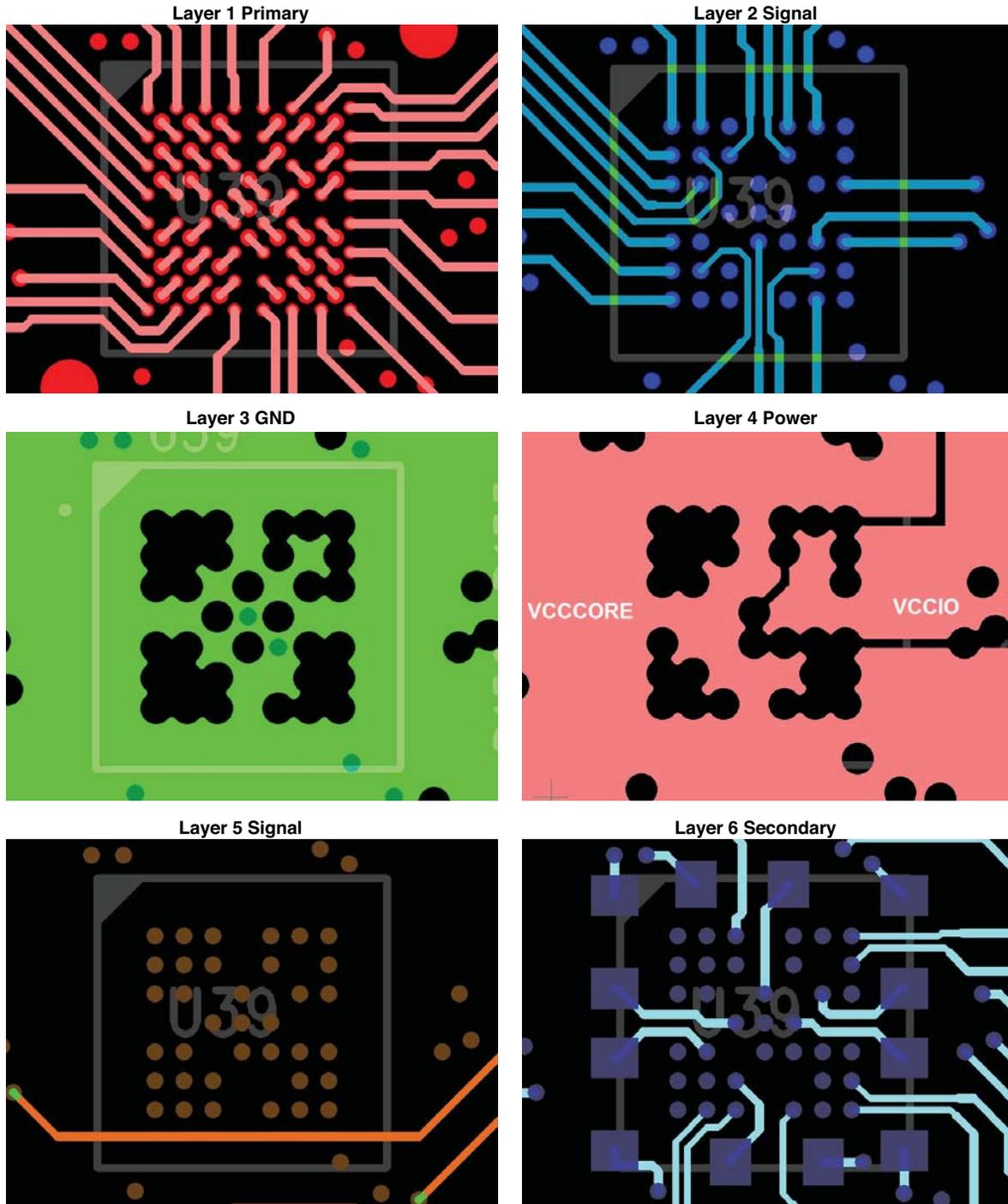
64-ball ucBGA BGA Breakout and Routing Example

This example places an ispMACH 4000ZE CPLD in a 4x4 mm, 0.4 mm pitch, 64-ball ucBGA package (LC4064ZE-UMN64) in an n-layer stack up with maximum I/O utilization. This example demonstrates a modified dogbone fanout technique to get access to all pins yet limiting number of layers and via schedules, while setting up layers to use reference planes for high-speed signal traces.

Figure 19-3. CAM Artwork Screen Shots, 64-Ball ucBGA

Layer 1 Primary

Layer 2 GND

Layer 3 Signal

Layer 4 Signal

Layer 5 Power

Layer 6 Secondary
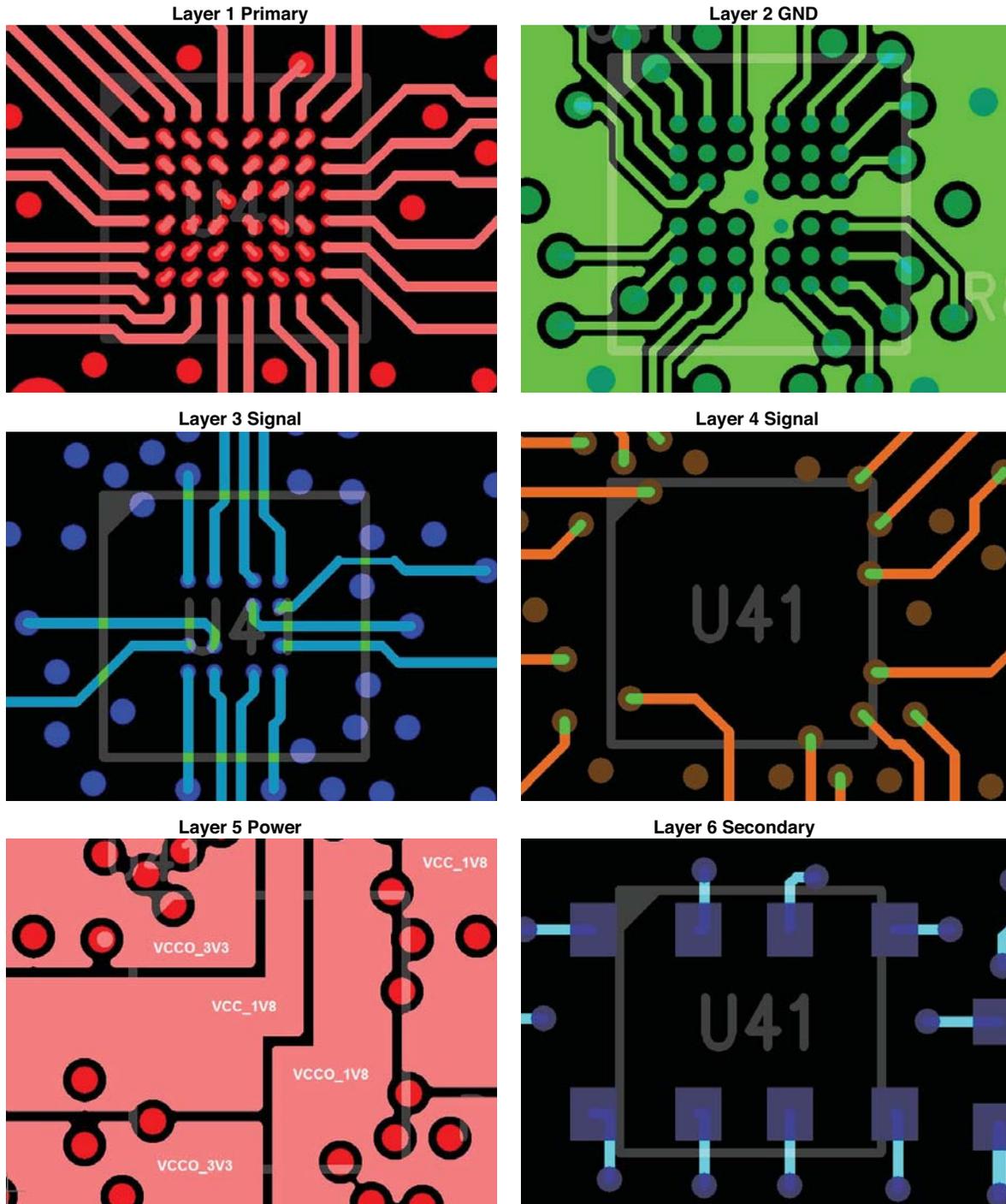
100-ball csBGA BGA Breakout and Routing Examples

These examples place a MachXO PLD in a 8x8 mm, 0.5 mm pitch, 100-ball csBGA package (LXMXO640-M100/MN100) into two fabrication scenarios. One for a 6-layer stack up with maximum I/O utilization and a 4-layer with about 15% fewer I/Os. The 4-layer (Example #2) design makes maximum use of via and trace geometry to reduce layer count and ease fabrication while still maintaining high I/O usage.

Figure 19-4. CAM Artwork Screen Shots, Example #1, 100-ball csBGA
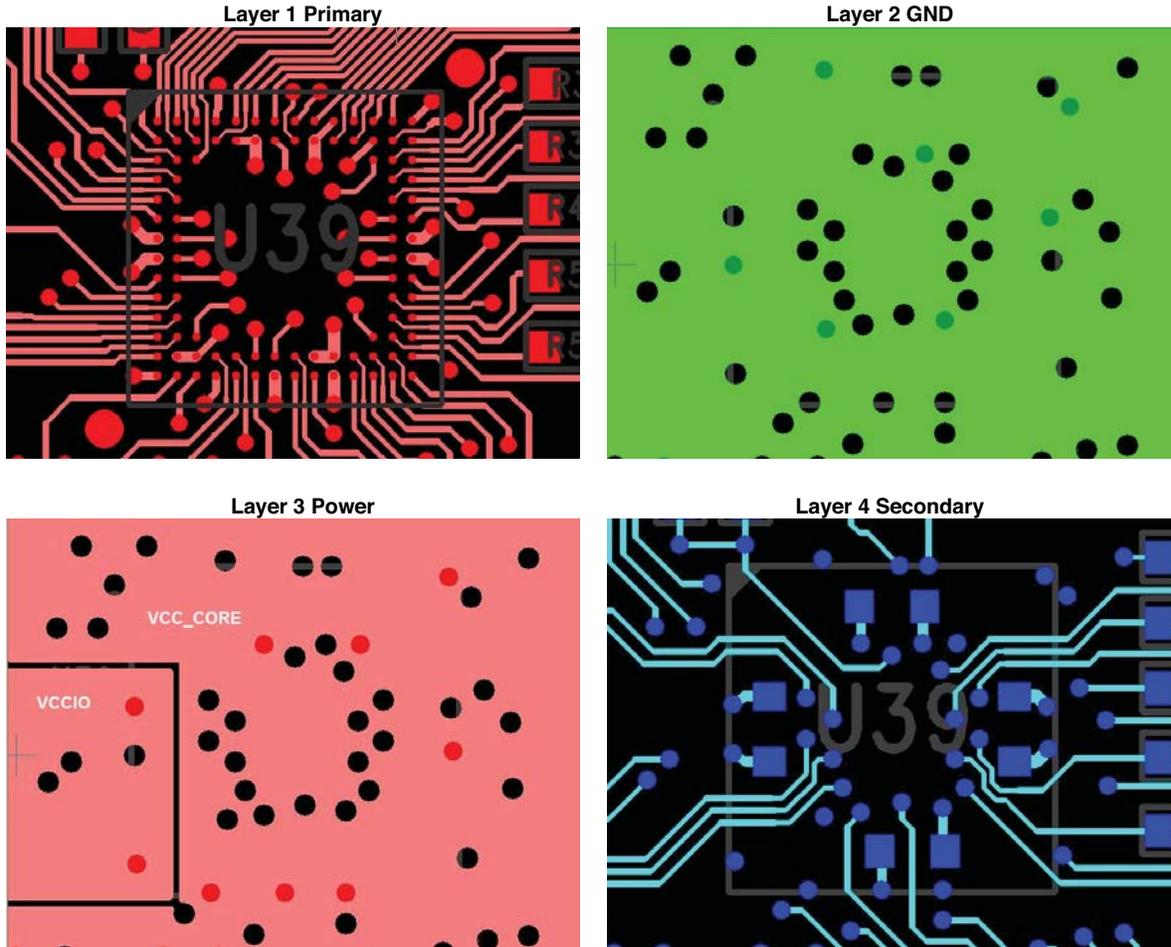
Figure 19-5. CAM Artwork Screen Shots, Example #2, 100-ball csBGA

Layer 1 Primary

Layer 2 GND

Layer 3 Power

Layer 4 Secondary
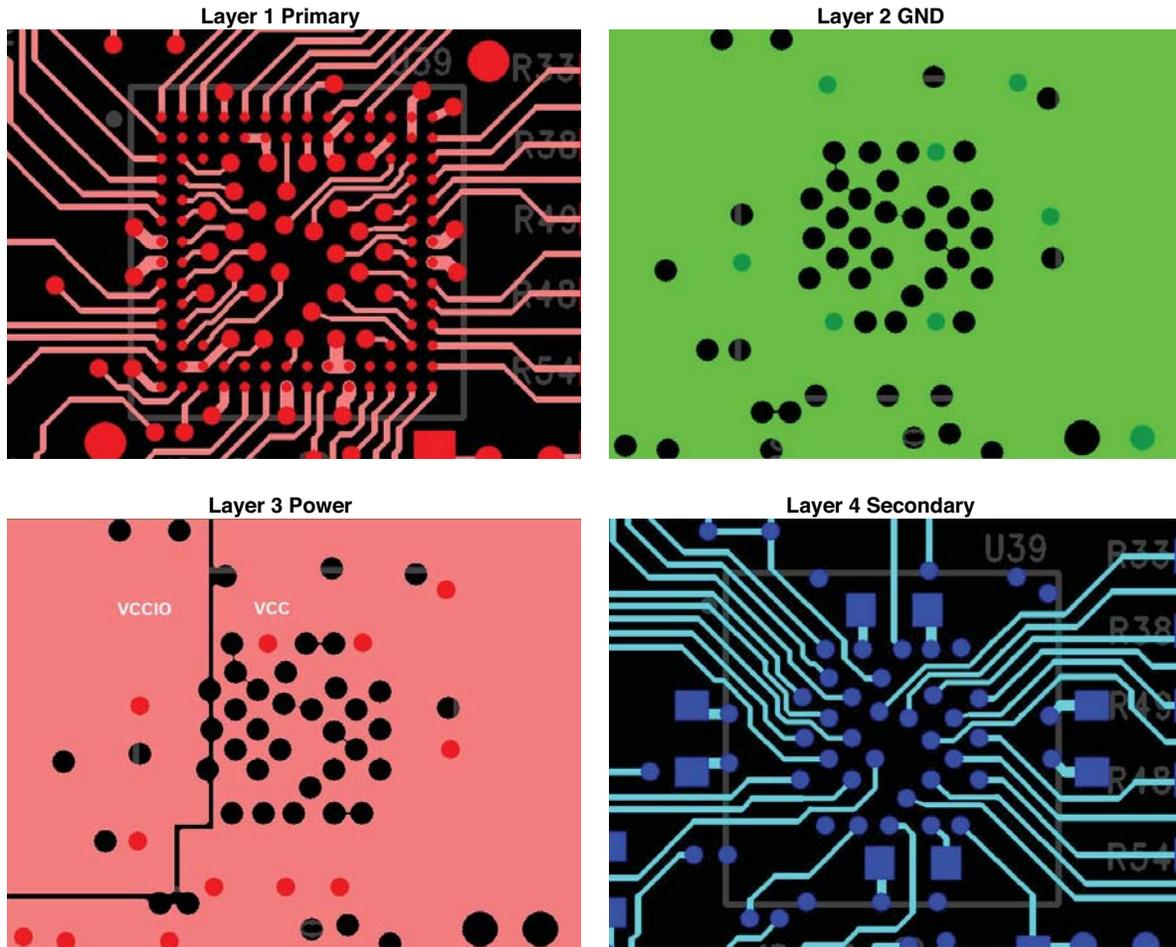
132-ball csBGA BGA Breakout Examples

These examples place a MachXO PLD in a 8x8 mm, 0.5 mm pitch, 132-ball csBGA package (LCMXO640-M132/MN132) into two fabrication scenarios. One for a 6-layer stack up with maximum I/O utilization and a 4-layer with about 15% fewer I/Os.

Figure 19-6. CAM Artwork Screen Shots, Example #1, 132-ball csBGA
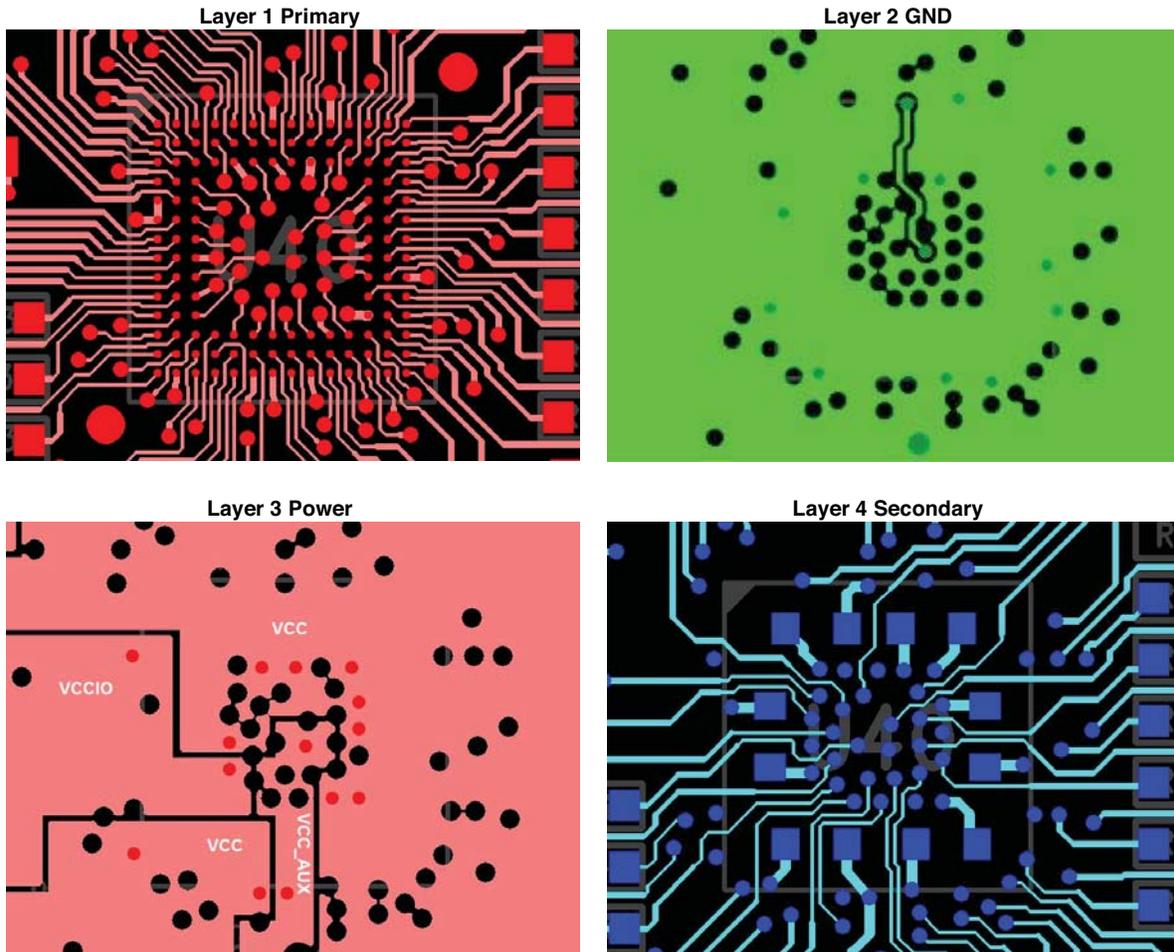
Figure 19-7. CAM Artwork Screen Shots, Example #2, 132-ball csBGA

Layer 1 Primary

Layer 2 GND

Layer 3 Power

Layer 4 Secondary
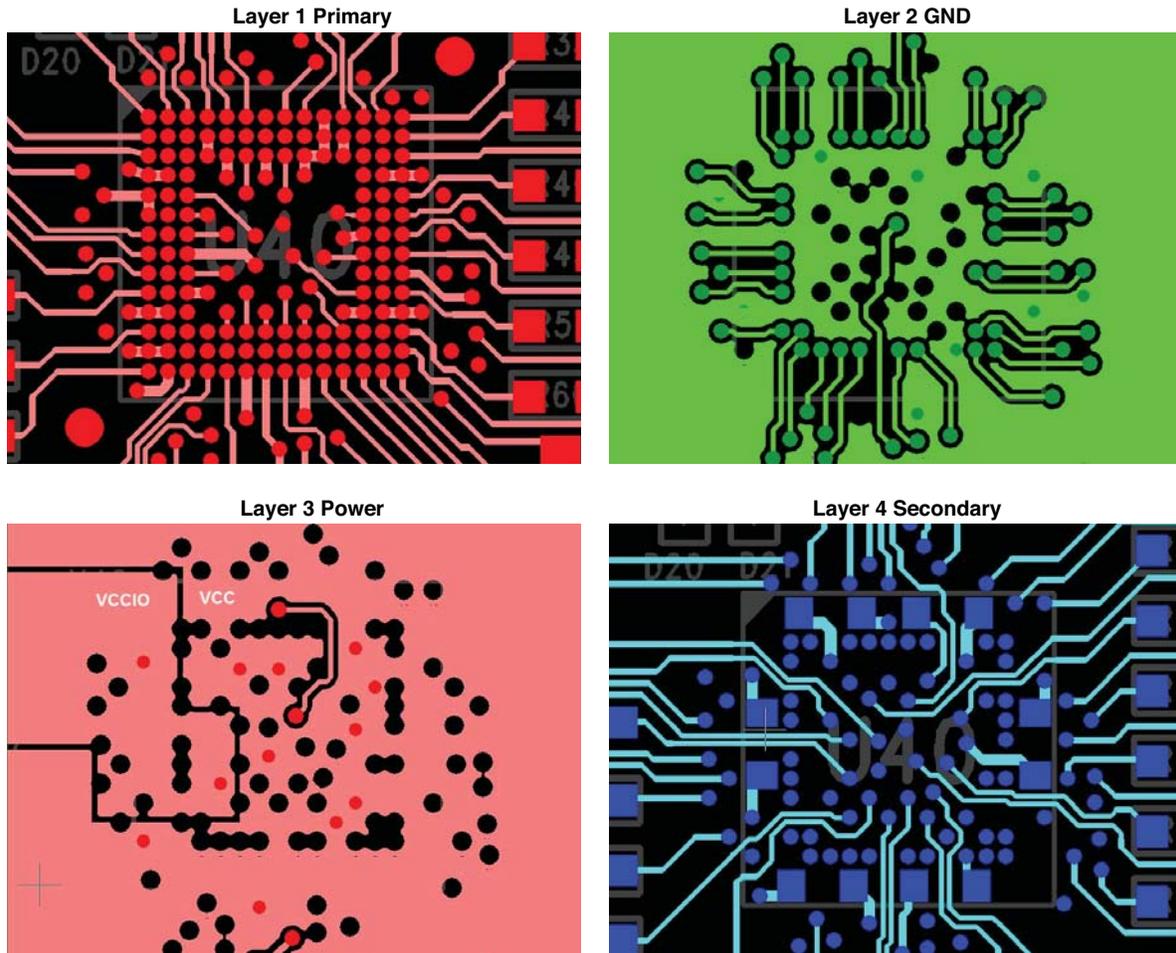
144-ball csBGA BGA Breakout Examples

These examples place an ispMACH 4000ZE in a 7x7 mm, 0.5 mm pitch, 144-ball csBGA package (LC4256ZE-MN144) into two fabrication scenarios. One for a 6-layer stack up with maximum I/O utilization and a 4-layer with about 5% fewer I/Os. The 6-layer (Example #1) design avoids uses of micro vias and takes advantage of removed pads on inner layers to route all pins out to 6 layers with good layer structure for high-speed signal integrity.

Figure 19-8. CAM Artwork Screen Shots, Example #1, 144-ball csBGA

Layer 1 Primary

Layer 2 GND

Layer 3 Signal

Layer 4 Signal

Layer 5 Power

Layer 6 Secondary
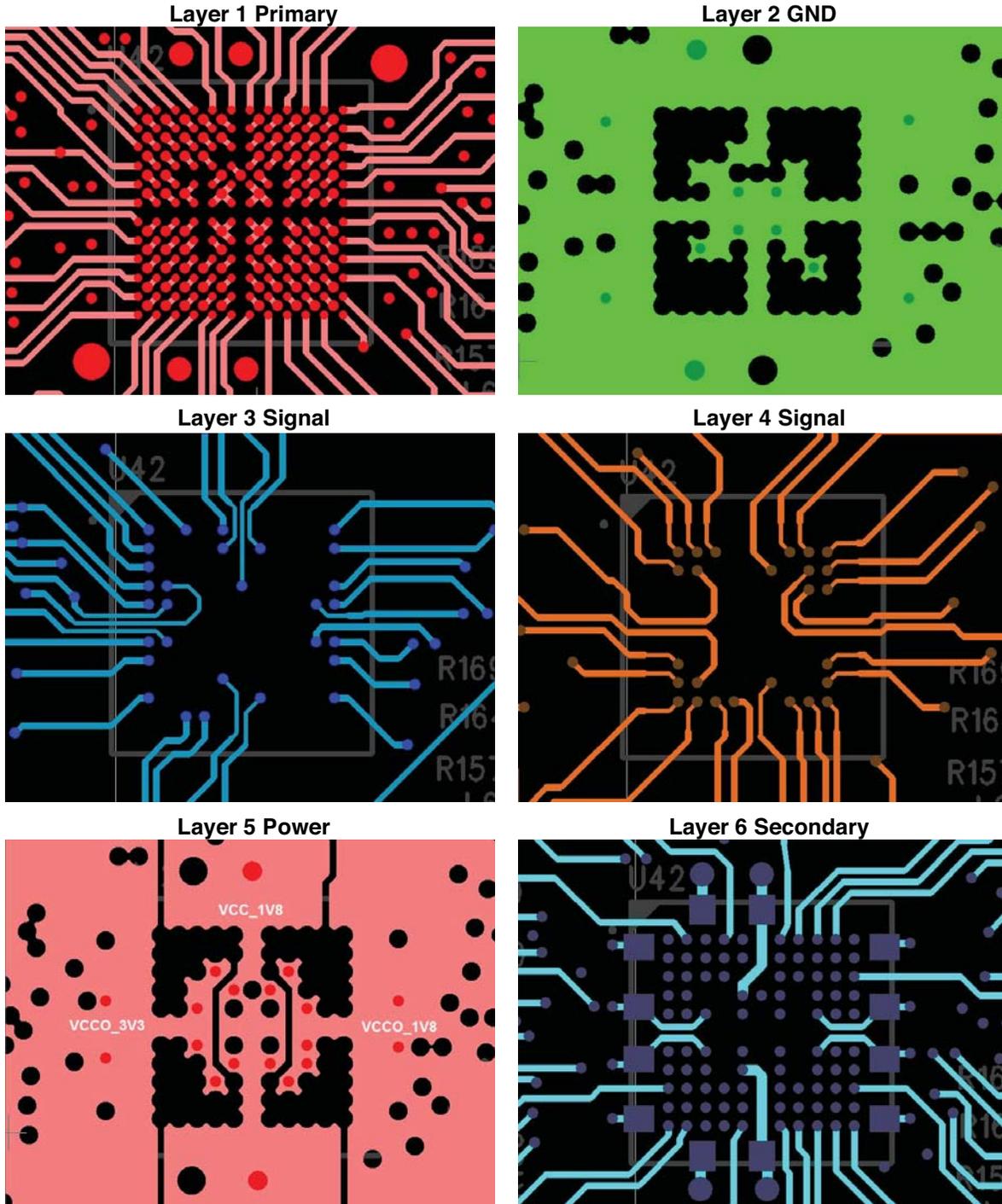
Figure 19-9. CAM Artwork Screen Shots, Example #2, 144-ball csBGA

Layer 1 Primary

Layer 2 GND

Layer 3 Power

Layer 4 Secondary
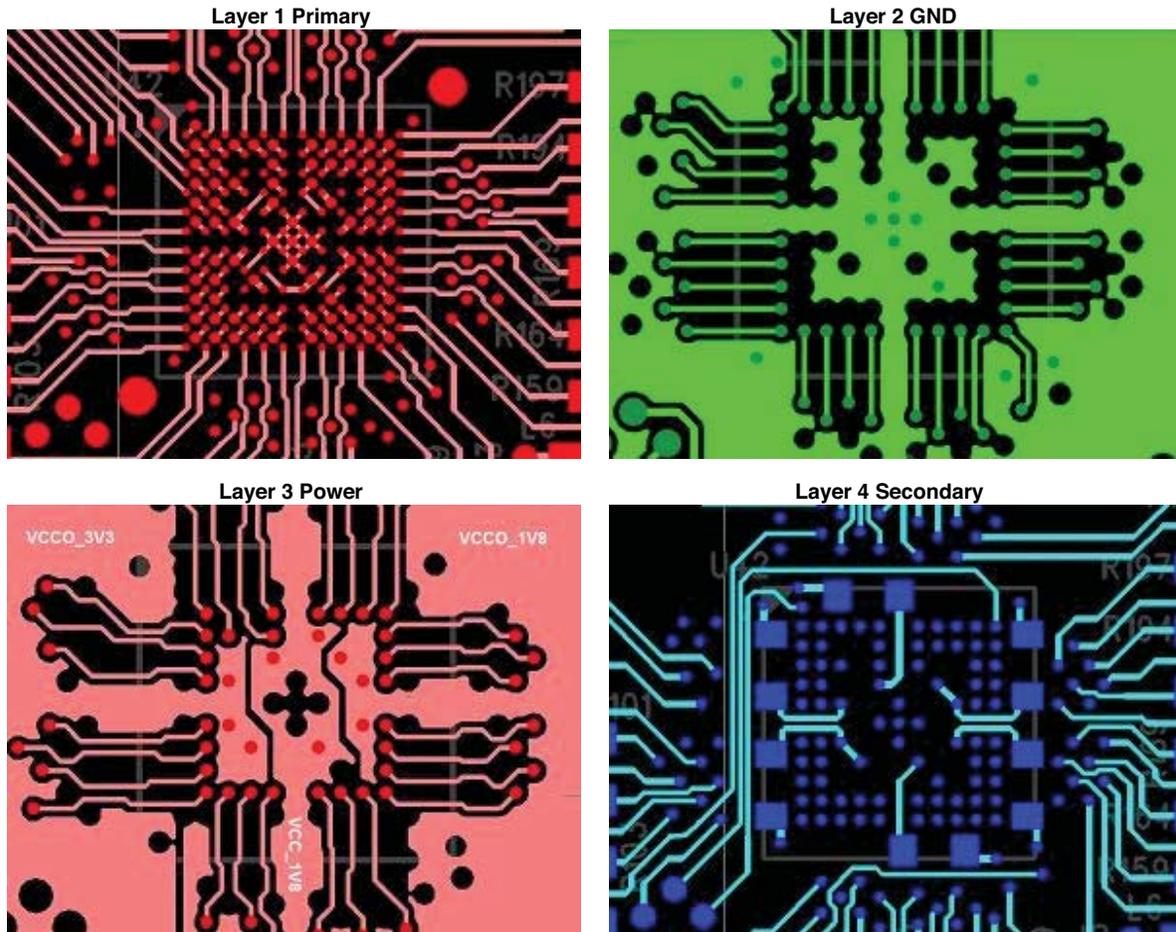
256-ball caBGA BGA Breakout Examples

This BGA breakout and routing example places a MachXO PLD in a 14x14 mm, 0.8 mm pitch, 256-ball caBGA package (LCMXO2280-B256/BN256) into two fabrication scenarios. One for a 6-layer stack up with maximum I/O utilization and a 4-layer with about 10% fewer I/Os. The 6-layer design (Example #1), demonstrates the best use of mechanically drill blind vias to place caps near power pins to minimize layers.

Figure 19-10. CAM Artwork Screen Shots, Example #1, 256-Ball caBGA

Layer 1 Primary   Layer 2 Signal

Layer 3 GND   Layer 4 Power

Layer 5 Signal   Layer 6 Secondary
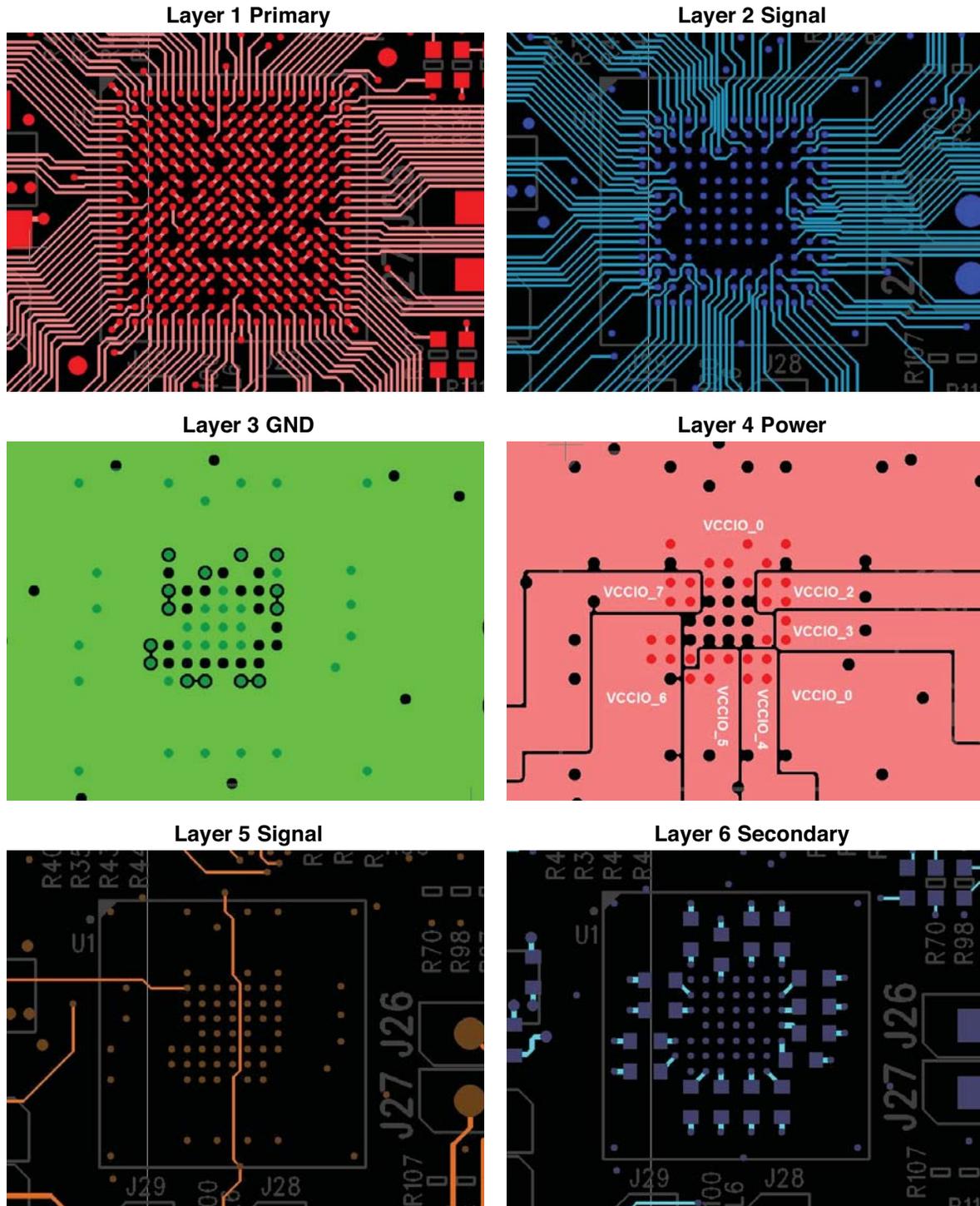
Figure 19-11. CAM Artwork Screen Shots, Example #2, 256-ball caBGA

Layer 1 Primary

Layer 2 GND

Layer 3 Power

Layer 4 Secondary
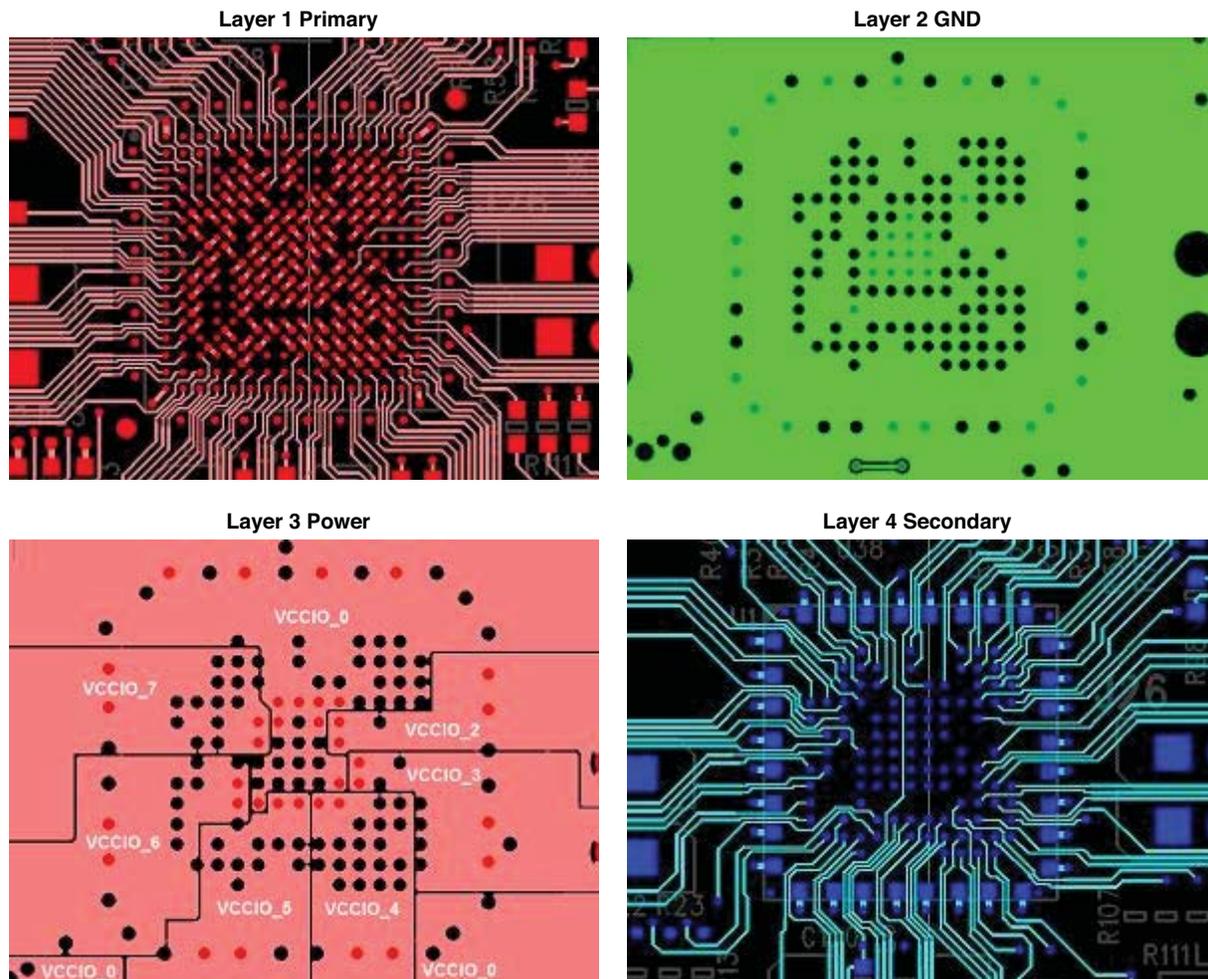
PCB Fabrication Cost and Design Rule Considerations

PCB fabrication cost is a key consideration for many electronics products. By reviewing the IC device package ball density and pitch, I/O signal requirements of your application, and the manufacturing constraints of your PCB fabrication facility you can better weigh the trade-offs between design decisions.

Choosing the best package for your application involves answering a few questions:

- What is the driving factor in the application? The smallest possible form factor or a low PCB cost?
- How many I/O signals does the application require?
- What PCB layer stack up will provide the best I/O density within budget?
- What layout design rules does the printed-circuit board (PCB) vendor support?
- How many PCB layers does the budget allow?

As the ball pitch becomes smaller with each new BGA generation, new PCB fabrication techniques and signal via type have been developed to handle the complexities. Micro vias, laser vias, filled, buried and blind vias, even buried and plated over vias. Complex boards use a combination of most of these.

Stack-up types, ordered by cost, high-to-low include:

• High-Density Interconnect (HDI) build up with micro vias

• Laminated with blind and buried vias

• Laminated with through vias

Figure 19-12. Stack-up Example
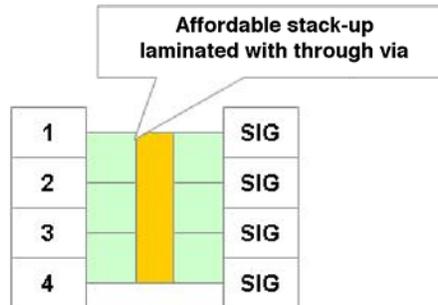
HDI is a "sandwich" with older-style larger geometry in the middle with fully drilled through holes and then a stack of fine geometry of blind, buried or mixed via, laminated both on the top and bottom of the middle stack-up. The laminated layers are thinner than traditional layers and allow finer drilling technology. Staggered micro vias allow vias within close tolerance or connected to a BGA pad to go down to the next layer or more to route away for escape routing or underneath the BGA device for further interconnect. HDI type interconnect is used on complex boards and takes extra steps in the processing flow due to special drilling, plating an laminations. It is a mix of older technology mixed with newer technology that results in a board that is highly routable.

Figure 19-13. HDI Stack-up with Staggered Micro Vias
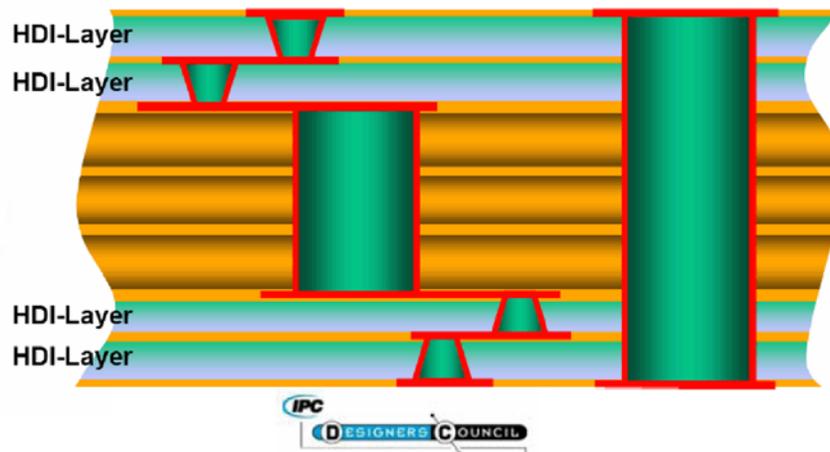
Advantages and Disadvantages of BGA Packaging

As pin counts increase and board space becomes more valuable, it is important to place as many circuits per square inch as possible. BGA offer the best I/O density for a given PCB area. Lattice offers a range of packages from a 4x4 mm 64-ball csBGA to a 33x33 mm 1704-ball fcBGA.

One of the greatest advantages of BGA packaging is that it can be supported with existing placement and assembly equipment. BGAs also offer significantly more misalignment tolerance and less susceptibility to co-planarity issues. Even if the solder paste is misaligned or the device is slightly offset, the BGA will self-center during the

reflow process. This is due to the surface tension of the solder and flux in its molten state pulling each ball into the center of the pad.

Figure 19-14. Misalignment of BGA Balls vs. QPF Leads
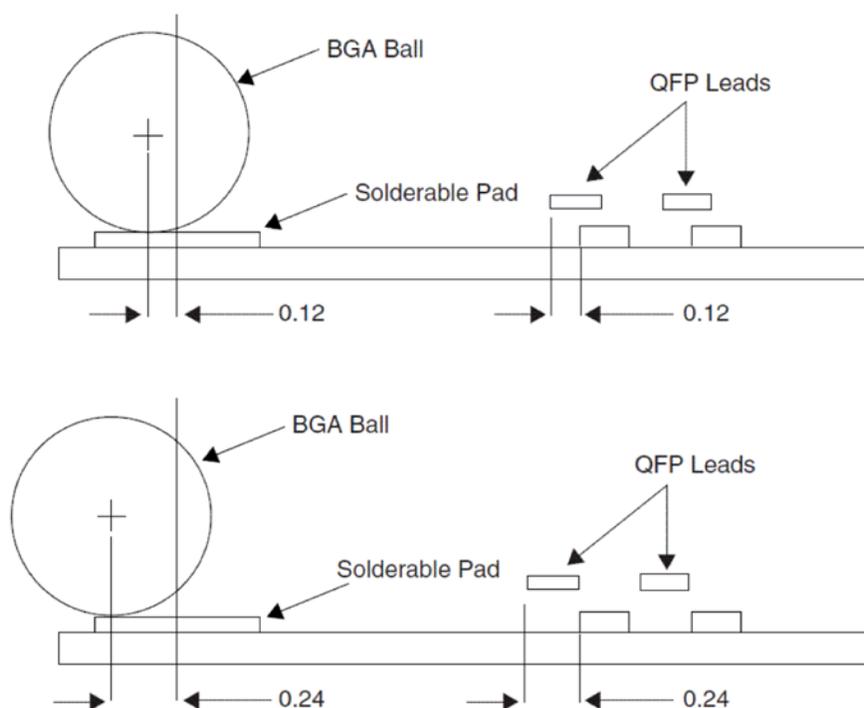
Controlling the oven re-flow profile is one of the most important assembly parameters for consistent and reliable BGA placement. Profiles are typically tested on a pre-run. One or two panels are run to dial in the process, then visual and X-ray inspection equipment are used for verification.

BGA packages present numerous benefits previously unobtainable in surface mount packaging technology. BGAs provide higher pin counts in a much smaller area than was previous possible. No longer is the package design limited to connections along the periphery on the outside quadrants of the package edge like a PQFP or TQFP outline. Fully populated ball grid arrays with pitches as small as 0.4 mm are available.

Some BGA devices are arranged with de-populated interconnect near or around the center. These are dependent on the die size and number of pins. The area void of interconnect in the middle of the array has some advantages, it can be used for escape routing vias or tying directly to the ground or power planes.

Although the packages can be quite complex and densely populated, all of these packages receive strict quality and reliability testing and are widely accepted today by designers and PCB fabrication/assembly houses. All of this is due to advances in equipment and technology that have allowed a smooth transition into the assembly flow.

BGA Package Test and Assembly

How can a pad/ball/pin be tested that can't be seen? All connections are hidden under the substrate at the ball interconnect, making it impossible to directly probe or test. To address this limitation, Lattice programmable devices provide JTAG, BSCAN, and boundary scan cells that allow electronic test and continuity of each pin with a boundary scan tester. This can be embedded into the system itself or driven externally from a high-speed test head. The boundary scan can test the pins or board for simple continuity tests or full functional test by shifting in test patterns through the JTAG port.

For debug or prototype boards it may be necessary to place test points, open vias, or pads to have access to a given set of pins in order to drive, over-drive or observe a given set of signals. These can be very small, as many

pogo pin type probes are extremely small and can handle GHz range DC frequency. Zero ohm resistors are also commonly used in first-run boards as a way to gain access to a pad or pin.

After assembly, BGA solder point quality and integrity is visually inspected with X-ray technology as part of the fabrication process. A special X-ray machine can look through the plastic package, substrate and silicon to directly view the BGA solder balls, vias, traces and pads.

Figure 19-15. Example of How Defects May Appear in an X-Ray Image
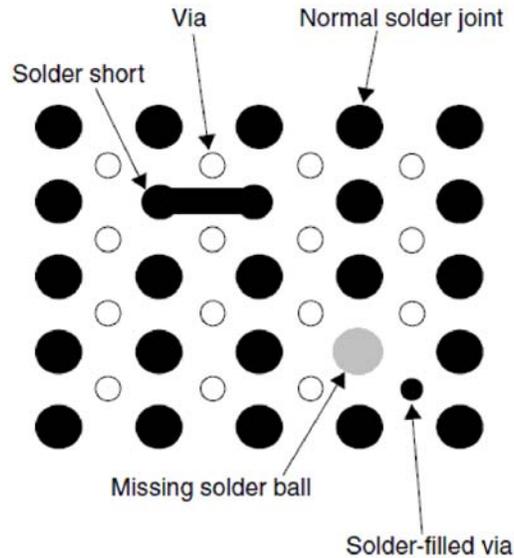
The X-ray image in Figure 19-16 shows proper alignment; no voids or defects are noted. Balls, vias and traces are visible.

Figure 19-16. X-Ray Inspection Plot of ispMACH 4000ZE 144-ball csBGA
(Photo Courtesy of CEM, Ltd., www.cemltd.com)
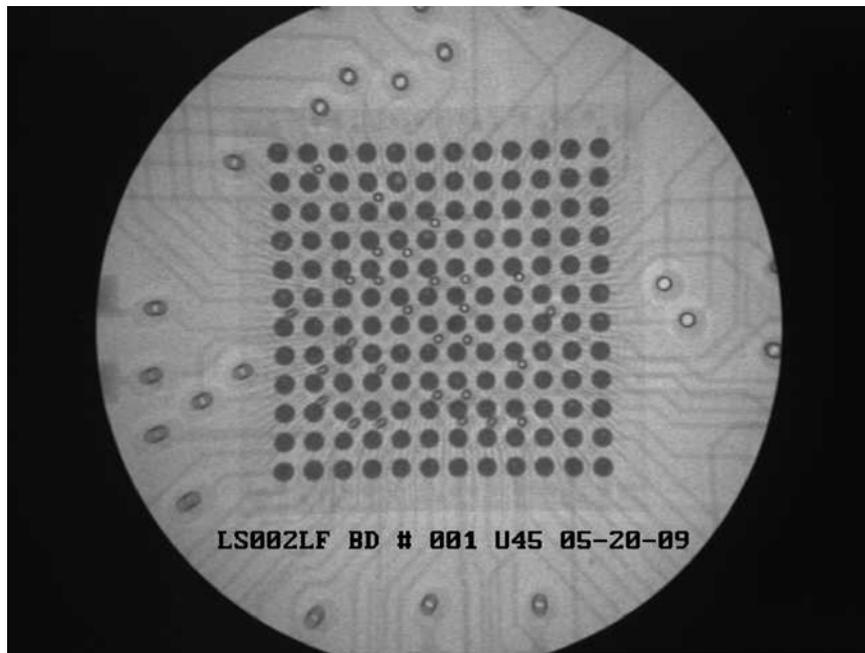
PCB cross-sectioning is another method used to verify BGA and PCB quality and reliability. After a new process has been developed or changed or when qualifying a new vendor, it is a good practice to get physical information from the vendor on their BGA reflow. When trace/space and drill or laser tolerances are nearing their limits, board yield can be as be as low as 50% for the bare board fab. Cross-sections give you a good idea if the process is correct but do not guarantee each batch or each board design will behave the same way due to layout dimensions, thermal issues, flux/paste and alignment, etc.

Figure 19-17 shows a BGA cross-section that uses a non-soldermask over bare copper-defined pad. (NSMD) pad.

Figure 19-17. BGA Cross-Section
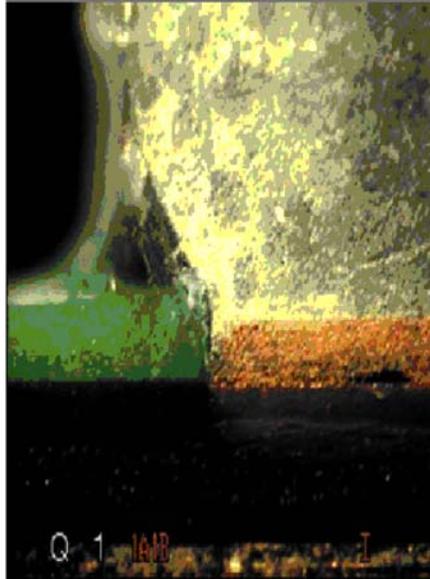
Figure 19-18 shows "offset" micro via stack routing between layers.

Figure 19-18. Cross-Section of Micro Vias
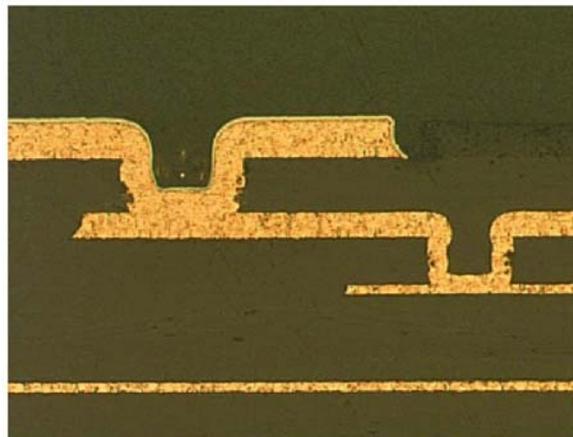
High-resolution video cameras are used for edge inspection to verify ball seating, distortion, solder wetting, flow, contaminates, etc. Figure 19-19 is a video view of a side/edge shot looking at BGA balls soldered down to the isp-MACH 4000ZE Pico Evaluation Board (www.latticesemi.com/4000ze-pico-kit), an FR4 4-layer PCB.

Figure 19-19. Edge View Camera Inspection
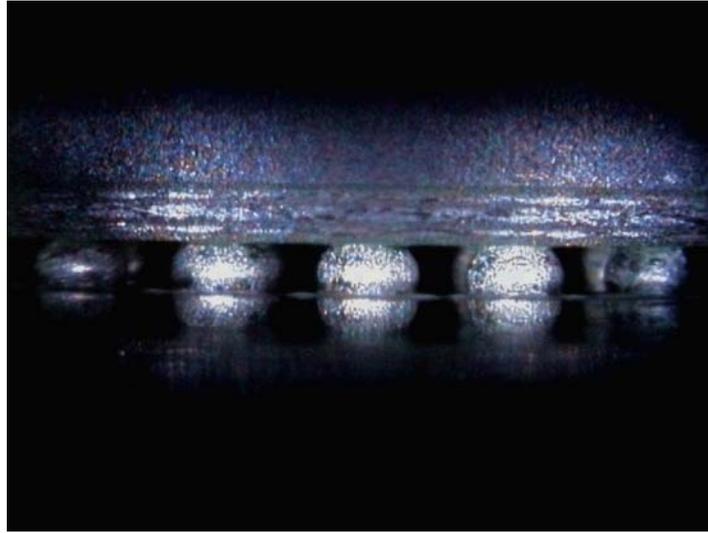
In the photos samples above, trained technicians and computer camera recognition equipment are used for inspection of the X-ray results, looking for voids, shorts, missing connections, contaminants, alignment or other gross failure mechanisms. For example, in Figure 19-19, the BGA ball connections appear to be squashed downward, with mild distortion, insuring that proper oven profile temperatures were achieved.

These technologies help in the successful placement and long term use of BGAs in the industry's latest products. Further advancements have been made in material content to conform to environmental issues, toxic materials and recycling. Another issue that relates to board design is the physical silkscreen logos and information related to recycling, lead content and other hazardous waste components, strict adherence must be paid to these requirements. Although a documentation and silkscreen issue, it can become a challenge to fit this information on the board in some cases due to component population and must be accounted for in overall board real estate.

PCB Design Support

Lattice provides a collection of PCB design resources at www.latticesemi.com/support/pcbdesignsupport.cfm including schematic libraries, PCB CAM viewers, technical notes, and BGA breakout and routing examples.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)

              +1-503-268-8001 (Outside North America)

e-mail: techsupport@latticesemi.com

Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
January 2005	01.0	Initial release.
November 2005	01.1	Figures updated.
June 2006	01.2	Removed NSMD content.
September 2006	01.3	Added note to BGA Board Layout Recommendations table.
		Reformatted BGA Package Types section in tabular format.
March 2008	01.4	Revised recommended Solder Mask Defined and Non Solder Mask Defined PCB solder pad dimensions to match industry standards.
May 2009	01.5	Updated BGA Board Layout Recommendations table and BGA Package Types table for 0.4 mm pitch ucBGA package.
February 2010	01.6	Edits to most sections and additional links and graphics added for each example.
March 2010	01.7	Replaced Lattice BGA Naming Conventions table with Lattice Semiconductor BGA Package Types table and SMD/NSMD Pad Recommendations table.

Introduction

Like all SRAM FPGAs the LatticeECP™ and LatticeEC™ devices need to be configured at power-up. This configuration can be done via:

1. Serial Peripheral Interface (SPI) boot memory
2. Traditional FPGA boot memory
3. JTAG
4. Microprocessor interface

If a boot memory is desired the SPI approach provides a number of advantages over traditional FPGA boot memory:

1. SPI devices are available from multiple vendors ensuring stable supply
2. The cost of SPI memory is up to 75% less than traditional FPGA boot memory
3. SPI memory is available in space saving 8-pin packages that are considerably smaller than packages used for traditional FPGA boot memory

Like all boot memories, SPI Serial Flash needs to be loaded with the FPGA configuration data. There are three options for programming an SPI memory used in conjunction with a LatticeECP/EC device. The SPI memory can be configured off-board using a stand-alone programmer, the memory can be programmed on-board using its SPI interface, or the memory can be programmed on-board via JTAG through the LatticeECP/EC device.

This technical note details the on-board configuration of SPI memory via the JTAG interface.

Related Documents

The following documents are available for download from the Lattice web site at www.latticesemi.com.

- *LatticeECP & EC – Low-Cost FPGA Configuration via Industry-Standard SPI Serial Flash*
- *LatticeECP/EC Family Handbook*
- Lattice technical note TN1053, *LatticeECP/EC sysCONFIG™ Usage Guide*
- *ispDOWNLOAD® Cable Data Sheet*

Hardware and Software Requirements

- An ispDOWNLOAD Cable, either USB or Parallel. Refer to the *ispDOWNLOAD Cable Data Sheet* for part numbers
- Properly installed ispLEVER® 4.2 or later
- Properly installed ispVM® System 14.3 or later

SPI/SPIX Differences

The majority of SPI Serial Flash on the market support a common read Operation Code (Op Code). LatticeECP/EC devices offer direct connection to these devices by hardwiring the read Op Code (03H) into the FPGA silicon. These devices are sometimes referred to as SPI3 devices because they support this common Read Op Code.

SPIX mode allows the LatticeECP/EC to easily interface to SPI Serial Flash devices that support a different read Op Code. This can be done with pull-up and pull-down resistors on the PCB wired to the SPID[7:0] pins, telling the

FPGA which Read Op Code that particular Flash device supports. If the configuration pins CFG2, CFG1, and CFG0 are 0, 0, 1 (respectively) at power-up the FPGA will use the Read Op Code hardwired on the PCB to access the Flash.

Table 21-1. Device Configuration Codes

CFG2	CFG1	CFG0	Configuration Mode
0	0	0	SPI Serial Flash
0	0	1	SPIX Serial Flash
1	0	0	Master Serial
1	0	1	Slave Serial
1	1	0	Master Parallel
1	1	1	Slave Parallel
X	X	X	ispJTAG™ (always available)

Note: The configuration mode pins indicate the type of device the FPGA will configure from at power-up. For SPI Serial Flash the mode pins should be set to '000' or '001'.

Table 21-2. Manufacturers of "SPI3" Compatible Flash

Manufacturer	Device Family
ST Microelectronics	M25P
NexFLASH	NX25P
Silicon Storage Technology	SST25VF
Saifun	SA25F
Spansion	S25FL
PMC	Pm25LV
Atmel	AT25F

Note: This is not meant to be an exhaustive list of manufacturers or device families.

SPI Serial Flash Sizing

As depicted in Table 21-3, the density of the FPGA determines the size requirement for the SPI Serial Flash. Smaller Flash sizes can be realized by using the compression option in ispLEVER.

Table 21-3. Selecting Flash Density

| Family	Device	Max Config Bits (Mb)	Required Boot Memory (Mb)	
			No Comp	Comp (typ)
LatticeECP/EC	EC1	0.6	1	25%
	EC3	1.1	2	25%
	ECP6/EC6	1.8	2	25%
	ECP10/EC10	3.1	4	25%
	ECP15/EC15	4.3	8	25%
	ECP20/EC20	5.3	8	25%
	ECP33/EC33	7.9	8	25%

Hardware

This section describes how to physically wire the SPI Serial Flash into a design.

SPI Serial Flash Interface

The standard pin-out for 8-pin SPI Serial Flash memories is shown below (top view):

Figure 21-1. 8-Pin SPI Flash Memory, Standard Pinout
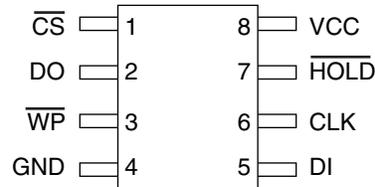
Note: V_{CCIO} for the bank that drives the signals to the SPI Serial Flash must match the SPI Serial Flash V_{CC} (today that voltage is 3.3V but this will change over time as Flash chip geometries decrease). For all LatticeECP/EC packages these signals are located in bank 3.

The SPI interface is a 4-wire serial interface comprised of the following signals:

1. **CS – Chip Select, Input:** Enables and disables device operation. When high the device is at standby power levels and the output is tri-stated. When low the device powers up and instructions can be written to and data read from the device.
2. **CLK – Serial Clock, Input**: Provides timing for the interface. The Serial Data Input (DI) is latched on the rising edge of CLK. Serial Data Output (DO) changes after the falling edge of CLK.
3. **DI – Serial Data In, Input**: When the device is enabled (CS is low) this pin allows instructions, addresses, and data to be serially written to the device. Data is latched on the rising edge of the CLK.
4. **DO – Serial Data Out, Output**: When the device is enabled (CS is low) this pin allows data and status to be serially read from the device. Data is shifted out on the falling edge of the CLK.

The SPI interface also supports the following two functions:

1. **HOLD – Input**: Allows device to be paused without de-selecting it. When HOLD is low DO will be tri-stated while DI and CLK are ignored. This function is useful when multiple devices are sharing the same SPI signals.
2. **WP – Write Protection, Input**: Used to prevent inadvertent writing of the Status Register Block Protect bits.

Note: The LatticeECP/EC SPI interface supports the basic 4-wire interface, but the user is free to implement these additional functions if desired.

ispJTAG Interface

The ispJTAG interface supports both IEEE 1149.1 Boundary Scan and IEEE 1532 In-System Configuration. Standard pinouts for 1x10, 1x8, and 2x5 download headers are shown in Table 21-4. The 1x10 header is preferred but ultimately the header chosen will depend on the available download cable. All new download cables have uncommitted "flywire" connections, so they can be attached to any of the header styles. Direction, in Table 21-4, refers to the cable, for example "output" indicates an output from the cable to the FPGA.

Table 21-4. Download Header Pinout

Pin Name	1x10	1x8	2x5	Direction	Description
V$_{CCJ}$	1	1	6	—	3.3V or 2.5V
TDO	2	2	7	Input	Test Data Out
TDI	3	3	5	Output	Test Data In
PROGRAMN	4	4	10	Output	Forces FPGA config, N/C
TRST	5	5	9	Output	Test Reset, N/C
TMS	6	6	3	Output	Test Mode Select
GND	7	7	2, 4, 8	—	Ground
TCK	8	8	1	Output	Test Clock
DONE	9			Input	FPGA configuration complete, optional
INITN	10			Input	FPGA ready for configuration, optional

1. **VCCJ** – V$_{CC}$ JTAG: Powers the ispDOWNLOAD Cable.
2. **TDO** – Test Data Out: Serial data read from the test device to the cable.
3. **TDI** – Test Data In: Serial data written from the cable to the device.
4. **PROGRAMN**: Initiates a configuration sequence when asserted low. Not used and should not be connected on the board.
5. **TRST** – Test Reset: Not used and should not be connected on the board.
6. **TMS** – Test Mode Select: Controls the IEEE 1149.1 state machine.
7. **TCK** – Test Clock: Clocks the IEEE 1149.1 state machine.
8. **GND**: Digital ground.
9. **DONE** – Optional, USB Only: Open drain, internal pull-up. A high indicates that the FPGA configuration sequence 9 was completed successfully.
10. **INITN** – Optional, USB Only: Open drain, internal pull-up. A high indicates that the FPGA is ready to be configured. A low indicates the FPGA is not ready to be configured, or an error occurred during configuration.

Note: Use of the DONE and INITN pins, while optional, does allow ispVM System to check that configuration completed successfully. If DONE or INITN are wired to the connector then the proper dialog box(es) must be checked in the Cable and I/O Port Setup section of ispVM System. See the Software section of this document for more details.

Schematic

The schematic in Figure 21-2 illustrates how to wire the ispJTAG connector, FPGA, and SPI Serial Flash.

Figure 21-2. Hardware Schematic

- The download header has standard 0.1 inch pin-to-pin spacing.

- The 4.7K pull-down resistors prevent spurious clock pulses during V_{CC} ramp-up. Place the resistors close to their clock line to keep the stub length as short as possible.

- The CCLK frequency can be as high as 50MHz, so keep this trace fairly short.

- V_{CCIO} for the bank that drives the signals to the SPI Serial Flash must match the SPI Serial Flash V_{CC} (today that voltage is 3.3V but this will change over time as Flash chip geometries decrease). For all packages these signals are located in bank 3.

- During configuration CCLK drives the SPI Serial Flash CLK pin, but once the FPGA completes configuration CCLK goes into tri-state. The CCLK pin is not accessible by user code so CCLK needs to be wired to a nearby General Purpose I/O pin (GPIO) to allow the FPGA fabric to supply a clock to the SPI Serial Flash. This pin is part of the Soft SPI Interface and is unique to each package. If the user embeds the Soft SPI Interface into their code then this pin, along with the other pins wired to the SPI Serial Flash, must be locked using the Pre-Map Preference Editor in ispLEVER. A complete list of these pins is found in Table 21-5.

- In addition to standard decoupling practices, place a decoupling capacitor close to the connector's V_{CCJ} pin. Any standard ceramic capacitor value may be used, for example 0.1 µF, 0.01 µF, etc.

Table 21-5. Pin Locations for Wiring ECP/EC to SPI Serial Flash[1]

LatticeECP/ EC Device	FPGA Pin Function	SPI Signal	672 fpBGA	484 fpBGA	256 fpBGA	208 PQFP	144 TQFP	100 TQFP
ECP33 EC33	GPIO	CLK	AB26	Y21				
	CCLK	CLK	V20	T20				
	CSSPIN	/CS	Y25	V21				
	SISPI	DI	W25	U21				
	SPIDO	DO	W26	V22				
ECP20 EC20	GPIO	CLK	AB26	Y21				
	CCLK	CLK	V20	T20				
	CSSPIN	/CS	Y25	V21				
	SISPI	DI	W25	U21				
	SPIDO	DO	W26	V22				
ECP15 EC15	GPIO	CLK		Y21	M13			
	CCLK	CLK		T20	L15			
	CSSPIN	/CS		V21	M16			
	SISPI	DI		U21	K16			
	SPIDO	DO		V22	J16			
ECP10 EC10	GPIO	CLK		Y21	M13	113		
	CCLK	CLK		T20	L15	130		
	CSSPIN	/CS		V21	M16	121		
	SISPI	DI		U21	K16	123		
	SPIDO	DO		V22	J16	124		
ECP6 EC6	GPIO	CLK		Y21	M13	113	77	
	CCLK	CLK		T20	L15	130	94	
	CSSPIN	/CS		V21	M16	121	85	
	SISPI	DI		U21	K16	123	87	
	SPIDO	DO		V22	J16	124	88	
EC3	GPIO	CLK			M13	113	77	52
	CCLK	CLK			L15	130	94	66
	CSSPIN	/CS			M16	121	85	57
	SISPI	DI			K16	123	87	59
	SPIDO	DO			J16	124	88	60
EC1	GPIO	CLK				113	77	52
	CCLK	CLK				130	94	66
	CSSPIN	/CS				121	85	57
	SISPI	DI				123	87	59
	SPIDO	DO				124	88	60

1. SPI is a four-wire interface; this table shows these wires plus the GPIO pin that needs to be wired to CCLK per Figure 21-2.

Software

The LatticeECP/EC FPGAs allow programming of the SPI Serial Flash via the ispJTAG port by using a small piece of code to redirect the ispJTAG 4-wire interface to the SPI Flash 4-wire interface.

With the Soft SPI Interface installed, ispJTAG is free to read or write the SPI Serial Flash while the FPGA is executing user code. This allows the user to perform functions such as background configuration updates.

Programming Procedure

In order to program SPI Serial Flash via ispJTAG the FPGA must contain the Soft SPI Interface. Programming the SPI Serial Flash with ispVM System and an ispDOWNLOAD cable makes this transparent to the user. The software simply loads a default Soft SPI Interface bitstream into the FPGA and then loads the user bitstream into the Flash. Once programing of the SPI Serial Flash is complete the FPGA configures itself by reading the Flash. Again, software makes all of this transparent to the user so that it is no different than programming any other serial boot device.

The following instructions describe the process of selecting the FPGA, selecting the SPI Serial Flash, and programming the SPI Serial Flash. The following screen shots are from ispVM System 15.0.

1. Connect the ispDOWNLOAD Cable to the appropriate header and apply power to the board.
2. Start the ispVM System software.
3. From the main window click on the **Scan** button located on the toolbar. The LatticeECP/EC device should be detected automatically (if it's not detected, check the ispJTAG connections and make sure the board is powered up). The resulting screen should be similar to Figure 21-3.

Figure 21-3. Main Window, Scan Complete
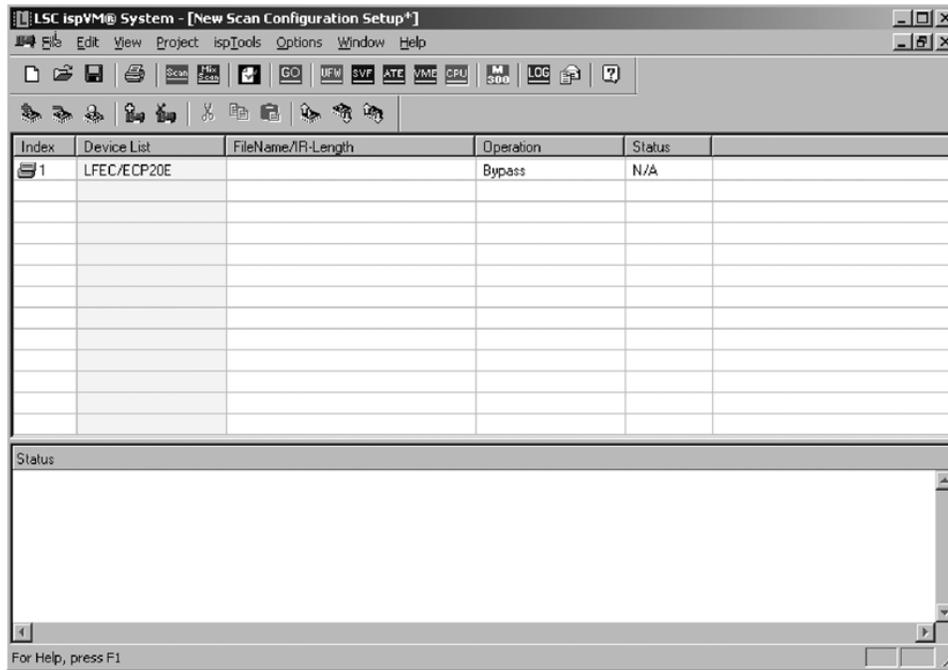
4. Double-click the number in the **Index** column and select the appropriate device to open the Device Information window, shown in Figure 21-4.

Figure 21-4. Device Information Window
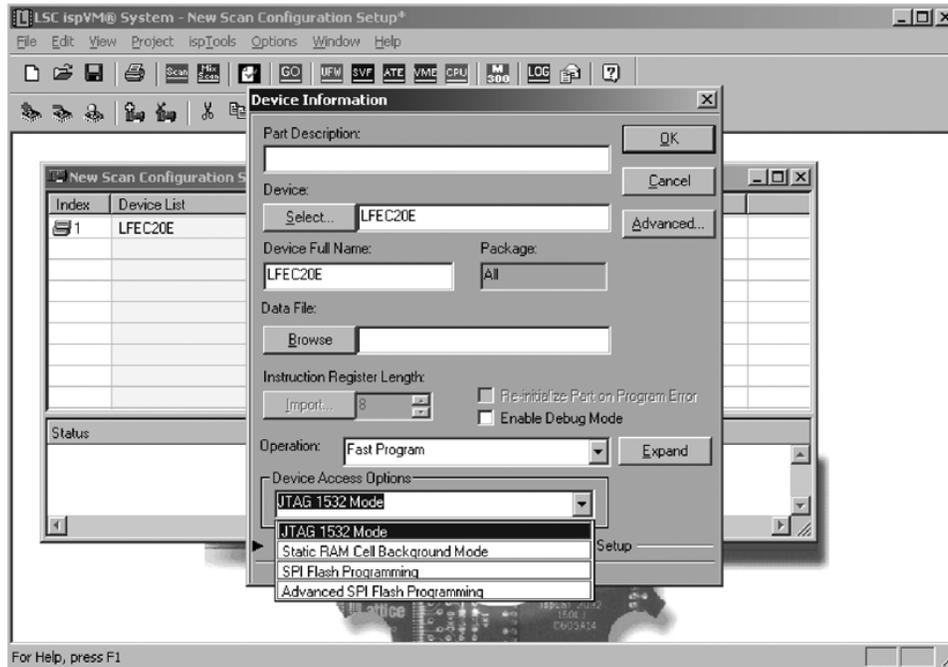
5. Under **Device Access Options**, select either **SPI Flash Programming** or **Advanced SPI Flash Programming**. To use the default Soft IP and program the SPI Serial Flash with a single LatticeECP/EC bitstream, select **SPI Flash Programming**. If the Soft IP was instantiated into the your design, or if you need to merge multiple bitstreams into a single SPI Serial Flash device, select **Advanced SPI Flash Programming**.

    a. **SPI Flash Programming**

        i. Under **Device Access Options**, select **SPI Flash Programming**. The SPI Serial Flash Device dialog shown in Figure 21-5 will be displayed.

Figure 21-5. SPI Serial Flash Device Dialog
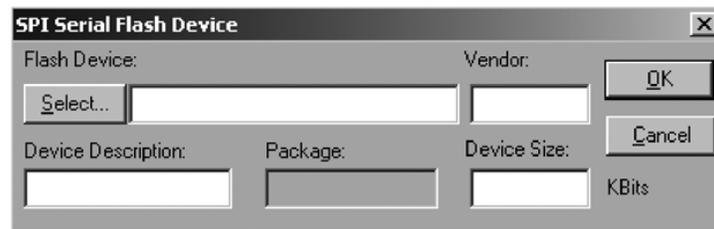
        ii. Click on the **Select** button to select the target SPI Serial Flash device. The SPI Serial Flash Select Device dialog shown in Figure 21-6 will be displayed.

Figure 21-6. Select SPI Serial Flash Device Dialog
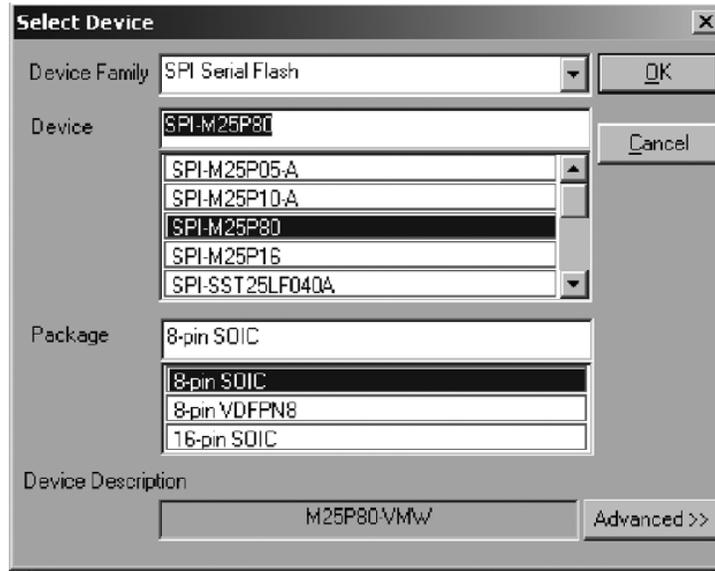
    iii. Select the target SPI Serial Flash device and click the **OK** button. The SPI Serial Flash Device dialog shown in Figure 21-7 will be displayed.

Figure 21-7. SPI Serial Flash Device Dialog
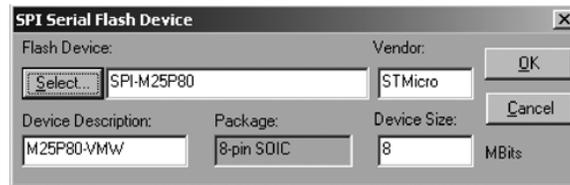
    iv. Click the **OK** button. The Device Information dialog shown in Figure 21-8 will be displayed.

*Figure 21-8. SPI Serial Flash Device Information Dialog*
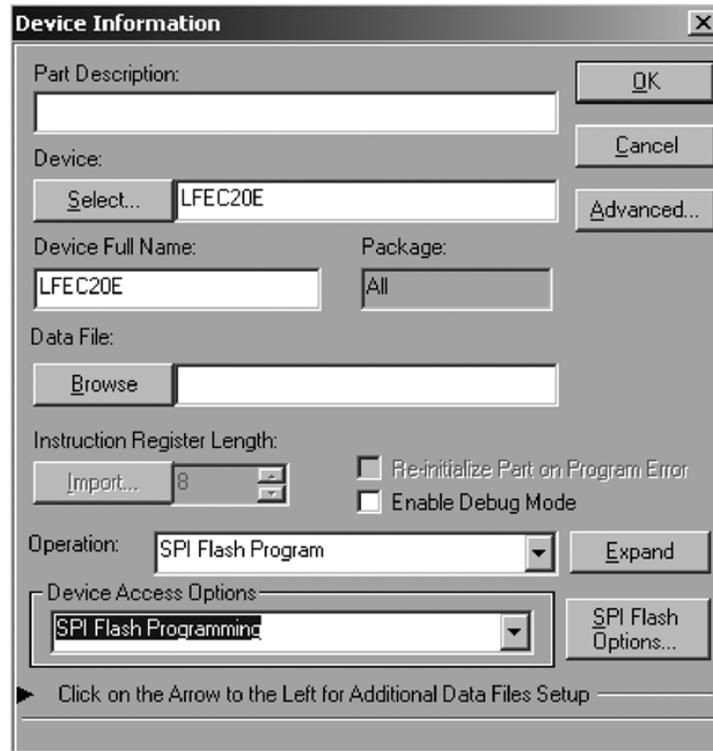
v. Under **Data File**, select the LatticeECP/EC bitstream to be programmed into the SPI Serial
    Flash device. Click the **OK** button. This will return you to the main ispVM window, shown in
    Figure 21-9. Proceed to step 6.

Figure 21-9. Main Window, SPI Programming
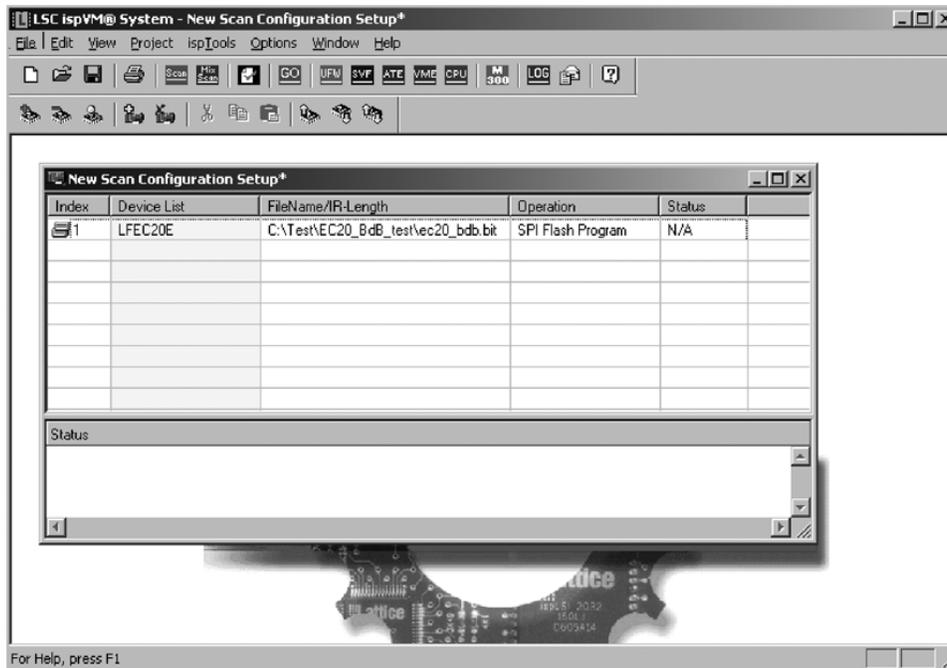
b. **Advanced SPI Flash Programming**

   i. Under **Device Access Options**, select **Advanced SPI FLASH Programming**, as shown in Figure 21-4.

   ii. The FPGA Loader Setup Dialog will be launched.

   iii. Click on **CPLD or FPGA Device**. It should look similar to Figure 21-10.

   iv. The default Soft IP will automatically be added as the **FPGA Loader Application Specific Data File**. If you instantiated the Soft IP into your own design, click on the **Browse** button to select your bitstream. Clicking on the **Default** button will reload the default Soft IP bitstream.

   v. Under **Operation**, select **Fast Program**.

Figure 21-10. Select FPGA Device
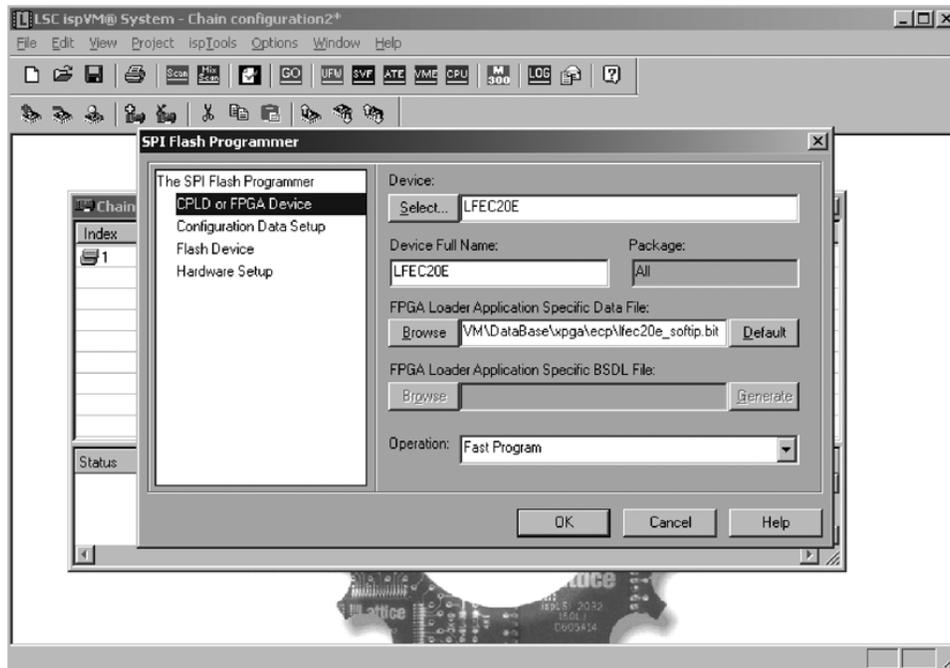
   vi. Click on **Configuration Data Setup** (see Figure 21-11).

   vii. Under **Configuration Data File**, click on **Browse** to locate the user configuration file created using ispLEVER. Double-click on the file name. Use the **Merging Multiple Configuration Data Files** option if you want to merge multiple bitstreams into the SPI Serial Flash device. Multiple bitstreams would be used to configure multiple FPGAs using one SPI Serial Flash.

Figure 21-11. Configuration Data Setup
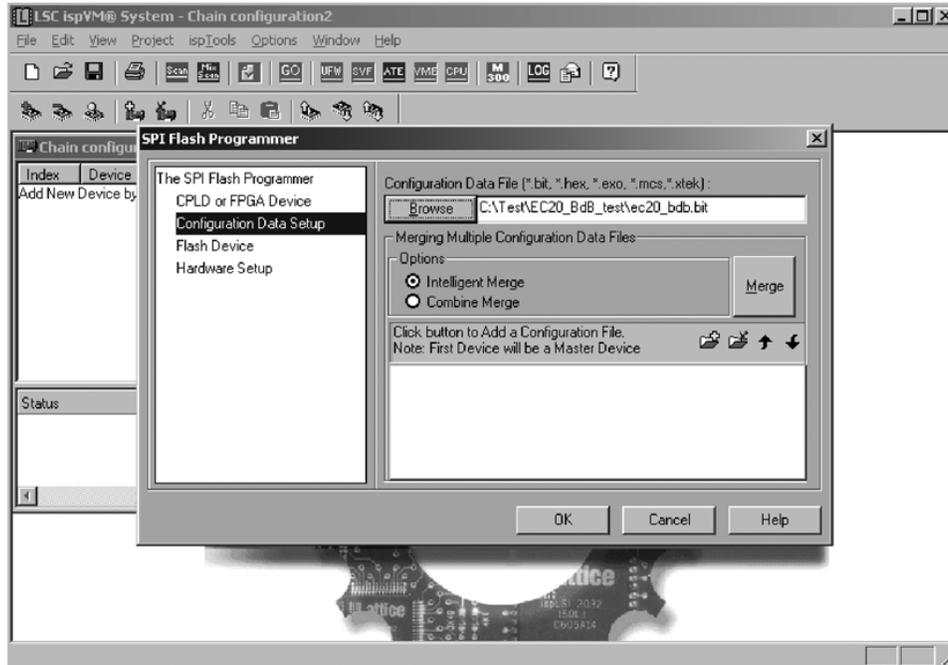
viii.Select **Flash Device** in the left window (see Figure 21-12).

ix. Under **Flash Device**, click **Select** to choose the desired device; in this case an ST Micro device was selected.

x. From the drop-down list under **Operation** select **Erase, Program, Verify**.

Figure 21-12. Select Flash Device

xi. Click **OK** to exit the FPGA Loader window. This will return you to the main ispVM window, shown in Figure 21-13.

Figure 21-13. Main Window, Advanced SPI Flash Programming

6. From the main project window, on the menu bar, click **Options > Cable** and **I/O Port Setup**.
7. Check that the proper cable type is selected (parallel or USB) and that the PROG, DONE, and INIT boxes (Figure 21-14) are properly selected/de-selected based on how the ispJTAG connector is wired up (see the Hardware Schematic section of this document).

Figure 21-14. Cable and I/O Port Setup

8. Click on **OK**.

9. From the main project window click the green **GO** button on the toolbar; this will begin the download process.

10. Upon successful download, in order to configure the FPGA with the new configuration, the user must cycle power to the FPGA or pulse the FPGA's Program pin low then high.

Including the SPI Interface in the FPGA Design

If the user wishes JTAG to have access to the SPI Serial Flash while the FPGA is operating, for instance to allow background configuration updates, then the Soft SPI Interface must be instantiated into the user code. Once a configuration bitstream containing the user code and the Soft SPI Interface has been created the programming sequences will be identical to those detailed above (see step 5).

Sample Code

The following code samples are simple VHDL and Verilog files that show how to instantiate the netlist file, i.e. the Soft SPI Interface. The netlist file should be placed in the same directory as the top design file. In the following examples the netlist file is called SPITOP.ngo. The netlist file, along with these sample source files, is freely available from the Lattice Semiconductor web site at www.latticesemi.com.

VHDL

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity design_top is
    port ( rst    : in std_logic;
           sclk   : in std_logic;
           cnt_out : out std_logic_vector(7 downto 0);

  •
  •
  •

        -- SPI Serial Flash pins

         SPI_C  : out std_logic;  -- clock
         SPI_D  : out std_logic;  -- data input
         SPI_SN : out std_logic;  -- chip select
         SPI_Q  : in  std_logic   -- data output
        );
end;

architecture behave of design_top is

--Instantiate the file

    component SPITOP
        port (
                SPI_PIN_C  : out std_logic;
                SPI_PIN_D  : out std_logic;
                SPI_PIN_SN : out std_logic;
                SPI_PIN_Q  : in  std_logic
                );
    end component;

--User code

    signal cnt: std_logic_vector(7 downto 0);

begin
    process(sclk, rst)
    begin
        if rst = '1' then
          cnt <= (others => '0');
     elsif rising_edge(sclk) then
            cnt <= cnt + 1;
      end if;

 end process;

 cnt_out <= cnt;

  •
  •
  •


-- SPITOP port map

 spi_ip: SPITOP port map
      (
        SPI_PIN_C  =>  SPI_C,
        SPI_PIN_D  =>  SPI_D,
        SPI_PIN_SN =>  SPI_SN,
        SPI_PIN_Q  =>  SPI_Q
        );

end behave;
```

Verilog

Here is the same design in Verilog.

```verilog
module design_top(rst, sclk, cnt_out, spi_c, spi_d, spi_sn, spi_q) ;

input        sclk, rst;
output [7:0] cnt_out;

// SPI Serial Flash pins

input        spi_q;
output       spi_d, spi_sn, spi_c;

reg [7:0]    cnt_out;

// User code

always @ (posedge sclk or posedge rst)
    begin
        if (rst)
            cnt_out = 2'h00;
        else
            cnt_out = cnt_out + 1;
    end

// Instantiate Soft SPI Interface

SPITOP spi_ip
        (
        .SPI_PIN_C(spi_c),
        .SPI_PIN_D(spi_d),
        .SPI_PIN_SN(spi_sn),
        .SPI_PIN_Q(spi_q)
        );

endmodule

module SPITOP (SPI_PIN_C, SPI_PIN_D, SPI_PIN_SN, SPI_PIN_Q);

output    SPI_PIN_D, SPI_PIN_SN, SPI_PIN_C ;
input     SPI_PIN_Q;

endmodule
```

The Soft SPI Interface is only needed to connect JTAG to the SPI Serial Flash interface. If the SPI Serial Flash will be used by the user design, for instance as scratch memory, and background access via JTAG will not be required, then the Soft SPI Interface is not needed, and the user code can access the SPI pins directly. Remember that the configuration memory must start at address zero; any user defined memory space must be located above the configuration data. It is recommended that an address above the maximum possible configuration size be chosen. For instance if an ECP/EC20 device is being used, select a scratch pad starting address above 5.3Mb (see Table 21-3 in this document).

Locking the Pins

The last thing the user needs to do is tell ispLEVER which pins are connected to the SPI Serial Flash device (see Figure 21-2 and Table 21-5). From the ispLEVER Project Navigator window click on the package name in the left window, then double click on the Pre-Map Preference Editor in the right window (see Figure 21-15).

Figure 21-15. Select Pre-Map Preference Editor
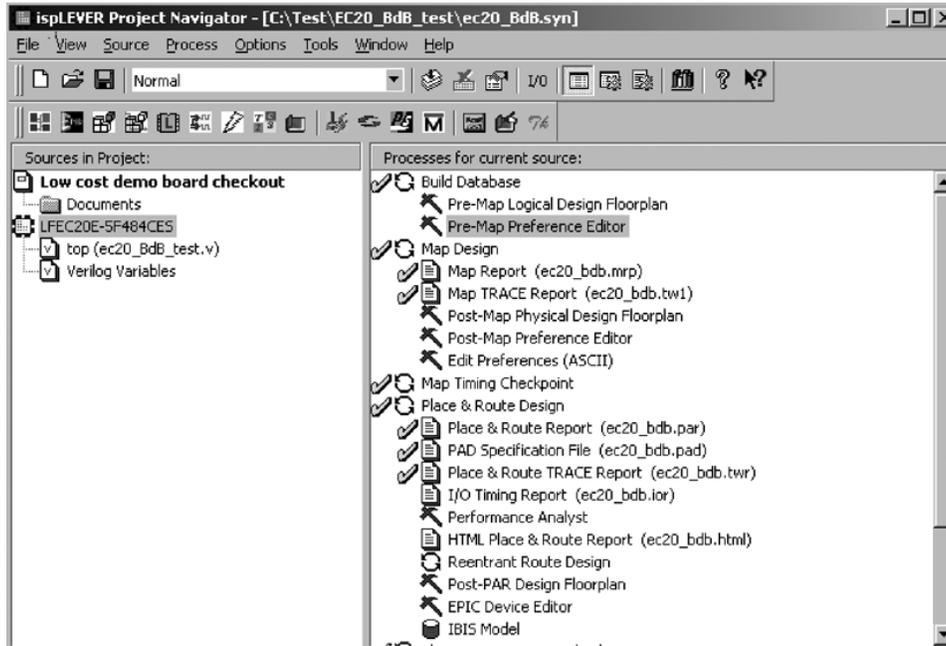
This will start the editor; it should look similar to Figure 21-16. The Pre-Map Preference Editor is where pin numbers and other attributes are assigned to the various I/O in a design. Figure 21-16 shows the proper pin selection for a 484 fpBGA, it also shows proper selection of the I/O type (in this case LVCMOS_3.3). Each package requires a different pin selection; refer to Table 21-5 and the schematic in Figure 21-2 (in the Hardware section of this document) for details.

While it is recommended that the pin listed in Table 21-5 be used as the GPIO to wire to CCLK, if the Soft IP is instantiated into the user design then any pin can be selected - with the following cautions:

1. The Default Soft IP is built to use the default pin (see step 5 above under Programming Procedures), if a different pin is chosen the Default Soft IP will not work, i.e. you must instantiate the Soft IP in your design.

2. Do not select the DOUT pin as the GPIO to wire to CCLK. DOUT is an output during configuration, as is CCLK. Selecting DOUT will cause contention.

3. Select a pin from a bank that has its V_{CCIO} connected to 3.3V.

Figure 21-16. Preference Editor
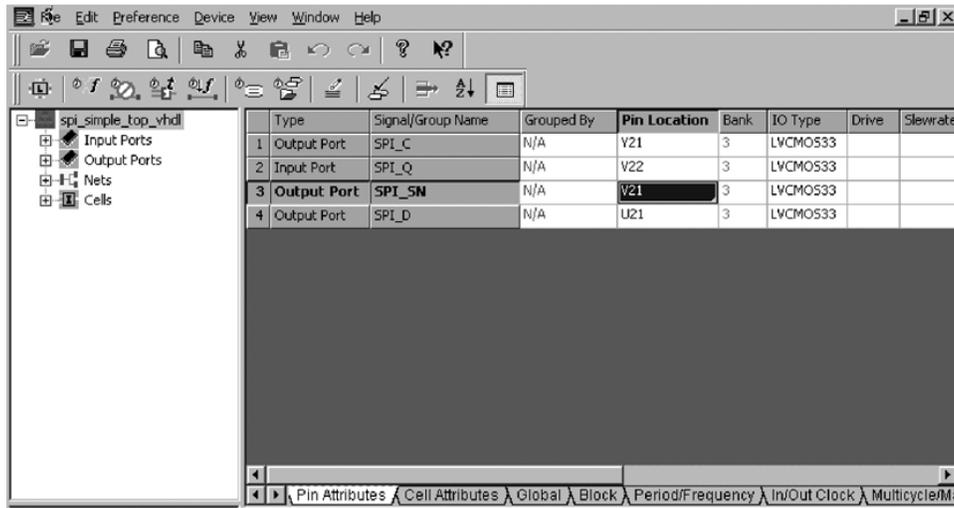
After all of the pin attributes have been entered, close the Preference Editor, scroll down the right window, and double click on Generate Bitstream Data (see Figure 21-17). Once the bitstream has been generated, go to the Programming Procedure section of this document.

Figure 21-17. Generate Bitstream Data
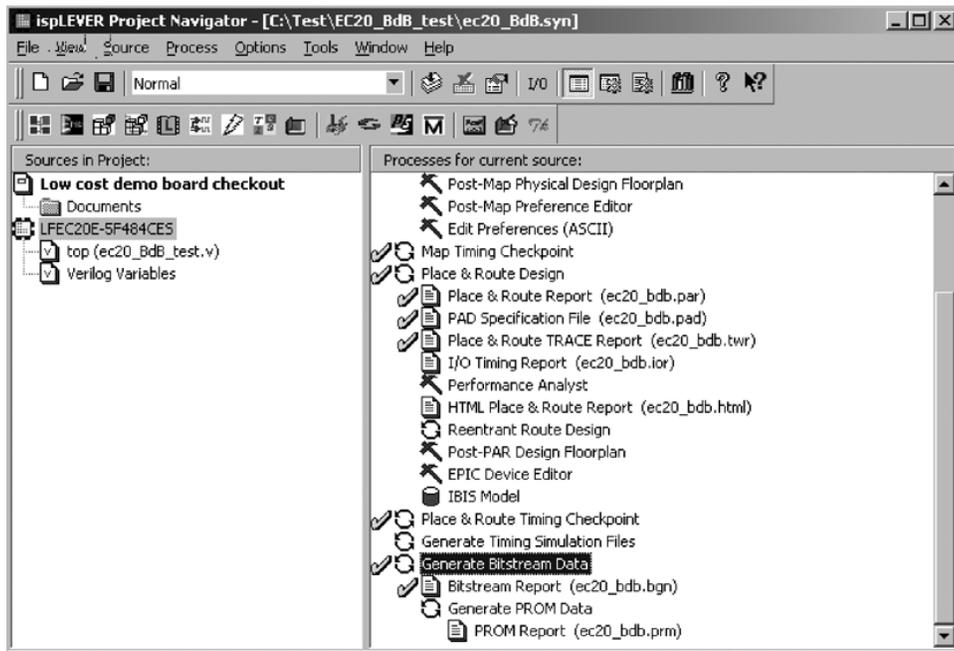
Design Notes

The following tips will help insure first pass success when using LatticeECP/EC devices with SPI Serial Flash.

The PROGRAMN pin can be left open, wired to a button, microprocessor, etc. The PROGRAMN pin should be high at power-up, do not tie this pin to a pull-down resistor. There is a weak pull-up on this pin, but if needed, add an external 10K ohm pull-up resistor.

The INITN pin can be left open, connected to a microprocessor, status register, or other LatticeECP/EC devices. Holding this pin low during configuration will keep the device from configuring. Do not tie this pin to a pull-down resistor. There is a weak pull-up on this pin but if needed add an external 10K ohm pull-up resistor. This pin can drive 8 mA. If driving an LED that requires higher current, use an external driver/buffer.

The DONE pin can be left open, connected to a microprocessor, status register, or other Lattice devices with DONE pins. If you connect this pin to other DONE pins then all of the DONE pins in the chain will need to be set to open drain (using ispLEVER or an attribute in your code) and a pull-up resistor of about 10K will need to be added. Holding this pin low during configuration will keep the LatticeECP/EC from waking up. Do not tie this pin to a pull-down resistor. There is a weak pull-up on this pin but, if needed, add an external 10K ohm pull-up resistor. This pin can drive 8 mA. If driving an LED that requires higher current, use an external driver/buffer.

All of the SPI pins are part of I/O bank 3; therefore V_{CCIO} for bank 3 must be connected to the same voltage as the SPI Serial Flash.

When utilizing SPI Serial Flash, use of the SPI pins as user pins is generally not recommended. If you must use one or more of the SPI pins as user I/O, do not change the I/O type or direction. For example, if, during configuration, the SPI pin you wish to use is an input you may only use the pin as an input, not an output, and it must be of type LVCMOS33.

If you set Config_Mode in the ispLEVER Preference Editor to SPI3, and you are instantiating the Soft IP, you will get warnings when you compile your code. This is due to the Soft IP requiring use of the SPI3 port as normal I/O during SPI Serial Flash programming. You can avoid these warnings by selecting None or JTAG instead of SPI3 for the Config_Mode.

If you set Config_Mode to SPI3, or you instantiate the Soft IP into your code (assigning the SPI pins in the Preference Editor or your code), the SPI pins will be protected from use by the Place and Route tools. If you set Config_Mode to None or JTAG, and you are not instantiating the Soft IP, the SPI pins will not be protected from use by the Place and Route tools. In this case, consider using the Prohibit Site "<pin number>" command on the SPI pins to keep the Place and Route tools from using these pins. This is particularly useful if your design is I/O bound and you want to allow Place and Route as much flexibility as possible. You can use the Package View in the Preference Editor to place the Prohibits.

Try to select an SPI Serial Flash that is supported by your version of ispVM programming software. Check for the latest SPI Serial Flash vendor support in the latest version of ispVM software. The ispVM software is available at no charge from the Lattice web site at www.latticesemi.com. If your SPI Serial Flash is not supported on the latest software, please contact Lattice Technical Support.

If you have selected a GPIO pin to drive the CCLK that is other than the pin recommended in Table 21-5 of this document you will need to instantiate the Soft IP into your code. See the section above entitled "Including the SPI Interface in the FPGA Design".

Conclusion

By combining the new low cost LatticeECP/EC family of devices with low cost, third party serial Flash, engineers can now take advantage of a very cost effective system solution. In addition to cost savings, the design also benefits from the space conscious 8-pin package.

This new capability, in addition to the traditional configuration methods, is fully supported by the latest Lattice tools.

For more information on Lattice's family of devices, visit our web site at www.latticesemi.com.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
             +1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Section III. LatticeECP/EC Family Handbook Revision History

LatticeECP/EC Family Handbook
Revision History

March 2010   **Data Sheet**

Revision History

Date	Handbook Revison Number	Change Summary
—	—	Previous Lattice releases.
January 2007	02.7	LatticeECP/EC Family Data Sheet updated to version 02.3.
		Technical note TN1051 updated to version 01.8.
		Technical note TN1049 updated to version 04.1.
		Technical note TN1052 updated to version 02.2.
		Technical note TN1074 updated to version 01.3.
February 2007	02.8	LatticeECP/EC Family Data Sheet updated to version 02.4.
May 2007	02.9	LatticeECP/EC Family Data Sheet updated to version 02.5.
		Technical note TN1049 updated to version 04.2.
		Technical note TN1050 updated to version 03.2.
November 2007	03.0	LatticeECP/EC Family Data Sheet updated to version 02.6.
		Technical note TN1049 updated to version 04.3.
		Technical note TN1052 updated to version 02.3.
February 2008	03.1	LatticeECP/EC Family Data Sheet updated to version 02.7.
March 2010	03.2	Technical note TN1052 updated to version 02.3.
		Technical note TN1074 updated to version 01.6.
March 2010	03.3	Technical note TN1074 updated to version 01.7.

Note: For detailed revision changes, please refer to the revision history for each document.