LatticeECP/EC Family Handbook

LatticeECP/EC Family Handbook Table of Contents

June 2004

Section I. LatticeECP/EC Family Data Sheet

Section I. LatticeECP/EC Family Data Sheet

LatticeECP/EC Family Data Sheet Introduction

June 2004 **Advance Data Sheet**

Features

- **Extensive Density and Package Options**
  - 1.5K to 41K LUT4s
  - 65 to 576 I/Os
  - Density migration supported
- **sysDSP™ Block (LatticeECP™ Versions)**
  - High performance multiply and accumulate
  - 4 to 10 blocks
    - 4 to 10 36x36 multipliers or
    - 16 to 40 18x18 multipliers or
    - 32 to 80 9x9 multipliers
- **Embedded and Distributed Memory**
  - 18 Kbits to 645 Kbits sysMEM™ Embedded Block RAM (EBR)
  - Up to 163 Kbits distributed RAM
  - Flexible memory resources:
    - Distributed and block memory
- **Flexible I/O Buffer**
  - Programmable sysIO™ 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
- **Dedicated DDR Memory Support**
  - Implements interface up to DDR333 (166MHz)
- **sysCLOCK™ PLLs**
  - Up to 4 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	LFEC40/ LFECP40
PFU/PFF Rows	12	16	24	32	40	44	64
PFU/PFF Columns	16	24	32	40	48	56	80
PFUs/PFFs	192	384	768	1280	1920	2464	5120
LUTs (K)	1.5	3.1	6.1	10.2	15.4	19.7	41.0
Distributed RAM (Kbits)	6	12	25	41	61	79	164
EBR SRAM (Kbits)	18	55	92	277	350	424	645
EBR SRAM Blocks	2	6	10	30	38	46	70
sysDSP Blocks[1]	—	—	4	5	6	7	10
18x18 Multipliers[1]	—	—	16	20	24	28	40
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	
672-ball fpBGA (27 x 27 mm)						400	496
900-ball fpBGA (31 x 31 mm)							576

1. LatticeECP devices only.

Introduction

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

The Lattice-ECP/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 from Lattice allows large complex designs to be efficiently implemented using the LatticeECP/EC family of FPGA devices. 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.

LatticeECP/EC Family Data Sheet Architecture

June 2004 Advance Data Sheet

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 sysIO 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, LatticeECP/EC Device (Top Level)

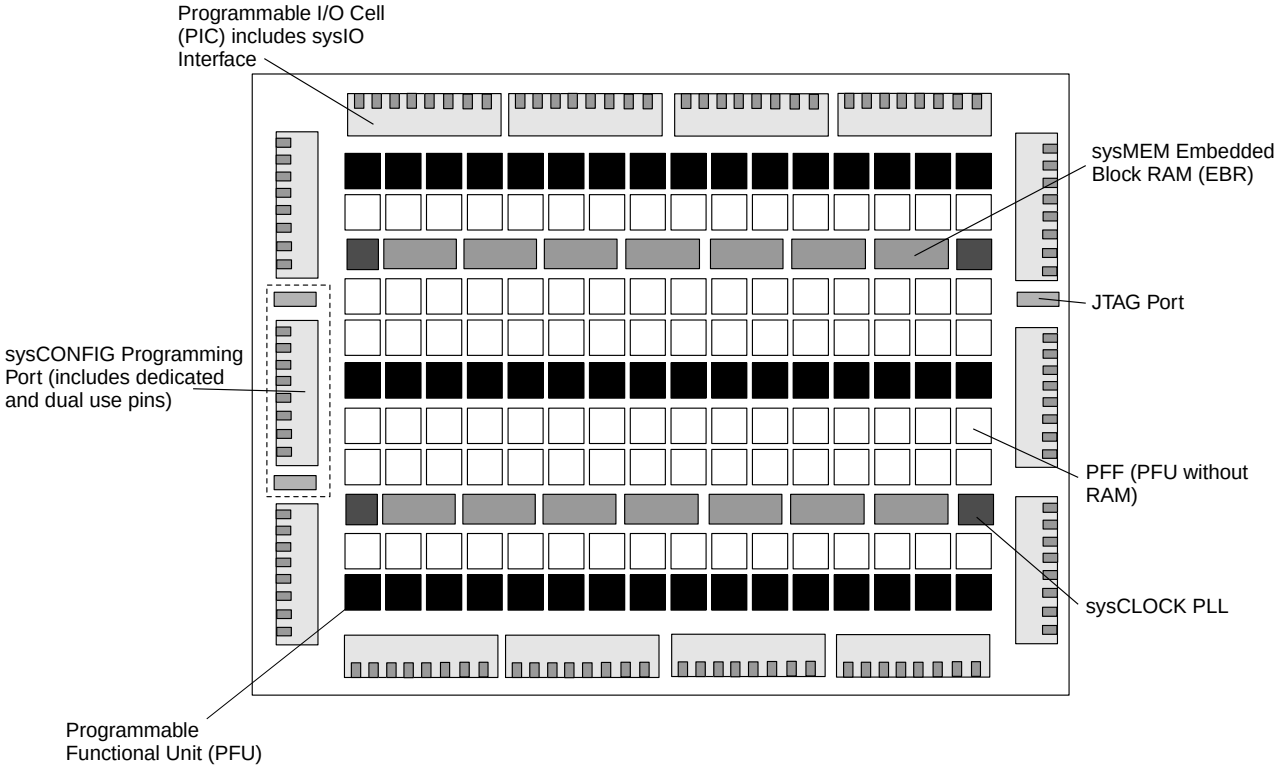
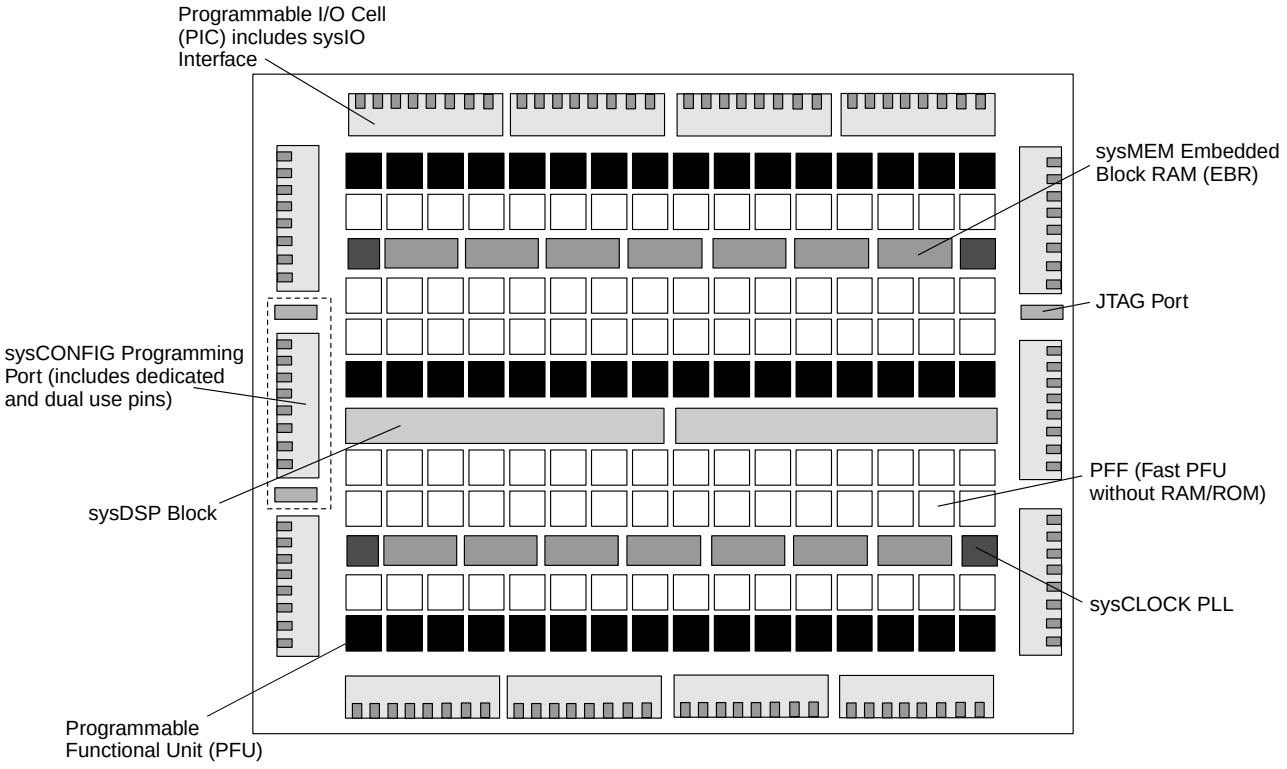


Figure 2-2. Simplified Block Diagram, LatticeECP-DSP Device (Top Level)

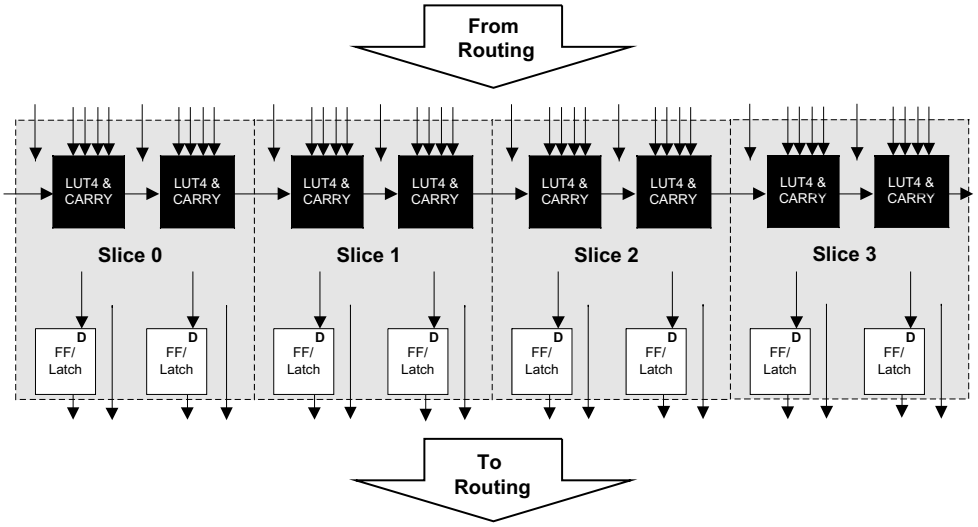


PFU and PFF Blocks

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

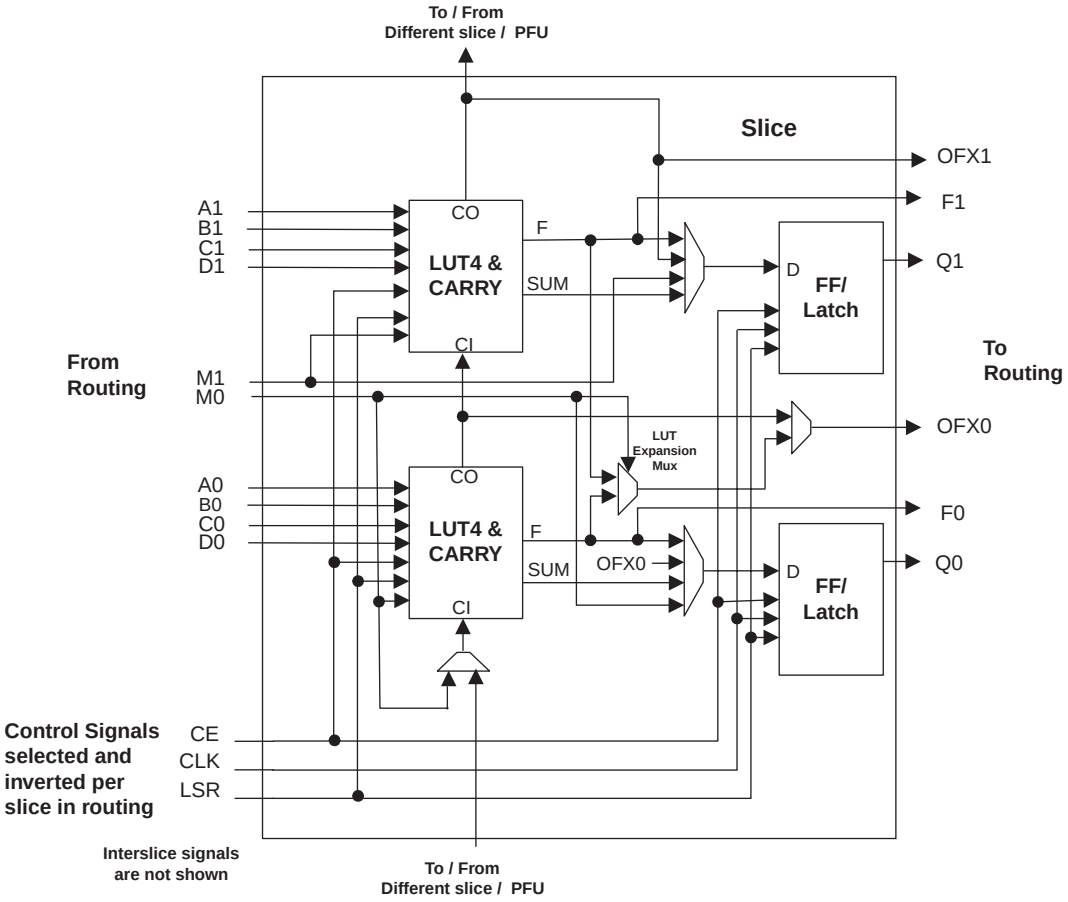


Table 2-1. Slice Signal Descriptions

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

Two additional signals: Carry Generate and Carry Propagate are generated per Slice in this mode, allowing fast arithmetic functions to be constructed by concatenating Slices.

RAM Mode: In this mode, 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.

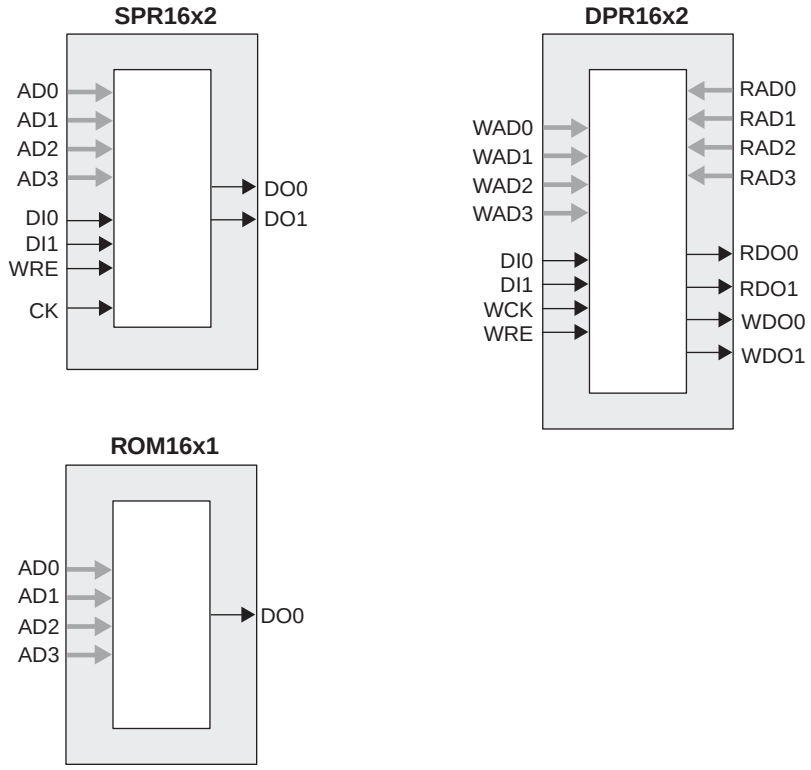
The Lattice design tools support the creation of a variety of different size memories. Where appropriate, the software will construct these using distributed memory primitives that represent the capabilities of the PFU. Table 2-3 shows the number of Slices required to implement different distributed 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 on using RAM in LatticeECP/EC devices, please see details of additional technical documentation at the end of this data sheet.

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

	SPR16x2	DPR16x2
Number of slices	1	2

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

Figure 2-5. Distributed Memory Primatives



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 allowing both short and long connections routing between PFUs.

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

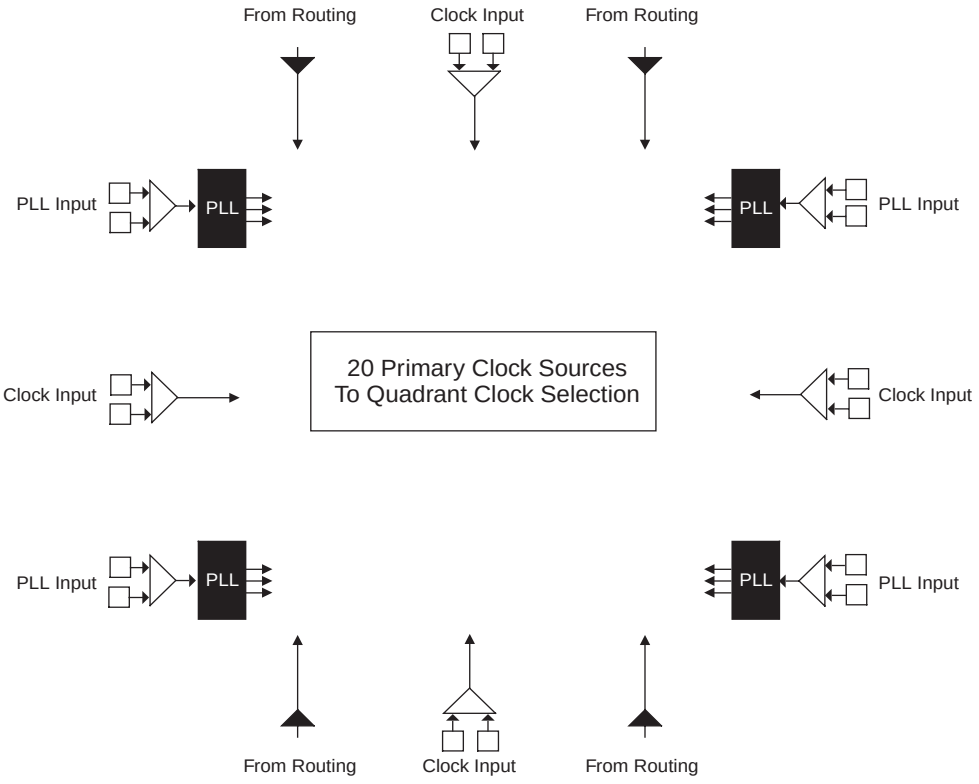
Clock Distribution Network

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. Clock Sources

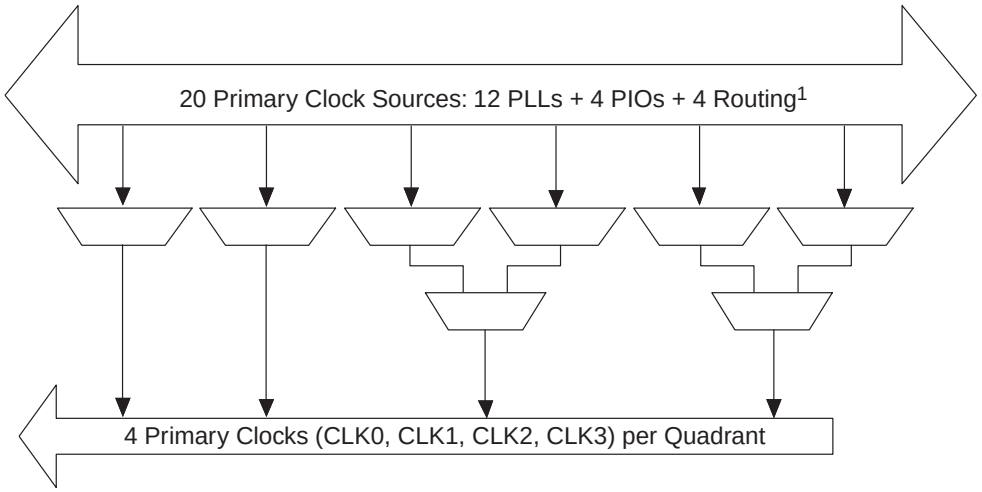


Note: Smaller devices have two PLLs.

Clock Routing

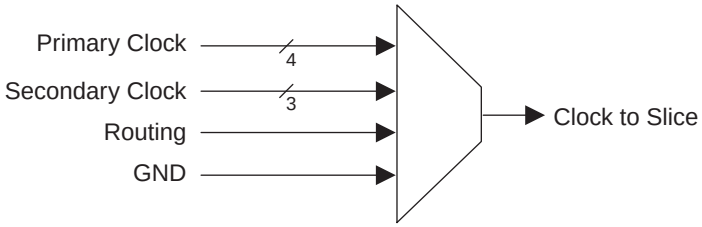
The clock routing structure in LatticeECP/EC devices consists of four Primary Clock lines and a Secondary Clock network per quadrant. The primary clocks are generated from MUXs located in each quadrant. Figure 2-7 shows this clock routing. The primary clock lines also feed into a secondary clock network (not shown). The secondary clock branches are tapped at every PFU. These secondary clock networks can also be used for controls and high fan out data. Each slice derives its clock from the primary clock lines, secondary clock lines and routing as shown in Figure 2-8.

Figure 2-7. Per Quadrant Clock Selection



1. Smaller devices have fewer PLL related lines.

Figure 2-8. Slice Clock Selection



sysCLOCK Phase Locked Loops (PLLs)

The PLL clock input, from pin or routing, feeds into an input clock divider. There are four sources of feedback signal to the feedback divider: from the clock net, from output of the post scalar divider, from the routing or from an external pin. There is a PLL_LOCK signal to indicate that VCO has locked on to the input clock signal. Figure 2-9 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 programmed 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, port 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 2-9. PLL Diagram

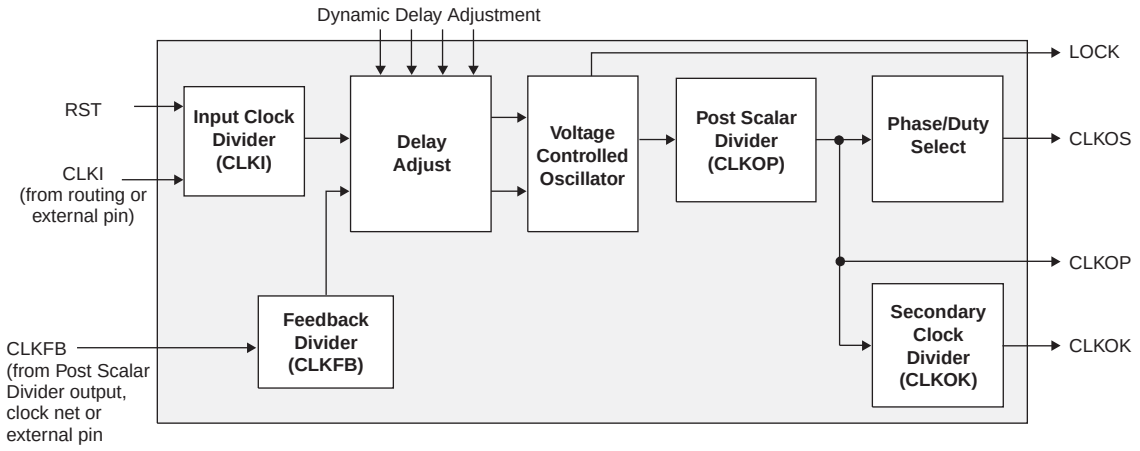


Figure 2-10 shows the available macros for the PLL. Table 2-5 provides signal description of the PLL Block.

Figure 2-10. PLL Primitive

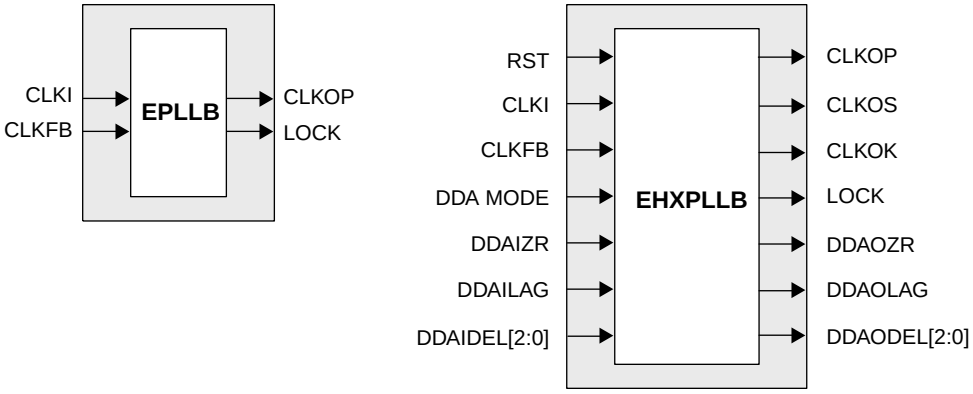


Table 2-5. PLL Signal Descriptions

Signal	I/O	Description
CLKI	I	Clock input from external pin or routing
CLKFB	I	PLL feedback input from PLL output, clocknet, routing or external pin
RST	I	“1” to reset input clock divider
CLKOS	O	PLL output clock to clock tree (phase shifted/duty cycle changed)
CLKOP	O	PLL output clock to clock tree (No phase shift)
CLKOK	O	PLL output to clock tree through secondary clock divider
LOCK	O	“1” indicates PLL LOCK to CLKI
DDAMODE	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”: Lag, “0”: Lead
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 on the PLL, please see details of additional technical documentation at the end of this data sheet.

sysMEM Memory

The LatticeECP/EC family of 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.

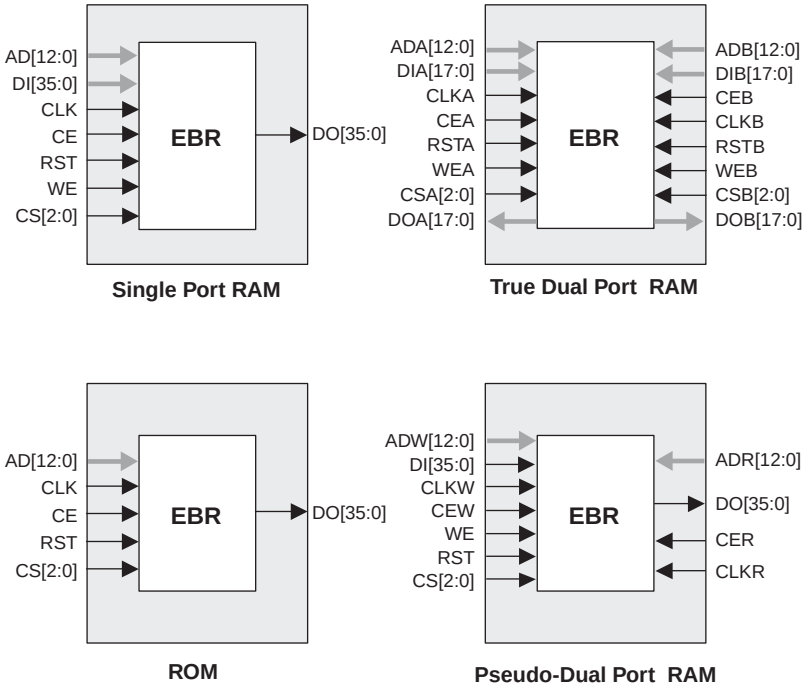
Memory Cascading

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

Single, Dual and Pseudo-Dual Port Modes

Figure 2-11 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 2-11. sysMEM EBR Primitives



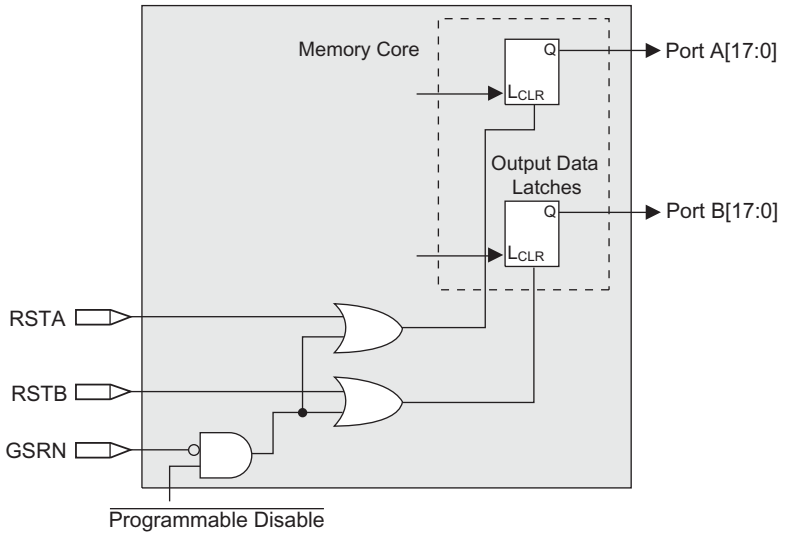
The EBR memory supports three forms of write behavior for single port or dual port operation:

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.
2. **Write Through** – a copy of the input data appears at the output of the same port, during a write cycle.
3. **Read-Before-Write** – when new data is being written, the old content of the address appears at the output.

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 2-12.

Figure 2-12. Memory Core Reset



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

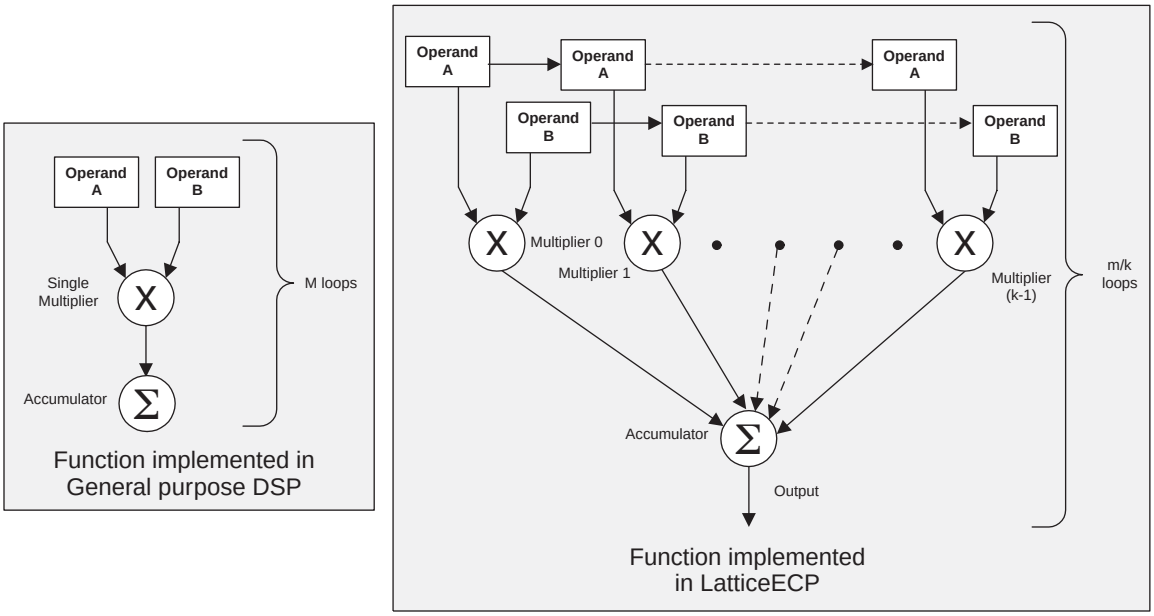
sysDSP Block

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 Compare to General DSP

Conventional general-purpose DSP chips typically contain one to four (Multiply and Accumulate) MAC units with fixed data-width multipliers; this leads to limited parallelism and limited throughput. Their throughput is increased by higher clock speeds. The 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 appropriate level of parallelism. Figure 2-13 compares the serial and the parallel implementations.

Figure 2-13. 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 in the width selected from the three available options x9, x18, and x36. A number of these elements are concatenated for highly parallel implementations of DSP functions. Table 2-1 shows the capabilities of the block.

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

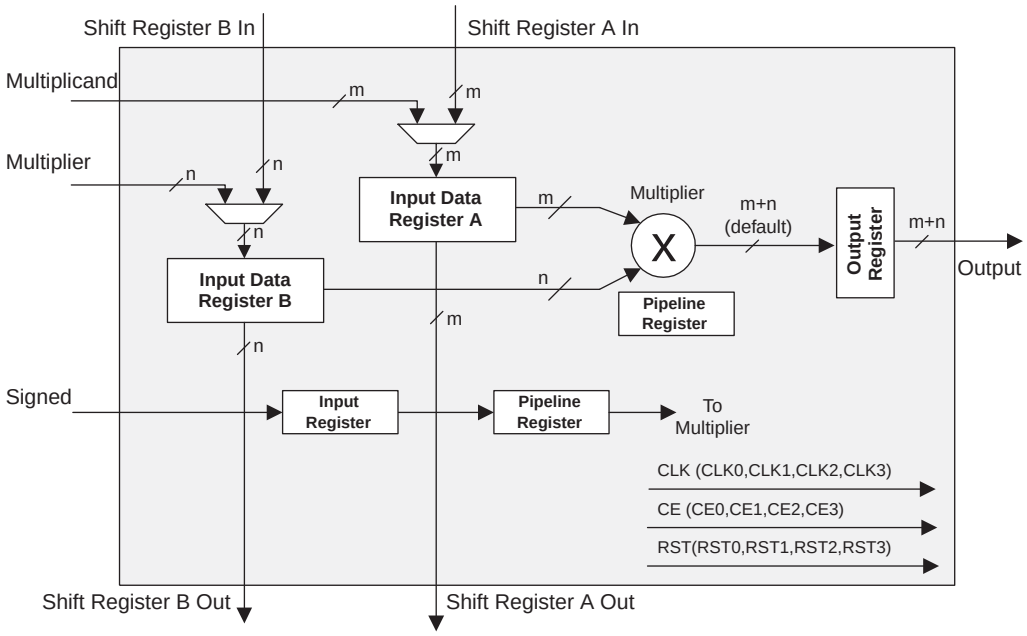
Width of Multiply	x9	x18	x36
MULT	8	4	1
MAC	2	1	—
MULTADD	4	2	—
MULTADDSUM	4	2	—

Some options are available in four elements. The input register in all the elements can be directly loaded or can be loaded as shift register from previous operand registers. 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 2-14 shows the MULT sysDSP element.

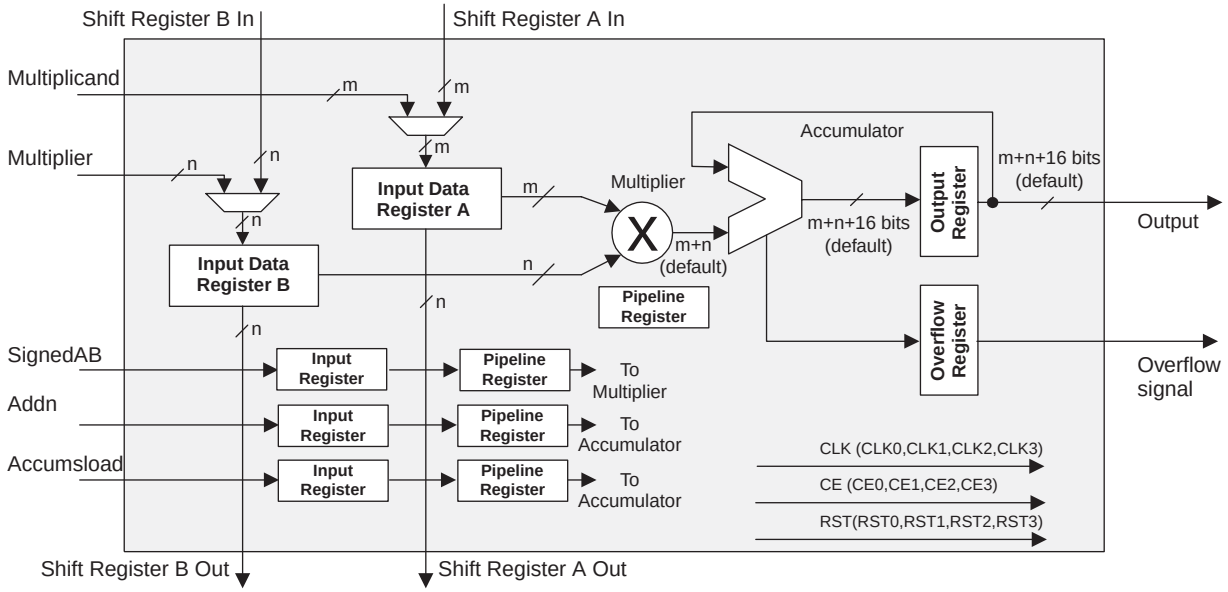
Figure 2-14. MULT sysDSP Element



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 2-15 shows the MAC sysDSP element.

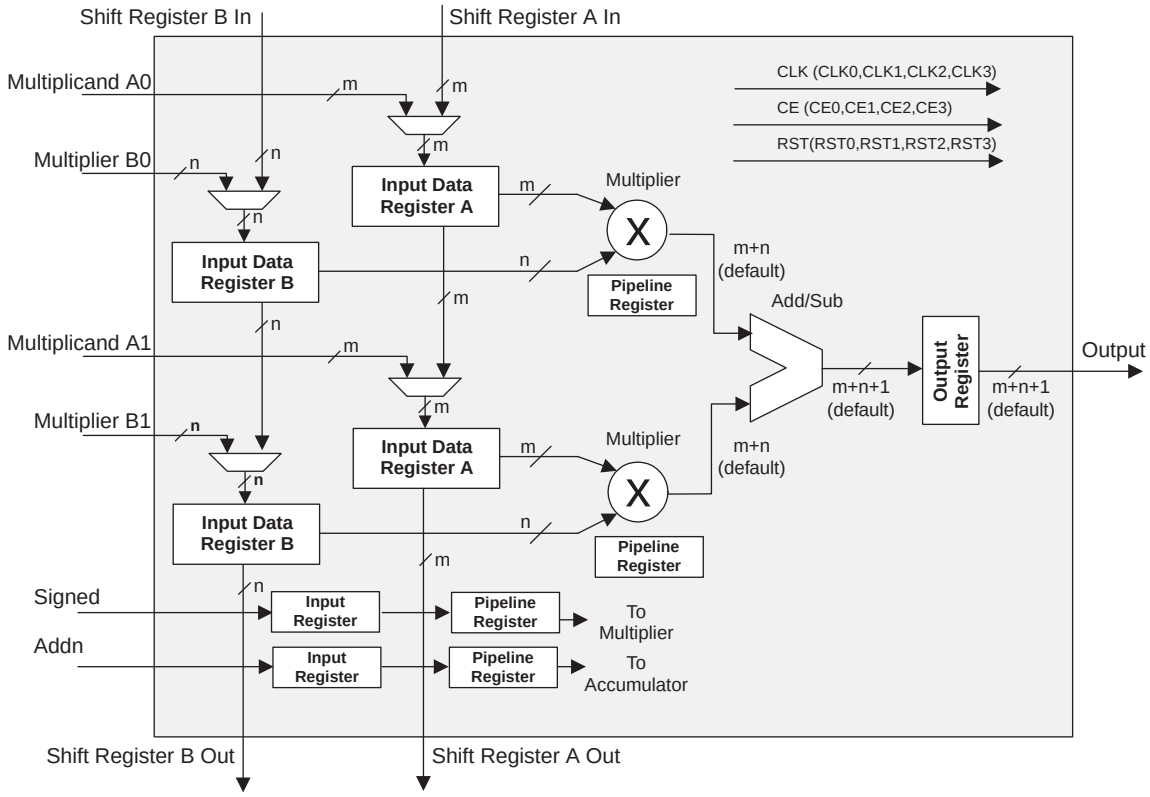
Figure 2-15. MAC sysDSP Element



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 2-16 shows the MULTADD sysDSP element.

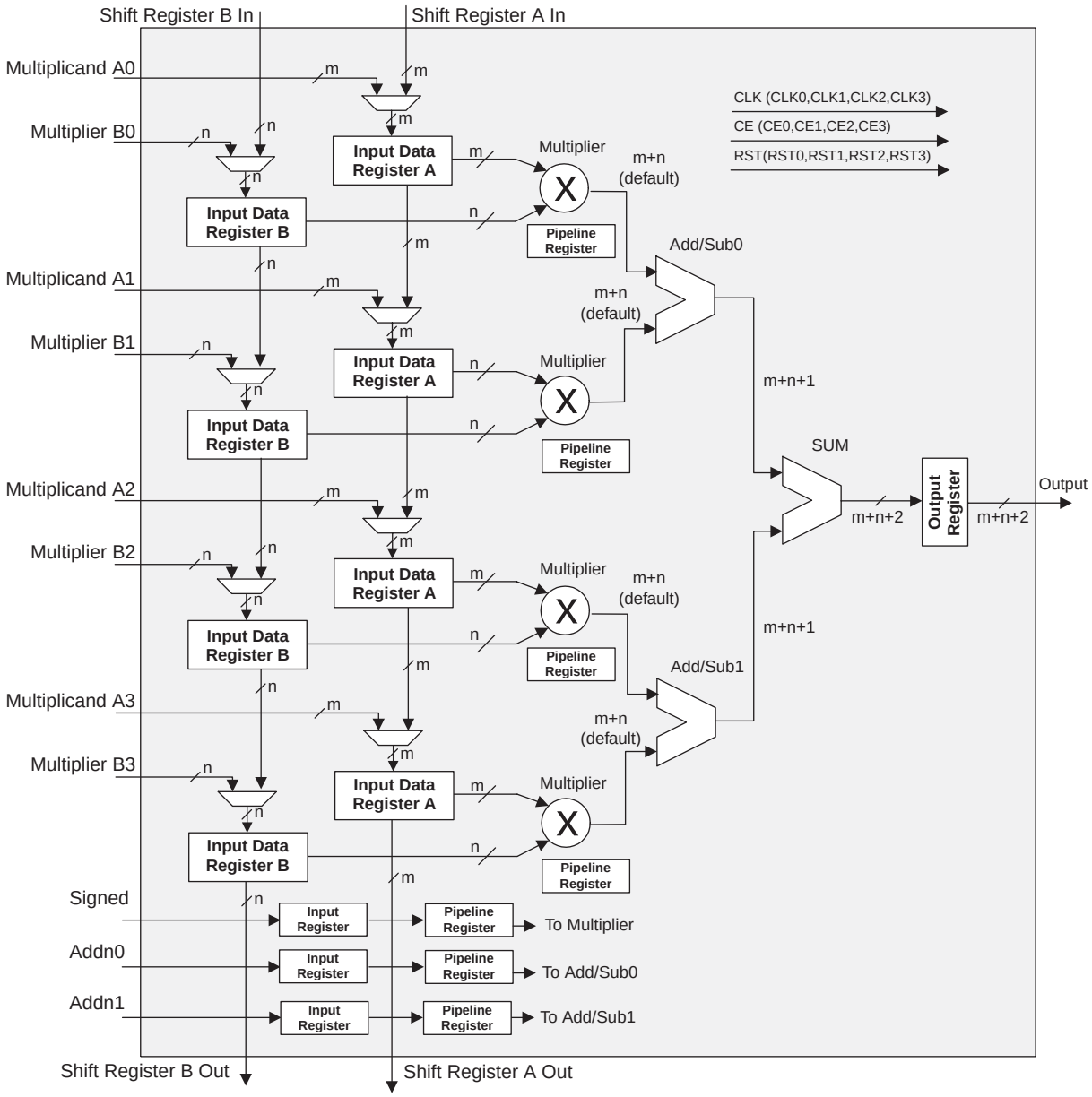
Figure 2-16. MULTADD



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 2-17 shows the MULTADDSUM sysDSP element.

Figure 2-17. 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.

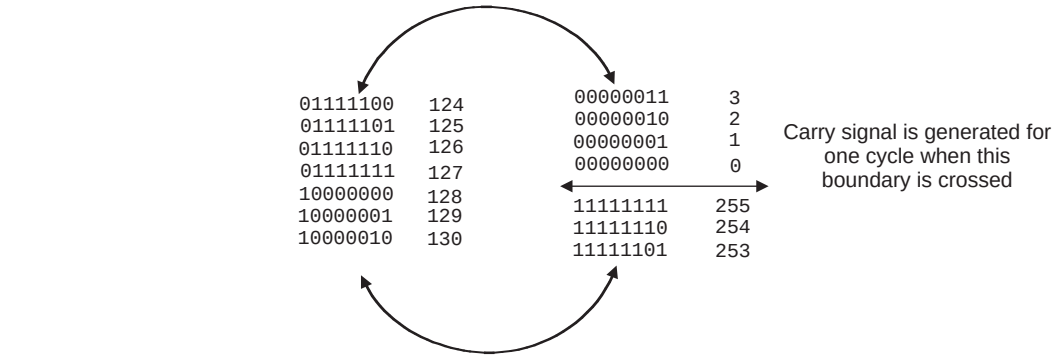
Table 2-8. An Example of Sign Extension

Number	Unsigned	Unsigned 9-bit	Unsigned 18-bit	Signed	Two's Complement Signed 9-Bits	Two's Complement Signed 18-bits
+5	0101	00000101	00000000 00000101	0101	00000101	00000000 00000101
-6	0110	00000110	00000000 00000110	1010	11111010	11111111 11111010

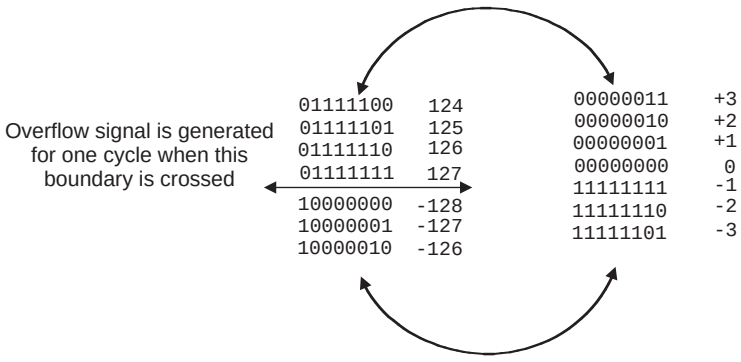
OVERFLOW Flag from MAC

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 signal for signed and unsigned operands are listed in Figure 2-18.

Figure 2-18. Accumulator Overflow/Underflow Conditions



Unsigned Operation



Signed Operation

ispLEVER Module Manager

The user can access the sysDSP block via the ispLEVER Module Manager, which 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
LFECP40	10	80	40	10

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
LFECP40	70	645

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

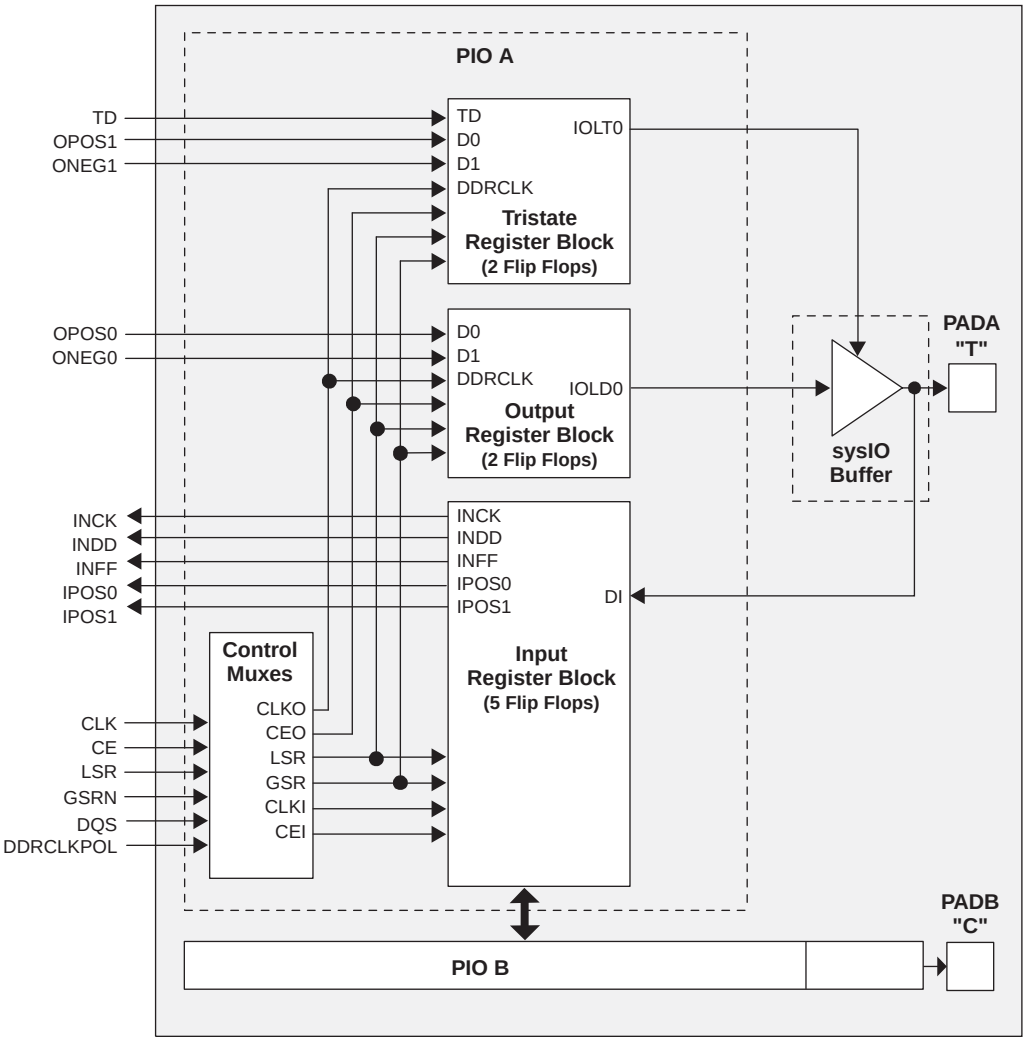
Device	DSP Block	DSP Performance MMAC
LFECP6	4	
LFECP10	5	
LFECP15	6	
LFECP20	7	
LFECP40	10	

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

Programmable I/O Cells (PIC)

Each PIC contains two PIOs connected to their respective sysIO Buffers which are then connected to the PADs as shown in Figure 2-19. The PIO Block supplies the output data (DO) and the Tri-state control signal (TO) to sysIO buffer, and receives input from the buffer.

Figure 2-19. PIC Diagram



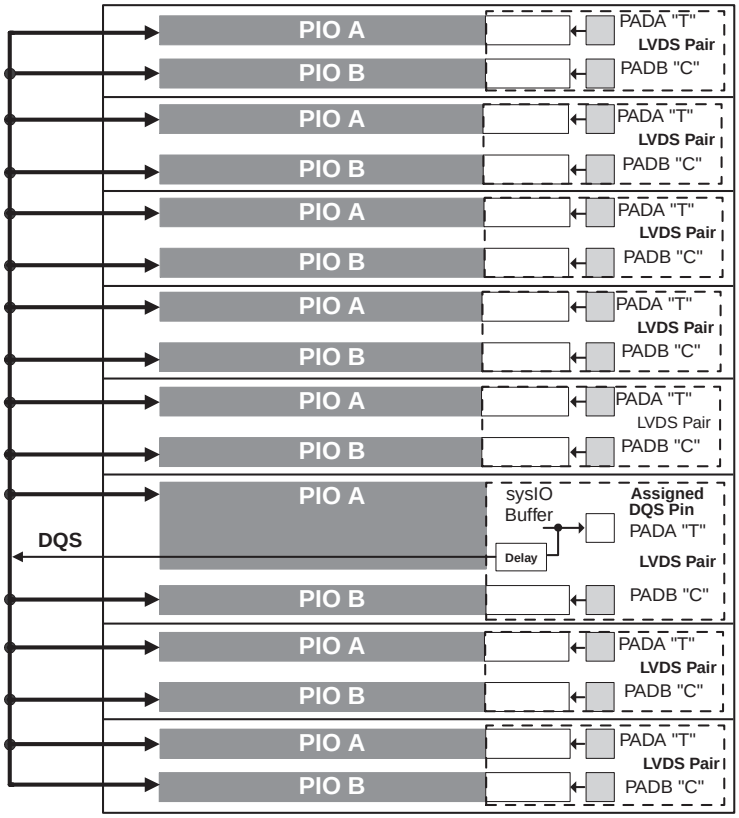
Two adjacent PIOs can be joined to provide a differential I/O pair (labeled as “T” and “C”) as shown in Figure 2-20. The PAD Labels “T” and “C” distinguish the two PIOs. Only the PIO pairs on the left and right edges of the device can be configured as LVDS transmit/receive pairs.

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. 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 2-20. 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 2-21 shows the diagram of the input register block.

Input signals are fed from the sysIO buffer to the input register block (as signal DI). If desired the input signal can bypass the register and delay elements and be used directly as a combinatorial signal (INDD), a clock (INCK) and in selected blocks the input to the DQS delay block. If 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 2-22 shows the input register waveforms for DDR operation and Figure 2-23 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 2-21. Input Register Diagram

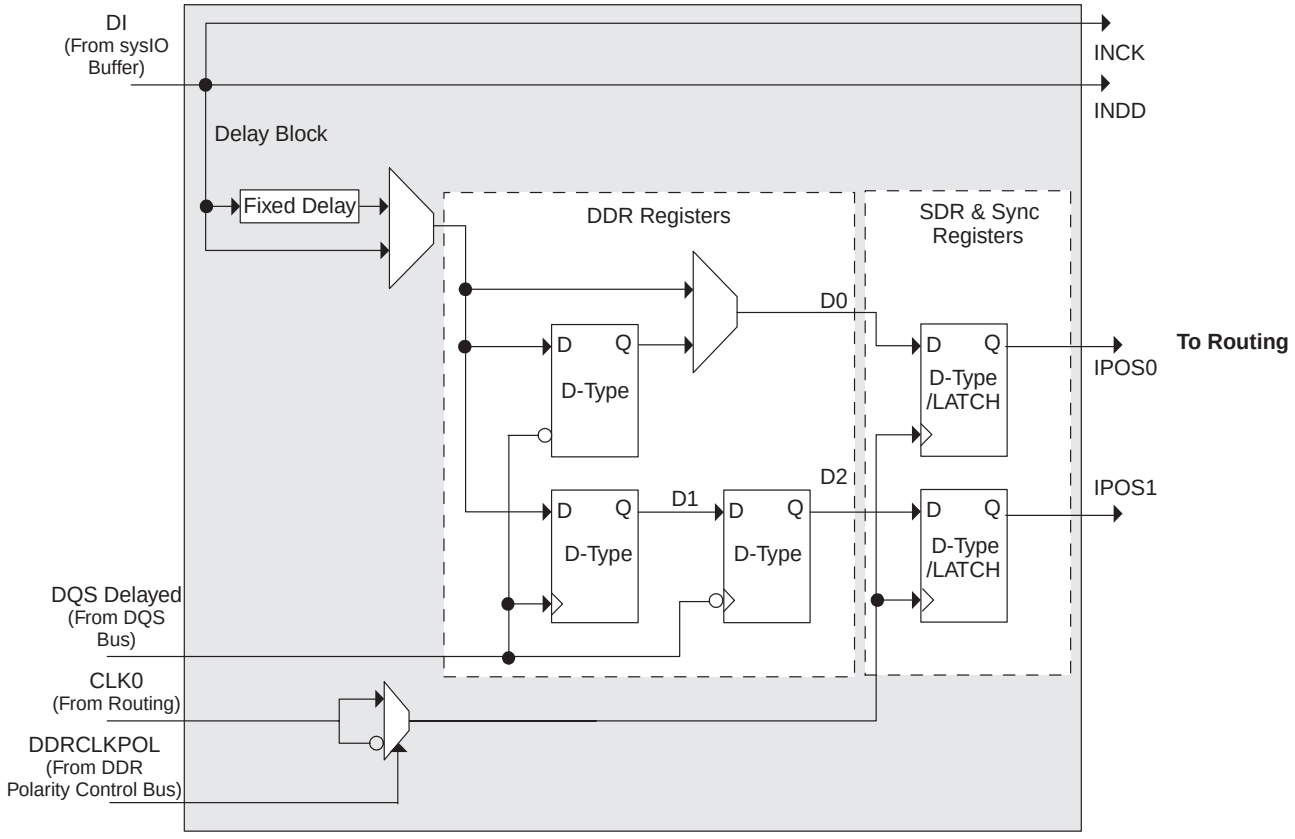


Figure 2-22. Input Register DDR Waveforms

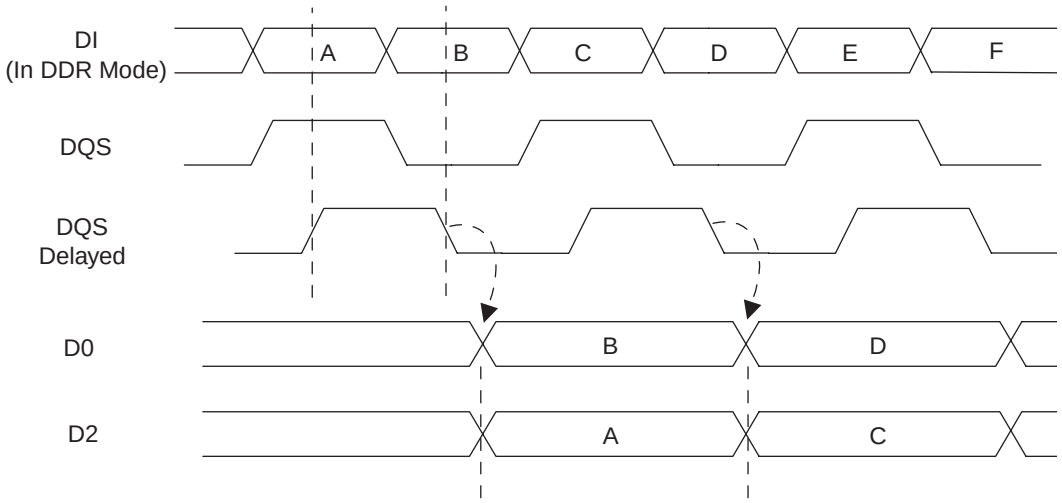
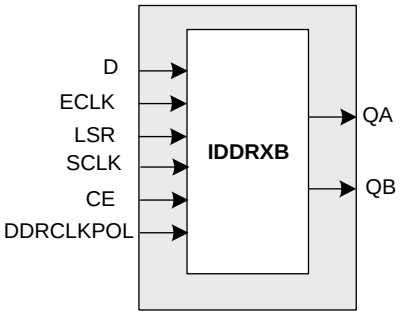


Figure 2-23. INDDRXB Primative



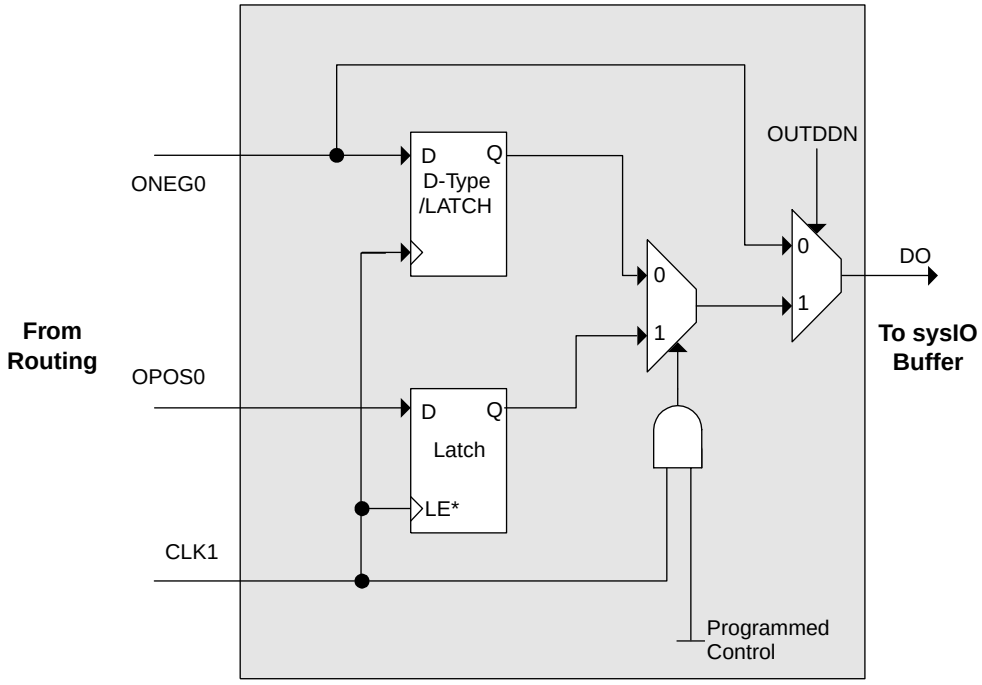
Output Register Block

The output register block provides the ability to register signals from the core of the device before they are passed to the sysIO buffers. The block contains a register for SDR operation that is combined with an additional latch for DDR operation. Figure 2-24 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).

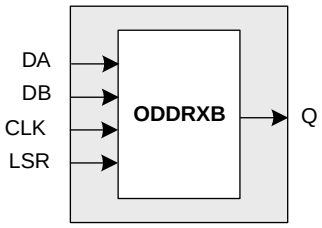
Figure 2-25 shows the design tool DDR primitives. The SDR output register has reset and clock enable available. The additional register for DDR operation does not have reset or clock enable available.

Figure 2-24. Output Register Block



*Latch is transparent when input is low.

Figure 2-25. ODDRXB Primative

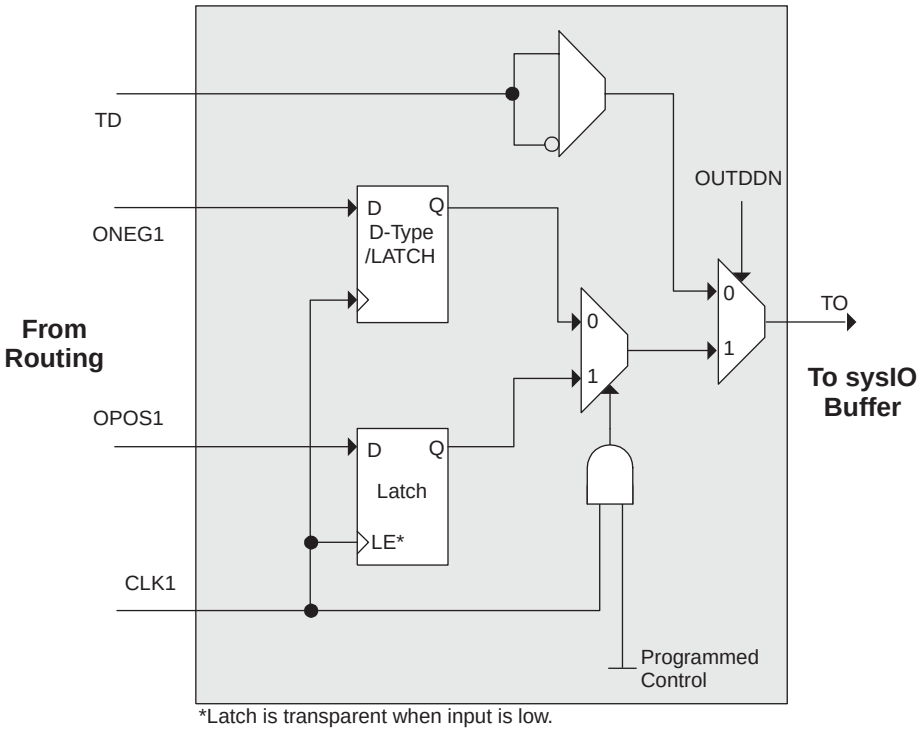


Tristate Register Block

The tristate register block provides the ability to register tri-state control signals from the core of the device before they are passed to the sysIO buffers. The block contains a register for SDR operation and an additional latch for DDR operation. Figure 2-26 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 2-26. 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 EC devices provide this capability. In addition to these registers, the EC 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 2-27 and 2-28 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 2-28. The DLL loop is compensated for temperature, voltage and process variations by the system clock and feedback loop.

Figure 2-27. DQS Local Bus.

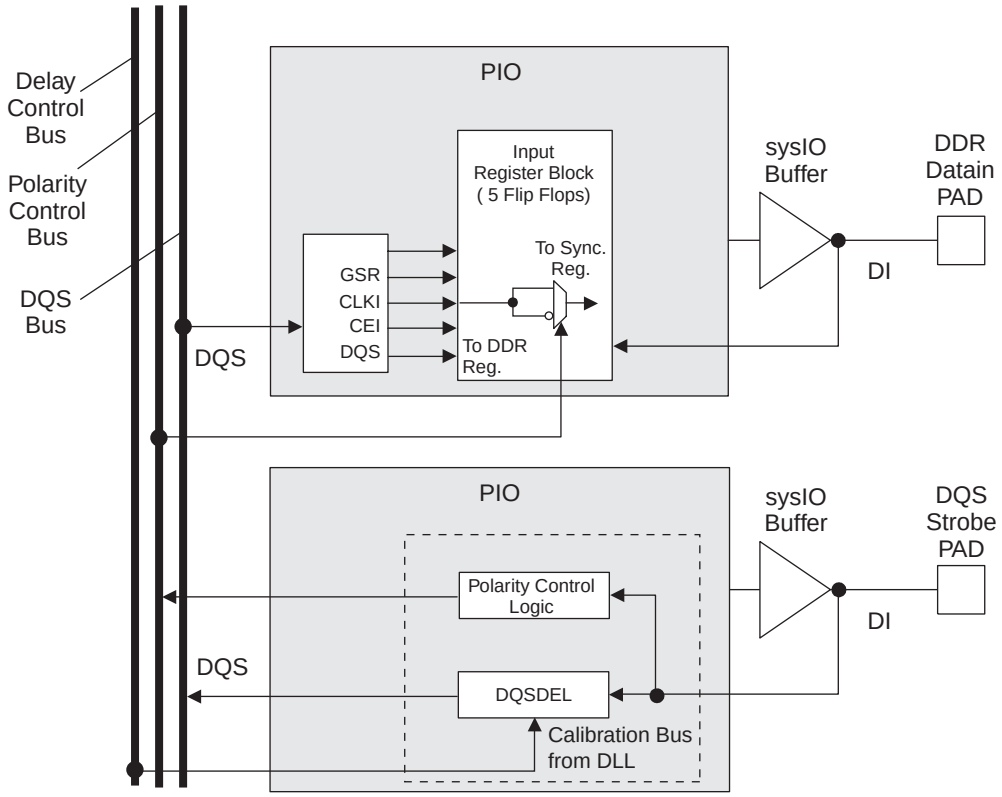
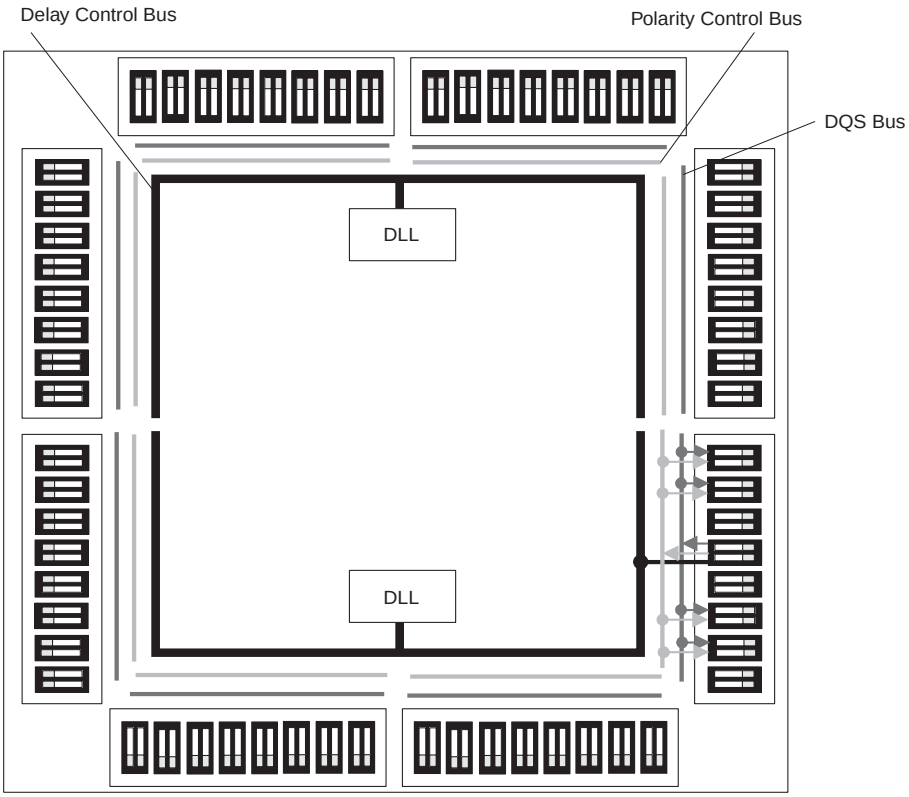


Figure 2-28. DLL Calibration Bus and DQS/DQS Transition Distribution



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.

sysIO Buffer

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

sysIO Buffer Banks

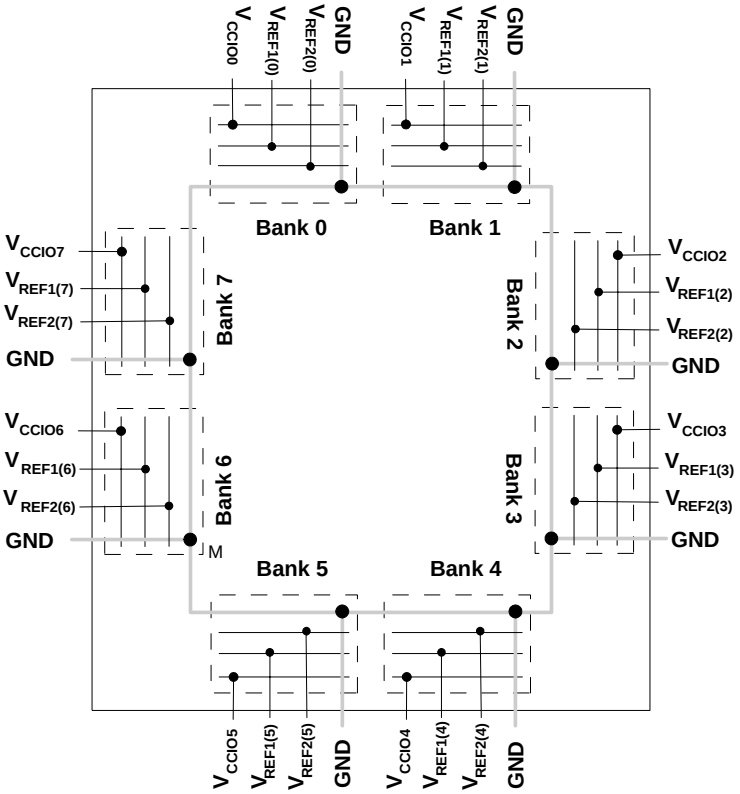
LatticeECP/EC devices have eight sysIO buffer banks; each is capable of supporting multiple I/O standards. Each sysIO bank has its own I/O supply voltage (V_{CCIO}), and two voltage references V_{REF1} and V_{REF2} resources allowing each bank to be completely independent from each other. Figure 2-29 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 2-29. LatticeECP/EC Banks



Note: N and M are the maximum number of I/Os per bank.

LatticeECP/EC devices contain two types of sysIO buffer pairs.

1. **Top and Bottom sysIO Buffer Pair (Single-Ended Outputs Only)**
   The sysIO 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 PCI clamp.

2. **Left and Right sysIO Buffer Pair (Differential and Single-Ended Outputs)**
   The sysIO buffer pairs in the left and right banks of the device consist of two single-ended output drivers, two sets of single-ended input buffers (both ratioed and referenced) and one differential output driver. 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.

Supported Standards

The LatticeECP/EC sysIO buffer supports both single-ended and differential standards. Single-ended standards can be further subdivided into LVCMOS, LVTTL and other standards. The buffers support the LVTTL, LVCMOS 1.2, 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, 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 on utilizing the sysIO buffer to support a variety of standards please see the details of additional technical information at the end of this data sheet.

Table 2-13. Supported Input Standards

Input Standard	V_{REF} (Nom.)	V_{CCIO}[1] (Nom.)
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	—	—

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

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 to 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 family of 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. When sysCONFIG mode is not used, these dual-use pins are available for 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 on device configuration, please see details of additional 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 on 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 serial clock for configuration. The oscillator and the master serial clock run continuously. The default value of the master serial clock is 2.5MHz. Table 2-15 lists all the available Master Serial Clock frequencies. When a different Master Serial Clock is selected during the design process, the following sequence takes place:

1. User selects a different Master Serial Clock frequency.
2. During configuration the device starts with the default (2.5MHz) Master Serial Clock frequency.
3. The clock configuration settings are contained in the early configuration bit stream.
4. The Master Serial Clock frequency changes to the selected frequency once the clock configuration bits are received.

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

Table 2-15. Selectable Master Serial 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.

LatticeECP/EC Family Data Sheet DC and Switching Characteristics

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

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}	Auxiliary Supply Voltage	3.135	3.465	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.

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

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

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. $0 \le V_{CC} \le V_{CC}$ (MAX), $0 \le V_{CCIO} \le V_{CCIO}$ (MAX) or $0 \le V_{CCAUX} \le V_{CCAUX}$ (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 Low leakage	$0 \le V_{IN} \le (V_{CCIO} - 0.2V)$	—	—	10	µA
		$(V_{CCIO} - 0.2V) \le V_{IN} \le 3.6V$	—	—	40	µA
I_{PU}	I/O Active Pull-up Current	$0 \le V_{IN} \le 0.7\ V_{CCIO}$	30	—	150	µA
I_{PD}	I/O Active Pull-down Current	V_{IL} (MAX) $\le V_{IN} \le V_{IH}$ (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	$0 \le V_{IN} \le V_{IH}$ (MAX)	—	—	150	µA
I_{BHLH}	Bus Hold High Overdrive current	$0 \le V_{IN} \le V_{IH}$ (MAX)	—	—	-150	µA
V_{BHT}	Bus Hold trip Points	$0 \le V_{IN} \le V_{IH}$ (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

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

Over Recommended Operating Conditions

Symbol	Parameter	Condition	Typ.	Max.	Units
I_{CC}	Core Power Supply Current	LFEC1			mA
		LFEC3			mA
		LFEC6/LFECP6			mA
		LFEC10/LFECP10			mA
		LFEC15/LFECP15			mA
		LFEC20/LFECP20	60		mA
		LFEC40/LFECP40			mA
I_{CCAUX}	Auxiliary Power Supply Current	LFEC1			mA
		LFEC3			mA
		LFEC6/LFECP6			mA
		LFEC10/LFECP10			mA
		LFEC15/LFECP15			mA
		LFEC20/LFECP20	15		mA
		LFEC40/LFECP40			mA
I_{CCPLL}	PLL Power Supply	LFEC1, LFEC3, LFEC6, LFECP6			mA
		LFEC10, LFEC15, LFEC20, LFEC40, LFECP10, LFECP15, LFECP20, LFECP40			mA
I_{CCIO}	Bank Power Supply Current		15		mA
I_{CCJ}	V_{CCJ} Power Supply Current		1		mA

1. For further information on supply current, please see details of additional technical documentation at the end of this data sheet.
2. Assumes all outputs are tristated, all inputs are configured as LVCMOS and held at the V_{CCIO} or GND.
3. Frequency 0MHz.

Initialization Supply Current[1]

Over Recommended Operating Conditions

Symbol	Parameter	Condition	Typ.	Max.	Units
I_{CC}	Core Power Supply Current	LFEC1			mA
		LFEC3			mA
		LFEC6/LFECP6			mA
		LFEC10/LFECP10			mA
		LFEC15/LFECP15			mA
		LFEC20/LFECP20			mA
		LFEC40/LFECP40			mA
I_{CCAUX}	Auxiliary Power Supply Current	LFEC1			mA
		LFEC3			mA
		LFEC6/LFECP6			mA
		LFEC10/LFECP10			mA
		LFEC15/LFECP15			mA
		LFEC20/LFECP20			mA
		LFEC40/LFECP40			mA
I_{CCPLL}	PLL Power Supply	LFEC1, LFEC3, LFEC6, LFECP6			mA
		LFEC10, LFEC15, LFEC20, LFEC40, LFECP10, LFECP15, LFECP20, LFECP40			mA
I_{CCIO}	Bank Power Supply Current				mA
I_{CCJ}	V_{CCJ} Power Supply Current				mA

1. Until DONE signal is active.

sysIO 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	2.5	1.89	1.15	1.25	1.35
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	3.625	—	—	—
LVPECL[1]	3.135	3.3	3.465	—	—	—
BLVDS[1]	2.375	2.5	2.625	—	—	—

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

sysIO 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.

sysIO 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 ≤ V_{THD}	V_{THD}/2	1.2	1.8	V
		200mV ≤ V_{THD}	V_{THD}/2	1.2	1.9	V
		350mV ≤ 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.

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 3-1 is one possible solution for bi-directional multi-point differential signals.

Figure 3-1. BLVDS Multi-point Output Example

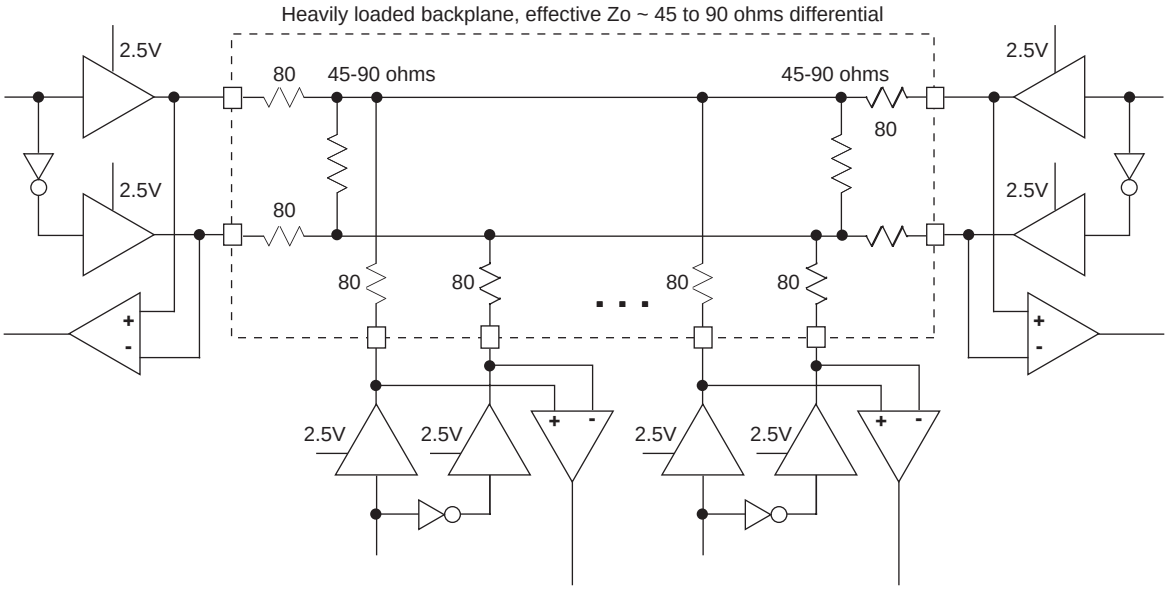


Table 3-1. BLVDS DC Conditions[1]

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 scheme shown in Figure 3-2 is one possible solution for point-to-point signals.

Figure 3-2. Differential LVPECL

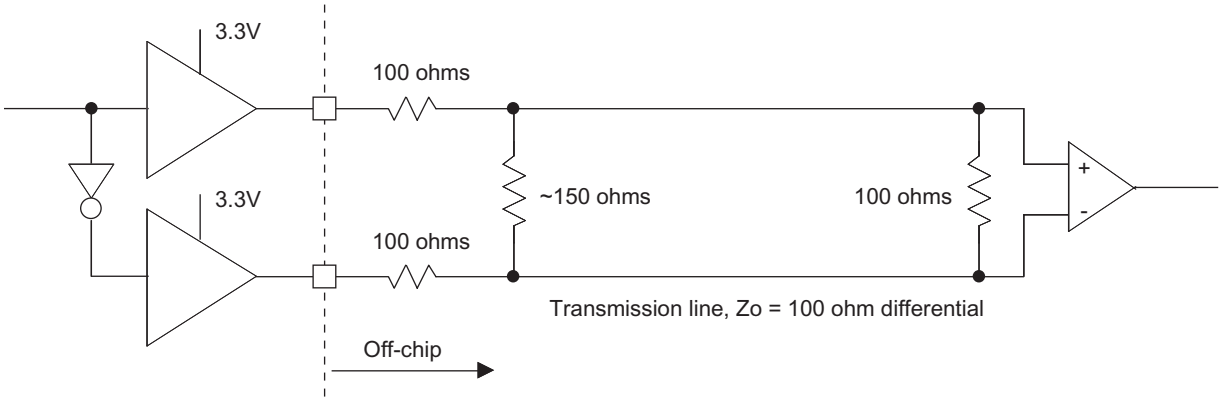


Table 3-2. LVPECL DC Conditions[1]

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 on LVPECL, BLVDS and other differential interfaces please see details of additional 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 scheme shown in Figure 3-3 is one possible solution for RSDS standard implementation. Use LVDS25E mode with suggested resistors for RSDS operation. Resistor values in Figure 3-3 are industry standard values for 1% resistors.

Figure 3-3. RSDS (Reduced Swing Differential Standard)

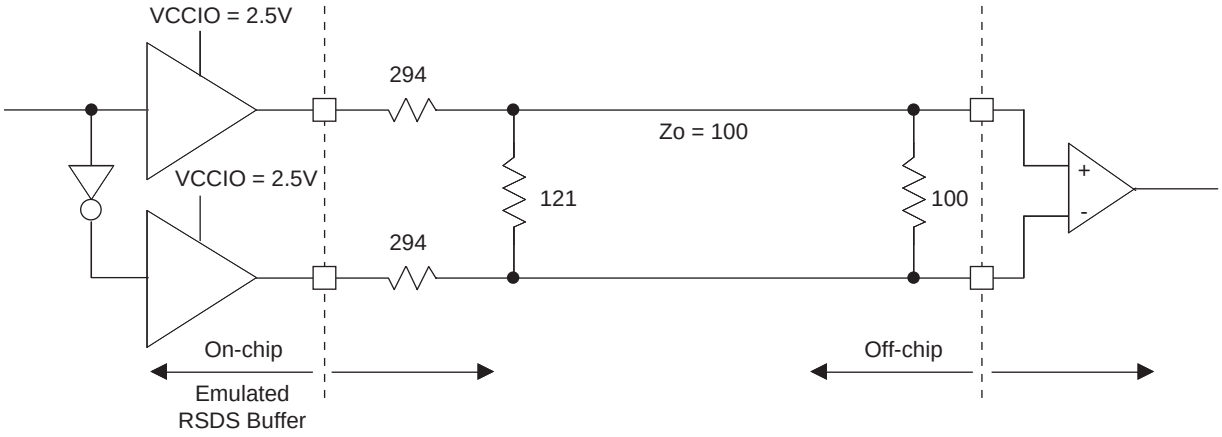


Table 3-3. RSDS DC Conditions

Parameter	Description	Typical	Units
Z_{OUT}	Output impedance		ohm
R_S	Driver series resistor		ohm
R_P	Driver parallel resistor		ohm
R_T	Receiver termination		ohm
V_{OH}	Output high voltage		V
V_{OL}	Output low voltage		V
V_{OD}	Output differential voltage		V
V_{CM}	Output common mode voltage		V
Z_{BACK}	Back impedance		ohm
I_{DC}	DC output current		mA

5V Tolerant Input Buffer

The input buffers of the LatticeECP/EC family of devices can support 5V signals by using a PCI Clamp and an external series resistor as shown in Figure 3-4. A suitable resistor can be selected by using the PCI Clamp Characteristic as shown in Figure 3-5.

Figure 3-4. 5 V Tolerant Input Buffer

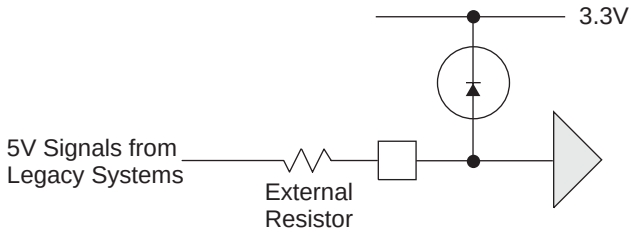
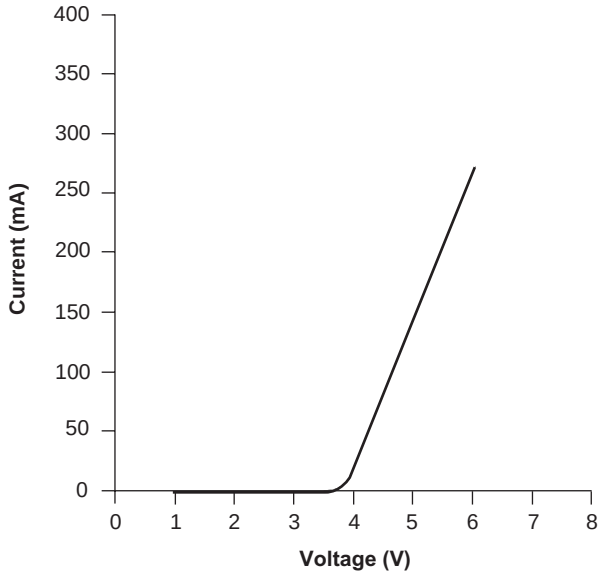


Figure 3-5. Typical PCI Clamp Current



Typical Building Block Function Performance

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

Function	-5 Timing	Units
Basic Functions		
16-bit decoder	6.8	ns
32-bit decoder	7.8	ns
64-bit decoder	8.4	ns
4:1 MUX	5.7	ns
8:1 MUX	5.9	ns
16:1 MUX	6.5	ns
32:1 MUX	6.9	ns
Combinatorial (pin to LUT to pin)	5.3	ns
Embedded Memory Functions		
Pin to EBR input register setup	0.0	ns
EBR output clock to pin	11.3	ns
Distributed PFU RAM		
Pin to PFU RAM register setup	0.0	ns
PFU RAM clock to pin	6.8	ns

Register-to-Register Performance

Function	-5 Timing	Units
Basic Functions		
16-bit decoder	263	MHz
32-bit decoder	230	MHz
64-bit decoder	211	MHz
4:1 MUX	500	MHz
8:1 MUX	375	MHz
16:1 MUX	360	MHz
32:1 MUX	373	MHz
8-bit adder	314	MHz
16-bit adder	251	MHz
64-bit adder	146	MHz
16-bit counter	360	MHz
32-bit counter	280	MHz
64-bit counter	180	MHz
64-bit accumulator	125	MHz
Embedded Memory Functions		
256x36 Single Port RAM	305	MHz
512x18 True-Dual Port RAM	308	MHz
Distributed Memory Functions		
16x2 Single Port RAM	455	MHz
64x2 Single Port RAM	244	MHz
128x4 Single Port RAM	196	MHz
32x2 Pseudo-Dual Port RAM	341	MHz
64x4 Pseudo-Dual Port RAM	303	MHz

Register-to-Register Performance (Continued)

Function	-5 Timing	Units
DSP Function		
9x9 Pipelined Multiply/Accumulate[1]	265	MHz
18x18 Pipelined Multiply/Accumulate[1]	226	MHz
36x36 Pipelined Multiply[1]	177	MHz

1. Applies to LatticeECP devices only.

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 3-4. Delay Derating Table for Internal Blocks

T_J °C Commercial	T_J °C Industrial	Power Supply Voltage		
		1.14V	1.2V	1.26V
—	-40			
—	-25			
0	15			
25	40			
85	100			
100	115			
110	125			
125	—			

LatticeECP/EC External Switching Characteristics

Over Recommended Operating Conditions

Parameter	Description	Device	-5		-4		-3		Units
			Min.	Max.	Min.	Max.	Min.	Max.	
General I/O Pin Parameters (Using Primary Clock without PLL)[1]									
t_{CO}	Clock to Output - PIO Output Register	LFEC20	—	6.75	-	8.43	—	11.25	ns
t_{SU}	Clock to Data Setup - PIO Input Register	LFEC20	0.00	—	0.00	—	0.00	—	ns
t_{H}	Clock to Data Hold - PIO Input Register	LFEC20	2.55	—	3.19	—	4.25	—	ns
t_{SU_DEL}	Clock to Data Setup - PIO Input Register with data input delay	LFEC20	2.85	—	3.42	—	3.99	—	ns
t_{H_DEL}	Clock to Data Hold - PIO Input Register with Input Data Delay	LFEC20	0.00	—	0.00	—	0.00	—	ns
f_{MAX_IO}	Clock Frequency of I/O and PFU Register	LFEC20	—		—		—		Mhz
DDR I/O Pin Parameters[2]									
t_{DVBDQ}	Data Valid Before DQS (DDR Read)	LFEC20	—		—		—		ps
t_{DVADQ}	Data Valid After DQS (DDR Read)	LFEC20		—		—		—	ps
t_{DQ_SK}	Data Skew (DDR Write)	LFEC20	—		—		—		ps
t_{DQS_JIT}	DQS Jitter (DDR Write)	LFEC20	—		—		—		ps
f_{MAX_DDR}	DDR Clock Frequency	LFEC20	—	166	—		—		MHz

1. General timing numbers based on LVCMOS2.5V, 12 mA.
2. DDR timing numbers based on SSTL I/O.

LatticeECP/EC Internal Timing Parameters[1]

Over Recommended Operating Conditions

Parameter	Description	-5		-4		-3		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
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)	—	0.55	—	0.66	—	0.77	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	0.08	—	0.10	—	0.11	—	ns
t_{HM_PFU}	Clock to Mux (M0,M1) input hold time	-0.06	—	-0.07	—	-0.08	—	ns
t_{SUD_PFU}	Clock to D input setup time	0.09	—	0.10	—	0.12	—	ns
t_{HD_PFU}	Clock to D input hold time	-0.04	—	-0.04	-	-0.05	—	ns
t_{CK2Q_PFU}	Clock to Q delay, D-type register configuration	—	0.43	—	0.51	—	0.60	ns
t_{LE2Q_PFU}	Clock to Q delay latch configuration	—	0.54	—	0.65	—	0.76	ns
t_{LD2Q_PFU}	D to Q throughput delay when latch is enabled	—	0.50	—	0.60	—	0.69	ns
PFU Memory Mode Timing								
t_{CORAM_PFU}	Clock to Output	—	0.43	—	0.51	—	0.60	ns
t_{SUDATA_PFU}	Data Setup Time	-0.25	—	-0.30	—	-0.34	—	ns
t_{HDATA_PFU}	Data Hold Time	-0.06	—	-0.07	—	-0.08	—	ns
t_{SUADDR_PFU}	Address Setup Time	-0.66	—	-0.79	—	-0.92	—	ns
t_{HADDR_PFU}	Address Hold Time	-0.27	—	-0.33	—	-0.38	—	ns
t_{SUWREN_PFU}	Write/Read Enable Setup Time	-0.30	—	-0.36	—	-0.42	—	ns
t_{HWREN_PFU}	Write/Read Enable Hold Time	-0.21	—	-0.25	—	-0.29	—	ns
PIC Timing								
PIO Input/Output Buffer Timing								
t_{IN_PIO}	Input Buffer Delay	—		—		—		ns
t_{OUT_PIO}	Output Buffer Delay	—		—		—		ns
IOLOGIC Input/Output Timing								
t_{SUI_PIO}	Input Register Setup Time (Data Before Clock)	—	0.12	—	0.14	—	0.17	ns
t_{HI_PIO}	Input Register Hold Time (Data after Clock)	—	-0.09	—	-0.11	—	-0.13	ns
t_{COO_PIO}	Output Register Clock to Output Delay	—	0.75	—	0.90	—	1.05	ns
t_{SUCE_PIO}	Input Register Clock Enable Setup Time	—	-0.02	—	-0.02	—	-0.03	ns
t_{HCE_PIO}	Input Register Clock Enable Hold Time	—	0.12	—	0.14	—	0.17	ns
t_{SULSR_PIO}	Set/Reset Setup Time	0.10	0.24	0.12	0.29	0.14	0.34	ns
t_{HLSR_PIO}	Set/Reset Hold Time	-0.24	-0.10	-0.29	-0.12	-0.34	-0.14	ns
EBR Timing								
t_{CO_EBR}	Clock to output from Address or Data	—	3.80	—	4.55	—	5.31	ns
t_{COO_EBR}	Clock to output from EBR output Register	—		—		—		ns
tS_{UDATA_EBR}	Setup Data to EBR Memory	-0.34	—	-0.41	—	-0.48	—	ns
t_{HDATA_EBR}	Hold Data to EBR Memory	0.37	—	0.44	—	0.52	—	ns
t_{SUADDR_EBR}	Setup Address to EBR Memory	-0.34	—	-0.41	—	-0.48	—	ns
t_{HADDR_EBR}	Hold Address to EBR Memory	0.37	—	0.45	—	0.52	—	ns
t_{SUWREN_EBR}	Setup Write/Read Enable to PFU Memory	-0.22	—	-0.26	—	-0.30	—	ns

LatticeECP/EC Internal Timing Parameters[1] (Continued)

Over Recommended Operating Conditions

Parameter	Description	-5		-4		-3		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
t_{HWREN_EBR}	Hold Write/Read Enable to PFU Memory	0.23	—	0.28	—	0.33	—	ns
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	-0.17	—	-0.20	—	-0.24	—	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	—		—		—		ns
t_{RSTSU}	Reset Signal Setup Time	1	—	1	—	1	—	ns
t_{RSTW}	Reset Signal Pulse Width	1.8	—	1.8	—	1.8	—	ns
DSP Block Timing[2]								
t_{SUI_DSP}	Input Register Setup Time	—		—		—		ns
t_{HI_DSP}	Input Register Hold Time	—		—		—		ns
t_{SUP_DSP}	Pipeline Register Setup Time	—		—		—		ns
t_{HP_DSP}	Pipeline Register Hold Time	—		—		—		ns
t_{SUO_DSP}	Output Register Setup Time	—		—		—		ns
t_{HO_DSP}	Output Register Hold Time	—		—		—		ns
t_{COI_DSP}	Input Register Clock to Output Time	—		—		—		ns
t_{COP_DSP}	Pipeline Register Clock to Output Time	—		—		—		ns
t_{COO_DSP}	Output Register Clock to Output Time	—		—		—		ns
$t_{COOVRFL\_DSP}$	Overflow Register Clock to Output Time	—		—		—		ns
$t_{SUADSUB}$	AdSub Setup Time	—		—		—		ns
t_{HADSUB}	AdSub Hold Time	—		—		—		ns
t_{SUSIGN}	Sign Setup Time	—		—		—		ns
t_{HSIGN}	Sign Hold Time	—		—		—		ns
$t_{SUACCSLOAD}$	Accumulator Load Setup Time	—		—		—		ns
$t_{HACCSLOAD}$	Accumulator Load Hold Time	—		—		—		ns

1. Internal parameters are characterized but not tested on every device.
2. These parameters apply to LatticeECP devices only.

Timing Diagrams

PFU Timing Diagrams

Figure 3-6. Slice Single/Dual Port Write Cycle Timing

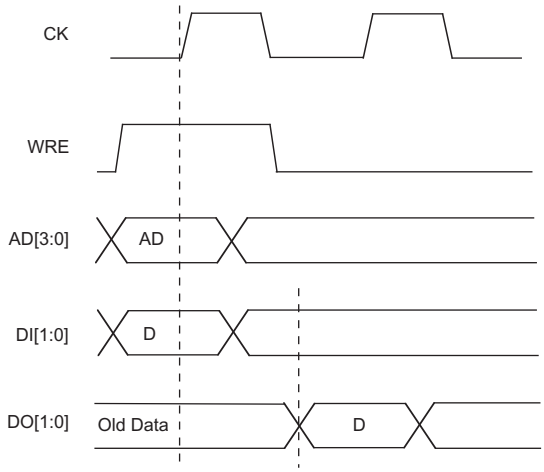
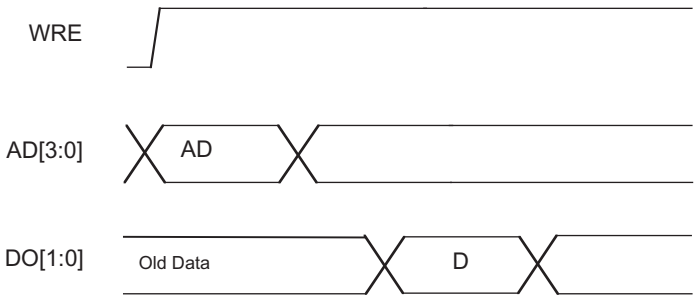
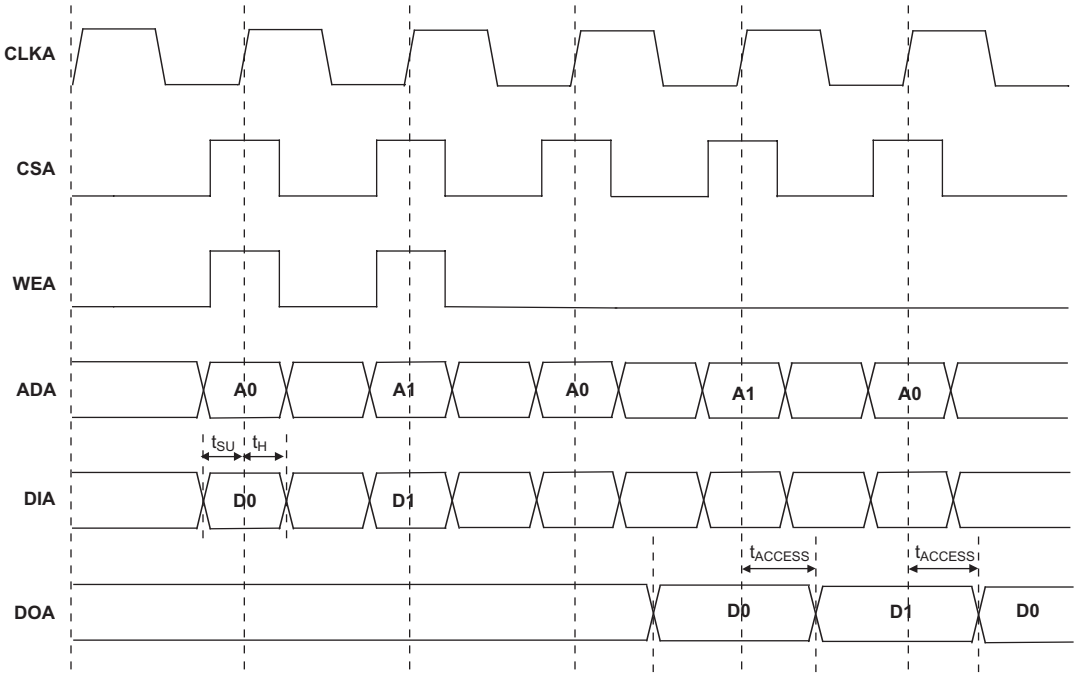


Figure 3-7. Slice Single /Dual Port Read Cycle Timing



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

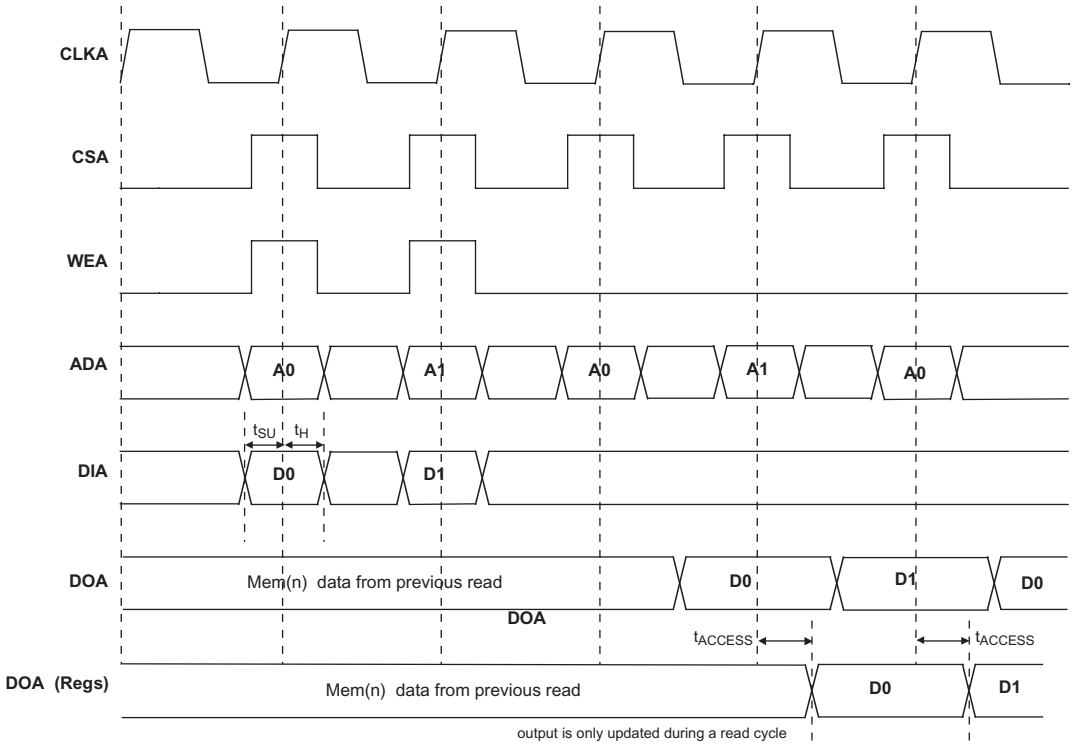
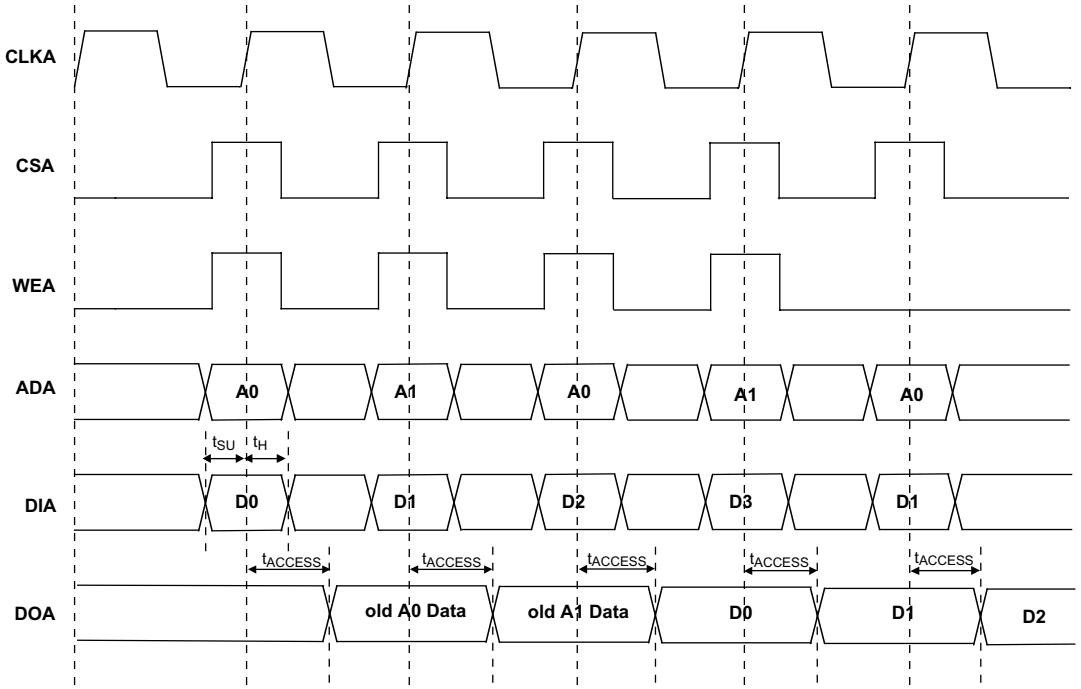
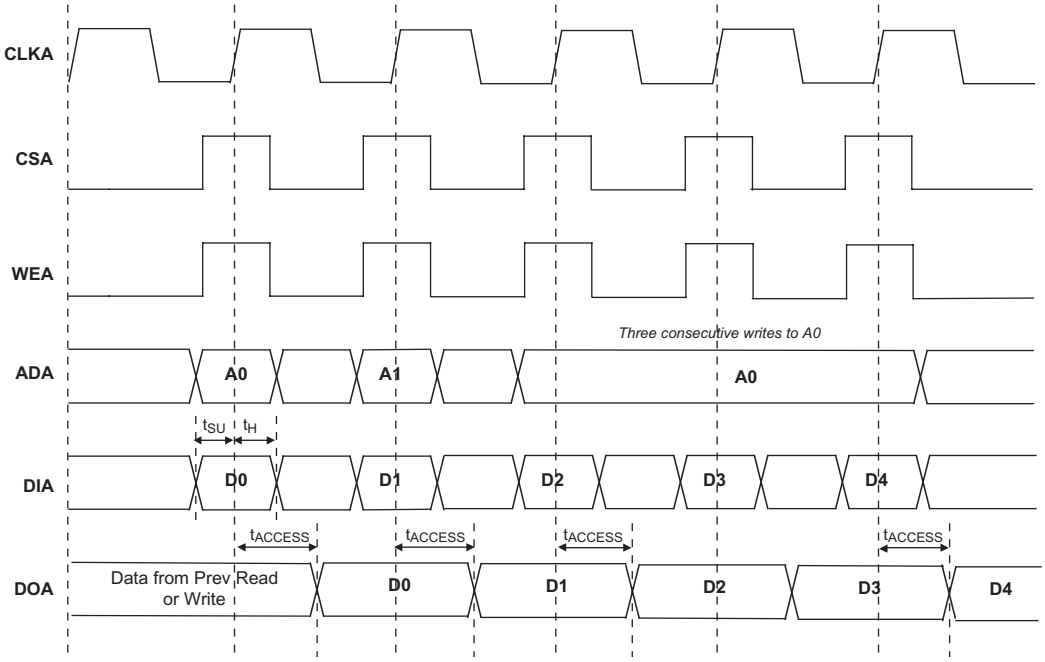


Figure 3-10. Read Before Write (SP Read/Write on Port A, Input Registers Only)



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]

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				ns
LVDS25	LVDS 2.5				ns
BLVDS25	BLVDS 2.5				ns
LVPECL33	LVPECL 3.3				ns
HSTL18_I	HSTL_18 class I				ns
HSTL18_II	HSTL_18 class II				ns
HSTL18_III	HSTL_18 class III				ns
HSTL18D_I	Differential HSTL 18 class I				ns
HSTL18D_II	Differential HSTL 18 class II				ns
HSTL18D_III	Differential HSTL 18 class III				ns
HSTL15_I	HSTL_15 class I				ns

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

Over Recommended Operating Conditions

Buffer Type	Description	-5	-4	-3	Units
HSTL15_II	HSTL_15 class II				ns
HSTL15_III	HSTL_15 class III				ns
HSTL15D_I	Differential HSTL 15 class I				ns
HSTL15D_III	Differential HSTL 15 class III				ns
SSTL33_I	SSTL_3 class I				ns
SSTL33_II	SSTL_3 class II				ns
SSTL33D_I	Differential SSTL_3 class I				ns
SSTL33D_II	Differential SSTL_3 class II				ns
SSTL25_I	SSTL_2 class I				ns
SSTL25_II	SSTL_2 class II				ns
SSTL25D_I	Differential SSTL_2 class I				ns
SSTL25D_II	Differential SSTL_2 class II				ns
SSTL18_I	SSTL_1.8 class I				ns
SSTL18D_I	Differential SSTL_1.8 class I				ns
LVTTL33_4mA	LVTTL 4mA drive				ns
LVTTL33_8mA	LVTTL 8mA drive				ns
LVTTL33_12mA	LVTTL 12mA drive				ns
LVTTL33_16mA	LVTTL 16mA drive				ns
LVTTL33_20mA	LVTTL 20mA drive				ns
LVCMOS33_4mA	LVCMOS 3.3 4mA drive				ns
LVCMOS33_8mA	LVCMOS 3.3 8mA drive				ns
LVCMOS33_12mA	LVCMOS 3.3 12mA drive				ns
LVCMOS33_16mA	LVCMOS 3.3 16mA drive				ns
LVCMOS33_20mA	LVCMOS 3.3 20mA drive				ns
LVCMOS25_4mA	LVCMOS 2.5 4mA drive				ns
LVCMOS25_8mA	LVCMOS 2.5 8mA drive				ns
LVCMOS25_12mA	LVCMOS 2.5 12mA drive	0.00	0.00	0.00	ns
LVCMOS25_16mA	LVCMOS 2.5 16mA drive				ns
LVCMOS25_20mA	LVCMOS 2.5 20mA drive				ns
LVCMOS18_4mA	LVCMOS 1.8 4mA drive				ns
LVCMOS18_8mA	LVCMOS 1.8 8mA drive				ns
LVCMOS18_12mA	LVCMOS 1.8 12mA drive				ns
LVCMOS18_16mA	LVCMOS 1.8 16mA drive				ns
LVCMOS15_4mA	LVCMOS 1.5 4mA drive				ns
LVCMOS15_8mA	LVCMOS 1.5 8mA drive				ns
LVCMOS12_2mA	LVCMOS 1.2 2mA drive				ns
LVCMOS12_6mA	LVCMOS 1.2 6mA drive				ns
LVCMOS12_4mA	LVCMOS 1.2 4mA drive				ns
PCI33	PCI33				ns

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

sysCLOCK PLL Timing

Over Recommended Operating Conditions

Parameter	Descriptions	Conditions	Min.	Max.	Units
f_{IN}	Input Clock Frequency (CLKI, CLKFB)		33	420	MHz
f_{OUT}	Output Clock Frequency (CLKOP, CLKOS)		33	420	MHz
f_{OUT2}	K-Divider Output Frequency (CLKOK)		0.258	210	MHz
f_{VCO}	PLL VCO Frequency		420	840	MHz
f_{PFD}	Phase Detector Input Frequency		33	—	MHz
AC Characteristics					
t_{DT}	Output Clock Duty Cycle	default duty cycle selected	45	55	%
t_{OPJIT}	Output Clock Period Jitter	$f_{OUT} \geq 100MHz$	—	+/- 100	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%	1	—	ns
t_{LOCK}[1]	PLL Lock-in Time		—	150	us
t_{PA}	Programmable Delay Unit		100	400	ps
t_R/t_F	Input Clock Rise/Fall Time	10% to 90%	—	1	ns
t_{IPJIT}	Input Clock Period Jitter		—	+/- 200	ps
t_{HI}	Input Clock High Time	90% to 90%	0.5	—	ns
t_{LO}	Input Clock Low Time	10% to 10%	0.5	—	ns

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

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		12	—	ns
t_{HCBDI}	Byte D[0:7] Hold Time to CCLK		0	—	ns
t_{CODO}	Clock to Dout in Flowthrough Mode	—			ns
t_{SUCS}	CS[0:1] Setup Time to CCLK		12	—	ns
t_{HCS}	CS[0:1] Hold Time to CCLK		0	—	ns
t_{SUWD}	Write Signal Setup Time to CCLK			—	ns
t_{HWD}	Write Signal Hold Time to CCLK			—	ns
t_{DCB}	CCLK to BUSY Delay Time	—	12		ns
t_{CORD}	Clock to Out for Read Data	—			ns
sysCONFIG Byte Slave Clocking					
t_{BSCH}	Byte Slave Clock Minimum High Pulse		6	—	ns
t_{BSCL}	Byte Slave Clock Minimum Low Pulse		6	—	ns
t_{BSCYC}	Byte Slave Clock Cycle Time		12	—	ns
t_{SUSCDI}	Din Setup time to CCLK Slave Mode		5	—	ns
t_{HSCDI}	Din Hold Time to CCLK Slave Mode		0	—	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		5	—	ns
t_{HMCDI}	Din Hold Time to CCLK Master Mode		0	—	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}	Initialization time of Internal CONFIG Circuit	—	5		ms
t_{VMC}	Time from t_{ICFG} to valid Master Clock	—	5		us
t_{PRGMRJ}	Program Pin Pulse Rejection	—	10		ns
t_{PRGM}	Low time to Start Configuration		25	—	ns
t_{DINIT}	INIT Delay Time		25	—	ns
$t_{DPPINIT}$	Delay Time from Program Low to INIT Low	—			ns
t_{DINITD}	Delay Time from Program Low to Done Low	—	37		ns
t_{IODISS}	User I/O Disable	—			ns
t_{IOENSS}	User I/O Enabled Time from GOE Being Released During Wake-up	—			ns
t_{MWC}	Additional Wake Master Clock Signals After Done Pin High	—	128		Typical cycle
sysCONFIG SPI Port					
t_{CFGX}	Init High to Clock Low	—	80		ns
t_{CSSPI}	Init High to CSSPIN Low	—	2		ns
t_{CSCCLK}	Clock Low to CSSPIN Low	—	0		ns
t_{SOCDO}	Clock Low to Output Valid	—	15		ns
t_{SOSU}	Data Setup Time		5	—	ns
t_{SOE}	CSSPIN Active Setup Time		0	—	ns
t_{CSPID}	CSSPIN Low to First Clock Edge Setup Time		400	—	ns

LatticeECP/EC sysCONFIG Port Timing Specifications

Over Recommended Operating Conditions

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

JTAG Port Timing Specifications

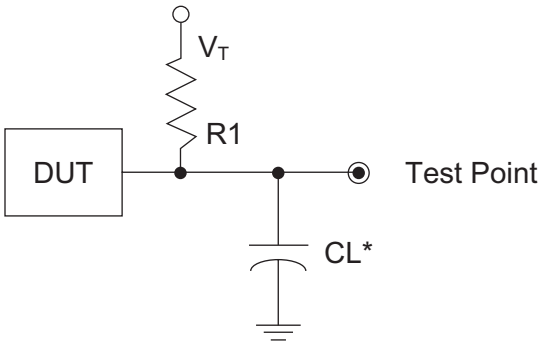
Over Recommended Operating Conditions

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 enalbe	—	10	ns
t_{BTCRS}	BSCAN test capture register setup time	8	—	ns
t_{BTCRH}	BSCAN test capture register hold time	10	—	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

Switching Test Conditions

Figure 3-12 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 Figure 3-5.

Figure 3-12. Output Test Load, LVTTL and LVCMOS Standards



*CL Includes Test Fixture and Probe Capacitance

Table 3-5. 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)	188	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	Descriptions
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 as assigned 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.
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	Descriptions
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	I/O	Generally not used.
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]	I/O	sysCONFIG Port Data I/O.
DOUT, CSON	O	Output for serial configuration data (rising edge of CCLK) when using sysCONFIG port.
DI	I	Input for serial configuration data (clocked with CCLK) when using sysCONFIG port. During configuration, a pull-up is enabled.

LFEC20/LFECP20 Pin Information Summary

Pin Type		Package	
		484 fpBGA	672 fpBGA
Single Ended User I/O		360	400
Differential Pair User I/O		180	200
Configuration	Dedicated	12	12
	Muxed	56	56
TAP		5	5
Dedicated (total without supplies)			
V_{CC}		20	32
V_{CCAUX}		12	20
V_{CCIO}	Bank0	4	6
	Bank1	4	6
	Bank2	4	6
	Bank3	4	6
	Bank4	4	6
	Bank5	4	6
	Bank6	4	6
	Bank7	4	6
GND		44	63
NC		3	96
Single Ended/ Differential I/O per Bank	Bank0	48	64
	Bank1	48	48
	Bank2	40	40
	Bank3	44	48
	Bank4	48	48
	Bank5	48	64
	Bank6	44	48
	Bank7	40	40
V_{CCJ}		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.

LFEC20/LFECP20 Power Supply and NC Connections

Signals	484 fpBGA	672 fpBGA
VCC	J6, J7, J16, J17, K6, K7, K16, K17, L6, L17, M6, M17, N6, N7, N16, N17, P6, P7, P16, P17	H8, H9, H10, H11, H16, H17, H18, H19, J9, J18, K8, K19, L8, L19, M19, N7, R7, R20, T19, U8, U19, V8, V18, V9, W8, W9, W10, W11, W16, W17, W18, W19
VCCIO0	G11, H9, H10, H11	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	R9, R10, R11, 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	G7, G8, G15, G16, H7, H16, R7, R16, T7, T8, T15, T16	G13,H7, H20, J8, J19, K7, L20, M7, M20, N20, P7, P20, T7, T8, T20, V7, V19, W20, Y7, Y13
GND	A1, A22, AB1, AB22, H8, H15, J9, J10, J11, J12, J13, J14, K9, K10, K11, K12, K13, K14, L9, L10, L11, L12, L13, L14, M9, M10, M11, M12, M13, M14, N9, N10, N11, N12, N13, N14, P9, P10, P11, P12, P13, P14, R8, R15	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	A2, A21, AB2	A25, B2, B23, B24, B25, B26, C2, C3, C19, C20, C21, C22, C23, C24, D3, D5, D20, D21, D22, D24, E5, E19, E21, E22, E24, E25, E26, F4, F5, F20, F22, F23, F24, F26, G5, G20, G26, H2, H3, H5, H6, H22, J2, J3, J7, J21, J22, J23, W5, W7, Y5, Y6, Y19, Y20, Y21, Y22, Y23, Y24, AA2, AA3, AA4, AA5, AA21, AA22, AA23, AA24, AB3, AB5, AB19, AB20, AB21, AB22, AB23, AB24, AC2, AC3, AC19, AC20, AC21, AC22, AD1, AD2, AD3, AD19, AD20, AD21, AD22, AD23, AD24, AD25, AD26, AE1, AE24, AE25, AE26, AF25

LFEC20/LFECP20 Logic Signal Connections: 484 & 672 fpBGA

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
PL11B	7	C		H6	J5
PL12A	7	T		G3	K4
PL12B	7	C		H4	K5
PL13A	7	T		J5	J6
PL13B	7	C		H5	K6
PL14A	7	T		F2	F1
PL14B	7	C		F1	G1
PL15A	7	T		E1	H1
PL15B	7	C		D1	J1
PL16A	7	T		H3	K2
PL16B	7	C		G2	K1
PL17A	7	T		H2	K3
PL17B	7	C		G1	L3
PL18A	7	T		J4	L2
PL18B	7	C		J3	L1
PL19A	7	T	LDQS19	J2	M3
PL19B	7	C		H1	M4
PL20A	7	T		K4	M1
PL20B	7	C		K5	M2
PL21A	7	T		K3	L4
PL21B	7	C		K2	L5
PL22A	7	T	PCLKT7_0	J1	N2
PL22B	7	C	PCLKC7_0	K1	N1
XRES	6			L3	N3
PL24A	6	T		L4	P1
PL24B	6	C		L5	P2

LFEC20/LFECP20 Logic Signal Connections: 484 & 672 fpBGA (Cont.)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PL25A	6	T		L2	L7
PL25B	6	C		L1	L6
PL26A	6	T		M4	N4
PL26B	6	C		M5	N5
PL27A	6	T		M1	R1
PL27B	6	C		M2	R2
PL28A	6	T	LDQS28	N3	P4
PL28B	6	C		M3	P3
PL29A	6	T		N5	M5
PL29B	6	C		N4	M6
PL30A	6	T		N1	T1
PL30B	6	C		N2	T2
PL31A	6	T		P1	R4
PL31B	6	C		P2	R3
PL32A	6	T		R6	N6
PL32B	6	C		P5	P5
PL33A	6	T		P3	P6
PL33B	6	C		P4	R5
PL34A	6	T		R1	U1
PL34B	6	C		R2	U2
PL35A	6	T		R5	T3
PL35B	6	C		R4	T4
PL36A	6	T	LDQS36	T1	R6
PL36B	6	C		T2	T5
PL37A	6	T		R3	T6
PL37B	6	C		T3	U5
PL38A	6	T			U3
PL38B	6	C			U4
PL39A	6	T			V1
PL39B	6	C			V2
TCK	6			T5	U7
TDI	6			U5	V4
TMS	6			T4	V5
TDO	6			U1	V3
VCCJ	6			U2	U6
PL41A	6	T	LLM0_PLLT_IN_A	V1	W1
PL41B	6	C	LLM0_PLLC_IN_A	V2	W2
PL42A	6	T	LLM0_PLLT_FB_A	U3	V6
PL42B	6	C	LLM0_PLLC_FB_A	V3	W6
PL43A	6	T		U4	Y1
PL43B	6	C		V5	Y2
PL44A	6	T		W1	W3
PL44B	6	C		W2	W4
PL45A	6	T	LDQS45	Y1	AA1

LFEC20/LFECP20 Logic Signal Connections: 484 & 672 fpBGA (Cont.)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PL45B	6	C		Y2	AB1
PL46A	6	T		AA1	Y4
PL46B	6	C		AA2	Y3
PL47A	6	T		W4	AC1
PL47B	6	C		V4	AB2
PL48A	6	T	VREF1_6	W3	AB4
PL48B	6	C	VREF2_6	Y3	AC4
PB2A	5	T			AB6
PB2B	5	C			AA6
PB3A	5	T			AC7
PB3B	5	C			Y8
PB4A	5	T			AB7
PB4B	5	C			AA7
PB5A	5	T			AC6
PB5B	5	C			AC5
PB6A	5	T	BDQS6		AB8
PB6B	5	C			AC8
PB7A	5	T			AE2
PB7B	5	C			AA8
PB8A	5	T			AF2
PB8B	5	C			Y9
PB9A	5	T			AD5
PB9B	5	C			AD4
PB10A	5	T		V7	AD8
PB10B	5	C		T6	AC9
PB11A	5	T		V8	AE3
PB11B	5	C		U7	AB9
PB12A	5	T		W5	AF3
PB12B	5	C		U6	AD9
PB13A	5	T		AA3	AE4
PB13B	5	C		AB3	AF4
PB14A	5	T	BDQS14	Y6	AE5
PB14B	5	C		V6	AA9
PB15A	5	T		AA5	AF5
PB15B	5	C		W6	Y10
PB16A	5	T		Y5	AD6
PB16B	5	C		Y4	AC10
PB17A	5	T		AA4	AF6
PB17B	5	C		AB4	AE6
PB18A	5	T		Y7	AF7
PB18B	5	C		W8	AB10
PB19A	5	T		W7	AE7
PB19B	5	C		U8	AD10
PB20A	5	T		W9	AD7

LFEC20/LFECP20 Logic Signal Connections: 484 & 672 fpBGA (Cont.)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PB20B	5	C		U9	AA10
PB21A	5	T		Y8	AF8
PB21B	5	C		Y9	AF9
PB22A	5	T	BDQS22	V9	AD11
PB22B	5	C		T9	Y11
PB23A	5	T		W10	AE8
PB23B	5	C		U10	AC11
PB24A	5	T		V10	AF10
PB24B	5	C		T10	AB11
PB25A	5	T		AA6	AE10
PB25B	5	C		AB5	AE9
PB26A	5	T		AA8	AA11
PB26B	5	C		AA7	Y12
PB27A	5	T		AB6	AE11
PB27B	5	C		AB7	AF11
PB28A	5	T		Y10	AF12
PB28B	5	C		W11	AE12
PB29A	5	T		AB8	AD12
PB29B	5	C		AB9	AC12
PB30A	5	T	BDQS30	AA10	AA12
PB30B	5	C		AA9	AB12
PB31A	5	T		Y11	AE13
PB31B	5	C		AA11	AF13
PB32A	5	T	VREF2_5	V11	AD13
PB32B	5	C	VREF1_5	V12	AC13
PB33A	5	T	PCLKT5_0	AB10	AF14
PB33B	5	C	PCLKC5_0	AB11	AE14
PB34A	4	T	WRITEN	Y12	AA13
PB34B	4	C	CS1N	U11	AB13
PB35A	4	T	VREF1_4	W12	AD14
PB35B	4	C	CSN	U12	AA14
PB36A	4	T	VREF2_4	W13	AC14
PB36B	4	C	D7	U13	AB14
PB37A	4	T	D5	AA12	AF15
PB37B	4	C	D6	AB12	AE15
PB38A	4	T	BDQS38	T13	AD15
PB38B	4	C	D4	V13	AC15
PB39A	4	T		W14	AF16
PB39B	4	C	D3	U14	Y14
PB40A	4	T		Y13	AE16
PB40B	4	C	D2	V14	AB15
PB41A	4	T		AA13	AF17
PB41B	4	C	D1	AB13	AE17
PB42A	4	T		AA14	Y15

LFEC20/LFECP20 Logic Signal Connections: 484 & 672 fpBGA (Cont.)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PB42B	4	C		Y14	AA15
PB43A	4	T		Y15	AD17
PB43B	4	C		W15	Y16
PB44A	4	T		V15	AD18
PB44B	4	C		T14	AC16
PB45A	4	T		AB14	AE18
PB45B	4	C		AB15	AF18
PB46A	4	T	BDQS46	AB16	AD16
PB46B	4	C		AA15	AB16
PB47A	4	T		AB17	AF19
PB47B	4	C		AA16	AA16
PB48A	4	T		AB18	AA17
PB48B	4	C		AA17	Y17
PB49A	4	T		AB19	AF21
PB49B	4	C		AA18	AF20
PB50A	4	T		W16	AE21
PB50B	4	C		U15	AC17
PB51A	4	T		V16	AF22
PB51B	4	C		U16	AB17
PB52A	4	T		Y17	AE22
PB52B	4	C		V17	AA18
PB53A	4	T		AB20	AE19
PB53B	4	C		AA19	AE20
PB54A	4	T	BDQS54	Y16	AA19
PB54B	4	C		W17	Y18
PB55A	4	T		AA20	AF23
PB55B	4	C		Y19	AA20
PB56A	4	T		Y18	AC18
PB56B	4	C		W18	AB18
PB57A	4	T		T17	AF24
PB57B	4	C		U17	AE23
PR48B	3	C	VREF2_3	W20	AC23
PR48A	3	T	VREF1_3	Y20	AC24
PR47B	3	C		AA21	AC25
PR47A	3	T		AB21	AC26
PR46B	3	C		W19	AB25
PR46A	3	T		V19	AA25
PR45B	3	C		Y21	AB26
PR45A	3	T	RDQS45	AA22	AA26
PR44B	3	C	RLM0_PLLC_IN_A	V20	W23
PR44A	3	T	RLM0_PLLT_IN_A	U20	W24
PR43B	3	C	RLM0_PLLC_FB_A	W21	W22
PR43A	3	T	RLM0_PLLT_FB_A	Y22	W21
PR42B	3	C	DI	V21	Y25

LFEC20/LFECP20 Logic Signal Connections: 484 & 672 fpBGA (Cont.)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PR42A	3	T	DOUT/CSON	W22	Y26
PR41B	3	C	BUSY	U21	W25
PR41A	3	T	D0	V22	W26
CFG2	3			T19	V24
CFG1	3			U19	V21
CFG0	3			U18	V23
PROGRAMN	3			V18	V22
CCLK	3			T20	V20
INITN	3			T21	V25
DONE	3			R20	U20
PR39B	3	C			V26
PR39A	3	T			U26
PR38B	3	C			U24
PR38A	3	T			U25
PR37B	3	C		T18	U23
PR37A	3	T		R17	U22
PR36B	3	C		R19	U21
PR36A	3	T	RDQS36	R18	T21
PR35B	3	C		U22	T25
PR35A	3	T		T22	T26
PR34B	3	C		R21	T22
PR34A	3	T		R22	T23
PR33B	3	C		P20	T24
PR33A	3	T		N20	R23
PR32B	3	C		P19	R25
PR32A	3	T		P18	R24
PR31B	3	C		P21	R26
PR31A	3	T		P22	P26
PR30B	3	C		N21	R21
PR30A	3	T		N22	R22
PR29B	3	C		N19	P25
PR29A	3	T		N18	P24
PR28B	3	C		M21	P23
PR28A	3	T	RDQS28	L20	P22
PR27B	3	C		L21	N26
PR27A	3	T		M20	M26
PR26B	3	C		M18	N21
PR26A	3	T		M19	P21
PR25B	3	C		M22	N23
PR25A	3	T		L22	N22
PR24B	3	C		K22	N25
PR24A	3	T		K21	N24
PR22B	2	C	PCLKC2_0	J22	L26
PR22A	2	T	PCLKT2_0	J21	K26

LFEC20/LFECP20 Logic Signal Connections: 484 & 672 fpBGA (Cont.)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PR21B	2	C		H22	M22
PR21A	2	T		H21	M23
PR20B	2	C		L19	M25
PR20A	2	T		L18	M24
PR19B	2	C		K20	M21
PR19A	2	T	RDQS19	J20	L21
PR18B	2	C		K19	L22
PR18A	2	T		K18	L23
PR17B	2	C		G22	L25
PR17A	2	T		F22	L24
PR16B	2	C		F21	K25
PR16A	2	T		E22	J25
PR15B	2	C		E21	J26
PR15A	2	T		D22	H26
PR14B	2	C		G21	H25
PR14A	2	T		G20	J24
PR13B	2	C		J18	K21
PR13A	2	T		H19	K22
PR12B	2	C		J19	K20
PR12A	2	T		H20	J20
PR11B	2	C		H17	K23
PR11A	2	T		H18	K24
PR9B	2	C	RUM0_PLLC_FB_A	D21	F25
PR9A	2	T	RUM0_PLLT_FB_A	C22	G25
PR8B	2	C	RUM0_PLLC_IN_A	G19	H23
PR8A	2	T	RUM0_PLLT_IN_A	G18	H24
PR7B	2	C		F20	H21
PR7A	2	T		F19	G21
PR6B	2	C		E20	D26
PR6A	2	T	RDQS6	D20	D25
PR5B	2	C		C21	F21
PR5A	2	T		C20	G22
PR4B	2	C		F18	G24
PR4A	2	T		E18	G23
PR3B	2	C		B22	C26
PR3A	2	T		B21	C25
PR2B	2	C	VREF1_2	E19	E23
PR2A	2	T	VREF2_2	D19	D23
PT57B	1	C		G17	A24
PT57A	1	T		F17	A23
PT56B	1	C		D18	E18
PT56A	1	T		C18	D19
PT55B	1	C		C19	F19

LFEC20/LFECP20 Logic Signal Connections: 484 & 672 fpBGA (Cont.)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PT55A	1	T		B20	B22
PT54B	1	C		D17	G19
PT54A	1	T	TDQS54	C16	B21
PT53B	1	C		B19	D18
PT53A	1	T		A20	C18
PT52B	1	C		E17	F18
PT52A	1	T		C17	A22
PT51B	1	C		F16	G18
PT51A	1	T		E16	A21
PT50B	1	C		F15	E17
PT50A	1	T		D16	B17
PT49B	1	C		B18	C17
PT49A	1	T		A19	D17
PT48B	1	C		B17	F17
PT48A	1	T		A18	E20
PT47B	1	C		B16	G17
PT47A	1	T		A17	B20
PT46B	1	C		B15	E16
PT46A	1	T	TDQS46	A16	A20
PT45B	1	C		A15	A19
PT45A	1	T		A14	B19
PT44B	1	C		G14	D16
PT44A	1	T		E15	C16
PT43B	1	C		D15	F16
PT43A	1	T		C15	A18
PT42B	1	C		C14	G16
PT42A	1	T		B14	B18
PT41B	1	C		A13	A17
PT41A	1	T		B13	A16
PT40B	1	C		E14	D15
PT40A	1	T		C13	B16
PT39B	1	C		F14	E15
PT39A	1	T		D14	C15
PT38B	1	C		E13	F15
PT38A	1	T	TDQS38	G13	G15
PT37B	1	C		A12	B15
PT37A	1	T		B12	A15
PT36B	1	C		F13	E14
PT36A	1	T		D13	G14
PT35B	1	C	VREF2_1	F12	D14
PT35A	1	T	VREF1_1	D12	E13
PT34B	1	C		F11	F14
PT34A	1	T		C12	C14
PT33B	0	C	PCLKC0_0	A11	B14

LFEC20/LFECP20 Logic Signal Connections: 484 & 672 fpBGA (Cont.)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PT33A	0	T	PCLKT0_0	A10	A14
PT32B	0	C	VREF1_0	E12	D13
PT32A	0	T	VREF2_0	E11	C13
PT31B	0	C		B11	A13
PT31A	0	T		C11	B13
PT30B	0	C		B9	F13
PT30A	0	T	TDQS30	B10	F12
PT29B	0	C		A9	A12
PT29A	0	T		A8	B12
PT28B	0	C		D11	A11
PT28A	0	T		C10	B11
PT27B	0	C		A7	D12
PT27A	0	T		A6	C12
PT26B	0	C		B7	B10
PT26A	0	T		B8	A10
PT25B	0	C		A5	G12
PT25A	0	T		B6	A9
PT24B	0	C		G10	E12
PT24A	0	T		E10	B9
PT23B	0	C		F10	F11
PT23A	0	T		D10	A8
PT22B	0	C		G9	D11
PT22A	0	T	TDQS22	E9	C11
PT21B	0	C		C9	B8
PT21A	0	T		C8	B7
PT20B	0	C		F9	E11
PT20A	0	T		D9	A7
PT19B	0	C		F8	G11
PT19A	0	T		D7	C7
PT18B	0	C		D8	G10
PT18A	0	T		C7	C6
PT17B	0	C		A4	C10
PT17A	0	T		B4	D10
PT16B	0	C		C4	F10
PT16A	0	T		C5	A6
PT15B	0	C		D6	E10
PT15A	0	T		B5	C9
PT14B	0	C		E6	G9
PT14A	0	T	TDQS14	C6	D9
PT13B	0	C		A3	A5
PT13A	0	T		B3	A4
PT12B	0	C		F6	F9
PT12A	0	T		D5	B6
PT11B	0	C		F7	E9

LFEC20/LFECP20 Logic Signal Connections: 484 & 672 fpBGA (Cont.)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PT11A	0	T		E8	C8
PT10B	0	C		G6	G8
PT10A	0	T		E7	B5
PT9B	0	C			A3
PT9A	0	T			A2
PT8B	0	C			F8
PT8A	0	T			B4
PT7B	0	C			E8
PT7A	0	T			B3
PT6B	0	C			D8
PT6A	0	T	TDQS6		G7
PT5B	0	C			C4
PT5A	0	T			C5
PT4B	0	C			E7
PT4A	0	T			D4
PT3B	0	C			F7
PT3A	0	T			D6
PT2B	0	C			D7
PT2A	0	T			E6

LatticeECP/EC Family Data Sheet Ordering Information

June 2004 **Advance Data Sheet**

Part Number Description

LFXXX XX – X – X XXXXXX X

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

Logic Capacity
- 1.5K LUTs = 1*
- 3K LUTs = 3*
- 6K LUTs = 6
- 10K LUTs = 10
- 15K LUTs = 15
- 20K LUTs = 20
- 40K LUTs = 40

Supply Voltage
- E = 1.2V

Note: Parts dual marked per table below.

*Not available in the LatticeECP Family.

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
- F900 = 900-ball fpBGA

Speed
- 3 = Slowest
- 4
- 5 = Fastest

Ordering Information

LatticeEC Commercial

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC1E-3 Q208C	112	-3	PQFP	208	COM	1.5K
LFEC1E-4 Q208C	112	-4	PQFP	208	COM	1.5K
LFEC1E-5 Q208C	112	-5	PQFP	208	COM	1.5K
LFEC1E-3 T144C	97	-3	TQFP	144	COM	1.5K
LFEC1E-4 T144C	97	-4	TQFP	144	COM	1.5K
LFEC1E-5 T144C	97	-5	TQFP	144	COM	1.5K
LFEC1E-3 T100C	65	-3	TQFP	100	COM	1.5K
LFEC1E-4 T100C	65	-4	TQFP	100	COM	1.5K
LFEC1E-5 T100C	65	-5	TQFP	100	COM	1.5K

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

LatticeEC Commercial (Continued)

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC3E-4 T144C	97	-4	TQFP	144	COM	3.1K
LFEC3E-5 T144C	97	-5	TQFP	144	COM	3.1K
LFEC3E-3 T100C	65	-3	TQFP	100	COM	3.1K
LFEC3E-4 T100C	65	-4	TQFP	100	COM	3.1K
LFEC3E-5 T100C	65	-5	TQFP	100	COM	3.1K
LFEC6E-3 F484C	224	-3	fpBGA	484	COM	6.1K
LFEC6E-4 F484C	224	-4	fpBGA	484	COM	6.1K
LFEC6E-5 F484C	224	-5	fpBGA	484	COM	6.1K
LFEC6E-3 F256C	192	-3	fpBGA	256	COM	6.1K
LFEC6E-4 F256C	192	-4	fpBGA	256	COM	6.1K
LFEC6E-5 F256C	192	-5	fpBGA	256	COM	6.1K
LFEC6E-3 Q208C	145	-3	PQFP	208	COM	6.1K
LFEC6E-4 Q208C	145	-4	PQFP	208	COM	6.1K
LFEC6E-5 Q208C	145	-5	PQFP	208	COM	6.1K
LFEC6E-3 T144C	97	-3	TQFP	144	COM	6.1K
LFEC6E-4 T144C	97	-4	TQFP	144	COM	6.1K
LFEC6E-5 T144C	97	-5	TQFP	144	COM	6.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC10E-3 F484C	288	-3	fpBGA	484	COM	10.2K
LFEC10E-4 F484C	288	-4	fpBGA	484	COM	10.2K
LFEC10E-5 F484C	288	-5	fpBGA	484	COM	10.2K
LFEC10E-3 F256C	192	-3	fpBGA	256	COM	10.2K
LFEC10E-4 F256C	192	-4	fpBGA	256	COM	10.2K
LFEC10E-5 F256C	192	-5	fpBGA	256	COM	10.2K
LFEC10E-3 Q208C	145	-3	PQFP	208	COM	10.2K
LFEC10E-4 Q208C	145	-4	PQFP	208	COM	10.2K
LFEC10E-5 Q208C	145	-5	PQFP	208	COM	10.2K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC15E-3 F484C	352	-3	fpBGA	484	COM	15.3K
LFEC15E-4 F484C	352	-4	fpBGA	484	COM	15.3K
LFEC15E-5 F484C	352	-5	fpBGA	484	COM	15.3K
LFEC15E-3 F256C	192	-3	fpBGA	256	COM	15.3K
LFEC15E-4 F256C	192	-4	fpBGA	256	COM	15.3K
LFEC15E-5 F256C	192	-5	fpBGA	256	COM	15.3K

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

LatticeEC Commercial (Continued)

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC40E-3 F900C	576	-3	fpBGA	900	COM	40.9K
LFEC40E-4 F900C	576	-4	fpBGA	900	COM	40.9K
LFEC40E-5 F900C	576	-5	fpBGA	900	COM	40.9K
LFEC40E-3 F672C	496	-3	fpBGA	672	COM	40.9K
LFEC40E-4 F672C	496	-4	fpBGA	672	COM	40.9K
LFEC40E-5 F672C	496	-5	fpBGA	672	COM	40.9K

LatticeECP Commercial

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP6E-3 F484C	224	-3	fpBGA	484	COM	6.1K
LFECP6E-4 F484C	224	-4	fpBGA	484	COM	6.1K
LFECP6E-5 F484C	224	-5	fpBGA	484	COM	6.1K
LFECP6E-3 F256C	192	-3	fpBGA	256	COM	6.1K
LFECP6E-4 F256C	192	-4	fpBGA	256	COM	6.1K
LFECP6E-5 F256C	192	-5	fpBGA	256	COM	6.1K
LFECP6E-3 Q208C	145	-3	PQFP	208	COM	6.1K
LFECP6E-4 Q208C	145	-4	PQFP	208	COM	6.1K
LFECP6E-5 Q208C	145	-5	PQFP	208	COM	6.1K
LFECP6E-3 T144C	97	-3	TQFP	144	COM	6.1K
LFECP6E-4 T144C	97	-4	TQFP	144	COM	6.1K
LFECP6E-5 T144C	97	-5	TQFP	144	COM	6.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP10E-3 F484C	288	-3	fpBGA	484	COM	10.2K
LFECP10E-4 F484C	288	-4	fpBGA	484	COM	10.2K
LFECP10E-5 F484C	288	-5	fpBGA	484	COM	10.2K
LFECP10E-3 F256C	192	-3	fpBGA	256	COM	10.2K
LFECP10E-4 F256C	192	-4	fpBGA	256	COM	10.2K
LFECP10E-5 F256C	192	-5	fpBGA	256	COM	10.2K
LFECP10E-3 Q208C	145	-3	PQFP	208	COM	10.2K
LFECP10E-4 Q208C	145	-4	PQFP	208	COM	10.2K
LFECP10E-5 Q208C	145	-5	PQFP	208	COM	10.2K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP15E-3 F484C	352	-3	fpBGA	484	COM	15.3K
LFECP15E-4 F484C	352	-4	fpBGA	484	COM	15.3K
LFECP15E-5 F484C	352	-5	fpBGA	484	COM	15.3K
LFECP15E-3 F256C	192	-3	fpBGA	256	COM	15.3K
LFECP15E-4 F256C	192	-4	fpBGA	256	COM	15.3K
LFECP15E-5 F256C	192	-5	fpBGA	256	COM	15.3K

LatticeECP Commercial (Continued)

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP20E-3 F676C	400	-3	fpBGA	676	COM	19.7K
LFECP20E-4 F676C	400	-4	fpBGA	676	COM	19.7K
LFECP20E-5 F676C	400	-5	fpBGA	676	COM	19.7K
LFECP20E-3 F484C	360	-3	fpBGA	484	COM	19.7K
LFECP20E-4 F484C	360	-4	fpBGA	484	COM	19.7K
LFECP20E-5 F484C	360	-5	fpBGA	484	COM	19.7K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP40E-3 F896C	576	-3	fpBGA	896	COM	40.9K
LFECP40E-4 F896C	576	-4	fpBGA	896	COM	40.9K
LFECP40E-5 F896C	576	-5	fpBGA	896	COM	40.9K
LFECP40E-3 F676C	496	-3	fpBGA	676	COM	40.9K
LFECP40E-4 F676C	496	-4	fpBGA	676	COM	40.9K
LFECP40E-5 F676C	496	-5	fpBGA	676	COM	40.9K

LatticeEC Industrial

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC1E-3 Q208I	112	-3	PQFP	208	IND	1.5K
LFEC1E-4 Q208I	112	-4	PQFP	208	IND	1.5K
LFEC1E-3 T144I	97	-3	TQFP	144	IND	1.5K
LFEC1E-4 T144I	97	-4	TQFP	144	IND	1.5K
LFEC1E-3 T100I	65	-3	TQFP	100	IND	1.5K
LFEC1E-4 T100I	65	-4	TQFP	100	IND	1.5K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC3E-3 F256I	160	-3	fpBGA	256	IND	3.1K
LFEC3E-4 F256I	160	-4	fpBGA	256	IND	3.1K
LFEC3E-3 Q208I	145	-3	PQFP	208	IND	3.1K
LFEC3E-4 Q208I	145	-4	PQFP	208	IND	3.1K
LFEC3E-3 T144I	97	-3	TQFP	144	IND	3.1K
LFEC3E-4 T144I	97	-4	TQFP	144	IND	3.1K
LFEC3E-3 T100I	65	-3	TQFP	100	IND	3.1K
LFEC3E-4 T100I	65	-4	TQFP	100	IND	3.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC6E-3 F484I	224	-3	fpBGA	484	IND	6.1K
LFEC6E-4 F484I	224	-4	fpBGA	484	IND	6.1K
LFEC6E-3 F256I	192	-3	fpBGA	256	IND	6.1K
LFEC6E-4 F256I	192	-4	fpBGA	256	IND	6.1K
LFEC6E-3 Q208I	145	-3	PQFP	208	IND	6.1K
LFEC6E-4 Q208I	145	-4	PQFP	208	IND	6.1K
LFEC6E-3 T144I	97	-3	TQFP	144	IND	6.1K

LatticeEC Industrial (Continued)

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC1E-3 Q208I	112	-3	PQFP	208	IND	1.5K
LFEC1E-4 Q208I	112	-4	PQFP	208	IND	1.5K
LFEC1E-3 T144I	97	-3	TQFP	144	IND	1.5K
LFEC1E-4 T144I	97	-4	TQFP	144	IND	1.5K
LFEC1E-3 T100I	65	-3	TQFP	100	IND	1.5K
LFEC1E-4 T100I	65	-4	TQFP	100	IND	1.5K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC6E-4 T144I	97	-4	TQFP	144	IND	6.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC10E-3 F484I	288	-3	fpBGA	484	IND	10.2K
LFEC10E-4 F484I	288	-4	fpBGA	484	IND	10.2K
LFEC10E-3 F256I	192	-3	fpBGA	256	IND	10.2K
LFEC10E-4 F256I	192	-4	fpBGA	256	IND	10.2K
LFEC10E-3 P208I	145	-3	PQFP	208	IND	10.2K
LFEC10E-4 P208I	145	-4	PQFP	208	IND	10.2K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC15E-3 F484I	352	-3	fpBGA	484	IND	15.3K
LFEC15E-4 F484I	352	-4	fpBGA	484	IND	15.3K
LFEC15E-3 F256I	192	-3	fpBGA	256	IND	15.3K
LFEC15E-4 F256I	192	-4	fpBGA	256	IND	15.3K

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

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFEC40E-3 F900I	576	-3	fpBGA	900	IND	40.9K
LFEC40E-4 F900I	576	-4	fpBGA	900	IND	40.9K
LFEC40E-3 F672I	496	-3	fpBGA	672	IND	40.9K
LFEC40E-4 F672I	496	-4	fpBGA	672	IND	40.9K

LatticeECP Industrial

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP6E-3 F484I	224	-3	fpBGA	484	IND	6.1K
LFECP6E-4 F484I	224	-4	fpBGA	484	IND	6.1K
LFECP6E-3 F256I	192	-3	fpBGA	256	IND	6.1K
LFECP6E-4 F256I	192	-4	fpBGA	256	IND	6.1K
LFECP6E-3 Q208I	145	-3	PQFP	208	IND	6.1K

LatticeECP Industrial (Continued)

LFECP6E-4 Q208I	145	-4	PQFP	208	IND	6.1K
LFECP6E-3 T144I	97	-3	TQFP	144	IND	6.1K
LFECP6E-4 T144I	97	-4	TQFP	144	IND	6.1K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP10E-3 F484I	288	-3	fpBGA	484	IND	10.2K
LFECP10E-4 F484I	288	-4	fpBGA	484	IND	10.2K
LFECP10E-3 F256I	192	-3	fpBGA	256	IND	10.2K
LFECP10E-4 F256I	192	-4	fpBGA	256	IND	10.2K
LFECP10E-3 Q208I	145	-3	PQFP	208	IND	10.2K
LFECP10E-4 Q208I	145	-4	PQFP	208	IND	10.2K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP15E-3 F484I	352	-3	fpBGA	484	IND	15.3K
LFECP15E-4 F484I	352	-4	fpBGA	484	IND	15.3K
LFECP15E-3 F256I	192	-3	fpBGA	256	IND	15.3K
LFECP15E-4 F256I	192	-4	fpBGA	256	IND	15.3K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP20E-3 F676I	400	-3	fpBGA	676	IND	19.7K
LFECP20E-4 F676I	400	-4	fpBGA	676	IND	19.7K
LFECP20E-3 F484I	360	-3	fpBGA	484	IND	19.7K
LFECP20E-4 F484I	360	-4	fpBGA	484	IND	19.7K

Part Number	I/Os	Grade	Package	Pins	Temp.	LUTs
LFECP40E-3 F896I	576	-3	fpBGA	896	IND	40.9K
LFECP40E-4 F896I	576	-4	fpBGA	896	IND	40.9K
LFECP40E-3 F676I	496	-3	fpBGA	676	IND	40.9K
LFECP40E-4 F676I	496	-4	fpBGA	676	IND	40.9K

LatticeECP/EC Family Data Sheet Supplemental Information

June 2004 **Advance Data Sheet**

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)
- ispTRACY Internal Logic Analyzer Guide (TN1054)
- LatticeECP/EC sysCLOCK PLL Design and Usage Guide (TN1049)
- Memory Usage Guide for LatticeECP/EC Devices (TN1051)
- LatticeECP/EC DDR Usage Guide (TN1050)
- Estimating Power Using Power Calculator for LatticeECP/EC Devices (TN1052)
- sysDSP/MAC Usage Guide (TN1057)
- LatticeECP/EC sysCONFIG Usage Guide (TN1053)
- IEEE 1149.1 Boundary Scan Testability in Lattice Devices

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

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

Section II. LatticeECP/EC Family Technical Notes

LatticeECP/EC sysIO Usage Guide

June 2004 Technical Note TN1056

Introduction

The LatticeECP™ and LatticeEC™ 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 sysIO interface contains multiple Programmable I/O Cells (PIC) blocks. Each PIC contains two Programmable I/Os (PIO), PIOA and PIOB, connected to their respective sysIO Buffers. Two adjacent PIOs can be joined to provide a differential I/O pair (labeled as "T" and "C").

Each Programmable I/O (PIO) includes a sysIO Buffer and I/O Logic (IOLOGIC). The LatticeECP/EC 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 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 LatticeECP/EC Family Data Sheet.

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

Supported sysIO Standards

The LatticeECP/EC 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) and open drain. Other single-ended standards supported include SSTL and HSTL. Differential standards supported include LVDS, BLVDS, LVPECL, Differential SSTL and Differential HSTL. Table 7-1 lists the sysIO standards supported in the LatticeECP/EC devices.

Table 7-1. Supported sysIO Standards

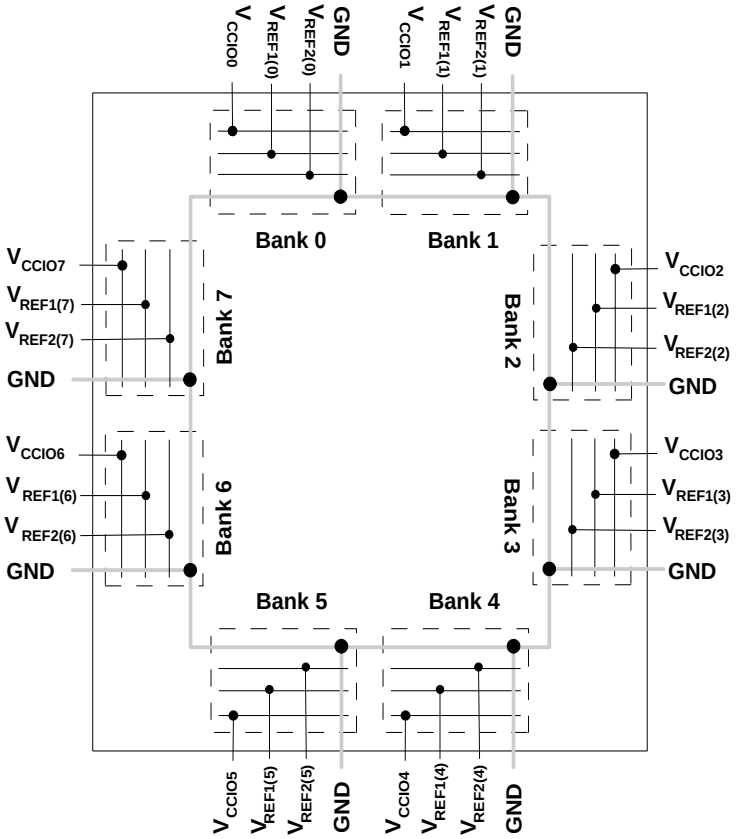
Standard	V_{CCIO}			V_{REF} (V)		
	Min.	Typ.	Max.	Min.	Typ.	Max.
LVCMOS 3.3	3.135	3.3	3.465	—	—	—
LVCMOS 2.5	2.375	2.5	2.625	—	—	—
LVCMOS 1.8	1.71	1.8	1.89	—	—	—
LVCMOS 1.5	1.425	1.5	1.575	—	—	—
LVCMOS 1.2	1.14	1.2	1.26	—	—	—
LVTTL	3.135	3.3	3.465	—	—	—
PCI	3.135	3.3	3.465	—	—	—
SSTL18 Class I	1.71	1.8	1.89	1.15	1.25	1.35
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	—	—	—

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

sysIO Banking Scheme

LatticeECP/EC 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 7-1 shows the eight banks and their associated supplies.

Figure 7-1. 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 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 7-2 shows the sysIO standards that the user can mix in the same bank.

Table 7-2. Mixed Voltage Support

	Input sysIO Standards					Output sysIO Standards				
V_{CCIO}	**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 7-3. 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, 2]	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, 2]	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, 2]	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, 2]
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.
2. This mode uses LVDS25E mode with alternative resistor pack values.

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 I/O voltage. The user must consider the maximum allowable current per bank and the package thermal limit current when selecting the drive strength.

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.

In addition to these configurations, each individual sysIO buffer also has a tristate control capability along with open drain control.

Open Drain Control

When configured as LVCMOS or LVTTL, the drivers support open drain operation on each I/O independently. When an I/O is configured as an open drain, the pull-up transistors on the pad are permanently disabled.

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

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.

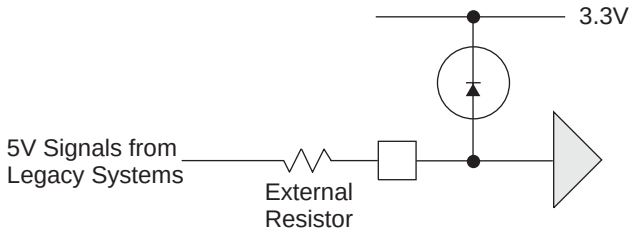
Programmable Input Delay

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.

5V Tolerant Input Buffers

All the I/Os have a clamp diode that is used to clamp the voltage at the input to $V_{CCIO.}$ This is especially useful for PCI I/O standards. This clamp diode can be used along with an external resistor to make an input 5V tolerant.

Figure 7-2. 5V Tolerant Input Buffer



The value of this external resistor will depend on the PCI clamp diode characteristics. Refer to the LatticeECP/EC Data Sheet for the voltage vs. current data for the PCI clamp diode. This data can be used to calculate the value of the external resistor as follows:

External Resistor Value = (5V - Max. Input Voltage)/Current at Max. Input Voltage

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 7-4 lists the available I/O types.

Table 7-4. O_TYPE Attribute Values

sysIO Signaling Standard	IO_TYPE
Default	LVCMOS18 (for UV) LVCMOS12 (for LV)
LVDS 2.5V	LVDS25
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[2] HSTL18D_III[2]
HSTL 15 Class I and III	HSTL15_I HSTL15_III
Differential HSTL 15 Class I and III	HSTL15D_I[2] HSTL15D_III[2]
SSTL 33 Class I and II	SSTL33_I SSTL33_II
Differential SSTL 33 Class I and II	SSTL33D_I[2] SSTL3D_II[2]
SSTL 25 Class I and II	SSTL25_I SSTL25_II
Differential SSTL 25 Class I and II	SSTL25D_I[2] SSTL25D_II[2]
SSTL 18 Class I	SSTL18_I[2]
Differential SSTL 18 Class I	SSTL18D_I
LVTTL	LVTTL33_OD
3.3V LVCMOS	LVCMOS33
2.5V LVCMOS	LVCMOS25
1.8V LVCMOS	LVCMOS18
1.5V LVCMOS	LVCMOS15
1.2V LVCMOS	LVCMOS12
3.3V LVCMOS, Open Drain	LVCMOS33_OD[2]

Table 7-4. O_TYPE Attribute Values (Continued)

sysIO Signaling Standard	IO_TYPE
2.5V LVCMOS, Open Drain	LVCMOS25_OD[2]
1.8V LVCMOS, Open Drain	LVCMOS18_OD[2]
1.5V LVCMOS, Open Drain	LVCMOS15_OD[2]
1.2V LVCMOS, Open Drain	LVCMOS12_OD[2]
3.3V PCI	PCI33

1. These differential standards are implemented by using complementary LVCMOS driver with external resistor pack.
2. IO_TYPE with a “OD” at the end indicate that the I/O is configures as an open drain output.

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

The programmable drive available on a pad will depend on the $V_{CCIO.}$ Table 7-5 shows the drive strength available for different $V_{CCIO.}$

Table 7-5. 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				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 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

FIXEDELAY

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

Values: TRUE, FALSE
Default: FALSE

DIN/DOUT

This attribute can be used when an I/O register needs to be assigned. Using DIN asserts an input register and using the DOUT attribute asserts an output register in the design. By default, the software will try to assign the I/O registers if applicable.

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 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.
- There are no restrictions on the number of I/Os that can support LVDS. LVDS can only be assigned to the TRUE pad. Please see the LatticeECP/EC Family Data Sheet 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 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 Family Data Sheet for a more detailed explanation of these LVDS implementations.

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 Family Data Sheet for further information on LVPECL implementation.

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 Family Data Sheet 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.

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 Family Data Sheet for a detailed explanation of Differential HSTL and SSTL implementation.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-408-826-6002 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Appendix A. HDL Attributes for Synplicity and Exemplar

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 you to use. Below are a list of all the sysIO attributes syntax and examples for Exemplar and Synplicity. This section only lists the sysIO buffer attributes for these devices. You can refer to the Exemplar and Synplicity user manuals for a complete list of synthesis attributes. These manuals are available through ispLEVER Software Help.

VHDL Synplicity/Exemplar

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

Syntax

Table 7-6. VHDL Attribute Syntax for Synplicity and Exemplar

Attribute	Syntax
IO_TYPE	Attribute IO_TYPE: string; Attribute IO_TYPE of Pinname: signal is “IO_TYPE 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”;
SLEWRATE	Attribute PULLMODE: string; Attribute PULLMODE of Pinname: signal is “Pullmode 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”;

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 “PCICLAMP”;

ATTRIBUTE PULLMODE OF portB: SIGNAL IS “UP”;

SLEWRATE
--*Attribute Declaration****

ATTRIBUTE SLEWRATE : string;

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

ATTRIBUTE SLEWRATE OF portB: SIGNAL IS “FAST”;

FIXEDDELAY
--*Attribute Declaration****

ATTRIBUTE FIXEDDELAY: string;

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

ATTRIBUTE FIXEDDELAY OF portB: SIGNAL IS “TRUE”;

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

ATTRIBUTE din : string;

ATTRIBUTE dout : string;

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

ATTRIBUTE din OF input_vector: SIGNAL IS “ “;

ATTRIBUTE dout OF output_vector: SIGNAL IS “ “;

LOC
--*Attribute Declaration****

ATTRIBUTE LOC : string;

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

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

Verilog Synplicity

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

Syntax

Table 7-7. Verilog Synplicity Attribute Syntax

Attribute	Syntax
IO_TYPE	PinType PinName /* synthesis IO_TYPE=”IO_Type Value”*/;
DRIVE	PinType PinName /* synthesis DRIVE=”Drive Value”*/;
PULLMODE	PinType PinName /* synthesis PULLMODE=”Pullmode Value”*/;
SLEWRATE	PinType PinName /* synthesis SLEWRATE=”Slewrate Value”*/;
FIXEDDELAY	PinType PinName /* synthesis FIXEDELAY=”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 portA /*synthesis IO_TYPE=”PCI33” PULLMODE =”PCICLAMP”*/;

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

DRIVE =”20”*/;

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

// Fixeddelay
input load /* synthesis FIXEDELAY=”TRUE” */;

// Place the flip-flops near the load input

input load /* synthesis din=”” */;

// Place the flip-flops near the outload output

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

//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 Exemplar

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

Syntax

Table 7-8. Verilog Exemplar Attribute Syntax

Attribute	Syntax
IO_TYPE	//exemplar attribute PinName IO_TYPE IO_TYPE Value
DRIVE	//exemplar attribute PinName DRIVE Drive Value
PULLMODE	//exemplar attribute PinName IO_TYPE Pullmode Value
SLEWRATE	//exemplar attribute PinName IO_TYPE Slewrate Value
FIXEDDELAY	//exemplar attribute PinName IO_TYPE Fixeddelay Value
LOC	//exemplar attribute PinName LOC pin_location

Example

//**IO_TYPE ***

//exemplar attribute portA IO_TYPE PCI33

//exemplar attribute portB IO_TYPE LVCMOS33

//exemplar attribute portC IO_TYPE SSTL25_II

//exemplar attribute portD IO_TYPE LVCMOS25_OD

//* Drive ***

//exemplar attribute portB DRIVE 20

//exemplar attribute portD DRIVE 8

//* Pullmode***

//exemplar attribute portB PULLMODE UP

//exemplar attribute portA PULLMODE PCICLAMP

//* Slewrate ***

//exemplar attribute portB SLEWRATE FAST

//exemplar attribute portD SLEWRATE SLOW

// ***Fixeddelay*

// exemplar attribute load FIXEDELAY TRUE

//*LOC***

//exemplar attribute portB loc E3

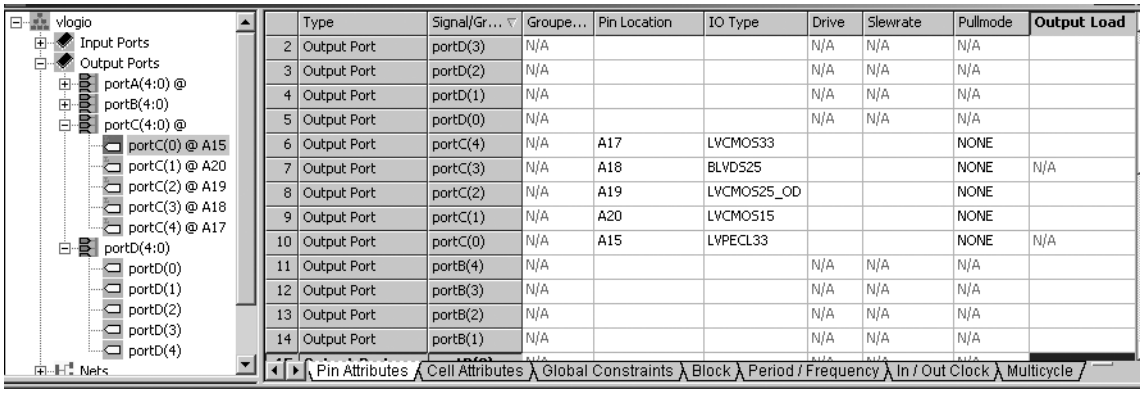
Appendix B. sysIO Attributes Using Preference Editor User Interface

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 preference file (.prf).

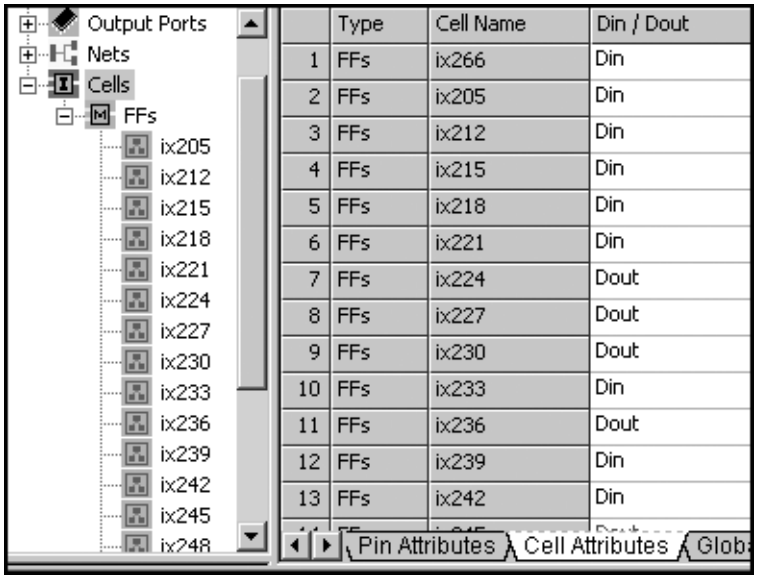
Figure 7-2 and Figure 7-3 show the Pin Attribute Sheet and the Cell Attribute Sheet views of the Preference Editor. For further information on how to use the Preference Editor, refer to the ispLEVER Help documentation located in the Help menu option of the software.

Figure 7-3. Pin Attributes Tab



	Type	Signal/Gr...	Groupe...	Pin Location	IO Type	Drive	Slewrate	Pullmode	Output Load
2	Output Port	portD(3)	N/A			N/A	N/A	N/A	
3	Output Port	portD(2)	N/A			N/A	N/A	N/A	
4	Output Port	portD(1)	N/A			N/A	N/A	N/A	
5	Output Port	portD(0)	N/A			N/A	N/A	N/A	
6	Output Port	portC(4)	N/A	A17	LVCMOS33			NONE	
7	Output Port	portC(3)	N/A	A18	BLVDS25			NONE	N/A
8	Output Port	portC(2)	N/A	A19	LVCMOS25_OD			NONE	
9	Output Port	portC(1)	N/A	A20	LVCMOS15			NONE	
10	Output Port	portC(0)	N/A	A15	LVPECL33			NONE	N/A
11	Output Port	portB(4)	N/A			N/A	N/A	N/A	
12	Output Port	portB(3)	N/A			N/A	N/A	N/A	
13	Output Port	portB(2)	N/A			N/A	N/A	N/A	
14	Output Port	portB(1)	N/A			N/A	N/A	N/A	

Figure 7-4. Cell Attributes Tab



	Type	Cell Name	Din / Dout
1	FFs	ix266	Din
2	FFs	ix205	Din
3	FFs	ix212	Din
4	FFs	ix215	Din
5	FFs	ix218	Din
6	FFs	ix221	Din
7	FFs	ix224	Dout
8	FFs	ix227	Dout
9	FFs	ix230	Dout
10	FFs	ix233	Din
11	FFs	ix236	Dout
12	FFs	ix239	Din
13	FFs	ix242	Din

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

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, DRIVE, PULLMODE, SLEWRATE.

Example
IOBUF PORT “port1” IO_TYPE=LVTTL33 DRIVE=8 PULLMODE=UP 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;

Memory Usage Guide for LatticeECP/EC Devices

May 2004 Technical Note TN1051

Introduction

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

The architecture of the LatticeECP/EC 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 utilize the memory primitives in two separate ways:

- **Via the Module Manager** – The Module Manager GUI allows users to specify the memory type and size that is required. The Module Manager takes this specification and constructs a netlist to implement the desired memory by using one or more of the memory primitives.
- **Via the PMI (Parameterizable Module Inferencing)** – PMI allows experienced users to skip the graphical interface and utilize the Configurable memory modules on the fly from the ispLEVER Project Navigator. The parameters and the control signals needed either in Verilog or VHDL can be set. The top-level design will have the parameters defined and signals declared so the interface can automatically generate the black box during synthesis.

The remainder of this document discusses these approaches, utilizing the Module Manager, PMI inference, memory modules and memory primitives.

Utilizing the Module Manager

Designers can utilize the Module Manager 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

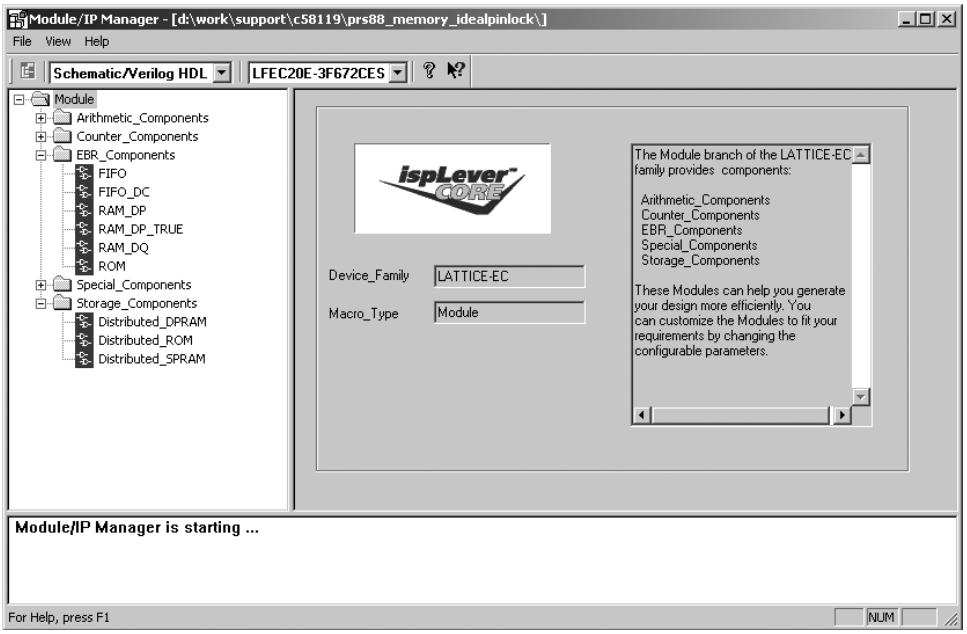
Module Manager Flow

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

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

This opens the Module Manager window as shown in Figure 8-1.

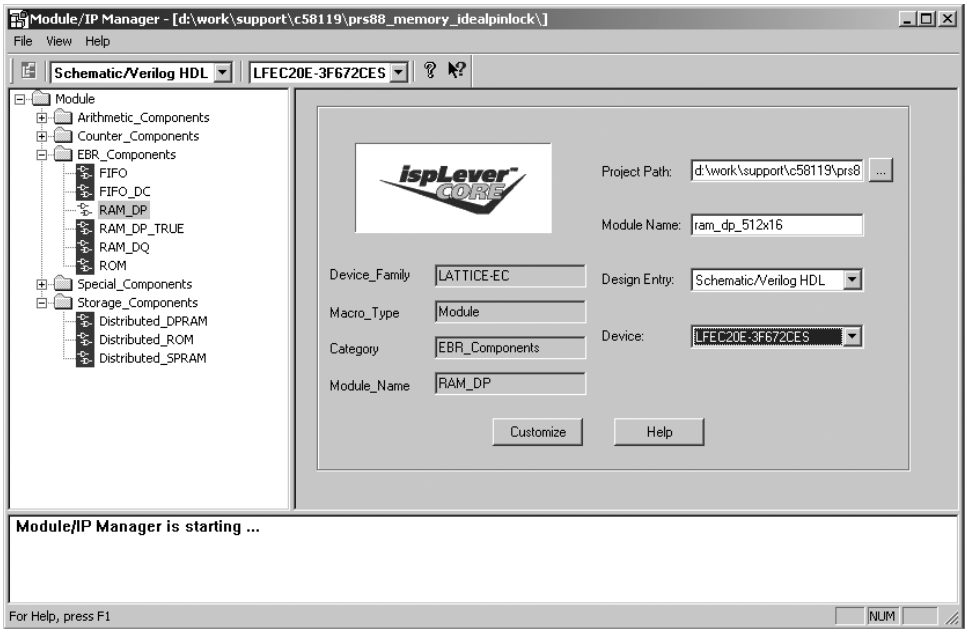
Figure 8-1. Module / IP Manager - Main Window



The left pane of this window has the Module Tree. The EBR-based Memory Modules are under the **EBR_Components** and the PFU-based Distributed Memory Modules are under **Storage_Components** as shown in Figure 8-1.

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 8-2.

Figure 8-2. Example Generating Pseudo Dual Port RAM (RAM_DP) Using Module Manager



In the right-hand pane, options like **Device Family**, **Macro Type**, **Category**, and **Module_Name** are device and selected module dependent. These cannot be changed in the Module Manager.

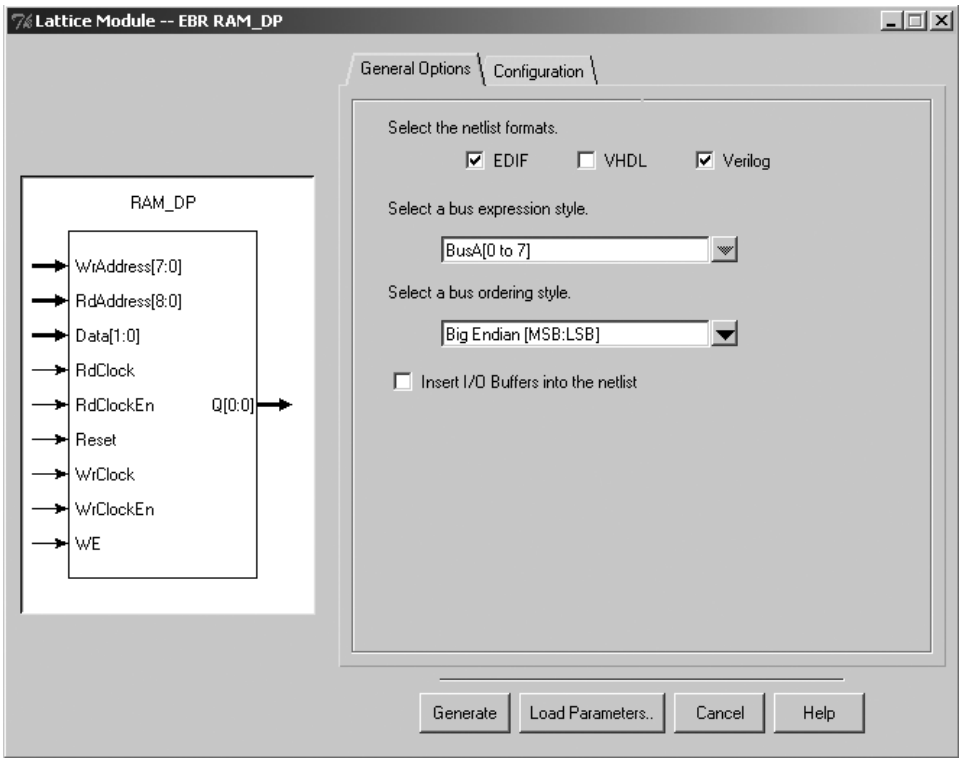
Users can change the directory where the generated module files will be placed by clicking the browse button in the **Project Path**.

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

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

The **Device** pull-down menu allows users to select different devices within the same family (LatticeEC in this example). Then click the **Customize** button. This opens another window (Figure 8-3) where the RAM can be customized.

Figure 8-3. Example Generating Pseudo Dual Port RAM (RAM_DP) Module Customization – General Options



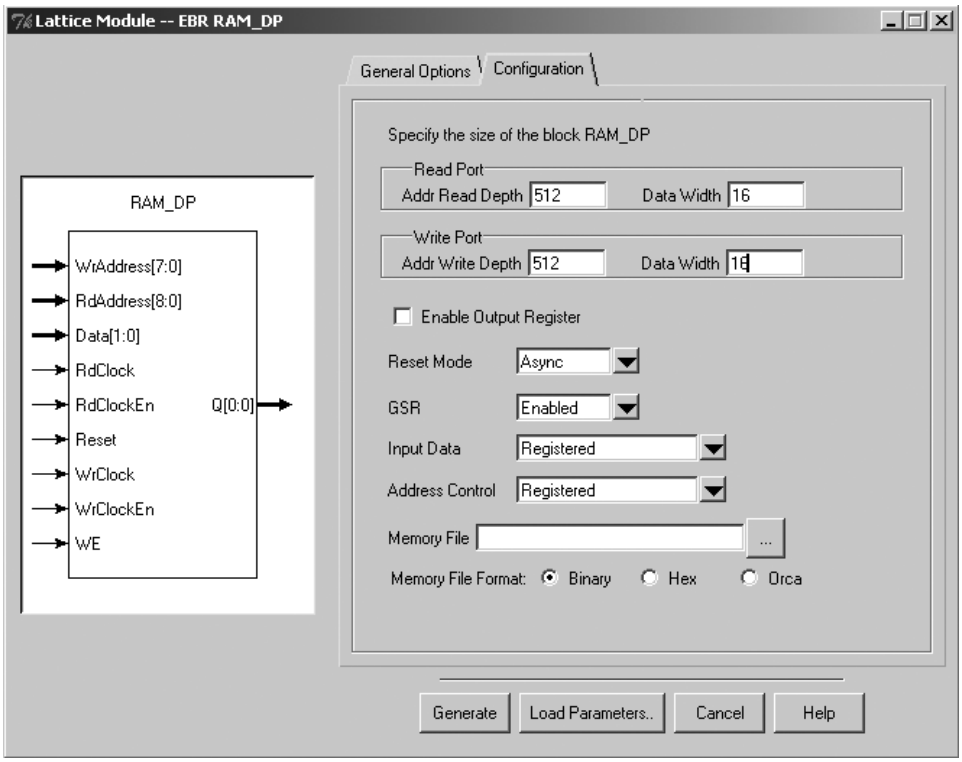
The the left-hand side of this window shows the block diagram of the module. The right-hand side has two tabs – the General Options tab and the Configuration tab.

In the **General Options** tab, users are given the option to generate an EDIF netlist along with Verilog-HDL or VHDL Module. The **Bus Expression Style** and the **Bus Ordering Style** can also be selected.

Checking **Insert I/O Buffers into the Netlist**, inserts buffers into the netlist that is generated.

The **Configuration** tab allows users to specify the address port sizes and data widths. Selecting the Configuration tab opens a window as shown in Figure 8-4.

Figure 8-4. 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 Configuration tab also has options to select the attributes. The **Reset Mode** can be selected to be Asynchronous Reset or Synchronous Reset. **GSR** or Global Set Reset can be selected 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 (ORCA®) 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.

Another button of importance here is **Load Parameters**. The Module Manager stores the parameters that the user has specified in an *<module_name>.lpc* file. This file is generated along with the module. Users can click on the Load Parameter button to load the parameters of a previously generated module to re-visit or make changes to it.

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.

Utilizing the PMI

Parameterizable Module Inferencing (PMI) allows experienced users to skip the graphical interface and utilize the configurable memory modules on-the-fly from the ispLEVER Project Navigator.

Users can instantiate a component of the memory module directly into their design, instead of using the module generated by the Module Manager. The instantiated component allows users to change the parameters and attributes of the module from within the Verilog-HDL or VHDL code. The instantiated component can be simulated too. The top level of the design will have the memory parameters defined and signals declared so the interface can automatically generate the black box during synthesis and ispLEVER can generate the netlist on the fly.

To include the component in the source code:

1. Create a design and open the source code in the Lattice Text Editor. The PMI flow is supported only through the Lattice Text Editor at present. With any other text editor, users can include the component using the Lattice Text Editor and then go back to the editor of their choice.
2. Click the cursor at the place where the PMI component needs to be inserted.
3. Click on the Templates Menu, and Select Insert. Alternatively, users can press F9 as the shortcut to this command.
4. This opens the **Insert Template** window. The left pane of this window is the **Template Files**, which allows users to choose the template for the device they are using. Also, users can select the Verilog-HDL or VHDL as per their requirement. The right pane of the window, which shows the **Template Names**, allows users to select the module component they are about to use.
5. Select the appropriate template in the Template Name pane and click **Insert**.
6. Click **Close** to close the **Insert Template** window.
7. The ports of the inserted template then need to be mapped appropriately in the user design. Once done, users can synthesize and place and route the design.

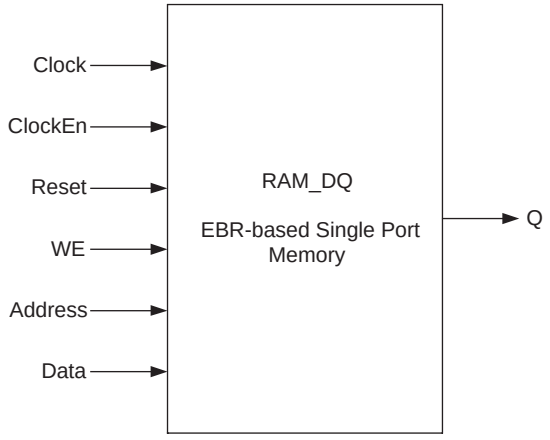
Memory Modules

Single Port RAM (RAM_DQ) – EBR Based

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

The Module Manager generates the memory module as shown in Figure 8-5.

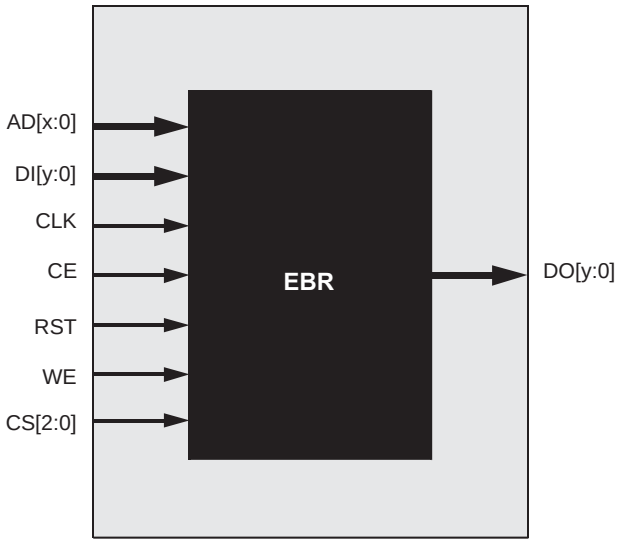
Figure 8-5. Single Port Memory Module generated by Module Manager



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 Module Manager 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 devices is shown in Figure 8-6.

Figure 8-6. Single Port RAM Primitive or RAM_DQ for LatticeECP/EC 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 8-1. The table lists the corresponding ports for the module generated by Module Manager and for the EBR RAM_DQ primitive.

Table 8-1. 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 8-2.

Table 8-2. 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 8-3 shows the various attributes available for the Single Port Memory (RAM_DQ). Some of these attributes are user selectable through the Module Manager GUI. For detailed attribute definitions, refer to Appendix A.

Table 8-3. Single Port RAM Attributes for LatticeECP/ EC Devices

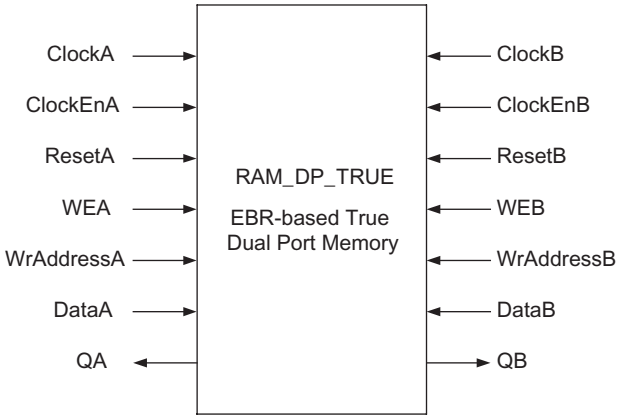
Attribute	Description	Values	Default Value	User Selectable Through Module Manager
DATA_WIDTH	Data Word Width	1, 2, 4, 9, 18, 36	1	YES
REGMODE	Register Mode (Pipelining)	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset type	ASYNC, SYNC	ASYNC	YES
CSDECODE	Chip Select Decode	000, 001, 010, 011, 100,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

True Dual Port RAM (RAM_DP_TRUE) – EBR Based

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

The Module Manager generates the memory module as shown in Figure 8-7.

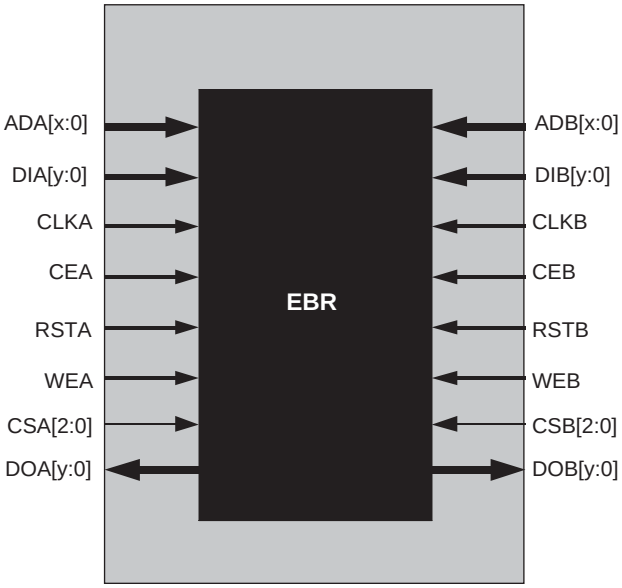
Figure 8-7. True Dual Port Memory Module Generated by Module Manager



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 devices, RAM_DP_TRUE, is shown in Figure 8-8.

Figure 8-8. True Dual Port RAM Primitive or RAM_DP_TRUE for LatticeECP/EC 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 8-4. The table lists the corresponding ports for the module generated by Module Manager and for the EBR RAM_DP_TRUE primitive.

Table 8-4. 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 8-5.

Table 8-5. True Dual Port Memory Sizes for 9K Memory for LatticeECP/EC 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 8-6 shows the various attributes available for True Dual Port Memory (RAM_DP_TRUE). Some of these attributes are user selectable through the Module Manager GUI. For detailed attribute definitions, refer to Appendix A.

Table 8-6. True Dual Port RAM Attributes for LatticeECP/EC

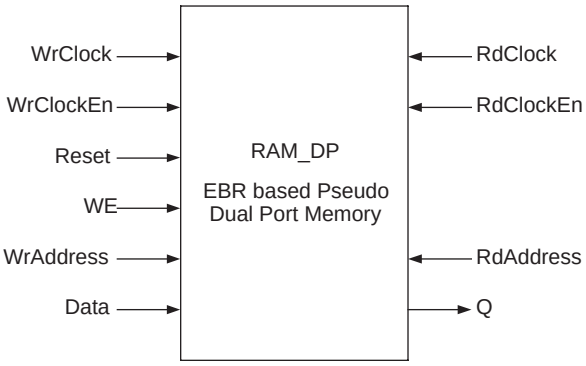
Attribute	Description	Values	Default Value	User Selectable Through Module Manager
DATA_WIDTH_A	Data Word Width Port A	1, 2, 4, 9, 18	1	YES
DATA_WIDTH_B	Data Word Width Port B	1, 2, 4, 9, 18	1	YES
REGMODE_A	Register Mode (Pipelining) for Port A	NOREG, OUTREG	NOREG	YES
REGMODE_B	Register Mode (Pipelining) 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	ENABLE, DISABLE	ENABLED	YES

Pseudo Dual Port RAM (RAM_DP) – EBR-Based

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

The Module Manager generates the memory module, as shown in Figure 8-9.

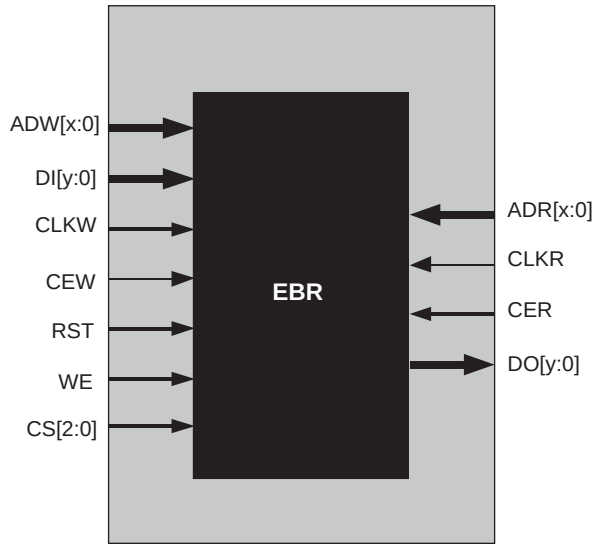
Figure 8-9. Pseudo Dual Port Memory Module Generated by Module Manager



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 devices is shown in Figure 8-10.

Figure 8-10. Pseudo Dual Port RAM primitive or RAM_DP for LatticeECP/EC 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 8-7. The table lists the corresponding ports for the module generated by the Module Manager and for the EBR RAM_DP primitive.

Table 8-7. 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 8-8.

Table 8-8. Pseudo-Dual Port Memory Sizes for 9K Memory for LatticeECP/EC 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 8-9 shows the various attributes available for the Pseudo Dual Port Memory (RAM_DP). Some of these attributes are user selectable through the Module Manager GUI. For detailed attribute definitions, refer to Appendix A.

Table 8-9. Pseudo-Dual Port RAM Attributes for LatticeECP/EC Devices

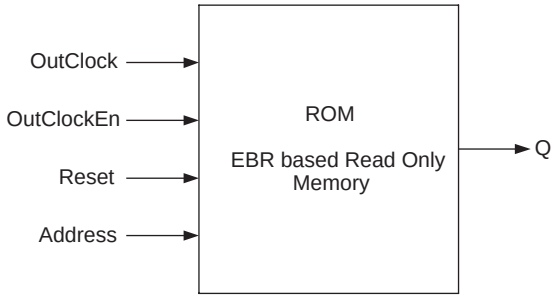
Attribute	Description	Values	Default Value	User Selectable Through Module Manager
DATA_WIDTH_W	Write Data Word Width	1, 2, 4, 9, 18, 36	1	YES
DATA_WIDTH_R	Read Data Word Width	1, 2, 4, 9, 18, 36	1	YES
REGMODE	Register Mode (Pipelining)	NOREG, OUTREG	NOREG	YES
RESETMODE	Selects the Reset type	ASYNC, SYNC	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	ENABLE, DISABLE	ENABLED	YES

Read Only Memory (ROM) – EBR Based

The EBR blocks in the LatticeECP/EC devices can be configured as Read Only Memory or ROM. The Module Manager 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.

The Module Manager generates the memory module as shown in Figure 8-11.

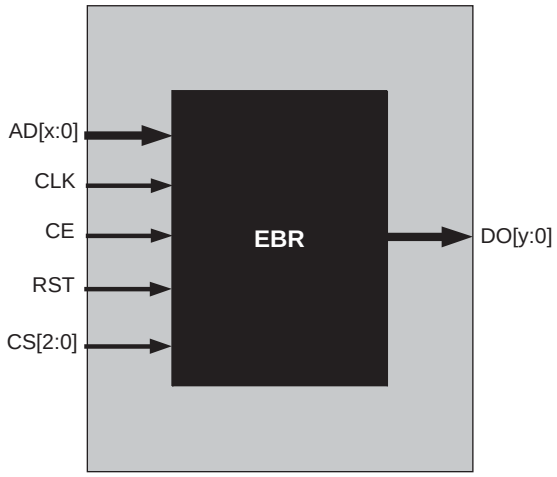
Figure 8-11. ROM - Read Only Memory Module Generated by Module Manager



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 devices is as shown in Figure 8-12.

Figure 8-12. ROM Primitive for LatticeECP/EC 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 8-10. The table lists the corresponding ports for the module generated by Module Manager and for the ROM primitive.

Table 8-10. 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 the Module Manager, 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.

First In First Out (FIFO, FIFO_DC) – EBR Based

The EBR blocks in the LatticeECP/EC 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. Module Manager allows users to generate the Verilog-HDL or VHDL along with an EDIF netlist for the memory size as per design requirement.

The Module Manager generates the FIFO and FIFO_DC memory module as shown in Figures 8-13 and 8-14.

Figure 8-13. FIFO Module Generated by Module Manager

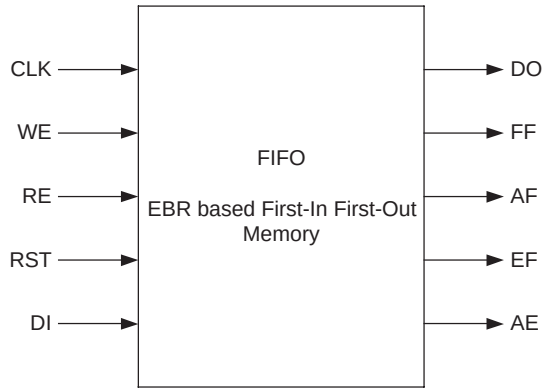
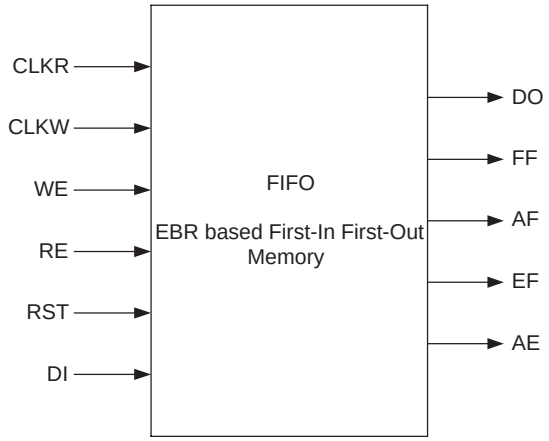


Figure 8-14. FIFO_DC Module Generated by Module Manager



LatticeECP/EC 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 8-11. 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 input and output registers of the RAM. It does not reset the contents of the memory.

The various supported sizes for the FIFO and FIFO_DC for EC and ECP are shown in Table 8-12.

Table 8-12. FIFO and FIFO_DC Data Widths Sizes for LatticeECP/EC 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]

Programmable Flag Values

The FIFO flags are programmable. The programmable ranges for the four FIFO flags are specified in Table 8-13.

Table 8-13. 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 offset value of the address at which the Almost Full Flag 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 12 in the Module/IP Manager.

FIFO Reset
A FIFO reset will clear the contents of the FIFO by resetting the read and write pointers as well as put the FIFO flags in the initial reset state.

Read Pointer Reset
The purpose of the read pointer reset is to indicate retransmit, and is most commonly used in “packetized” communications. In this application, the user must keep careful track of when a packet is written into or read from the FIFO.

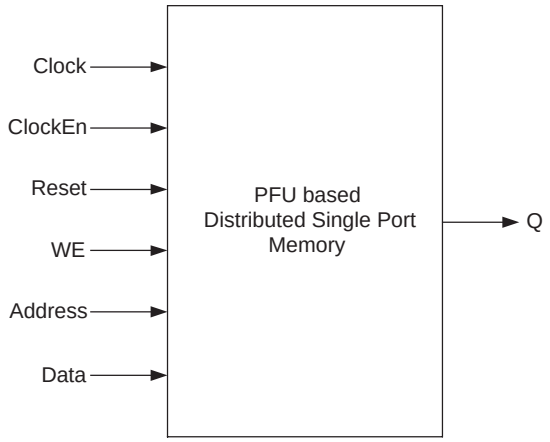
Register Mode
There are two modes for registering and pipelining the read and write cycles of the memory. In the minimum mode, a single set of input registers allows synchronous write cycles into the memory array with the other register banks bypassed. The additional mode includes using the output registers.

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.

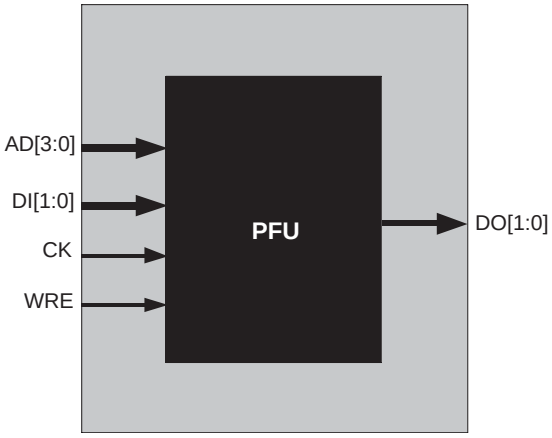
Figure 8-15 shows the Distributed Single Port RAM module as generated by the Module Manager.

Figure 8-15. Distributed Single Port RAM Module Generated by Module Manager



The generated module makes use of the 4-input LUT available in the PFU. Additional logic like Clock and Reset is generated by utilizing the resources available in the PFU. The basic Distributed Single Port RAM primitive for the LatticeECP/EC devices is shown in Figure 8-16.

Figure 8-16. Distributed Single Port RAM (Sync_Single-Port_RAM) for LatticeECP/EC Devices



Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by the Module Manager when the user wants to enable the output registers in the Module Manager configuration.

The various ports and their definitions for the memory are included in Table 8-14. The table lists the corresponding ports for the module generated by Module Manager and for the primitive.

Table 8-14. PFU based Distributed Single port RAM Port Definitions

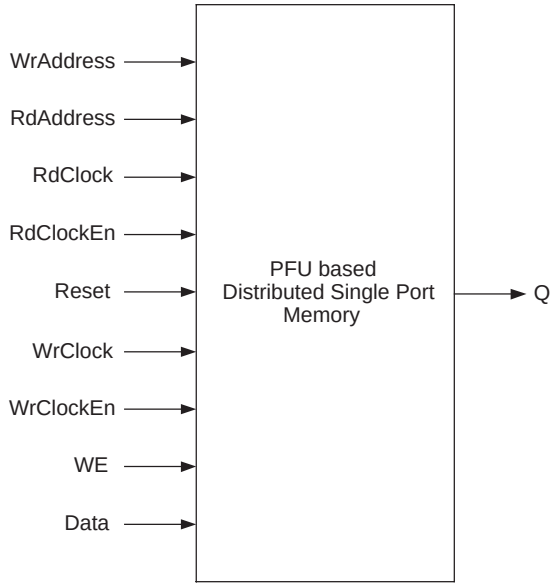
Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State
Clock	CK	Clock	Rising Clock Edge
ClockEn	-	Clock Enable	Active High
Reset	-	Reset	Active High
WE	WRE	Write Enable	Active High
Address	AD[3:0]	Address	—
Data	DI[1:0]	Data In	—
Q	DO[1:0]	Data Out	—

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.

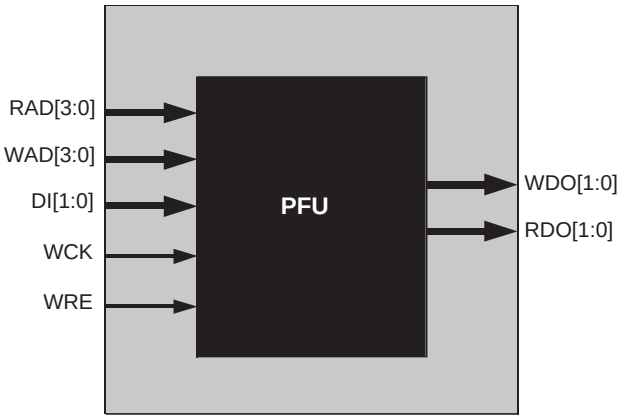
Figure 8-17 shows the Distributed Single Port RAM module as generated by the Module Manager.

Figure 8-17. Distributed Dual Port RAM Module Generated by the Module Manager



The generated module makes use of a 4-input LUT available in the PFU. Additional logic like Clock and Reset is generated by utilizing the resources available in the PFU. The basic Distributed Dual Port RAM primitive for the LatticeECP/EC devices is shown in Figure 8-18.

Figure 8-18. PFU-based Distributed Dual Port RAM for Lattice-EC/ECP Devices



Ports such as Read Clock (RdClock) and Read Clock Enable (RdClockEn) are not available in the hardware primitive. These are generated by the Module Manager when the user wants the to enable the output registers in the Module Manager configuration.

The various ports and their definitions for the memory are included in Table 8-15. The table lists the corresponding ports for the module generated by Module Manager and for the primitive.

Table 8-15. PFU-based Distributed Dual-Port RAM Port Definitions

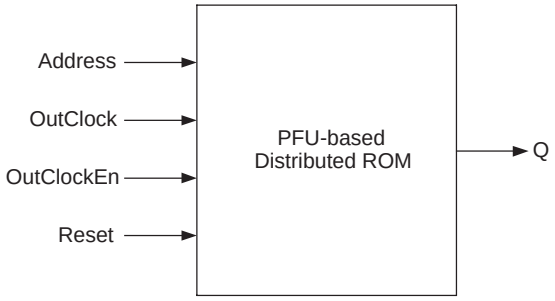
Port Name in Generated Module	Port Name in the EBR Block Primitive	Description	Active State	WrAddress
WAD[23:0]	Write Address	—	RdAddress	RAD[3:0]
Read Address	—	RdClock	—	Read Clock
Rising Clock Edge	RdClockEn	—	Read Clock Enable	Active High
WrClock	WCK	Write Clock	Rising Clock Edge	WrClockEn
—	Write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

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.

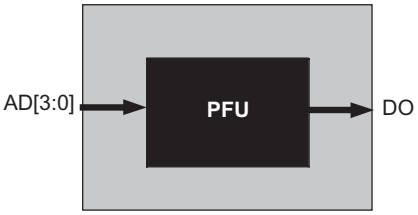
Figure 8-19 shows the Distributed Single Port RAM module as generated by the Module Manager.

Figure 8-19. Distributed ROM Generated by Module Manager



The generated module makes use of the 4-input LUT available in the PFU. Additional logic like Clock and Reset is generated by utilizing the resources available in the PFU. The basic Distributed Dual Port RAM primitive for the LatticeECP/EC devices is shown in Figure 8-20.

Figure 8-20. PFU-based Distributed ROM (Sync_ROM) for LatticeECP/EC Devices



Ports such as Out Clock (OutClock) and Out Clock Enable (OutClockEn) are not available in the hardware primitive. These are generated by the Module Manager when the user wants the to enable the output registers in the Module Manager configuration.

The various ports and their definitions for the memory are included in Table 8-16. The table lists the corresponding ports for the module generated by Module Manager and for the primitive.

Table 8-16. 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	—

Initializing Memory

In each of the memory modes it is possible to specify the power-on state of each bit in the memory array. This allows the memory to be used as ROM, if desired. Each bit in the memory array can have 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. The Module Manager 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. RAMs 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. The Module Manager makes memory initialization possible both through the synthesis and simulation flows.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-408-826-6002 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

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.

In WRITETHROUGH mode, the output data is updated with the input data during the write cycle.

In READBEFOREWRITE mode, the output data port is updated with the existing data stored in the write address, during a write cycle.

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.

GSR

GSR or Global Set/ Reset attribute is used to enable or disable the global set/reset for RAM element.

LatticeECP/EC DDR Usage Guide

June 2004 Technical Note TN1050

Introduction

LatticeECP™ and LatticeEC™ 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 devices to implement both generic DDR and DDR memory interfaces.

Generic DDR Implementation

The LatticeECP/EC device families support the DDR memory interface with on-chip dedicated circuitry. In addition to the DDR memory interface, the user can use the I/O Logic Registers to implement the normal DDR data write operation. The generic DDR data write operations can be implemented using the I/O Logic Registers similar to the memory interface implementation. The ODDRXB primitives can be used as shown in Figure 9-12.

The generic DDR data read operation can be implemented using the FPGA core registers. The dedicated DQS circuitry and the input capture registers in IOLOGIC are dedicated to do a DDR memory interface. The following sections discuss the DDR Memory Interface in detail.

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 9-1 and 9-2 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 9-1. Typical DDR Interface

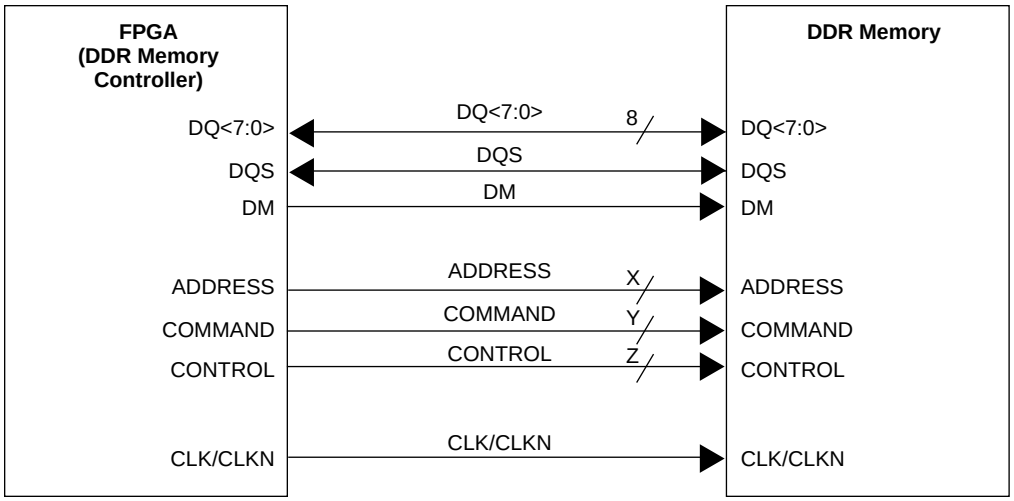


Figure 9-2. DQ-DQS During READ

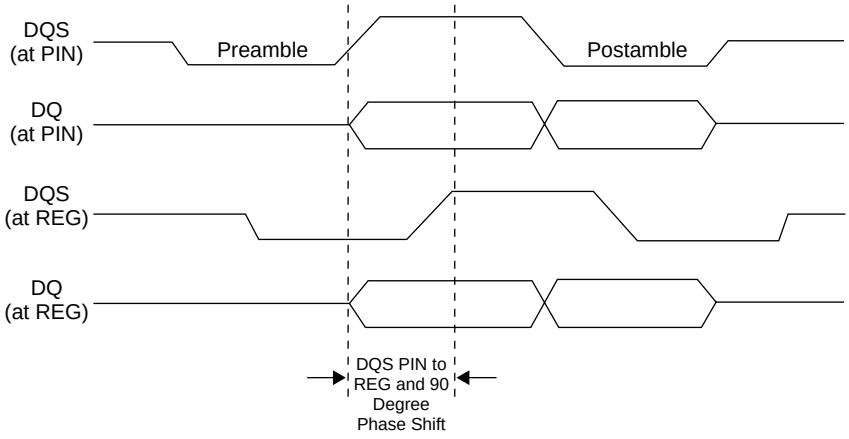
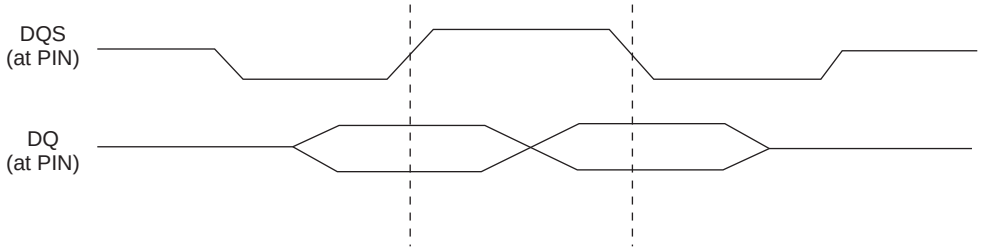


Figure 9-3. DQ-DQS During WRITE



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 devices.

DQ-DQS Grouping

When interfacing to the DDR SDRAM memory, the designer needs to use the dedicated DQ-DQS groups available on the device. For the LatticeECP/EC devices, there is one dedicated DQS pin for every 16 I/Os. Any eight of these I/O can be used to assign the DQ data pins. The ninth I/O of this group of 16 I/Os is the dedicated DQS pin. When not interfacing with the memory these pins can be used as general purpose I/Os. Each of these dedicated DQS

pins are internally connected to the dedicated DQS phase shift circuitry. The LatticeECP/EC Family Data Sheet shows in detail the pin locations of this grouping.

DDR Software Primitives and Related Attributes

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

DQSDLL

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 9-4 shows the primitive symbol.

Figure 9-4. DQSDLL Symbol

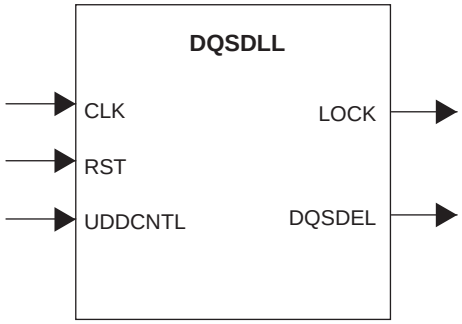


Table 9-1 provides a description of the ports.

Table 9-1. 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 update signal to the DLL that will update the dynamic delay.
LOCK	O	Indicates when the DLL is in phase
DQSDEL	O	The 6-bit 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, but the user can modify this delay by using the following delay adjustment settings. Each of these settings is available as a software attribute.

- **DEL_VAL** – This is a 6-bit, 64-step delay element that can be used to adjust the dynamic delay output.
- **DEL_ADJ** – This configuration bit can be set to either “PLUS” or “MINUS”. This allows adjustment of the delay element controls coming from the DLL. If this bit is set to “PLUS” this means addition of delay. If it is set to “MINUS” this means delay subtraction.
- **LOCK_SENSITIVITY** – This DLL 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 greater or less sensitivity to jitter. Above 150MHz, it is recommended that the LOCK_SENSITIVITY bit be programmed “HIGH” (more sensitive). For 100MHz, it is recommended that the bit be programmed LOW (more tolerant). For 133MHz, the LOCK_SENSITIVITY bit can go either way.

Table 9-2 lists the attributes available with the DQSDLL primitive.

Table 9-2. DQSDLL Attributes

Attribute	Values	Default	Definition
DEL_VAL	0..63	0	64-step delay element
DEL_ADJ	PLUS, MINUS	PLUS	Allows adjustment of the delay element controls coming from the DLL
LOCK_SENSITIVITY	HIGH, LOW	LOW	Determines high or low sensitivity to jitter

Note:

1. By default the delay programmed into this DQSDLL is 90 degrees (DEL_VAL=0). You only need to use this attribute when delay other than 90 degrees is desired.

VHDL Usage:

```
--Component declaration
component DQSDLL
   port(
      CLK     : in STD_LOGIC;
      UDDCNTL : in STD_LOGIC;
      DQSDEL  : out STD_LOGIC;
      RST     : in STD_LOGIC;
      LOCK    : out STD_LOGIC);
end component;


--Attribute declaration
attribute DEL_VAL          : string;
attribute DEL_ADJ          : string;
attribute LOCK_SENSITIVITY : string;


--Attribute Assignment
attribute DEL_VAL of U9      :label is “PLUS”;
attribute DEL_ADJ of U9      :label is “10”;
attribute LOCK_SENSITIVITY of U9:label is “HIGH”;
```

```
--Module instantiation
U9: DQSDLL
       PORT MAP( CLK     =>  CLK,
                 UDDCNTL =>  UDDCNTL,
                 RST     =>  RESET,
                 DQSDEL  =>  dqsdel,
                 LOCK    =>  DLL_LOCK);
```

Verilog Usage

```
// Attribute Assignment
defparam U9.DEL_VAL = “10”,
         U9.DEL_ADJ = “PLUS”,
         U9.LOCK_SENSITIVITY = “HIGH”;

//Module instantiation
DQSDLL U9 (.CLK (CLK), .UDDCNTL(UDDCNTL), .RST(RESET), .DQSDEL(dqsdel),
           .LOCK(DLL_LOCK));
```

DQSBUF

This primitive implements the DQS Delay and the DQS transition detector logic. Figure 9-5 shows the DQSBUFB function. The preamble detect signal is also generated within this primitive.

Figure 9-5. DQSBUFB Function

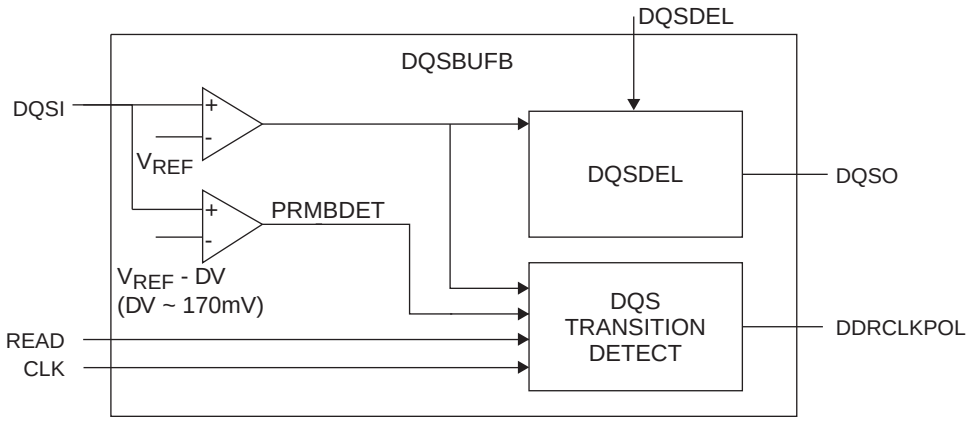


Figure 9-6 shows the primitive symbol and its ports.

Figure 9-6. DQSBUFB Symbol

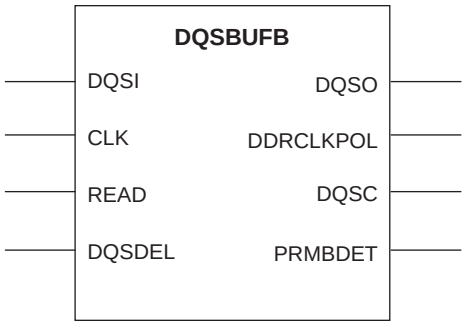


Table 9-3 provides a description of the I/O ports associated with the DQSBUFB primitive.

Table 9-3. 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).
2. If the delay in the DQSDLL is manually assigned then in order to see this dynamic delay in the DQSBUFB block, software requires that you to also assign the attributes DEL_VAL and DEL_ADJ with the DQSBUF primitive.

VHDL Usage:

```
--Component declaration
component DQSBUFB
   port(
      DQSI     : in STD_LOGIC;
      CLK      : in STD_LOGIC;
      READ     : in STD_LOGIC;
      DQSDEL   : in STD_LOGIC;
      DDRCLKPOL: in STD_LOGIC;
      DQSC     :  out STD_LOGIC;
      PRMBDET  : out STD_LOGIC;
      DQSO     : out STD_LOGIC);
end component;


--Module instantiation
U8 : DQSBUFB PORT MAP( DQSI => DQS,
                 CLK   => CLK,
                 READ  => READIN,
                 DQSDEL => dqsdel,
                 DDRCLKPOL =>ddrclkpol_sig,
                 DQSC => DQSCIB,
                 PRMBDET =>PRMBDET,
                 DQSO =>dqsbuf);
```

Verilog Usage:

```
//Module instantiation
DQSBUFB U8 (.DQSI(DQS), .CLK(CLK), .READ(READ), .DQSDEL(dqsdel),
              .DDRCLKPOL (ddrclkpol_sig),.DQSC(DQSCIB), .PRMBDET(PRMBDET),
              .DQSO(dqsbuf));
```

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 9-7 shows the primitive symbol and the I/O ports.

Figure 9-7. IDDRXB Symbol

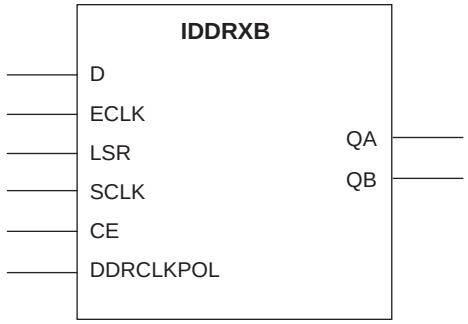


Table 9-4 provides a description of all I/O ports associated with the IDDRXB primitive.

Table 9-4. 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 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.

VHDL Usage:

```
--Component declaration
component IDDRXB
   port(
      D        : in STD_LOGIC;
      ECLK     : in STD_LOGIC;
      SCLK     : in STD_LOGIC;
      CE       : in STD_LOGIC;
      DDRCLKPOL: in STD_LOGIC;
      LSR      : in STD_LOGIC;
      QA       : out STD_LOGIC;
      QB       : out STD_LOGIC);
end component;


--Module instantiation
UL0 : IDDRXB PORT MAP( D        => DQ(0),
                ECLK     => dqsbuf,
                SCLK     => CLK,
                CE       => vcc_net,
                DDRCLKPOL=>ddrclkpol_sig,
                LSR      =>RESET,
                QA       =>INDDR_DA(0),
                QB       => INDDR_DB(0))
```

Verilog Usage:

```
//Module instantiation
IDDRXB UL0 (.D(DQ[0]), .ECLK(dqsbuf), .SCLK(CLK), .CE(vcc_net),
              .DDRCLKPOL(ddrclkpol_sig),.LSR(RESET),
              .QA(INDDR_DA[0]),  .QB(INDDR_DB[0]));
```

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 9-8 shows the ODDRXB primitive symbol and its I/O ports.

Figure 9-8. ODDRXB Symbol

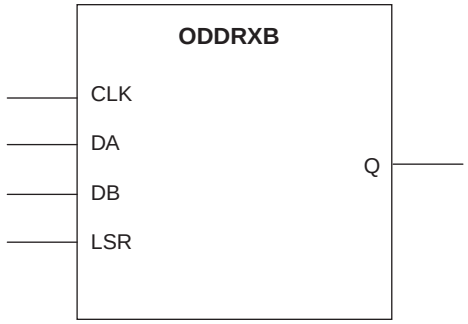


Table 9-5 provides a description of all I/O ports associated with the ODDRXB primitive.

Table 9-5. 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. For the tri-state block, DDR registers do not have CE support. CE and LSR support is available for the regular (non-DDR) mode only. LSR is available in tristate DDRX mode (while reading). The LSR will default to set when used in the tristate mode.
3. When asserting RESET during DDR writes, note that this will only reset the FFs and not the latches.

Memory Read Implementation

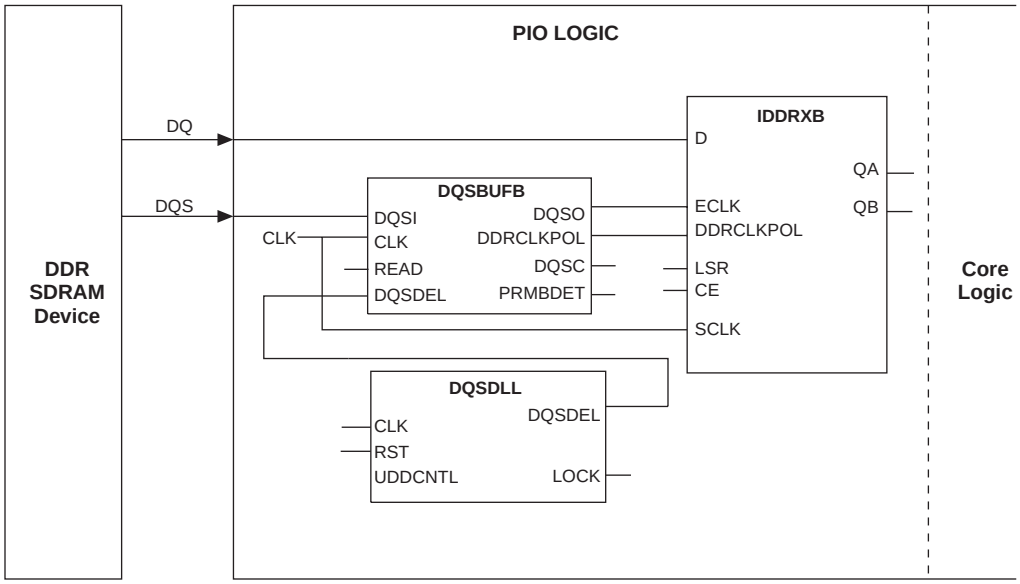
The LatticeECP/EC devices contain a variety of features to simplify implementation of the read portion of a DDR interface:

- DLL compensated DQS delay elements
- DDR input registers
- Automatic DQS to system clock domain transfer circuitry

The LatticeECP/EC Family Data Sheet details these circuit elements.

Three primitives in the Lattice ispLEVER® design tools represent the capability of these three elements. The DQSDLL 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 9-9 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 device.

Figure 9-9. Software Primitive Implementation for Memory READ

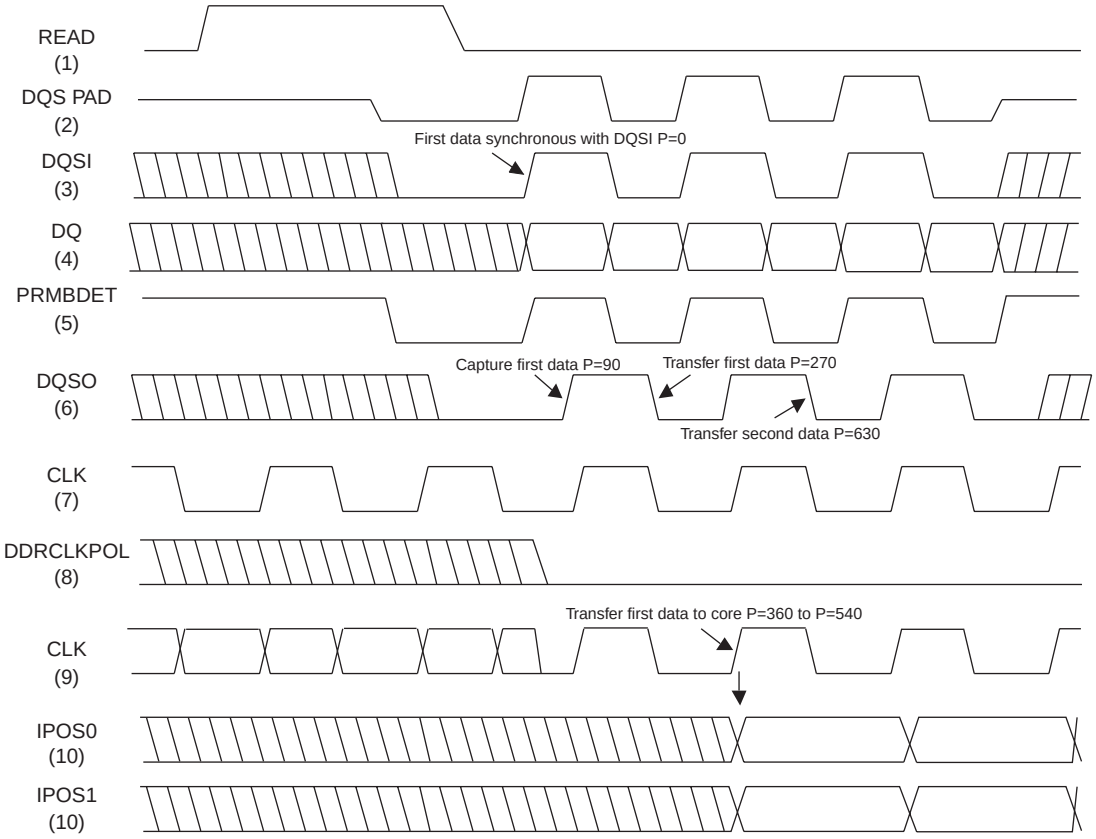


Read Timing Waveforms

Figures 9-10 and 9-11 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 two synchronization registers based on the relative phases of DQSI (before delay) and CLK.

- Case 1: If CLK = 0 on the first DQSI transition, then DDRCLKPOL = 0; hence no inversion required. See Figure 9-10.
- Case 2: If CLK=1 on the first DQS strobe transition, then DDRCLKPOL = 1. The system clock (CLK) needs to be inverted before it is used for synchronization. See Figure 9-11.

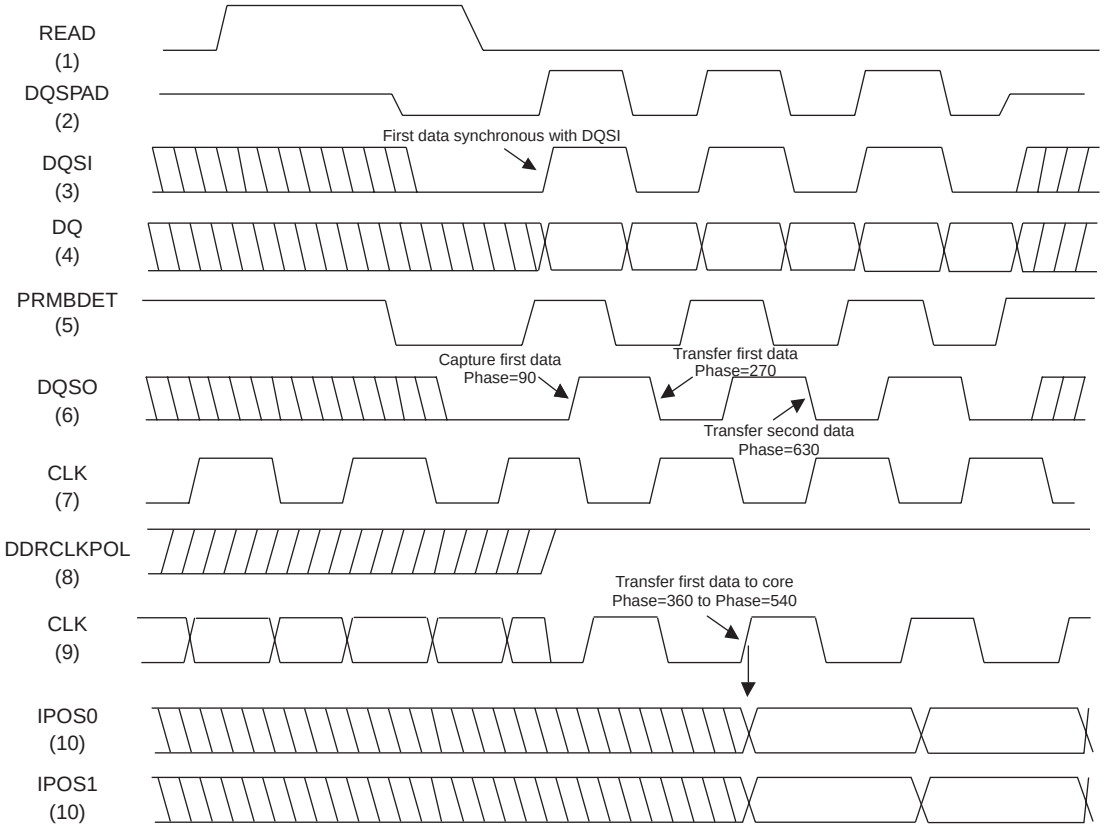
Figure 9-10. READ Data Transfer when DDRCLKPOL=0



Notes -

(1) READ is internally generated and should go high when the READ command (to control the DDR-RAM) is initially asserted. READ is used to generate the clock edge to sample the state of the CLK.
(2) DQSPAD is the DQS strobe at the PAD from the DDR memory.
(3) DQSI is the DQS Strobe signal at the input of the DQSBUFB primitive
(4) DQ is the DDR data coming from the DDR memory. This data bus is edge aligned to the DQSI.
(5) PRMBDET is the Preable detect signal generated in the DQSBUFB primitive.
(6) DQSO is the 90 degree delayed DQS. This is the output of the DQSBUFB primitive and is used to clock the first set of DDR registers. This is connected to the ECLK input of IDDRXB primitive.
(7) CLK is the clock coming from the core logic.
(8) DDRCLKPOL signal is generated in by the DQS Transition detect logic represented in software by DQSBUFB primitive.
In this case 1, the DDRCLKPOL=0 as the CLK is LOW at the 1st DQSI transition.
(9) As DDRCLKPOL = 0, the CLK is not inverted. This CLK is connected to the SCLK input of the IDDRXB primitive. The DQ data will be synchronized to this CLK before it enters the FPGA.
(10) IPOS0 and IPOS1 are the DDR data both synchronized to the positive edge of CLK.

Figure 9-11. Read Data Transfer When DDRCLKPOL=1



Notes -

(1) READ is internally generated and should go high when the READ command (to control the DDR-RAM) is initially asserted. READ is used to generate the clock edge to sample the state of the CLK.
(2) DQSPAD is the DQS strobe at the PAD from the DDR memory.
(3) DQSI is the DQS Strobe signal at the input of the DQSBUFB primitive
(4) DQ is the DDR data coming from the DDR memory. This data bus is edge aligned to the DQSI.
(5) PRMBDET is the Preable detect signal generated in the DQSBUFB primitive.
(6) DQSO is the 90 degree delayed DQS. This is the output of the DQSBUFB primitive and is used to clock the first set of DDR registers. This is connected to the ECLK input of IDDRXB primitive.
(7) CLK is the clock coming from the core logic.
(8) DDRCLKPOL signal is generated in by the DQS Transition detect logic represented in software by DQSBUFB primitive.
In this case 2, the DDRCLKPOL=1 as the CLK is HIGH at the 1 st DQSI transition.
(9) As DDRCLKPOL = 1 , the CLK is inverted before enters the IDDRXB. This CLK is connected to the SCLK input of the IDDRXB primitive. The DQ data will be synchronized to this CLK before it enters the FPGA.
(10) IPOS0 and IPOS1 are the DDR data both synchronized to the positive edge of CLK.

Memory Write Implementation

To implement the write portion of a DDR memory interface two streams 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.

The LatticeECP/EC 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 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 9-12.

CLKP and CLKN are generated using the CLK+90 from the PLL. CLKN is generated by inverting the CLK+90 (=CLK+270) as shown in Figure 9-12.

The CLKP and CLKN can also be generated using the differential SSTL25 II buffers. The CLKP can be generated using the ODDRXB primitive as shown in Figure 9-12 and fed into a SSTL25 differential output buffer to generate a CLKN output. Generating the CLKN in this manner prevents any skew between the two signals.

The ADDR, CMD signal needs to be center aligned with relative to CLKN. The ADDR/CMD signal is a SDR signal.

In order to meet DDR interface specification for t_{DSS} and t_{DSH} parameters, defined as DQS falling to CLKN rising setup and hold times. This is met by making CLKN and DQS identical in phase. DQS is inverted to match CLKN (= 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.

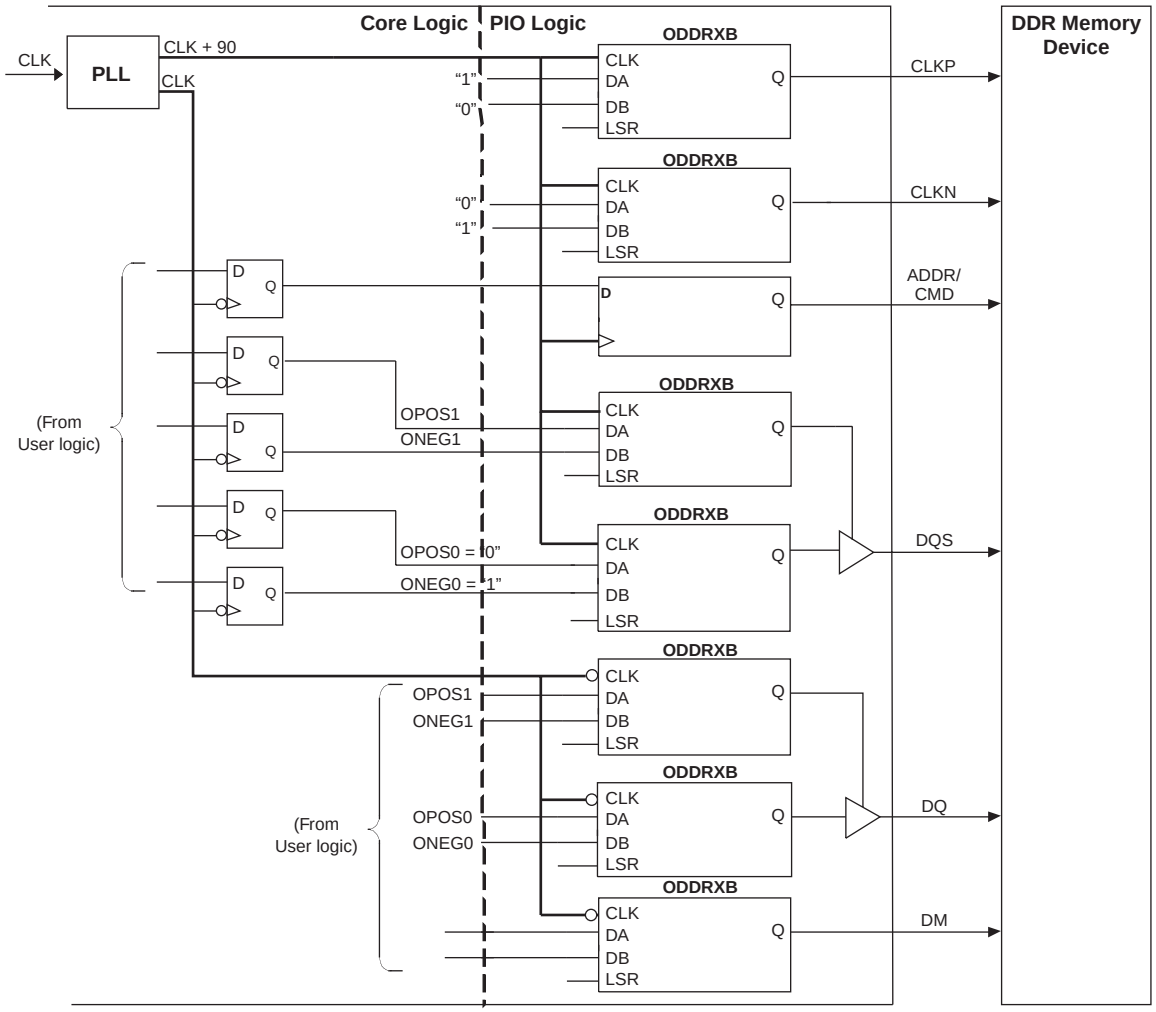
The data DQ and DM needs to be delayed by 90° as it leaves the FPGA. This is to center the data and data mask relative to the DQS when it reaches the DDR memory. This can be done by inverting the CLK to the DQ and DM data.

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 9-12 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. The ADDR/CMD is generated using an IO SDR register. Appendix A and B provides example code to implement the complete I/O section of a memory interface within a LatticeECP/EC device.

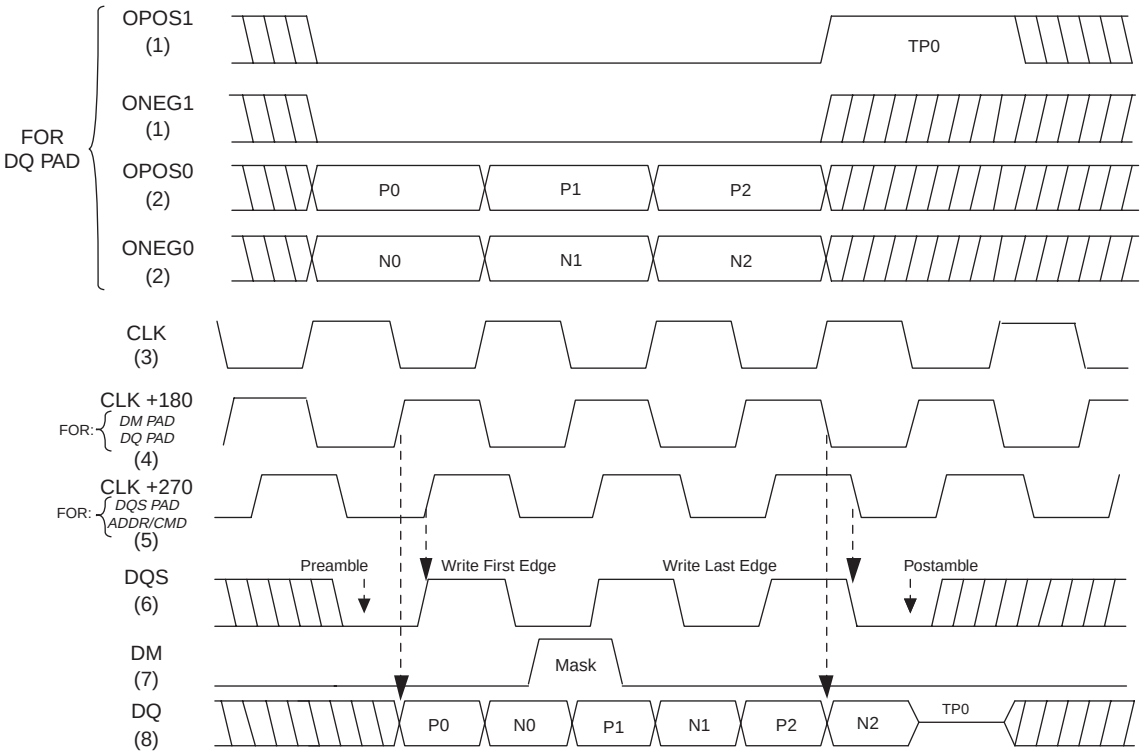
Figure 9-12. Software Primitive Implementation for Memory Write



Write Timing Waveforms

Figures 9-13 and 9-14 show 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 9-13. DDR Write Data Transfer for DQ Data



Notes -

(1) OPOS1 and ONEG1 are the tristate inputs to the tristate block(ODDRXB)

(2) OPOS0 and ONEG0 are the 2 sets of data that are aligned with the CLK.

(3) CLK is the core clock that is used to generate all the DDR outputs in the user logic.

(4) CLK +180 is the inverted CLK. This clock is inverted to transmit the OPOS0 and OPOS1 data.

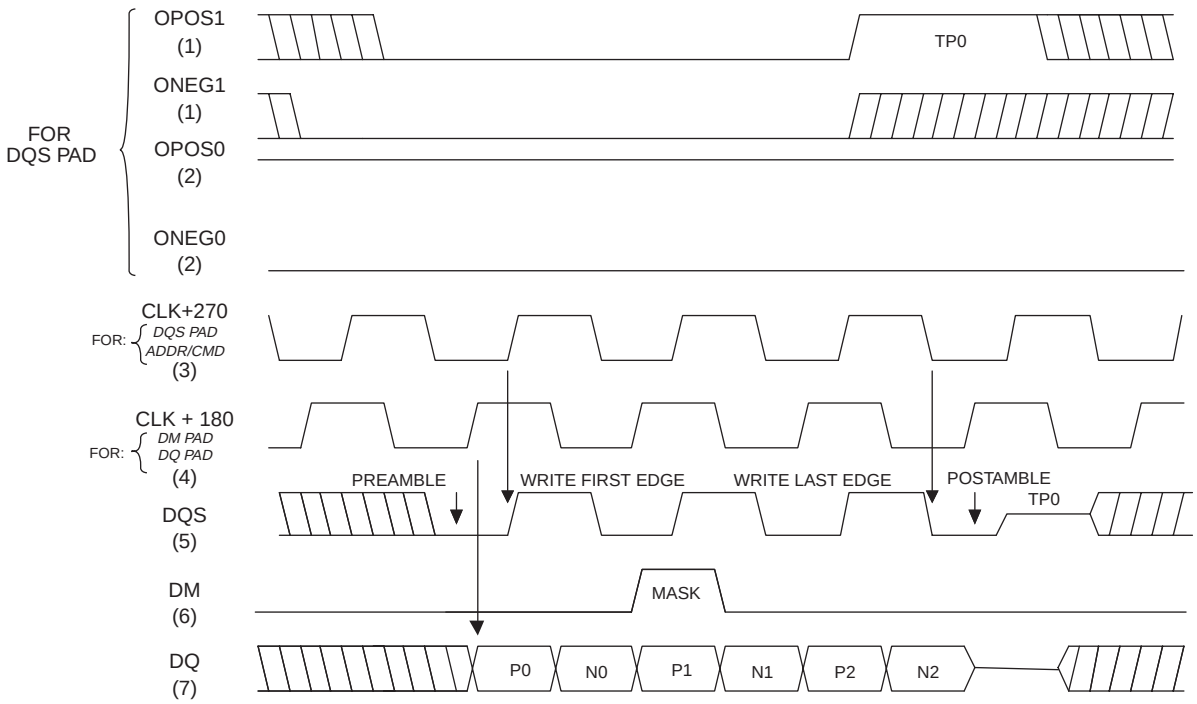
(5) CLK +270 is the phase of the CLK used to generate the DQS strobe output and the Address / Command output signals.

(6) DQS is the strobe output that is sent to the memory device. Figure 9 show DQS generation.

(7) The DM signal is the Data Mask signal and should be in phase with the DQ data output.

(8) DQ is DDR data, this data needs to be center aligned with the DQS strobe signal when it reaches the memory.

Figure 9-14. DDR Write Data Transfer for DQS Strobe



Notes -

(1) OPOS1 and ONEG1 are the tristate inputs to the tristate block (ODDRXB) used to generate tristate signal for the DQS output

(2) OPOS0 and ONEG0 in this case are tied to “0” and “1” respectively to generate the DQS strobe.

(3) CLK +270 phase of the CLK is used to clock the DQS strobe output.

(4) CLK +180 is the inverter core clock. This clock is used to clock the DDR output Data DQ and Data mask signal DM as shown in Figure 8.

(5) DQS is the strobe output that is sent to the memory device.

(6) The DM signal is the Data Mask signal and should be in phase with the DQ data output.

(7) DQ is DDR data, this data needs to be 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 devices.

- The LatticeECP/EC devices have dedicated DQ- DQS groups. Please refer to the logical signal connections of the groups in the LatticeECP/EC Family Data Sheet 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. Also all the DDR interfaces implemented on the top half or the bottom half of the device should run at the same interface frequency.
- The DDR SDRAM interface supports the SSTL25 IO standard, hence the interface pins should be assigned as SSTL25 I/O type.
- When implementing the DDR interface, the VREF1 of the bank is used to provide the reference voltage for the interface pins.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-408-826-6002 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Appendix A. Verilog Example of DDR Input and Output Modules

DDR Input Module

```
module ddr_in (DQ, DQS, CLK, RESET, UDDCNTL, READ, INDDR_DA, INDDR_DB, DQSCIB, PRMBDET, DLL_LOCK,
DDRCLKPOL);
input [7:0] DQ;
input DQS;
input CLK, RESET, UDDCNTL, READ;
output [7:0]INDDR_DA;
output[7:0] INDDR_DB;
output DQSCIB, PRMBDET, DLL_LOCK, DDRCLKPOL;
wire vcc_net,gnd_net;
wire dqsbuf, dqsdel, CLK, ddrclkpol_sig;

wire [7:0] INDDR_DA_sig;
wire [7:0] INDDR_DB_sig;

reg [7:0]INDDR_DA;
reg [7:0] INDDR_DB;

assign vcc_net = 1'b1;
assign gnd_net = 1'b0;
assign DDRCLKPOL = ddrclkpol_sig;

IDDRXB UL0 (.D(DQ[0]), .ECLK(dqsbuf), .SCLK(CLK), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
            .LSR(RESET), .QA(INDDR_DA_sig[0]), .QB(INDDR_DB_sig[0]));

IDDRXB UL1 (.D(DQ[1]), .ECLK(dqsbuf), .SCLK(CLK), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
            .LSR(RESET), .QA(INDDR_DA_sig[1]), .QB(INDDR_DB_sig[1]));

IDDRXB UL2 (.D(DQ[2]), .ECLK(dqsbuf), .SCLK(CLK), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
            .LSR(RESET), .QA(INDDR_DA_sig[2]), .QB(INDDR_DB_sig[2]));

IDDRXB UL3 (.D(DQ[3]), .ECLK(dqsbuf), .SCLK(CLK), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
            .LSR(RESET), .QA(INDDR_DA_sig[3]), .QB(INDDR_DB_sig[3]));

IDDRXB UL4 (.D(DQ[4]), .ECLK(dqsbuf), .SCLK(CLK), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
            .LSR(RESET), .QA(INDDR_DA_sig[4]), .QB(INDDR_DB_sig[4]));

IDDRXB UL5 (.D(DQ[5]), .ECLK(dqsbuf), .SCLK(CLK), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
            .LSR(RESET), .QA(INDDR_DA_sig[5]), .QB(INDDR_DB_sig[5]));

IDDRXB UL6 (.D(DQ[6]), .ECLK(dqsbuf), .SCLK(CLK), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
            .LSR(RESET), .QA(INDDR_DA_sig[6]), .QB(INDDR_DB_sig[6]));

IDDRXB UL7 (.D(DQ[7]), .ECLK(dqsbuf), .SCLK(CLK), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
            .LSR(RESET), .QA(INDDR_DA_sig[7]), .QB(INDDR_DB_sig[7]));

always@(posedge CLK)
 begin
  INDDR_DA = INDDR_DA_sig;
  INDDR_DB = INDDR_DB_sig;
 end
```

```
DQSBUFB U8 (.DQSI(DQS), .CLK(CLK), .READ(READ), .DQSDEL(dqsdel), .DDRCLKPOL(ddrclkpol_sig),
            .DQSC(DQSCIB), .PRMBDET(PRMBDET), .DQSO(dqsbuf));

DQSDLL U9 (.CLK(CLK), .UDDCNTL(UDDCNTL), .RST(RESET), .DQSDEL(dqsdel), .LOCK(DLL_LOCK));

endmodule
```

DDR Output Module

```
module ddr_out (DQ, DQS, REFCLK, RESET, CLKOUTN, CLKOUTP, OUTDDR_data_DA, OUTDDR_data_DB,
DQS_data_DA, DQS_data_DB);

output [7:0] DQ;
output DQS ;
input REFCLK, RESET;
input [7:0] OUTDDR_data_DA, OUTDDR_data_DB;
input DQS_data_DA, DQS_data_DB;
output CLKOUTN, CLKOUTP;

wire vcc_net, gnd_net;
wire [7:0] OUTDDR_data_DA, OUTDDR_data_DB;
wire RESET, REFCLK;
wire [7:0] DQ;

reg DQS_data_DA_Q, DQS_data_DB_Q;
reg DQS_data_DA_Q1, DQS_data_DB_Q1;
reg [7:0] OUTDDR_data_DA_Q, OUTDDR_data_DB_Q;


assign vcc_net = 1'b1;
assign gnd_net = 1'b0;

wire pllclk, zero_dly, lead_lag, clkok, clklock;
wire [3:0] plldly;


xpll_s pllInst0 (.CLKI(REFCLK), .CLKFB(pllclk), .RST(RESET),
.DDAMODE(gnd_net),   .DDAIZR(gnd_net),   .DDAILAG(vcc_net),   .DDAIDEL0(gnd_net),   .DDAIDEL1(gnd_net),
.DDAIDEL2(gnd_net),
.CLKOP(clk), .CLKOS(clkp), .CLKOK(clkok), .LOCK(clklock),
.DDAOZR(zero_dly),        .DDAOLAG(lead_lag),        .DDAODEL0(plldly[0]),        .DDAODEL1(plldly[1]),
.DDAODEL2(plldly[2]));


always @ (posedge RESET or posedge clk)
begin
if (RESET)// reset logic
begin
    DQS_data_DA_Q = 0;
    DQS_data_DB_Q = 0;
end

else
begin
    DQS_data_DA_Q = DQS_data_DA;
    DQS_data_DB_Q = DQS_data_DB;
end
end
```

```
always @ (posedge RESET or negedge clk)
begin
if (RESET)
begin
    DQS_data_DA_Q1 = 0;
    DQS_data_DB_Q1 = 0;
end

else
begin
    DQS_data_DA_Q1 = DQS_data_DA_Q;
    DQS_data_DB_Q1 = DQS_data_DB_Q;
end
end

always @ (posedge RESET or posedge clk)
begin
if (RESET)
 begin
    OUTDDR_data_DA_Q = 0;
    OUTDDR_data_DB_Q = 0;
 end
else
 begin
    OUTDDR_data_DA_Q = OUTDDR_data_DA;
    OUTDDR_data_DB_Q = OUTDDR_data_DB;
 end
end

//-----------DDR OUTPUT-------------------------------------
ODDRXB O0 (.DA(OUTDDR_data_DA_Q[0]), .DB(OUTDDR_data_DB_Q[0]), .LSR(RESET), .CLK(~clk), .Q(DQ[0]));
ODDRXB O1 (.DA(OUTDDR_data_DA_Q[1]), .DB(OUTDDR_data_DB_Q[1]), .LSR(RESET), .CLK(~clk), .Q(DQ[1]));
ODDRXB O2 (.DA(OUTDDR_data_DA_Q[2]), .DB(OUTDDR_data_DB_Q[2]), .LSR(RESET), .CLK(~clk), .Q(DQ[2]));
ODDRXB O3 (.DA(OUTDDR_data_DA_Q[3]), .DB(OUTDDR_data_DB_Q[3]), .LSR(RESET), .CLK(~clk), .Q(DQ[3]));
ODDRXB O4 (.DA(OUTDDR_data_DA_Q[4]), .DB(OUTDDR_data_DB_Q[4]), .LSR(RESET), .CLK(~clk), .Q(DQ[4]));
ODDRXB O5 (.DA(OUTDDR_data_DA_Q[5]), .DB(OUTDDR_data_DB_Q[5]), .LSR(RESET), .CLK(~clk), .Q(DQ[5]));
ODDRXB O6 (.DA(OUTDDR_data_DA_Q[6]), .DB(OUTDDR_data_DB_Q[6]), .LSR(RESET), .CLK(~clk), .Q(DQ[6]));
ODDRXB O7 (.DA(OUTDDR_data_DA_Q[7]), .DB(OUTDDR_data_DB_Q[7]), .LSR(RESET), .CLK(~clk), .Q(DQ[7]));

//--------------- DQS Generation-----------------------------------
ODDRXB O8 (.DA(DQS_data_DA_Q1), .DB(DQS_data_DB_Q1), .LSR(RESET), .CLK(clkp), .Q(DQS));

//--------------- CLKOUTP and CLKOUTN Generation---------------------
ODDRXB O9 (.DA(vcc_net), .DB(gnd_net), .LSR(RESET), .CLK(clkp), .Q(CLKOUTP));
ODDRXB O10 (.DA(gnd_net), .DB(vcc_net), .LSR(RESET), .CLK(clkp), .Q(CLKOUTN));


endmodule
```

Appendix B. VHDL Example of a DDR Memory Interface Using LatticeECP/EC Devices

DDR Input Module

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
library ec;
use ec.components.all;

entity ddr_in is
        port(DQ  : in std_logic_vector(7 downto 0);
             DQS : in std_logic;
             CLK : in std_logic;
             RESET : in std_logic;
             UDDCNTL :in std_logic;
              READ: in std_logic;
              INDDR_DA: out std_logic_vector(7 downto 0);
              INDDR_DB: out std_logic_vector(7 downto 0);
              DQSCIB : out std_logic;
              PRMBDET : out std_logic;
              DLL_LOCK: out std_logic);
end ddr_in;

architecture structure of ddr_in is

component IDDRXB
   port(
      D  : in STD_LOGIC;
      ECLK   : in STD_LOGIC;
      SCLK   : in STD_LOGIC;
      CE  : in STD_LOGIC;
      DDRCLKPOL: in STD_LOGIC;
      LSR:  in STD_LOGIC;
      QA    : out STD_LOGIC;
      QB    : out STD_LOGIC);
end component;

component DQSBUFB
   port(
      DQSI  : in STD_LOGIC;
      CLK   : in STD_LOGIC;
      READ   : in STD_LOGIC;
      DQSDEL  : in STD_LOGIC;
      DDRCLKPOL: out STD_LOGIC;
      DQSC:  out STD_LOGIC;
      PRMBDET    : out STD_LOGIC;
      DQSO    : out STD_LOGIC);
end component;

component DQSDLL
   port(
      CLK   : in STD_LOGIC;
      UDDCNTL   : in STD_LOGIC;
      DQSDEL  : out STD_LOGIC;
      RST: in STD_LOGIC;
      LOCK    : out STD_LOGIC);
end component;
```

```
attribute DEL_VAL : string;
attribute DEL_ADJ : string;
attribute LOCK_SENSITIVITY : string;

attribute DEL_VAL of U8:label is "10";
attribute DEL_ADJ of U8:label is "PLUS";

attribute DEL_VAL of U9:label is "10";
attribute DEL_ADJ of U9:label is "PLUS";
attribute LOCK_SENSITIVITY of U9:label is "HIGH";


signal vcc_net : std_logic;
signal gnd_net : std_logic;
signal dqsbuf, dqsdel,ddrclkpol_sig: std_logic;
signal INDDR_DA_Q: std_logic_vector(7 downto 0);
signal INDDR_DB_Q: std_logic_vector(7 downto 0);

begin
        vcc_net <= '1';
        gnd_net <= '0';

UL0 : IDDRXB PORT MAP( D         => DQ(0),
               ECLK      => dqsbuf,
               SCLK      => CLK,
               CE        => vcc_net,
               DDRCLKPOL => ddrclkpol_sig,
               LSR       => RESET,
               QA        => INDDR_DA_Q(0),
               QB        => INDDR_DB_Q(0));

UL1 : IDDRXB PORT MAP( D         => DQ(1),
                       ECLK      => dqsbuf,
                       SCLK      => CLK,
               CE        => vcc_net,
               DDRCLKPOL => ddrclkpol_sig,
               LSR       => RESET,
               QA        => INDDR_DA_Q(1),
               QB        => INDDR_DB_Q(1));

        UL2 : IDDRXB PORT MAP( D         => DQ(2),
               ECLK      => dqsbuf,
               SCLK      => CLK,
               CE        => vcc_net,
               DDRCLKPOL => ddrclkpol_sig,
               LSR       => RESET,
               QA        => INDDR_DA_Q(2),
               QB        => INDDR_DB_Q(2));

UL3 : IDDRXB PORT MAP( D         => DQ(3),
               ECLK      => dqsbuf,
               SCLK      => CLK,
               CE        => vcc_net,
               DDRCLKPOL => ddrclkpol_sig,
               LSR       => RESET,
               QA        => INDDR_DA_Q(3),
               QB        => INDDR_DB_Q(3));
```

```
        UL4 : IDDRXB PORT MAP (D         => DQ(4),
               ECLK      => dqsbuf,
               SCLK      => CLK,
               CE        => vcc_net,
               DDRCLKPOL => ddrclkpol_sig,
               LSR       => RESET,
               QA        => INDDR_DA_Q(4),
               QB        => INDDR_DB_Q(4));

        UL5 : IDDRXB PORT MAP( D         => DQ(5),
               ECLK      => dqsbuf,
               SCLK      => CLK,
               CE        => vcc_net,
               DDRCLKPOL => ddrclkpol_sig,
               LSR       => RESET,
               QA        => INDDR_DA_Q(5),
               QB        => INDDR_DB_Q(5));

        UL6 : IDDRXB PORT MAP( D          => DQ(6),
               ECLK       => dqsbuf,
               SCLK       => CLK,
               CE         => vcc_net,
               DDRCLKPOL  => ddrclkpol_sig,
               LSR        => RESET,
               QA         => INDDR_DA_Q(6),
               QB         => INDDR_DB_Q(6));

        UL7 : IDDRXB PORT MAP( D          => DQ(7),
               ECLK       => dqsbuf,
               SCLK       => CLK,
               CE         => vcc_net,
               DDRCLKPOL  => ddrclkpol_sig,
               LSR        => RESET,
               QA         => INDDR_DA_Q(7),
               QB         => INDDR_DB_Q(7));

U8 : DQSBUFB PORT MAP( DQSI       => DQS,
               CLK        => CLK,
               READ       => READ,
               DQSDEL     => dqsdel,
               DDRCLKPOL  => ddrclkpol_sig,
               DQSC       => DQSCIB,
               PRMBDET    => PRMBDET,
               DQSO       => dqsbuf);

        U9: DQSDLL PORT MAP( CLK        =>  CLK,
                             UDDCNTL    =>  UDDCNTL,
                             RST        =>  RESET,
                             DQSDEL     =>  dqsdel,
                             LOCK       =>  DLL_LOCK);
 reg0 :process  (RESET, CLK)
     begin
        if  RESET = '1' then
            INDDR_DA <= (others => '0');
            INDDR_DB <= (others => '0');
elsif  rising_edge(CLK) then
    INDDR_DA <= INDDR_DA_Q;
    INDDR_DB <= INDDR_DB_Q;
end if;
```

```
end  process reg0;
end structure;
```

DDR Output Module

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
library ec;
use ec.components.all;

entity ddr_example is
        port(DQ  : out std_logic_vector(7 downto 0);
              DQS : out std_logic;
              REFCLK : in std_logic;
              RESET : in std_logic;
              CLKOUTN: out std_logic;
              CLKOUTP: out std_logic;
              OUTDDR_data_DA: in std_logic_vector(7 downto 0);
              OUTDDR_data_DB: in std_logic_vector(7 downto 0);
              DQS_data_DA: in std_logic;
      DQS_data_DB: in std_logic;
              CLKLOCK  : out std_logic);
end ddr_example;

architecture structure of ddr_example is
 signal clk : std_logic;
 signal clkp : std_logic;
 signal pllclk : std_logic;
 signal zero_dly : std_logic;
 signal lead_lag : std_logic;
 signal clkok : std_logic;
 signal plldly : std_logic_vector(3 downto 0);

signal DQS_data_DA_Q, DQS_data_DB_Q :std_logic;
signal DQS_data_DA_Q1, DQS_data_DB_Q1 :std_logic;
signal OUTDDR_data_DA_Q, OUTDDR_data_DB_Q:std_logic_vector(7 downto 0);
signal vcc_net : std_logic;
signal gnd_net : std_logic;
signal clkn : std_logic;


--DDR Output
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;
component xpll_s
    port(
        RST       : in STD_LOGIC;
        CLKI      : in STD_LOGIC;
        CLKFB     : in STD_LOGIC;
        DDAMODE   : in STD_LOGIC;
        DDAIZR    : in STD_LOGIC;
        DDAILAG   : in STD_LOGIC;
```

```
        DDAIDEL2  : in STD_LOGIC;
        DDAIDEL1  : in STD_LOGIC;
        DDAIDEL0  : in STD_LOGIC;
        CLKOP     : out STD_LOGIC;
        CLKOS     : out STD_LOGIC;
        CLKOK     : out STD_LOGIC;
        LOCK      : out STD_LOGIC;
        DDAOZR    : out STD_LOGIC;
        DDAOLAG   : out STD_LOGIC;
        DDAODEL2  : out STD_LOGIC;
        DDAODEL1  : out STD_LOGIC;
        DDAODEL0  : out STD_LOGIC);
 end component;
 --PLL Input frequency
 attribute FIN : string;
 attribute FIN of expll : label is "133.0";

begin
        vcc_net <= '1';
        gnd_net <= '0';

   expll: xpll_s PORT MAP(  CLKI     => REFCLK,
                            RST      => RESET,
                            CLKFB    => pllclk,
                            DDAMODE  => gnd_net,
                            DDAIZR   => gnd_net,
                            DDAILAG  => gnd_net,
                            DDAIDEL2 => gnd_net,
                            DDAIDEL1 => gnd_net,
                            DDAIDEL0 => gnd_net,
                            CLKOP    => clk,
                            CLKOS    => clkp,
                            CLKOK    => clkok,
                            DDAOZR   => zero_dly,
                            DDAOLAG  => lead_lag,
                            DDAODEL2 => plldly(0),
                            DDAODEL1 => plldly(1),
                            DDAODEL0 => plldly(2),
                           LOCK     => CLKLOCK);
   clkn<= not clk;

    -- In the Core logic
     reg0 :process  (RESET, clk)
     begin
        if  RESET = '1' then
            DQS_data_DA_Q <= '0';
    DQS_data_DB_Q <= '0';
            DQS_data_DA_Q1 <= '0';
    DQS_data_DB_Q1 <= '0';
    OUTDDR_data_DA_Q <= (others => '0');
            OUTDDR_data_DB_Q <= (others => '0');

elsif  rising_edge(clk) then

    DQS_data_DA_Q <= DQS_data_DA;
    DQS_data_DB_Q <= DQS_data_DB;

    OUTDDR_data_DA_Q <= OUTDDR_data_DA;
    OUTDDR_data_DB_Q <= OUTDDR_data_DB;
```

```
elsif  falling_edge(clk) then
    DQS_data_DA_Q1 <= DQS_data_DA_Q;
    DQS_data_DB_Q1 <= DQS_data_DB_Q;

end if;
end  process reg0;


--**********DDR OUTPUT*************
    O0 : ODDRXB PORT MAP( DA => OUTDDR_data_DA_Q(0), DB  => OUTDDR_data_DB_Q(0), LSR => RESET,
         CLK  => clkn, Q => DQ(0));
    O1 : ODDRXB PORT MAP( DA => OUTDDR_data_DA_Q(1), DB  => OUTDDR_data_DB_Q(1), LSR => RESET,
         CLK  => clkn, Q => DQ(1));
    O2 : ODDRXB PORT MAP( DA => OUTDDR_data_DA_Q(2), DB  => OUTDDR_data_DB_Q(2), LSR => RESET,
         CLK  => clkn, Q => DQ(2));
    O3 : ODDRXB PORT MAP( DA => OUTDDR_data_DA_Q(3), DB  => OUTDDR_data_DB_Q(3), LSR => RESET,
         CLK  => clkn, Q => DQ(3));
    O4 : ODDRXB PORT MAP( DA => OUTDDR_data_DA_Q(4), DB  => OUTDDR_data_DB_Q(4), LSR => RESET,
         CLK  => clkn, Q => DQ(4));
    O5 : ODDRXB PORT MAP( DA => OUTDDR_data_DA_Q(5), DB  => OUTDDR_data_DB_Q(5), LSR => RESET,
         CLK  => clkn, Q => DQ(5));
    O6 : ODDRXB PORT MAP( DA => OUTDDR_data_DA_Q(6), DB  => OUTDDR_data_DB_Q(6), LSR => RESET,
         CLK  => clkn, Q => DQ(6));
    O7 : ODDRXB PORT MAP( DA => OUTDDR_data_DA_Q(7), DB  => OUTDDR_data_DB_Q(7), LSR => RESET,
         CLK  => clkn, Q => DQ(7));

--********DQS Generation*************
    O8: ODDRXB PORT MAP( DA => DQS_data_DA_Q1, DB => DQS_data_DB_Q1, LSR => RESET, CLK => clkp,
        Q => DQS);

--******** CLKOUTP and CLKOUTN Generation*********
    O9 : ODDRXB PORT MAP( DA => vcc_net, DB => gnd_net, LSR => RESET, CLK =>clkp, Q => CLKOUTP);
    O10: ODDRXB PORT MAP( DA => gnd_net, DB => vcc_net, LSR => RESET, CLK => clkp, Q => CLKOUTN);

end structure;
```

Appendix C. 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 8 DQ data bits.
2. 4-bit, 8-bit and 16-bit configurations. For 16-bit configurations, each data byte must have independent DQS strobe.
3. The memory device uses SSTL2 I/O interface standard.
4. Data transfer rate DDR333 or DDR266.
5. Clock transfer rate of 167MHz or 133MHz.

Table lists the DDR SDRAM part numbers that can be used with the LatticeEC device.

Please note these part numbers are chosen based on the criteria stated above and have not necessarily been validated in hardware.

Table 9-6. 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	DDR333 DDR266	167MHz 133MHz
Micron 256MB	MT46V64M4TG	64Mx4	DDR333 DDR266	167MHz 133MHz
Micron 256MB	MT46V32M8FG	32Mx8	DDR333 DDR266	167MHz 133MHz
Micron 256MB	MT46V32M8TG	32Mx8	DDR333 DDR266	167MHz 133MHz
Micron 256MB	MT46V16M16FG	16Mx16	DDR266	133MHz
Micron 256MB	MT46V16M16TG	16Mx16	DDR333 DDR266	167MHz 133MHz
Micron 512MB	MT46V128M4FN	128Mx4	DDR333 DDR266	167MHz 133MHz
Micron 512MB	MT46V128M4TG	128Mx4	DDR333 DDR266	167MHz 133MHz
Micron 512MB	MT46V64M8FN	64Mx8	DDR333 DDR266	167MHz 133MHz
Micron 512MB	MT46V64M8TG	64Mx8	DDR333 DDR266	167MHz 133MHz
Micron 512MB	MT46V32M16FN	32Mx16	DDR333 DDR266	167MHz 133MHz
Micron 512MB	MT46V32M16TG	32Mx16	DDR333 DDR266	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

Table 9-6. List of Compatible DDR SDRAM (Continued)

DDR SDRAM Vendor	Part Number	Configuration	Max Data Rate	Clock Speed
Samsung 256 Mb E-die	K4H560438E-TC/LB3	64Mx4	DDR333	167MHz
	K4H560438E-NC/LB3	64Mx4	DDR333	167MHz
	K4H560438E-GC/LB3	64Mx4	DDR333	167MHz
	K4H560838E-TC/LB3	32Mx8	DDR333	167MHz
	K4H560838E-NC/LB3	32Mx8	DDR333	167MHz
	K4H560838E-GC/LB3	32Mx8	DDR333	167MHz
Samsung 512Mb B die	K4H510838B-TC/LB3	64Mx8	DDR333	167MHz
	K4H510838B-NC/LB3	64Mx8	DDR333	167MHz
	K4H511638B-TC/LB3	32Mx16	DDR333	167MHz
	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	167MHz
	HYB25D128160CE	8Mx16	DDR333	167MHz
	HYB25D128160CC	8Mx16	DDR333	167MHz
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	167MHz
	HYB25D256160BT	16Mx16	DDR333 DDR266	167MHz 133MHz
	HYB25D256160CT	16Mx16	DDR333	167MHz
	HYB25D256160CE	16Mx16	DDR333	167MHz
	HYB25D256400BC	64Mx16	DDR333 DDR266	167MHz 133MHz
	HYB25D256400CF	64Mx16	DDR333	167MHz
	HYB25D256400CC	64Mx16	DDR333	167MHz
	HYB25D256160BC	16Mx16	DDR333 DDR266	167MHz 133MHz
	HYB25D256160CC	16Mx16	DDR333	167MHz

Table 9-6. List of Compatible DDR SDRAM (Continued)

DDR SDRAM Vendor	Part Number	Configuration	Max Data Rate	Clock Speed
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	167MHz
	HYB25D512800BE	64Mx8	DDR333	167MHz
	HYB25D512160AT	32Mx16	DDR333 DDR266	167MHz 133MHz
	HYB25D512160BT	32Mx16	DDR333	167MHz
	HYB25D512160BE	32Mx16	DDR333	167MHz
	HYB25D512400BC	128Mx4	DDR333	167MHz
	HYB25D512400BF	128Mx4	DDR333	167MHz
	HYB25D512800BC	64Mx8	DDR333	167MHz
	HYB25D512800BF	64Mx8	DDR333	167MHz
	HYB25D512160BC	32Mx16	DDR333	167MHz
	HYB25D512160BF	32Mx16	DDR333	167MHz

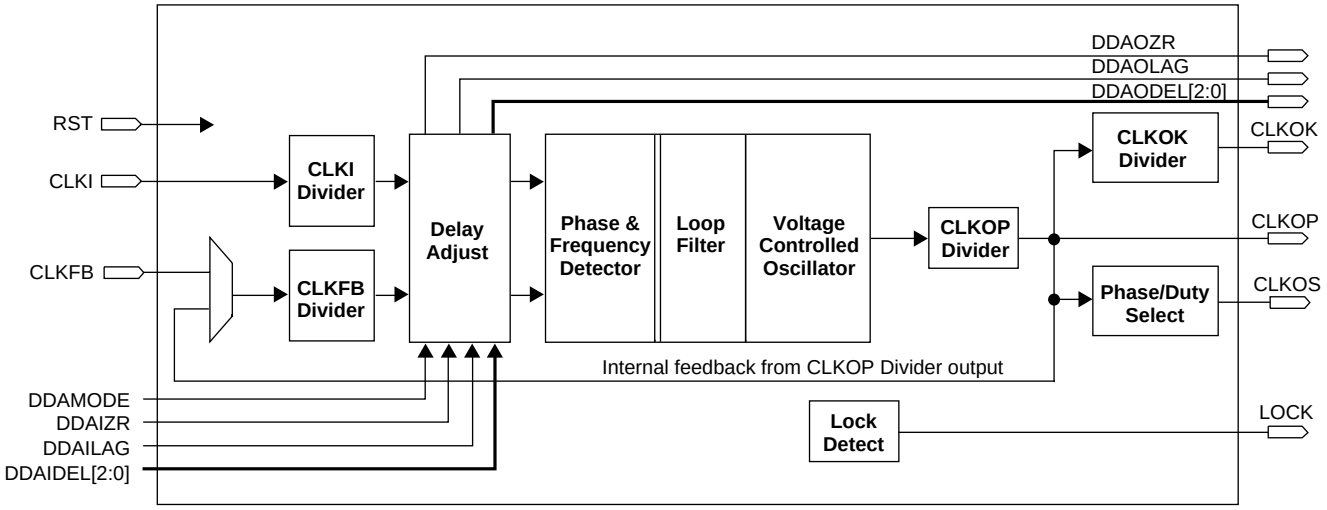
LatticeECP/EC sysCLOCK PLL Design and Usage Guide

June 2004 Technical Note TN1049

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™ and LatticeEC™ device families to help designers manage clocks within their designs. The PLL components in the LatticeECP/EC device families share the same architecture. This technical note describes the features and functionalities of the PLLs and their configuration in the ispLEVER® design tool. Figure 10-1 shows the block diagram of the PLL.

Figure 10-1. LatticeECP/EC 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 12. 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 12. 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. The divided value can be 2, 4, 6,..., 22, 24.

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 input is provided by an internally generated reset function (node). It resets the CLKI Divider. The RST signal is not required, and if not used will be set to logic 0. The PLL_RST signal is active high. The RST signal must be asserted for the minimum reset pulse width and de-asserted within the reset recovery time before the clock, as defined in the device data sheet. Asserting PLL reset when the CLKI Divider is set to 1 will not affect the operation of the PLL. The RST pin is most commonly used when multiple sysCLOCK PLLs are dividing the same input clock and a reset signal is needed to synchronize the PLLs.

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 (for EHXPLLB Only)
Refer to Table 10-2 and page 10-6 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.

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.

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.

WAKE_ON_LOCK
The WAKE_ON_LOCK cell determines if the device will wait for the PLL to lock before beginning the wake-up process. If the attribute is set to “ON”, the device will not wake up until the LOCK signal for the given PLL is active.

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 PLL Primitive Definitions

Two PLL primitives may be used for LatticeECP/EC PLL implementation. Users can choose either primitive depending on the design requirement of the PLL. The definitions of the PLL I/O ports are shown in Table 10-2. Some of the features are optional as shown in the table below.

Figure 10-2 shows the LatticeECP/EC PLL primitives library symbols and Table 10-1 lists the signals.

The EPLLB is a scaled down version of the PLL and is for users who do not need to use all the high performance features. Users may choose not to use optional features available in this primitive.

The EHXPLLB includes all features available in the PLL including Dynamic Delay Adjustment. Some of the features are optional as shown in Table 10-1.

Figure 10-2. LatticeECP/EC PLL Primitive Symbols

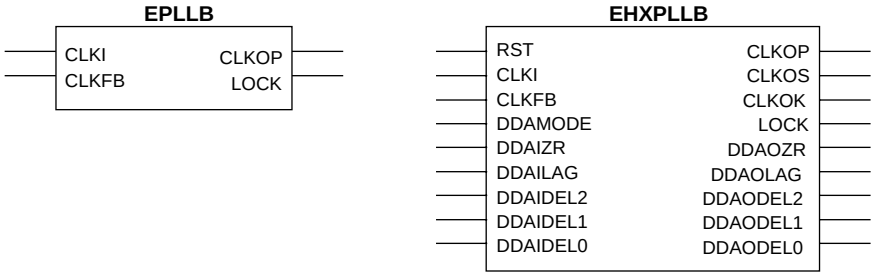


Table 10-1. Signal Usage in EPLLB and EHXPLLB Primitives

Signal	EPLLB	EHXPLLB	Optional
CLKI	X	X	No
CLKFB	X	X	No
CLKOP	X	X	Yes[1]
CLKOS	—	X	Yes[1]
CLKOK	—	X	Yes[1]
LOCK	X	X	Yes
RST	—	X	Yes
DDAMODE	—	X	Yes
DDAIZR		X	Yes
DDAILAG	—	X	Yes
DDAIDEL(0:2)	—	X	Yes
DDAOZR		X	Yes
DDAOLAG	—	X	Yes
DDAODEL(0:2)	—	X	Yes

1. At least one of these output clocks must be used.

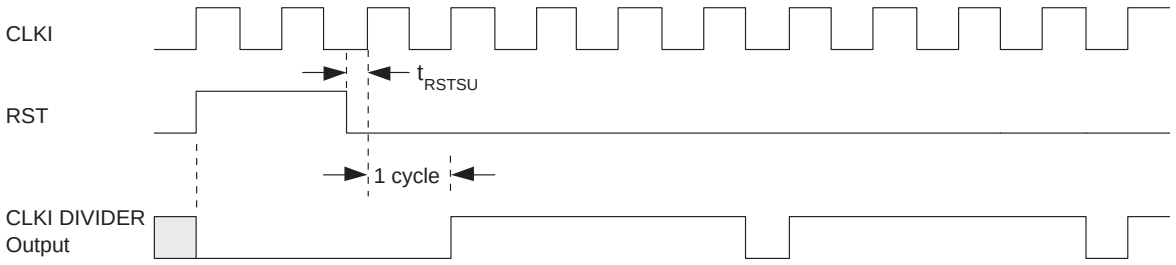
Table 10-2. LatticeECP/EC PLL I/O Definitions

Signal	I/O	Description	Freq. (MHz)[2]
CLKI	I	PLL clock input from either internal logic or dedicated clock input pin.	33 to 420
CLKFB[1]	I	Feedback clock input from either internal node, internal feedback from CLKOP or dedicated external feedback pin.	33 to 420
RST	I	“1” to Reset CLKI-divider (M-divider).	—
CLKOP	O	PLL output clock to clock tree (no phase shift).	33 to 420
CLKOS	O	PLL output clock to clock tree (programmable phase shift/duty cycle).	33 to 420
CLKOK	O	PLL output to clock tree (through K-divider for lower frequency clock).	0.258 to 210
LOCK	O	“1” indicates PLL locked to CLKI.	—
DDAMODE	I	DDA Mode. “1” Pin control (dynamic), “0”: Fuse Control (static).	—
DDAIZR	I	DDA Delay Zero. “1”: delay = 0, “0”: delay = [DDILAG + DDAIDEL].	—
DDAILAG	I	DDA Lag/Lead. “1”: Lag, “0”: Lead.	—
DDAIDEL[2:0]	I	DDA Delay	—
DDAOZR	O	DDA Delay Zero Output	—
DDAOLAG	O	DDA Lag/Lead Output	—
DDAODEL[2:0]	O	DDA Delay Output	—

1. For internal feedback, user does not specify CLKFB and software implements internal feedback automatically.
2. Please refer to data sheet for latest data.

RST: Resets the timing of CLKI Divider Output referenced to CLKI as shown in Figure 10-3. The reset operation is used to synchronize the clock output phase when multiple PLLs are cascaded. This RST signal does not reset the divider value.

Figure 10-3. RST Timing Diagram (Example M=4)



PLL Attributes Definitions

Both EPLLB and EHXPLLB can be configured through attributes in the source code. The following section details these attributes and their usage.

Table 10-3. LatticeECP/EC PLL Attributes

Parameter	Description	Value	EPLLB	EHXPLLB	Default
FIN	Input Frequency (MHz)	33.0000 to 420.0000	Yes	Yes	—
CLKI_DIV	CLKI Divider Setting	1 to 12	Yes	Yes	1
CLKFB_DIV	CLKFB Divider Setting	1 to 12	Yes	Yes	1
CLKOP_DIV	CLKOP Divider Setting	2,4,6,.., 22, 24	Yes	Yes	Note 1
CLKOK_DIV	CLKOK Divider Setting	2,4,6,..,126,128	—	Yes	2
FDEL	Fine Delay Adjust	-8 to 8	Yes	Yes	0
PHASEADJ	Coarse Phase Shift Selection (O)	0, 45, 90...315	—	Yes	0
DUTY	Duty Cycle Selection (1/8 increment)	1 to 7	—	Yes	4
WAKE_ON_LOCK	Wake up after PLL locked	On/Off	Yes	Yes	Off
DELAY_CNTL	Delay Control	Dynamic/Static	—	Yes	Static

1. CLKOP_DIV value is calculated to maximize the FVCO within the specified range based on CLKI_DIV and CLKFB_DIV values for optimum performance.

Dynamic Delay Adjustment (for EHXPLLB only)

The Dynamic Delay Adjustment is controlled by the DDAMODE input. This feature is available in the EHXPLLB primitive only. When the DDAMODE input is set to "1", the delay control is done through the inputs, DDAIZR, DDAILAG and DDAIDEL(2:0). For this mode, the attribute "DELAY_CNTL" must be set to "DYNAMIC". Table 10-4 shows the delay adjustment values based on the attribute/input settings.

In this mode, the PLL may come out of lock due to the abrupt change of phase. To ensure that the PLL is back in lock, it is recommended to wait for the period t_{LOCK} specified in the data sheet.

Table 10-4. Delay Adjustment

DDAMODE = 1: Dynamic Delay Adjustment			DELAY 1 t_{DLY} = 250ps (nominal)	DDAMODE = 0
DDAIZR	DDAILAG	DDAIDEL[2:0]		Equivalent FDEL Value
0	0	111	Lead 8 t_{DLY}	-8
0	0	110	Lead 7 t_{DLY}	-7
0	0	101	Lead 6 t_{DLY}	-6
0	0	100	Lead 5 t_{DLY}	-5
0	0	011	Lead 4 t_{DLY}	-4
0	0	010	Lead 3 t_{DLY}	-3
0	0	001	Lead 2 t_{DLY}	-2
0	0	000	Lead 1 t_{DLY}	-1
1	Don't Care	Don't Care	No delay	0
0	1	000	Lag 1 t_{DLY}	1
0	1	001	Lag 2 t_{DLY}	2
0	1	010	Lag 3 t_{DLY}	3
0	1	011	Lag 4 t_{DLY}	4
0	1	100	Lag 5 t_{DLY}	5
0	1	101	Lag 6 t_{DLY}	6
0	1	110	Lag 7 t_{DLY}	7
0	1	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

Table 10-5. Frequency Limits

Parameter	Description	Values, Ranges
f_{IN}	CLKI, CLKFB Frequency	33MHz ≤ f_{IN} ≤ 420MHz
f_{OUT}	CLKOP, CLKOS Frequency	33MHz ≤ f_{OUT} ≤ 420MHz
f_{OUTK}	CLKOK Frequency	258KHz ≤ f_{OUT} ≤ 210MHz
f_{VCO}	VCO Operating Frequency	420MHz ≤ f_{VCO} ≤ 840MHz
CLKI Divider	Input Clock Divider	1 to 12
CLKFB Divider	Feedback Clock Divider	1 to 12
CLKOP Divider	Output Clock Divider	2,4,6,8,..,22, 24
CLKOK Divider	CLKOK Output Clock Divider	2,4,6,8,..,126,128
Maximum (N * V)	CLKFB_DIV * CLKOP_DIV	24
f_{PFD} (f_{IN} / M) (Hz)	PFD input Frequency	33MHz ≤ f_{IN}/M ≤ 420MHz

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.

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.

Please refer to Figure 10-1 for the discussion below.

f_{VCO} Constraint

From the loop:

$$f_{OUT} = f_{IN} * (N/M) \quad (1)$$

From the loop:

$$f_{VCO} = f_{OUT} * V \quad (2)$$

Substitute (1) in (2) yields:

$$f_{VCO} = f_{IN} * (N/M) * V \quad (3)$$

Arrange (3):

$$f_{IN} = (f_{VCO} / (V*N))*M \quad (4)$$

From equation (4):

$$f_{INMIN} = ((f_{VCOMIN} /(V*N))*M \quad (5)$$

$$f_{INMAX} = (f_{VCOMAX}/(V*N))*M \quad (6)$$

f_{PFD} Constraint

From the loop:

$$f_{PFD} = f_{IN} / M \quad (7)$$

$$f_{IN} = f_{PFD} * M$$

$$f_{INMIN} = f_{PFDMIN} * M = 33 * M \quad (8)$$

Equation (5) becomes:

$$\mathbf{f_{INMIN} = ((f_{VCOMIN} / (V*N))*M, \text{ if below } 33 * M \text{ round up to } 33 * M} \quad (9)$$

From the loop:

$$f_{INMAX} = f_{PFDMAX} * M = 420 * M \quad (10)$$

From the table, $f_{INMAX} = 420$

Equation (6) becomes:

$$\mathbf{f_{INMAX} = (f_{VCOMAX} / (V*N))*M, \text{ if above } 420 \text{ round down to } 420} \quad (11)$$

From equation (1):

$$\mathbf{f_{OUTMIN} = f_{INMIN} * (N/M), \text{ if below } 33 * N \text{ round up to } 33 * N} \quad (12)$$

$$\mathbf{f_{OUTMAX} = f_{INMAX} * (N/M), \text{ if above } 420 \text{ round down to } 420} \quad (13)$$

$$f_{OUTKMIN} = f_{OUTMIN} / K$$

$$f_{OUTKMAX} = f_{OUTMAX} / K$$

Example

Assume:

f_{IN} = 40MHz, M = 2, N = 3, V = 5

Then:

f_{OUT} = 40 * 3 / 2 = 60 (within range)
f_{VCO} = 60 * 5 = 300 (out of range)
f_{PFD} = 40 / 2 = 20 or 60 / 3 = 20 (out of range)

Let's assume M =1. Then:

f_{OUT} = 40 * 3 / 1 = 120 (within range)
f_{VCO} = 120 * 5 = 600 (out of range)
f_{PFD} = 40 / 1 or 120 / 3 = 40 (within range)

In this case, V=6 will satisfy all conditions.

PLL Usage in Module Manager and HDL

Including sysCLOCK PLLs in a Design

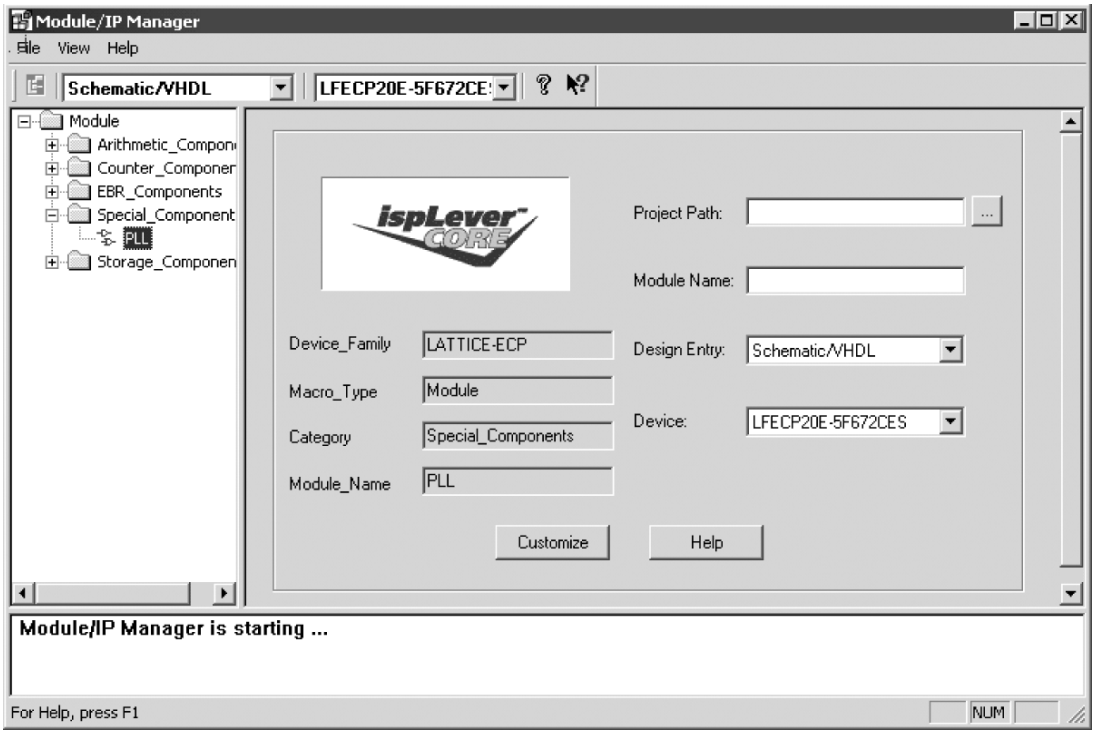
The sysCLOCK PLL capability can be accessed either through the Module Manager or directly instantiated in a design's source code. The following sections describe both methods.

Module Manager Usage

The LatticeECP/EC PLL is fully supported in Module Manager in the ispLEVER software. The Module Manager 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 10-4 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 "General Options" window as shown in Figure 10-5.

Figure 10-4. Module IP Manager Main Window



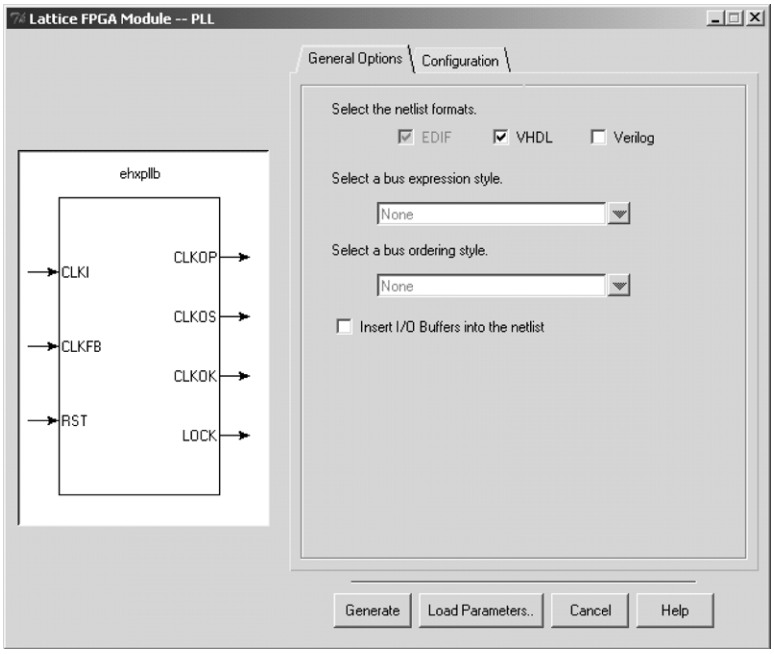
General Options Tab

The General Options Tab provides the ability to define the following:

- Netlist format type: EDIF, VHDL or Verilog
- Synthesizer: Synplicity or Exemplar

Clicking 'Generate' creates a VHDL (module name.vhd) or Verilog (module name.v) file in the working directory that instantiates the core. At the same time a parameter file (module name.lpc) file is created in the working directory. The load parameters button can be used to reload configurations from previously created parameter files (*.lpc files).

Figure 10-5. General Options Tab



The General Options Tab also provides the ability to define the following:

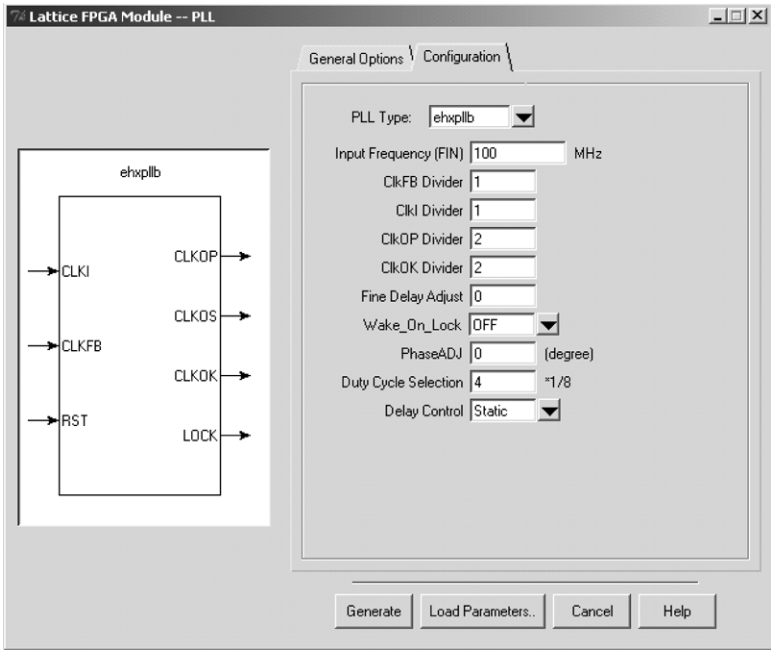
- Bus expression style
- Bus ordering style
- Insertion of I/O Buffers in the netlist

These options are reserved for future use.

Configuration Tab

The Configuration Tab lists all user accessible attributes. Default values are set initially. Figure 10-6 shows the Configuration Tab. The 'Generate' and 'Load parameters' buttons operate in same way as in the General Options Tab.

Figure 10-6. Configuration Tab



Direct Instantiation Into Source Code

If desired, the Module Manager can be bypassed and the sysCLOCK PLL instantiated directly in the source code. Appendix A provides examples of source code generated by the Module Manager. These examples can be used as templates for directly instantiating the sysCLOCK PLL in the source code.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-408-826-6002 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Appendix A. Source Code Examples Generated by Module Manager

EPLLB Module (Verilog)

```
`timescale 1 ns / 100 ps
module  epll ( CLKI, CLKFB, CLKOP, LOCK);

input   CLKI;
input   CLKFB;
output   CLKOP;
output   LOCK;

defparam PllInst0.FIN = “33”;
defparam PllInst0.CLKI_DIV = “1”;
defparam PllInst0.CLKOP_DIV = “16”;
defparam PllInst0.CLKFB_DIV = “2”;
defparam PllInst0.FDEL = “-5”;
defparam PllInst0.WAKE_ON_LOCK = “OFF”;
EPLLB PllInst0(.LOCK(LOCK), .CLKOP(CLKOP), .CLKFB(CLKFB), .CLKI(CLKI));

endmodule
```

EPLLB Module (VHDL)

```
--VHDL netlist generated by scuba
--timeStamp 2004 5 06 10 59 34
USE STD.TEXTIO.ALL;
Library IEEE, EC;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE EC.COMPONENTS.ALL;
USE WORK.ALL;
ENTITY epll IS

        PORT (
        CLKI                :IN    std_logic;
        CLKFB               :IN    std_logic;
        CLKOP               :OUT   std_logic;
        LOCK                :OUT   std_logic);

END epll;

ARCHITECTURE V OF epll IS
  COMPONENT EPLLB
        GENERIC  (
        FIN : string :=  “33”;
        CLKI_DIV : string := “1”;
        CLKOP_DIV : string :=  “12”;
        CLKFB_DIV : string := “2”;
        FDEL : string :=  “0”;
        WAKE_ON_LOCK : string :=  “ON”);
        PORT (
        CLKI                :IN    std_logic;
        CLKFB               :IN    std_logic;
        CLKOP               :OUT   std_logic;
        LOCK                :OUT   std_logic);
  END COMPONENT;
```

```
BEGIN
    PllInst0: EPLLB
        GENERIC MAP (
        FIN => “33”,
        CLKI_DIV => “1”,
        CLKOP_DIV => “12”,
        CLKFB_DIV => “2”,
        FDEL => “0”,
        WAKE_ON_LOCK => “ON”)
        PORT MAP (
        LOCK                =>  LOCK,
        CLKOP               =>  CLKOP,
        CLKFB               =>  CLKFB,
        CLKI                =>  CLKI);

END V;
```

EHXPLLB Module (Verilog)

```
`timescale 1 ns / 100 ps
module  pllx ( CLKI, CLKFB, RST, DDAMODE, DDAIZR, DDAILAG, DDAIDEL0, DDAIDEL1,
               DDAIDEL2, CLKOP, CLKOS, CLKOK, LOCK, DDAOZR, DDAOLAG, DDAODEL0,
               DDAODEL1, DDAODEL2);

input   CLKI, CLKFB, RST, DDAMODE, DDAIZR, DDAILAG, DDAIDEL0, DDAIDEL1, DDAIDEL2;
output   CLKOP, CLKOS, CLKOK, LOCK, DDAOZR, DDAOLAG, DDAODEL0, DDAODEL1, DDAODEL2;

defparam PllInst0.FIN = “100”;
defparam PllInst0.CLKI_DIV = “1”;
defparam PllInst0.CLKOP_DIV = “2”;
defparam PllInst0.CLKFB_DIV = “1”;
defparam PllInst0.FDEL = “0”;
defparam PllInst0.WAKE_ON_LOCK = “OFF”;
defparam PllInst0.CLKOK_DIV = “2”;
defparam PllInst0.PHASEADJ = “270”;
defparam PllInst0.DUTY = “4”;
defparam PllInst0.DELAY_CNTL = “STATIC”;

EHXPLLB PllInst0(.DDAODEL2(DDAODEL2), .DDAODEL1(DDAODEL1), .DDAODEL0(DDAODEL0),
               .DDAOLAG(DDAOLAG), .DDAOZR(DDAOZR), .LOCK(LOCK), .CLKOK(CLKOK),
               .CLKOS(CLKOS), .CLKOP(CLKOP), .DDAIDEL2(DDAIDEL2), .DDAIDEL1(DDAIDEL1),
               .DDAIDEL0(DDAIDEL0), .DDAILAG(DDAILAG), .DDAIZR(DDAIZR),
               .DDAMODE(DDAMODE), .RST(RST), .CLKFB(CLKFB), .CLKI(CLKI));

endmodule
```

EHXPLLB Module (VHDL)

```
--VHDL netlist generated by scuba
--timeStamp 2004 3 24 11 14 21
USE STD.TEXTIO.ALL;
Library IEEE, EC;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE EC.COMPONENTS.ALL;
USE WORK.ALL;
```

```
ENTITY ehxpllb_top IS

        PORT (
        CLKI                :IN    std_logic;
        CLKFB               :IN    std_logic;
        RST                 :IN    std_logic;
        DDAMODE             :IN    std_logic;
        DDAIZR              :IN    std_logic;
        DDAILAG             :IN    std_logic;
        DDAIDEL0            :IN    std_logic;
        DDAIDEL1            :IN    std_logic;
        DDAIDEL2            :IN    std_logic;
        CLKOP               :OUT   std_logic;
        CLKOS               :OUT   std_logic;
        CLKOK               :OUT   std_logic;
        LOCK                :OUT   std_logic;
        DDAOZR              :OUT   std_logic;
        DDAOLAG             :OUT   std_logic;
        DDAODEL0            :OUT   std_logic;
        DDAODEL1            :OUT   std_logic;
        DDAODEL2            :OUT   std_logic);

END ehxpllb_top;

ARCHITECTURE V OF ehxpllb_top IS
  COMPONENT EHXPLLB
        GENERIC  (
              FIN : string :=  “100”;
              CLKI_DIV : string := “1”;
              CLKOP_DIV : string :=  “1”;
              CLKFB_DIV : string := “1”;
              FDEL : string :=  “0”;
              CLKOK_DIV : string :=  “2”;
              PHASEADJ : string :=  “0”;
              DUTY : string :=  “4”;
              DELAY_CNTL : string :=  “STATIC”;
              WAKE_ON_LOCK : string :=  “OFF”);
        PORT (
              CLKI                :IN    std_logic;
              CLKFB               :IN    std_logic;
              RST                 :IN    std_logic;
              DDAMODE             :IN    std_logic;
              DDAIZR              :IN    std_logic;
              DDAILAG             :IN    std_logic;
              DDAIDEL0            :IN    std_logic;
              DDAIDEL1            :IN    std_logic;
              DDAIDEL2            :IN    std_logic;
              CLKOP               :OUT   std_logic;
              CLKOS               :OUT   std_logic;
              CLKOK               :OUT   std_logic;
              LOCK                :OUT   std_logic;
              DDAOZR              :OUT   std_logic;
              DDAOLAG             :OUT   std_logic;
              DDAODEL0            :OUT   std_logic;
              DDAODEL1            :OUT   std_logic;
              DDAODEL2            :OUT   std_logic);
  END COMPONENT;
```

```
BEGIN
    PllInst0: EHXPLLB
        GENERIC MAP (
              FIN => “100”,
              CLKI_DIV => “1”,
              CLKOP_DIV => “1”,
              CLKFB_DIV => “1”,
              FDEL => “0”,
              WAKE_ON_LOCK => “OFF”,
              CLKOK_DIV => “2”,
              PHASEADJ => “0”,
              DUTY => “4”,
              DELAY_CNTL => “STATIC”)
        PORT MAP (
              DDAODEL2             =>  DDAODEL2,
              DDAODEL1             =>  DDAODEL1,
              DDAODEL0             =>  DDAODEL0,
              DDAOLAG              =>  DDAOLAG,
              DDAOZR               =>  DDAOZR,
              LOCK                 =>  LOCK,
              CLKOK                =>  CLKOK,
              CLKOS                =>  CLKOS,
              CLKOP                =>  CLKOP,
              DDAIDEL2             =>  DDAIDEL2,
              DDAIDEL1             =>  DDAIDEL1,
              DDAIDEL0             =>  DDAIDEL0,
              DDAILAG              =>  DDAILAG,
              DDAIZR               =>  DDAIZR,
              DDAMODE              =>  DDAMODE,
              RST                  =>  RST,
              CLKFB                =>  CLKFB,
              CLKI                 =>  CLKI);

END V;
```

Appendix B. A Complete Project Example with Test Bench for Modelsim in VHDL

Top Module

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
-- synthesis translate_off
library ec;
use ec.all;
-- synthesis translate_on

entity plltest is
       port (
              sysrst        :in std_logic;
              sysclk        :in std_logic;
              clklock       :out std_logic;
              clkout        :out std_logic;
              pllreg        :out std_logic_vector(15 downto 0)
              );
end;

architecture behave of plltest is

component spll
        PORT (
              CLKI          :IN std_logic;
              CLKFB         :IN std_logic;
              CLKOP         :OUT std_logic;
              LOCK          :OUT std_logic);
END component;

signal pllclk: std_logic;
signal pllreg_int : std_logic_vector(15 downto 0);

begin

PLLInst0: spll
port map (
CLKI         => sysclk,
        CLKFB        => pllclk,
        CLKOP        => pllclk,
        LOCK         => clklock
        );

clkout <= pllclk;

pllinst1: process(pllclk, sysrst)
       begin
              if (sysrst = '0') then
                     pllreg_int <= (others => '0');
              elsif rising_edge(pllclk) then
                     pllreg_int <= pllreg_int + 1;
              end if;
       end process;
       pllreg <= pllreg_int;

end behave;
```

PLL Module

```
--VHDL netlist generated by scuba
--timeStamp 2004 3 31 10 54 03
USE STD.TEXTIO.ALL;
Library IEEE, EC;
USE IEEE.std_logic_1164.ALL;
USE IEEE.std_logic_arith.ALL;
USE EC.COMPONENTS.ALL;
USE WORK.ALL;
ENTITY spll IS

        PORT (
        CLKI                :IN    std_logic;
        CLKFB               :IN    std_logic;
        CLKOP               :OUT   std_logic;
        LOCK                :OUT   std_logic);

END spll;

ARCHITECTURE V OF spll IS
  COMPONENT EPLLB
        GENERIC  (
        FIN : string :=  “50”;
        CLKI_DIV : string := “1”;
        CLKOP_DIV : string :=  “3”;
        CLKFB_DIV : string := “5”;
        FDEL : string :=  “0”;
        WAKE_ON_LOCK : string :=  “OFF”);
        PORT (
        CLKI                :IN    std_logic;
        CLKFB               :IN    std_logic;
        CLKOP               :OUT   std_logic;
        LOCK                :OUT   std_logic);
  END COMPONENT;

BEGIN
    PllInst0: EPLLB
        GENERIC MAP (
        FIN => “50”,
        CLKI_DIV => “1”,
        CLKOP_DIV => “3”,
        CLKFB_DIV => “5”,
        FDEL => “0”,
        WAKE_ON_LOCK => “OFF”)
        PORT MAP (
        LOCK                 =>  LOCK,
        CLKOP                =>  CLKOP,
        CLKFB                =>  CLKFB,
        CLKI                 =>  CLKI);

END V;
```

Test Bench

```
-- VHDL Test Bench Created from source file plltest.vhd -- 03/31/04  10:24:18
-- Notes:
-- 1) This testbench template has been automatically generated using types
-- std_logic and std_logic_vector for the ports of the unit under test.
-- Lattice recommends that these types always be used for the top-level
-- I/O of a design in order to guarantee that the testbench will bind
-- correctly to the timing (post-route) simulation model.
-- 2) To use this template as your testbench, change the filename to any
-- name of your choice with the extension .vhd, and use the “source->import”
-- menu in the ispLEVER Project Navigator to import the testbench.
-- Then edit the user defined section below, adding code to generate the
-- stimulus for your design.
--
LIBRARY ieee;
LIBRARY generics;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;
USE generics.components.ALL;

ENTITY plltest_tb IS
END plltest_tb;

ARCHITECTURE behavior OF plltest_tb IS

COMPONENT plltest
PORT(
       sysrst: in std_logic;
       sysclk : IN std_logic;
       clklock: out std_logic ;
       clkout   : out std_logic;
       pllreg:out std_logic_vector(15 downto 0)
       );
END COMPONENT;

signal sysrst : std_logic := ‘0’;
SIGNAL sysclk :  std_logic := ‘0’;
SIGNAL clklock : std_logic;
SIGNAL clkout: std_logic;
signal pllcnt:std_logic_vector(15 downto 0);

constant clk_cyc0 : time := 30 ns;

BEGIN
sysclk <= not sysclk after clk_cyc0/2;

uut: plltest PORT MAP(
       sysrst => sysrst,
       sysclk => sysclk,
       clklock => clklock,
       clkout => clkout,
       pllreg => pllcnt
);
```

```
-- *** Test Bench - User Defined Section ***
   tb : PROCESS
   BEGIN
      wait for 10 ns;
      sysrst <= '1';
      wait; -- will wait forever
   END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

LatticeECP-DSP sysDSP Usage Guide

June 2004 Technical Note TN1057

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 11-1 provides an example of the performance and area benefits of this approach.

Table 11-1. sysDSP Block vs. LUT-based Multipliers

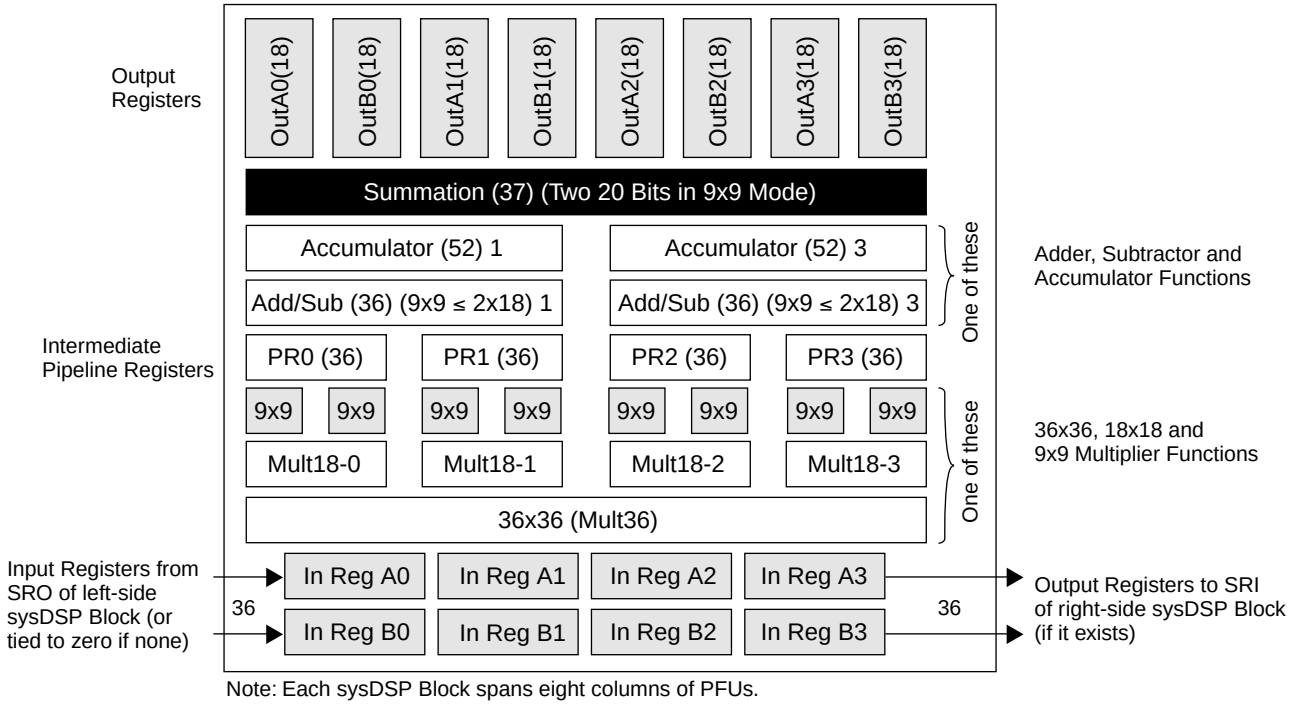
Multiplier Width	Register Pipelining	LatticeECP LFECP20E-5 Uses One DSP Block		LatticeEC LFEC20E-5 Uses LUTs	
		f_{MAX}	LUTs	f_{MAX}	LUTs
9x9	Input, Multiplier, Output	235	0	76	174
18x18	Input, Multiplier, Output	211	0	50	608
36x36	Input, Multiplier, Output	177	0	35	2225

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 11-1.

Figure 11-1. 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.

- The Module/IP Manager 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 the Module/IP Manager

The Module/IP Manager allows you to graphically specify sysDSP elements. Once the element is specified, an HDL file is generated which can be instantiated in a design. The Module/IP Manager allows users to configure all ports and set all available parameters. The following show modules which target the sysDSP Block. For design examples using the Module Manager, 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 the Module Manager:

- MULT (Multiplier)
- MAC (Multiplier Accumulate)
- MULTADD (Multiplier Add/Subtract)
- MULTADDSUM (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 11-2 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 11-3, allows finer control over the register. In the Advanced mode you can control each register with independent clocks, clock enables and resets.

Figure 11-2. MULT Mode Basic Set-up

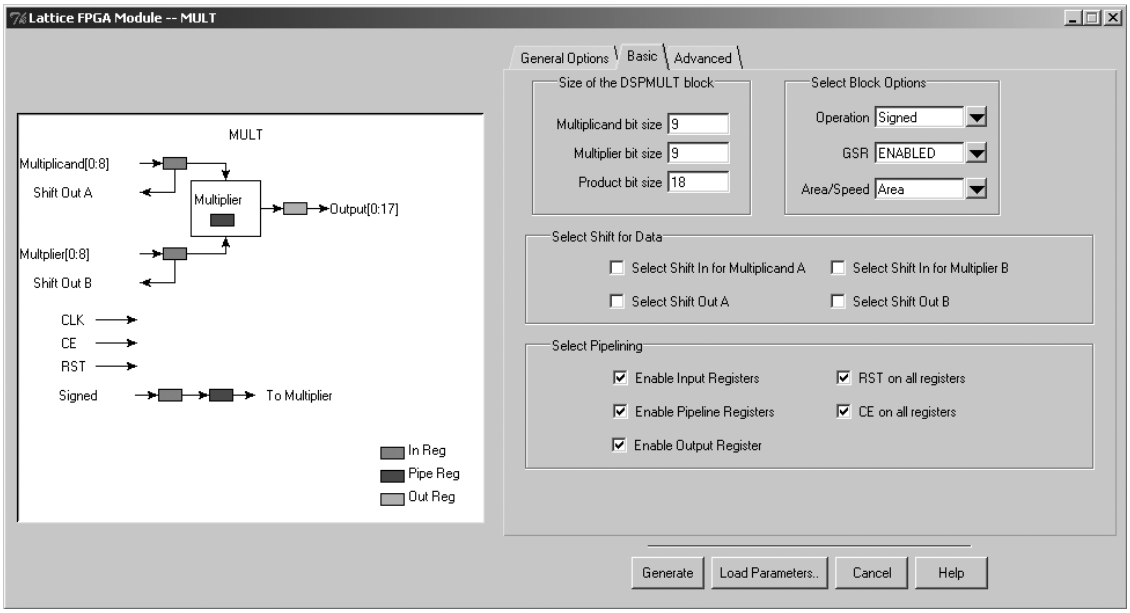
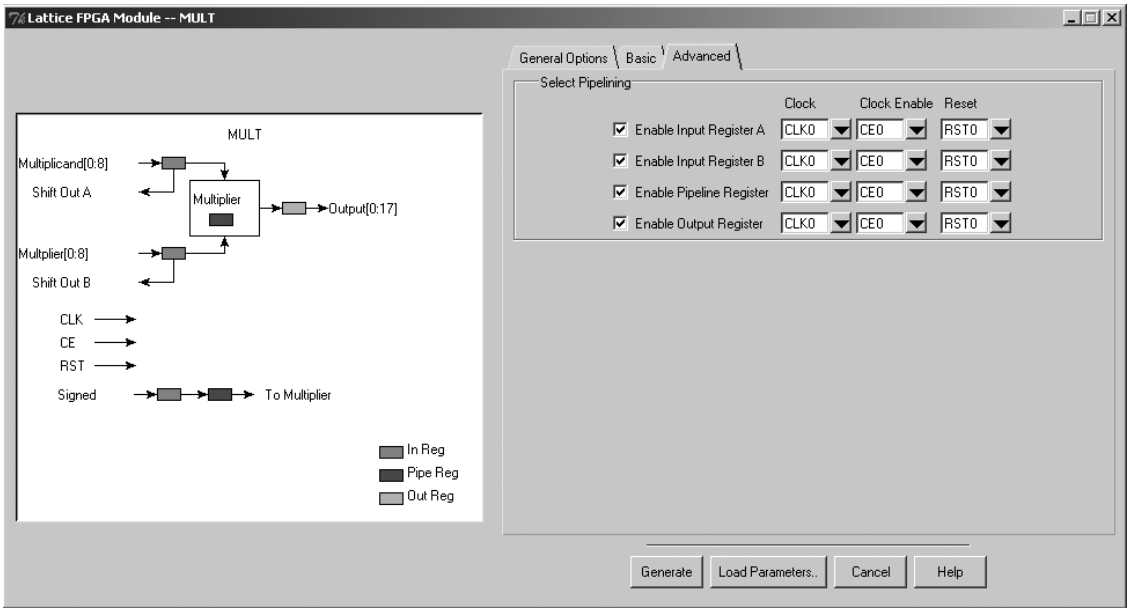


Figure 11-3. MULT Mode Advanced Set-up



MAC Module

The MAC Module configures multiply accumulate elements to be packed into the primitives. The Basic mode, shown in Figure 11-4, 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 11-5, allows finer control over the registers. In the Advanced mode you can control each register with independent clocks, clock enables and resets.

Figure 11-4. MAC Mode Basic Set-up

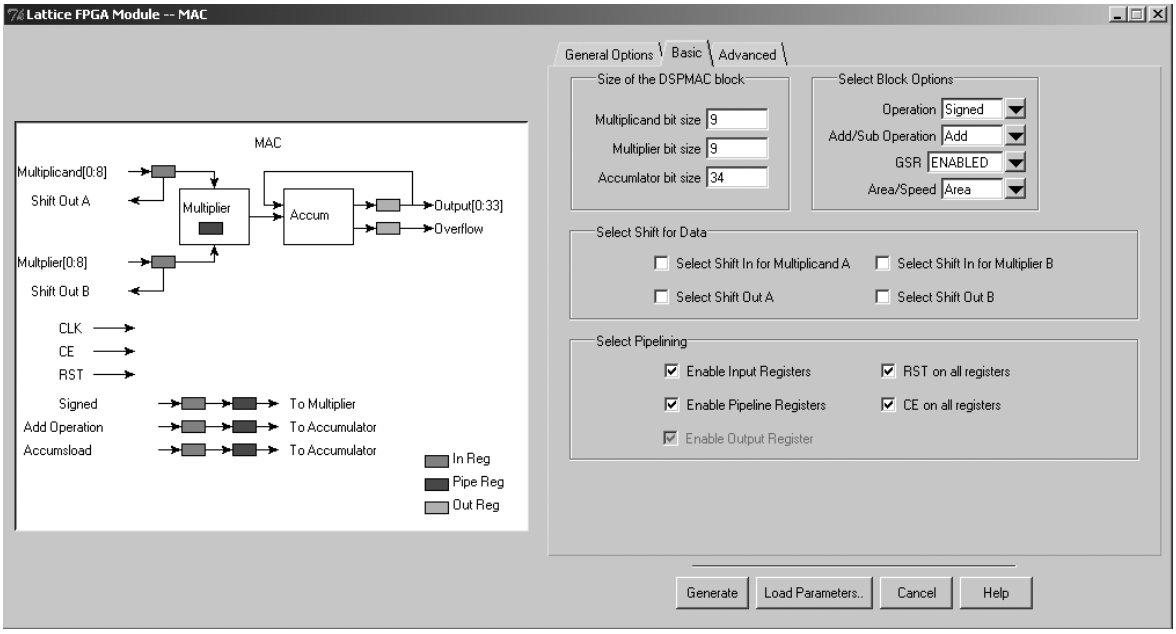
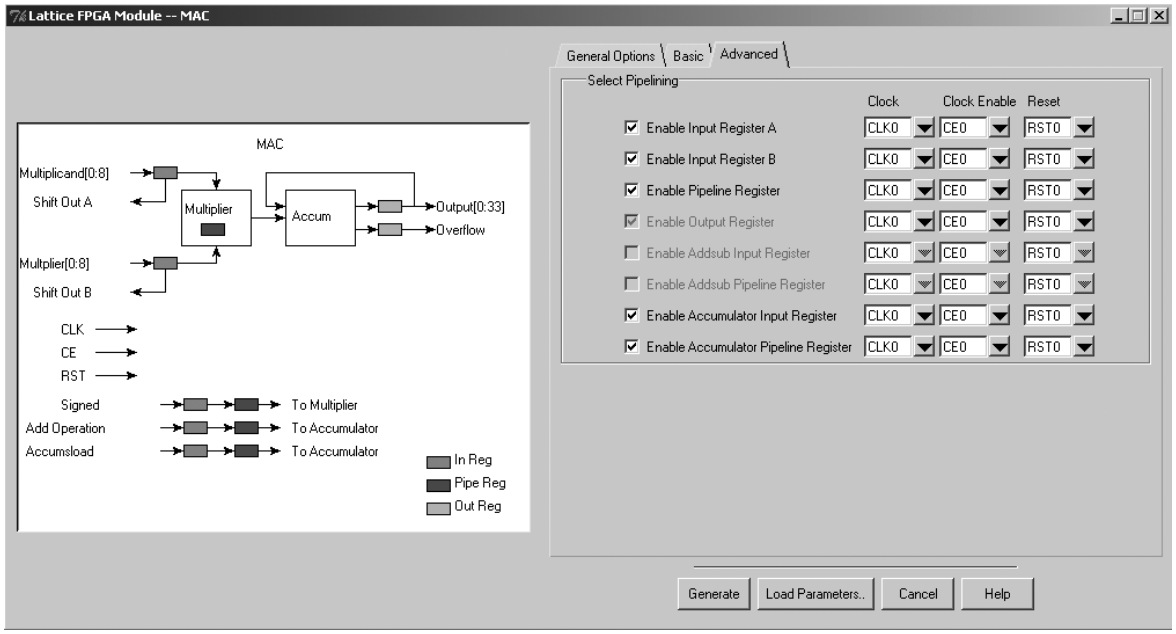


Figure 11-5. MAC Mode Advanced Set-up



MULTADD Module

The MULTADD GUI configures Multiplier Addition/Subtraction elements to be packed into the primitives MULT18X18ADDSUB or MULT9X9ADDSUB. The Basic mode, shown in Figure 11-6, consists of one clock (optional), one clock enable and one reset tied to all registers. Using the MULTADD 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 11-7, provides finer control over the registers. In the Advanced mode you can control each register with independent clocks, clock enables and resets.

Figure 11-6. MULTADD Mode Basic Set-up

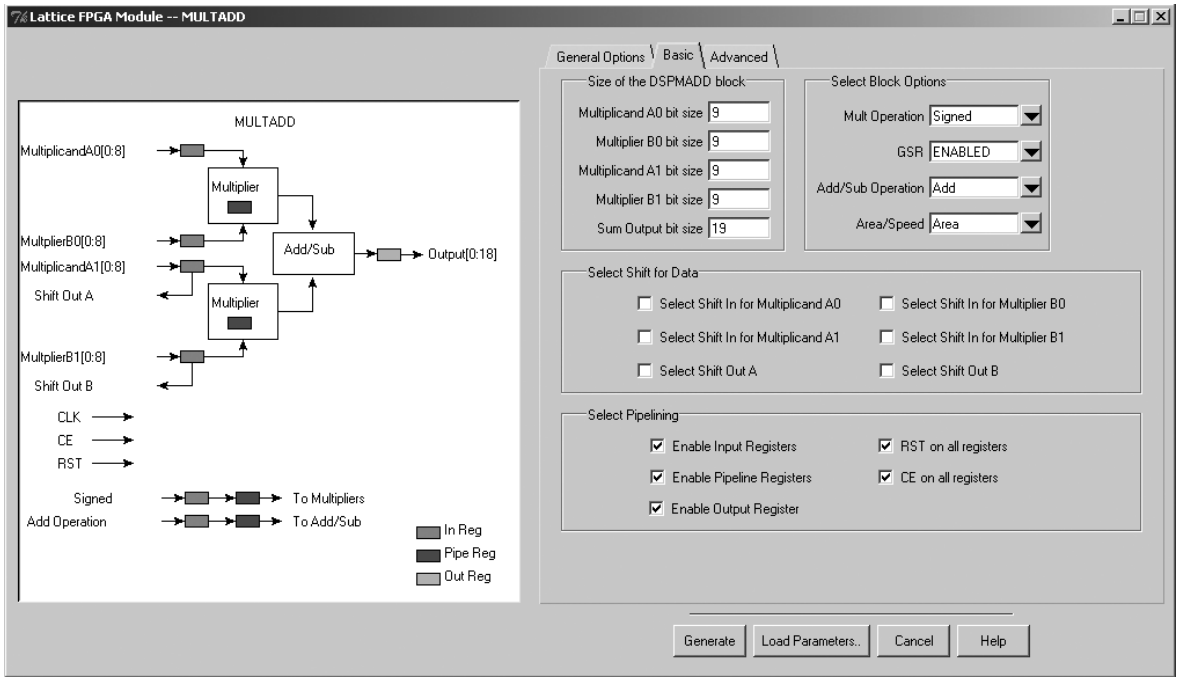
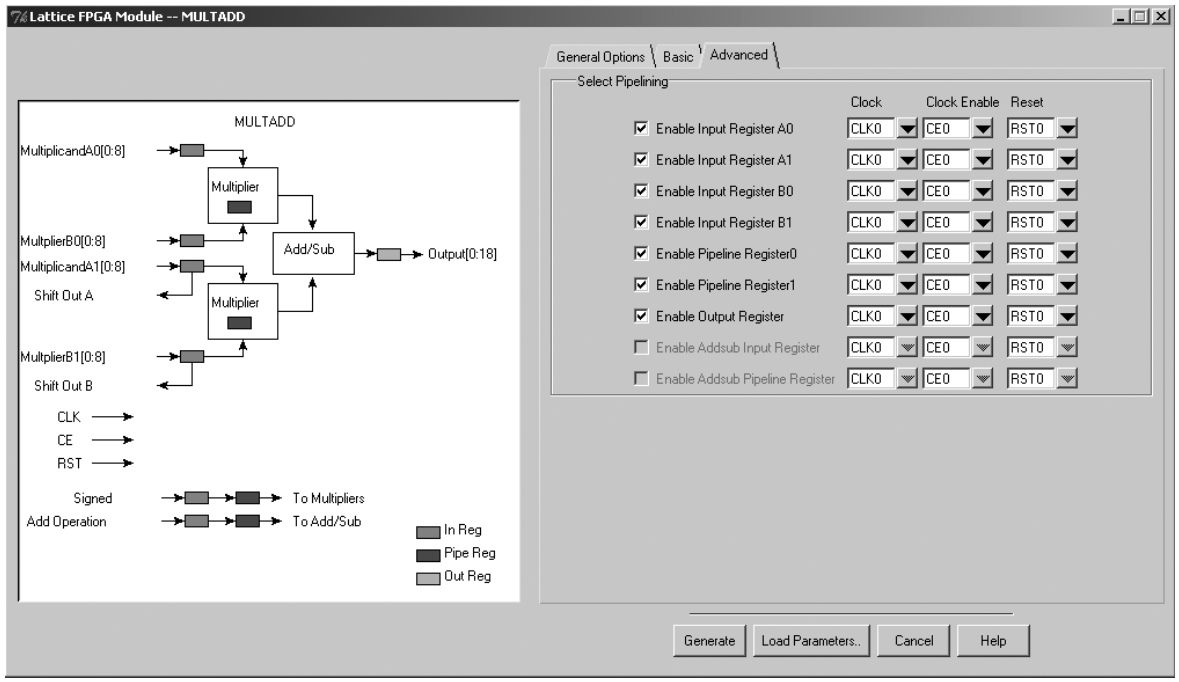


Figure 11-7. MULTADD Mode Advanced Set-up



MULTADDSUM Module

The MULTADDSUM GUI configures Multiplier Addition/Subtraction Addition elements to be packed into the primitives MULT18X18ADDSUBSUM or MULT9X9ADDSUBSUM. The Basic mode, shown in Figure 11-8, consists of one clock (optional), one clock enable and one reset tied to all registers. Using the MULTADDSUM 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 11-9, provides finer control over the registers. In the Advanced mode you can control each register with independent clocks, clock enables and resets.

Figure 11-8. MULTADDSUM Mode Basic Set-up

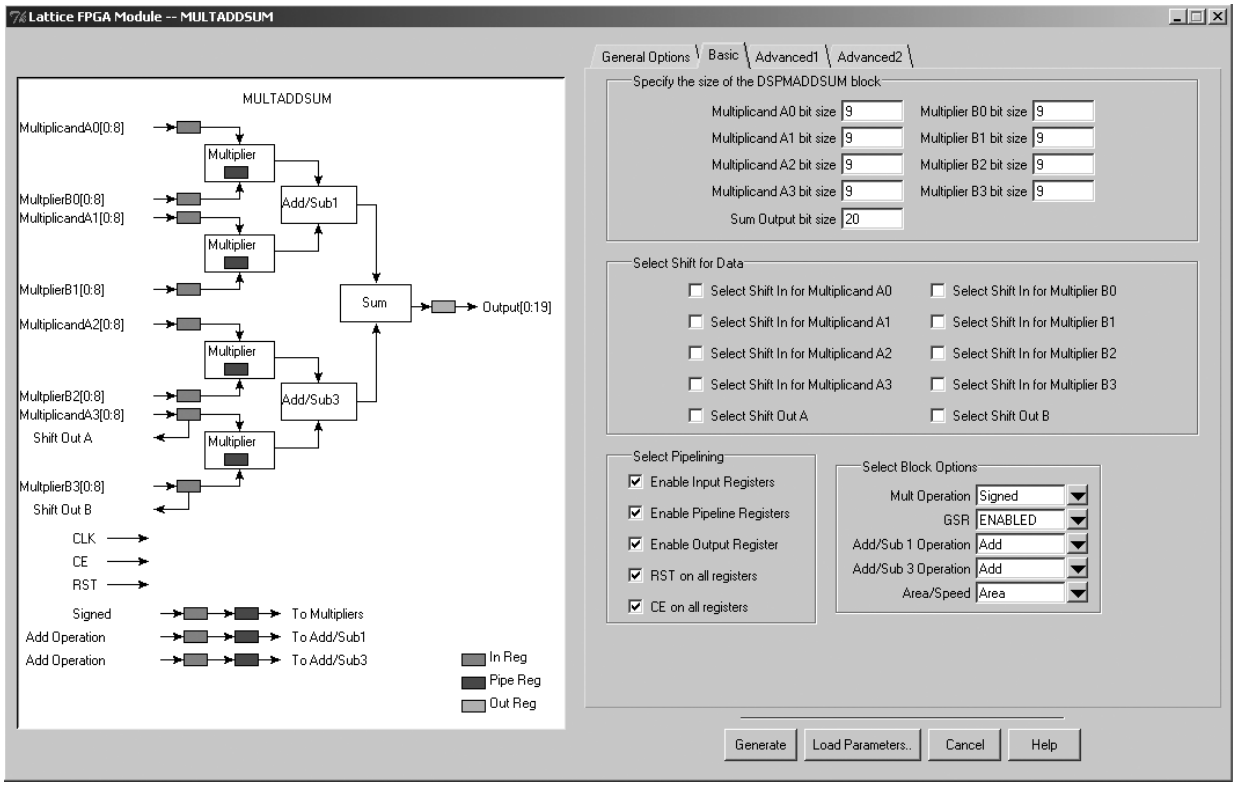


Figure 11-9. MULTADDSUM Mode Advance 1 Set-up

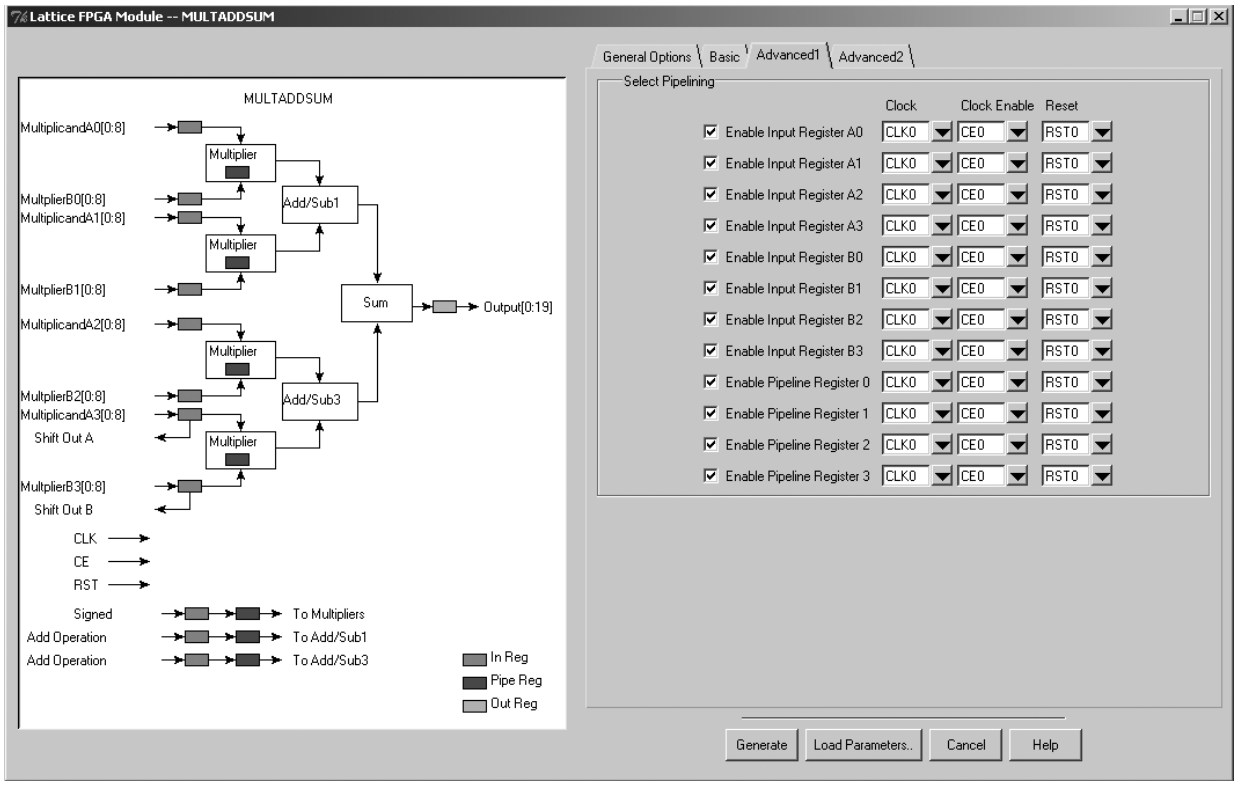
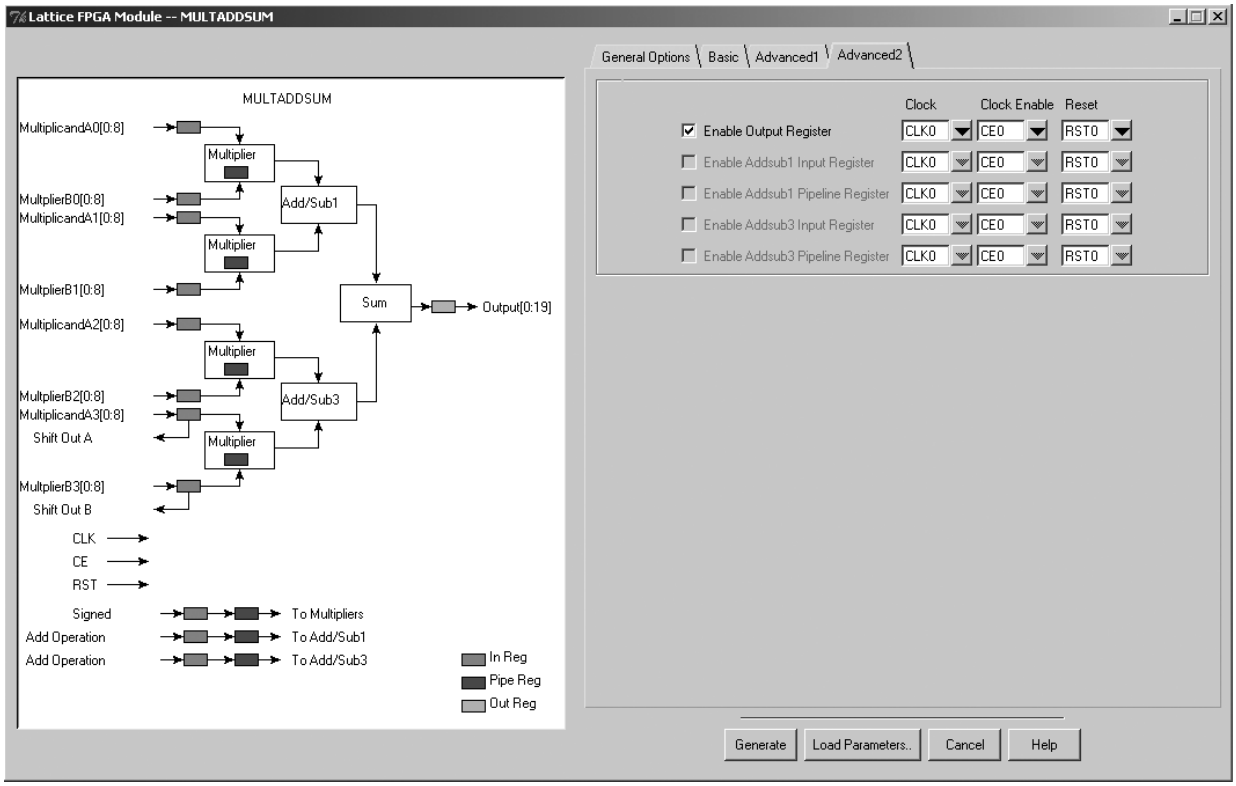


Figure 11-10. MULTADDSUM Mode Advance 2 Set-up



Targeting the sysDSP Block Using by Inference

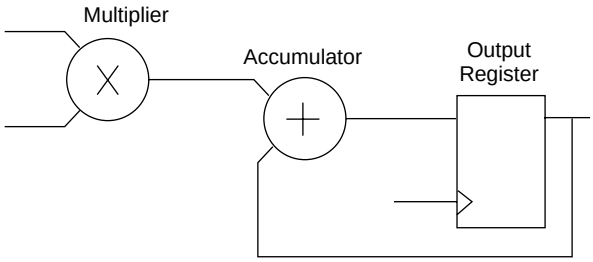
The Inferencing flow enables the design tools to infer sysDSP Blocks from a HDL design. It is important to note that when using the Inferencing flow; unless the code style matches the sysDSP Block results will not be optimal results. Consider the following example:

```
// 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
```

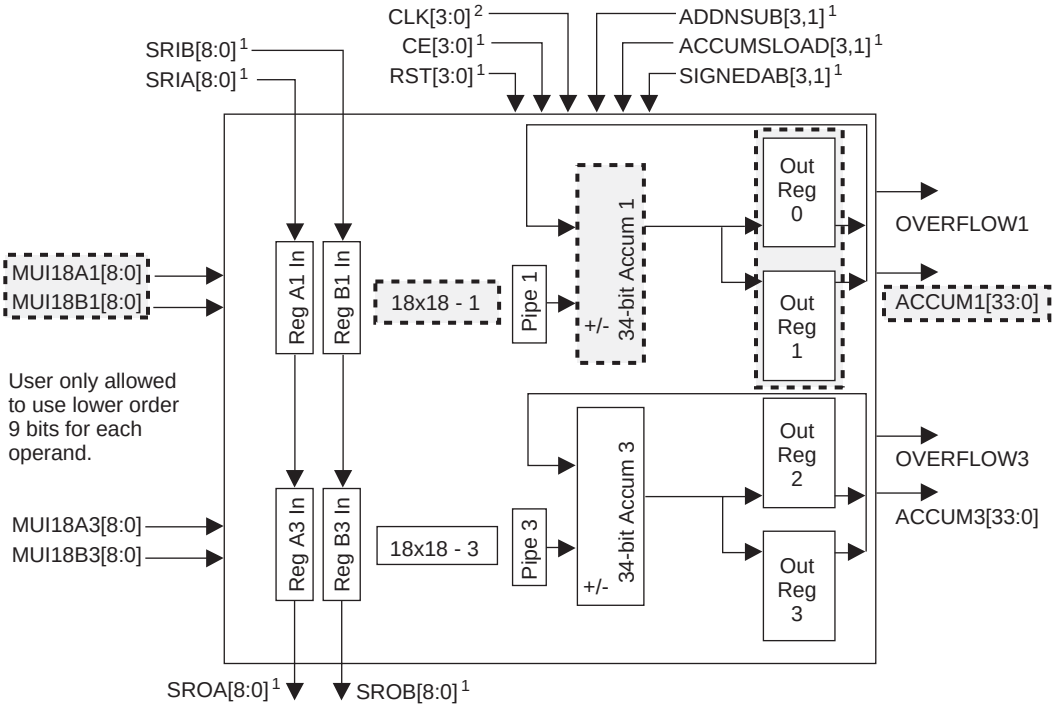
The above RTL will infer the following block diagram.

Figure 11-11. 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 11-12. 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 ispGENERATOR 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.

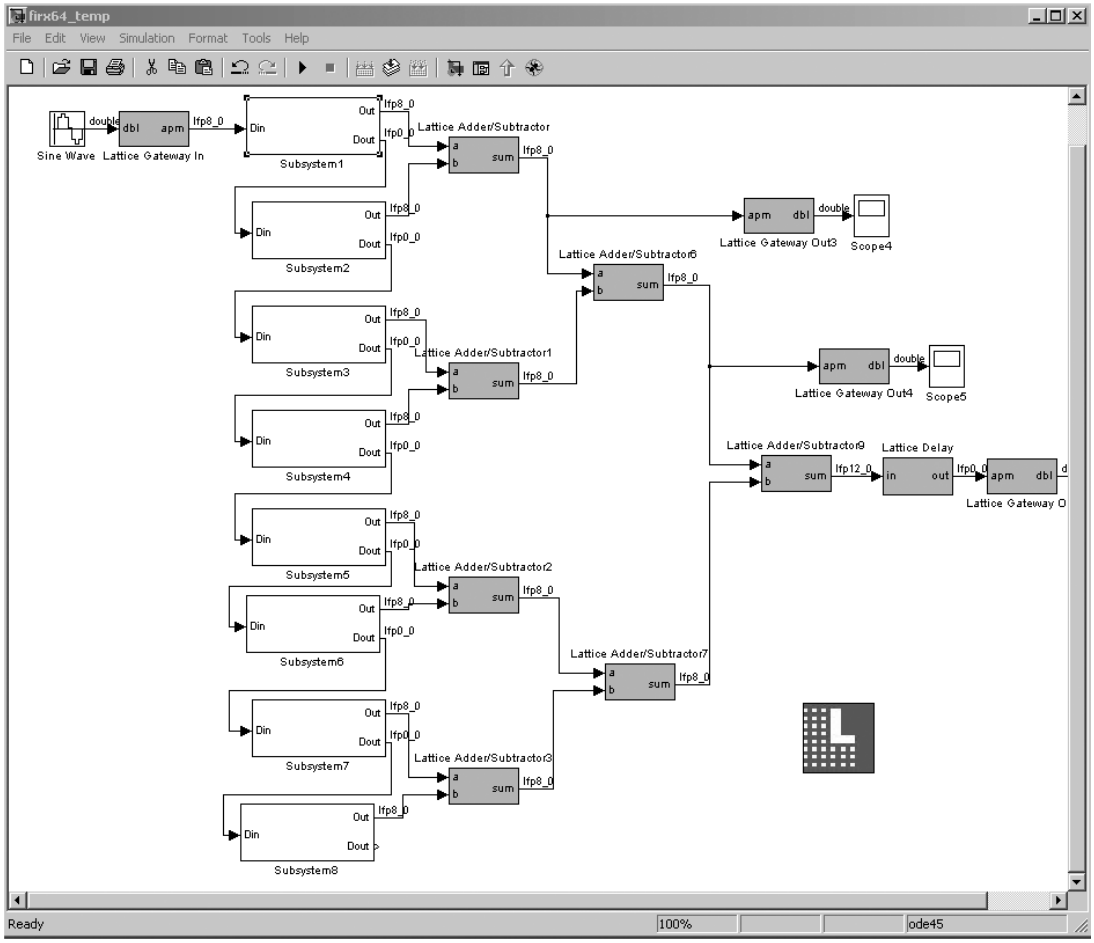
Gateways In and Out

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 11-13 is an example.

ispGENERATOR

The ispGENERATOR block is used to convert fixed point Simulink design into HDL files which can be instantiated in your HDL design. The ispGENERATOR Block is identified by the Lattice logo and can be seen in Figure 21.

Figure 11-13. 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)
+1-408-826-6002 (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,SRIA14,SRIA1
5,SRIA16,SRIA17,
SRIB0,SRIB1,SRIB2,SRIB3,SRIB4,SRIB5,SRIB6,SRIB7,SRIB8,SRIB9,SRIB10,SRIB11,SRIB12,SRIB13,SRIB14,SRIB1
5,SRIB16,SRIB17,
SROA0,SROA1,SROA2,SROA3,SROA4,SROA5,SROA6,SROA7,SROA8,SROA9,SROA10,SROA11,SROA12,SROA13,SROA14,SROA1
5,SROA16,SROA17,
SROB0,SROB1,SROB2,SROB3,SROB4,SROB5,SROB6,SROB7,SROB8,SROB9,SROB10,SROB11,SROB12,SROB13,SROB14,SROB1
5,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,ACCUMSLOAD,
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,ADDNSUB
3,
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,ACCUMSLOAD,
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,ADDNSUB3,
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”};
```

Estimating Power Using the Power Calculator for LatticeECP/EC Devices

June 2004 Technical Note TN1052

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 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 are listed below.

Total DC Power (Resource)

= Total DC Power of Used Portion + Total DC Power of Unused Portion

= [DC Leakage per resource when Used * $N_{RESOURCE}$]

+ [DC Leakage per resource when Unused * ($N_{TOTAL\ RESOURCE}$ - $N_{RESOURCE}$)]

Where,

$N_{TOTAL\ RESOURCE}$	is the total number of resources in a device
$N_{RESOURCE}$	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 Quiescent Power in the Power Calculator.

The AC power, on the other hand, is governed by the equation

Total AC Power (Resource)

$= K_{RESOURCE} * f_{MAX} * AF_{RESOURCE} * N_{RESOURCE}$

Where,

$N_{TOTAL\ RESOURCE}$	is the total number of Resources in a device,
$N_{RESOURCE}$	is the number of Resources used in the design,
$K_{RESOURCE}$	is the Power constant for the Resource. The units of this factor are mW/MHz
f_{MAX}	is the max frequency at which the Resource is running. Frequency is measured in terms of MHz
$AF_{RESOURCE}$	is the activity factor for the Resource group. Activity Factor is in terms of the 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)

= Total DC Power of Used Portion + Total DC Power of Unused Portion

= [DC Leakage per Slice when Used * N_{SLICE}]
+ [DC Leakage per Slice when Unused * ($N_{TOTAL\ SLICE}$ - N_{SLICE})]

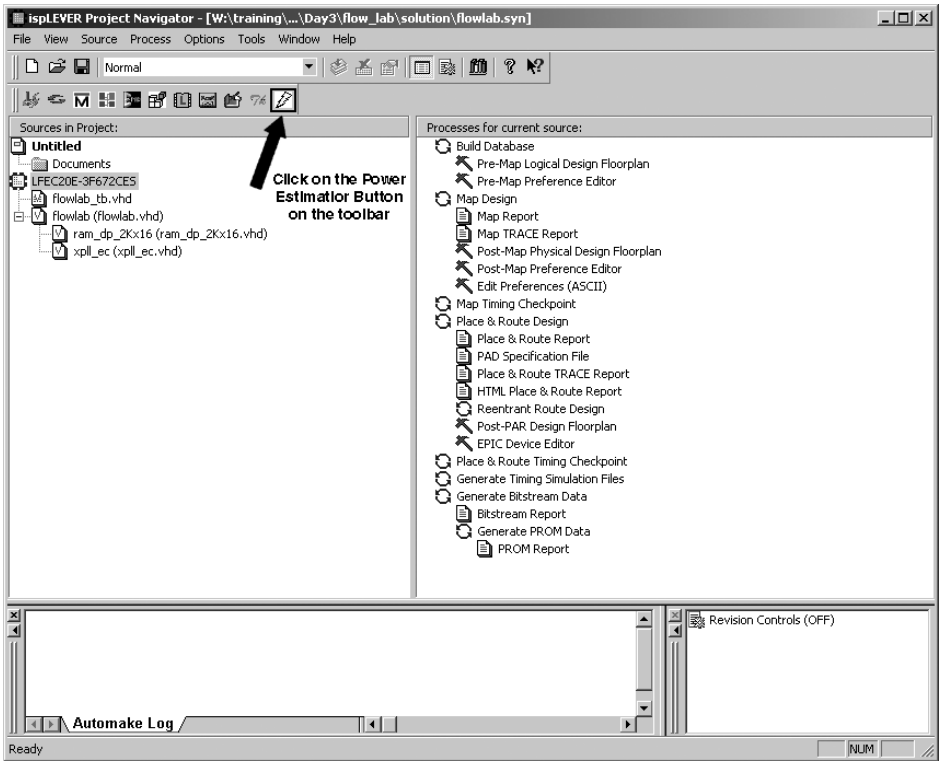
Total AC Power (Slice)

$= K_{SLICE} * f_{MAX} * AF_{SLICE} * N_{SLICE}$

Starting the Power Calculator

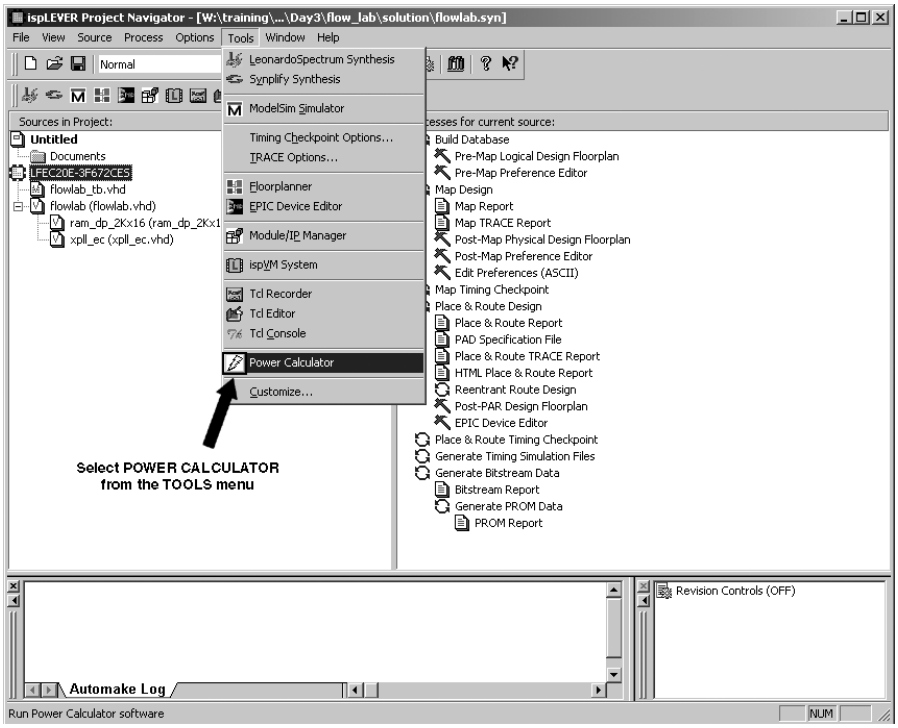
The user can launch the Power Calculator by one of the two methods. The first method is by clicking the Power Calculator button in the toolbar as shown in Figure 12-1.

Figure 12-1. Starting Power Calculator from Toolbar



Alternatively, users can launch 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

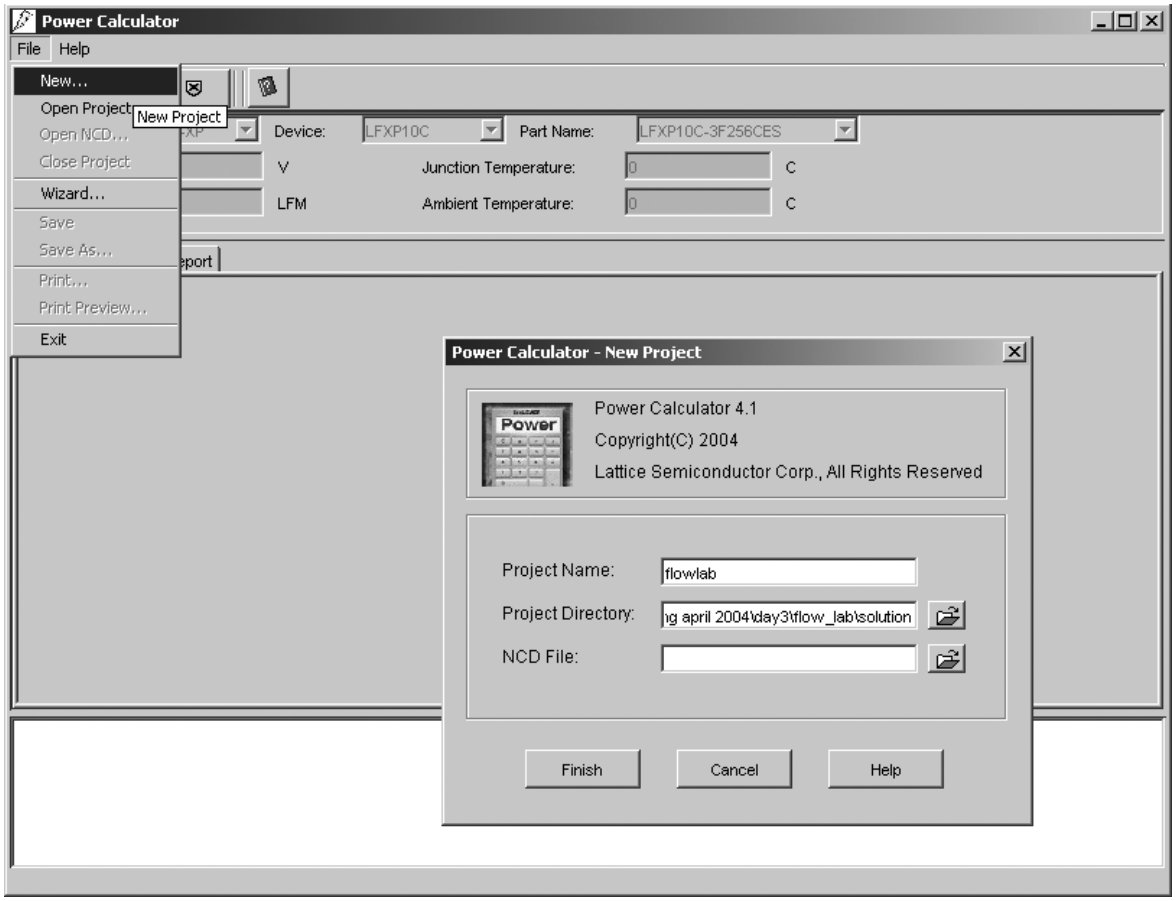


The Power Calculator does not support some of Lattice's older devices. The toolbar button and menu item is only present when supported devices are selected.

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)



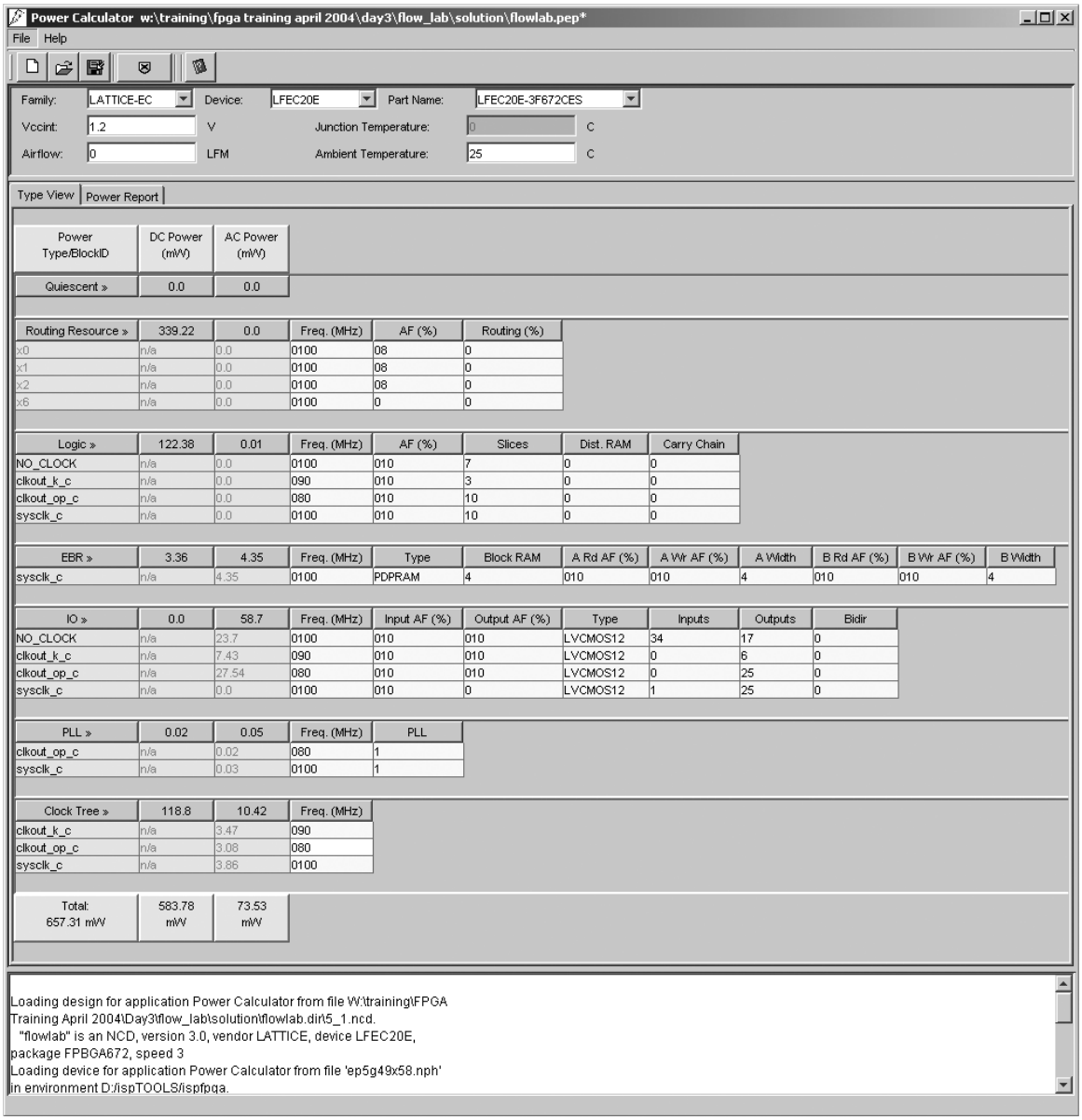
The Start Project window is used to create a new Power Calculator Project (*.pep project). Three pieces of data are input in the Start Project window.

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 of the window shows information about the device family, device and the part number as it appears in the Project Navigator.

Core Voltage, $V_{CC,}$ is labeled as V_{CCINT} in this part of the window. The power calculation depends upon the number of other voltage supplies in addition to the core voltage supply, $V_{CC.}$ V_{CCIO} voltage supplies for the I/Os also contribute to the final power consumption by the device. Other voltage supplies like V_{CCAUX} and V_{CCJ} also affect the final power consumption.

The present version of the tool uses the default values of these voltages supplies. For example, the LFEC6E-3 F484C device from the LatticeEC™ family, which is a 1.2V device, will use 1.2V as the V_{CC} to calculate power con-

sumption. For other Power Calculator assumptions, refer to the Power Calculator Assumptions section at the end of this document.

The top pane of this window also includes information about the Junction Temperature (°C), which is calculated as a function of the Ambient Temperature (°C), also included here. Air flow is also included.

Upon opening up the Power Calculator, the second and third columns, which are shaded blue in the tool, provide the DC (or Static) and AC (or dynamic) power consumption, respectively.

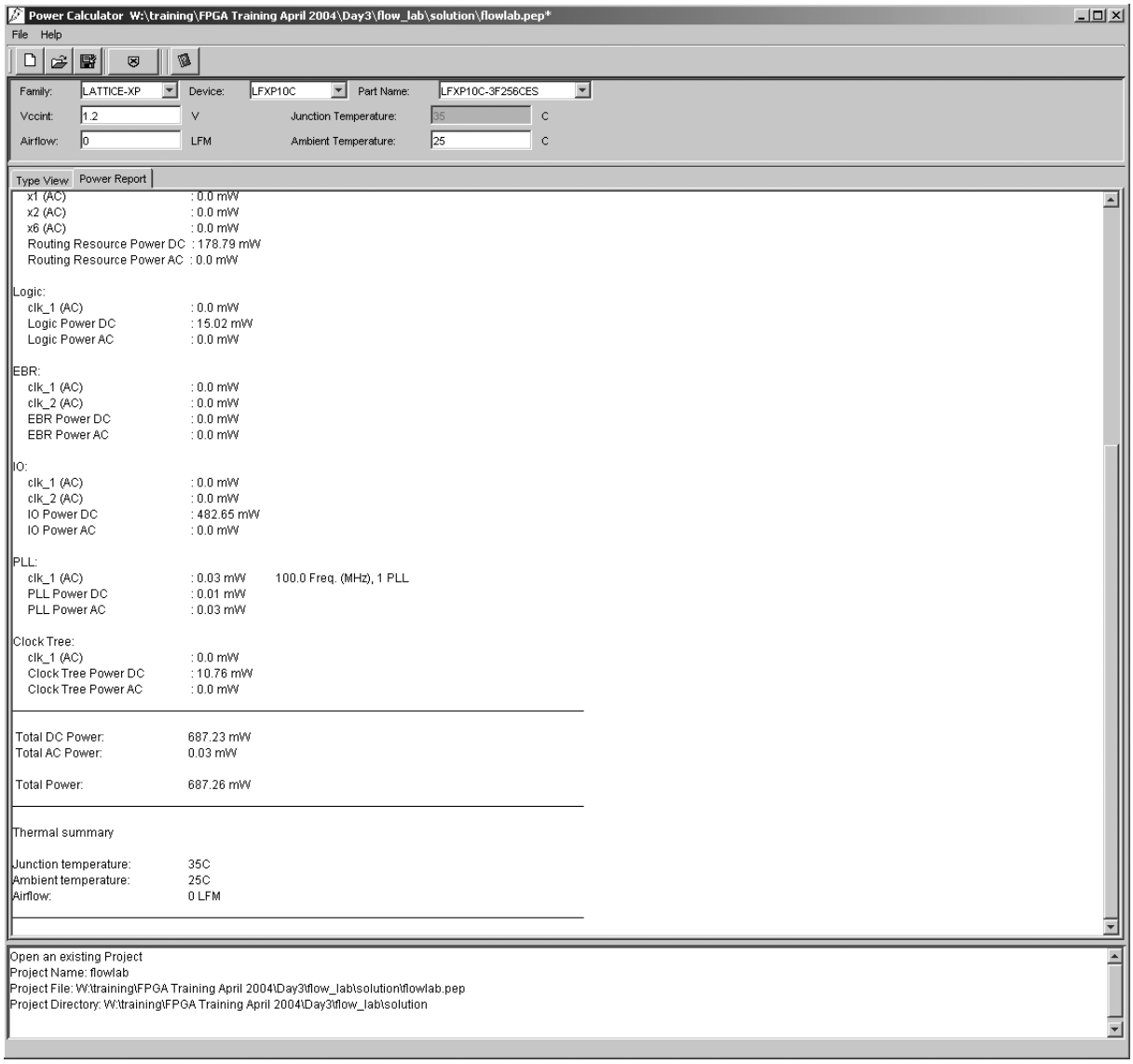
The first row shows the Quiescent Power, which is the DC power of a device with no resource utilization.

The columns to the right of these, which are shaded yellow in the tool, show factors like $f_{MAX,}$ resource utilization, and activity factors (AF%) that affect the static and dynamic power consumptions.

Using the tabs, power estimation is reported in two ways:

1. TYPE VIEW, tabular format as shown in Figure 12-4.
2. POWER REPORT, a text-based power estimation report, shown in Figure 12-5.

Figure 12-5. Power Calculator Main Window (Power Report View)

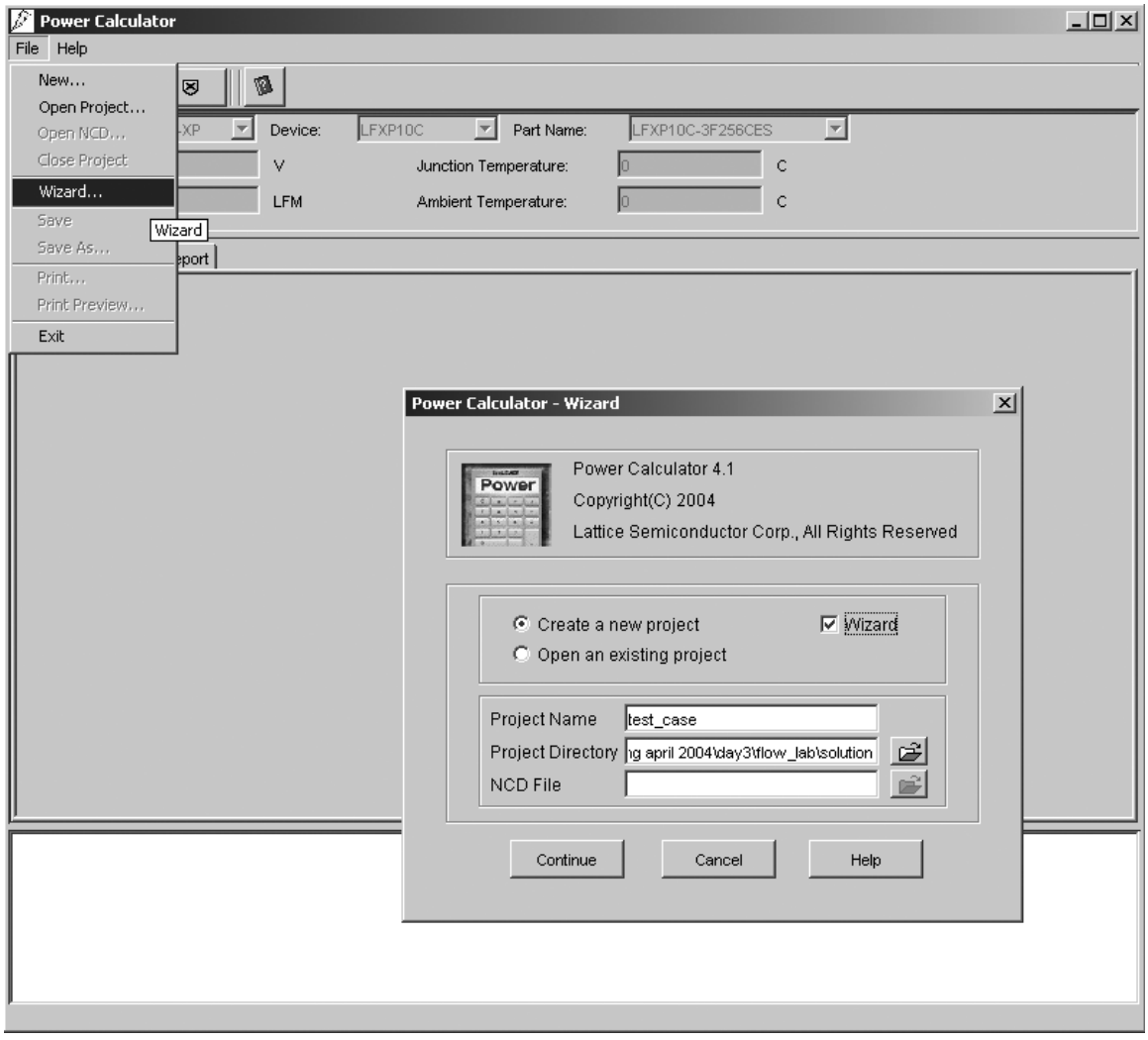


Power Calculator Wizard

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.

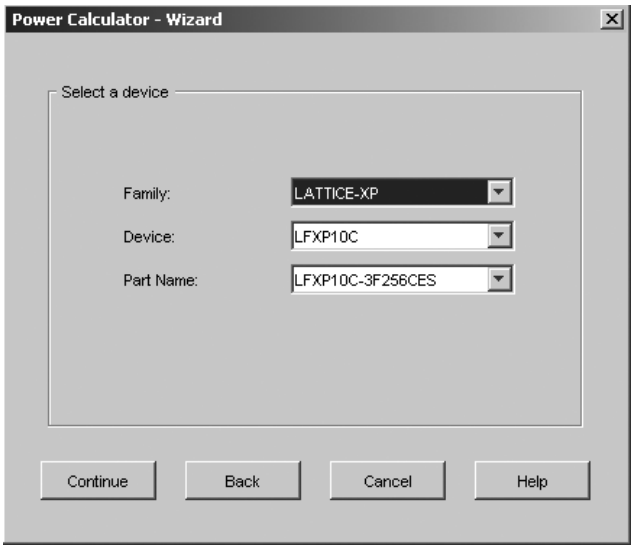
To start the Power Calculator in the wizard mode, go to the **File** menu and select **Wizard**. Alternatively, click on the **Wizard** button and get the **Power Calculator - Wizard** window, as shown in Figure 12-6. Select the option **Create a new Project** and check the **Wizard** check box in the 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 number 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

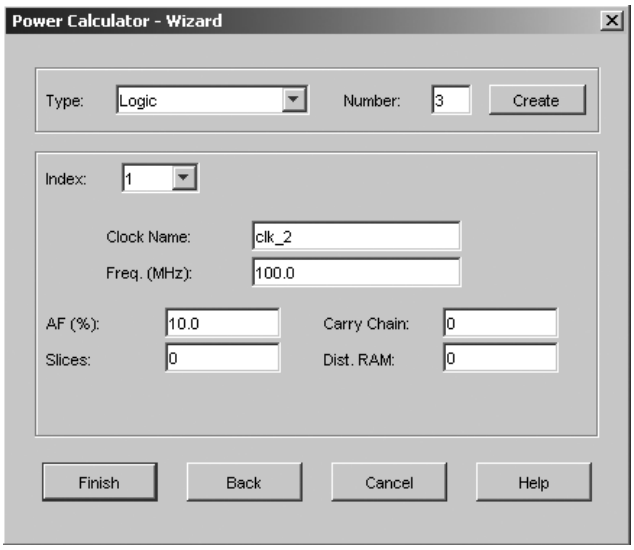


Figure 12-9. Power Calculator Wizard Mode Window - Resource Specification - EBR

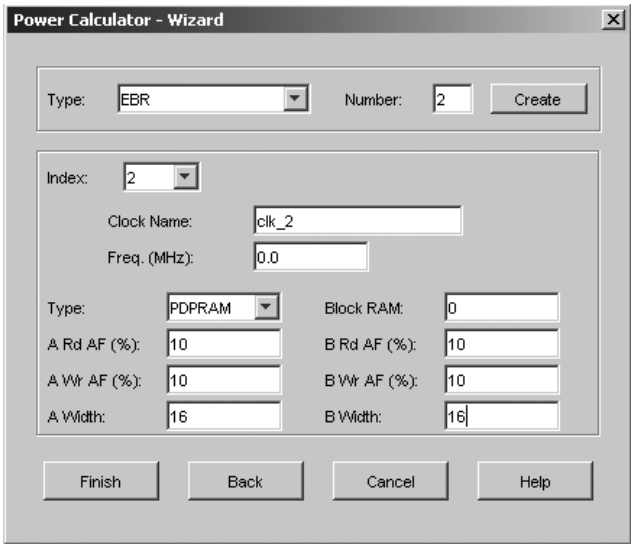


Figure 12-10. Power Calculator Wizard Mode Window - Resource Specification - PLL

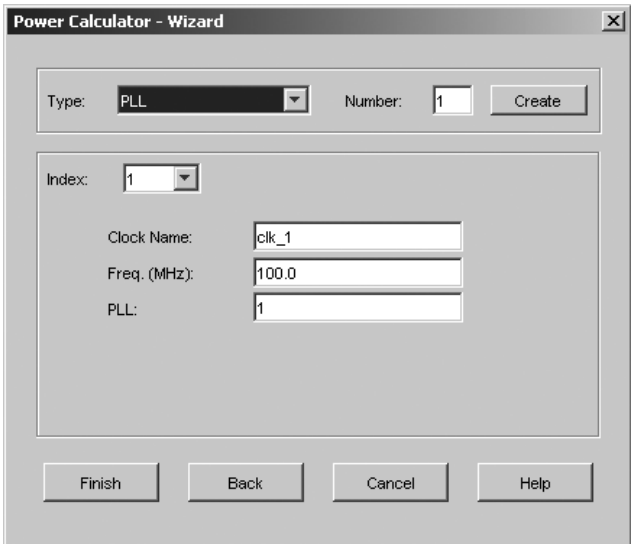


Figure 12-11. Power Calculator Wizard Mode Window - Resource Specification - Routing Resources

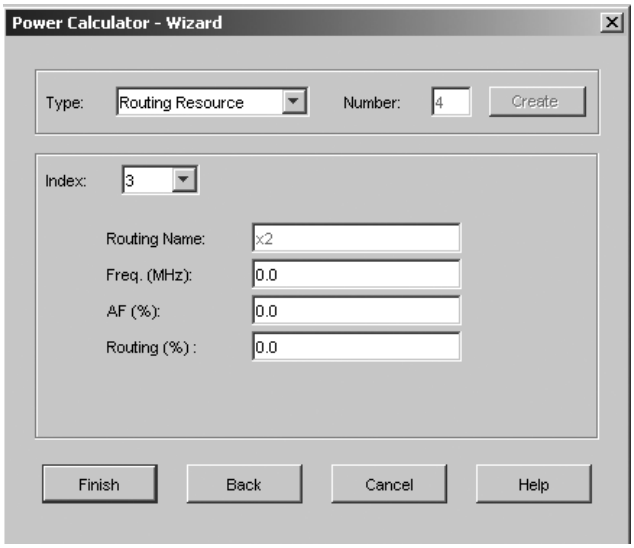


Figure 12-12. Power Calculator Wizard Mode Window - Resource Specification - I/Os

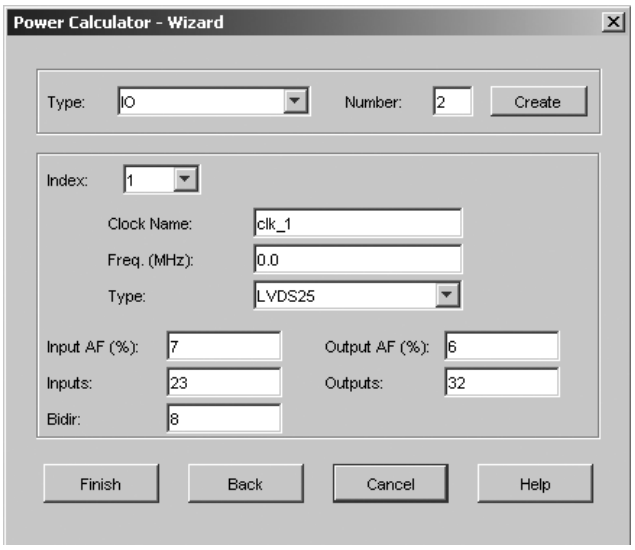
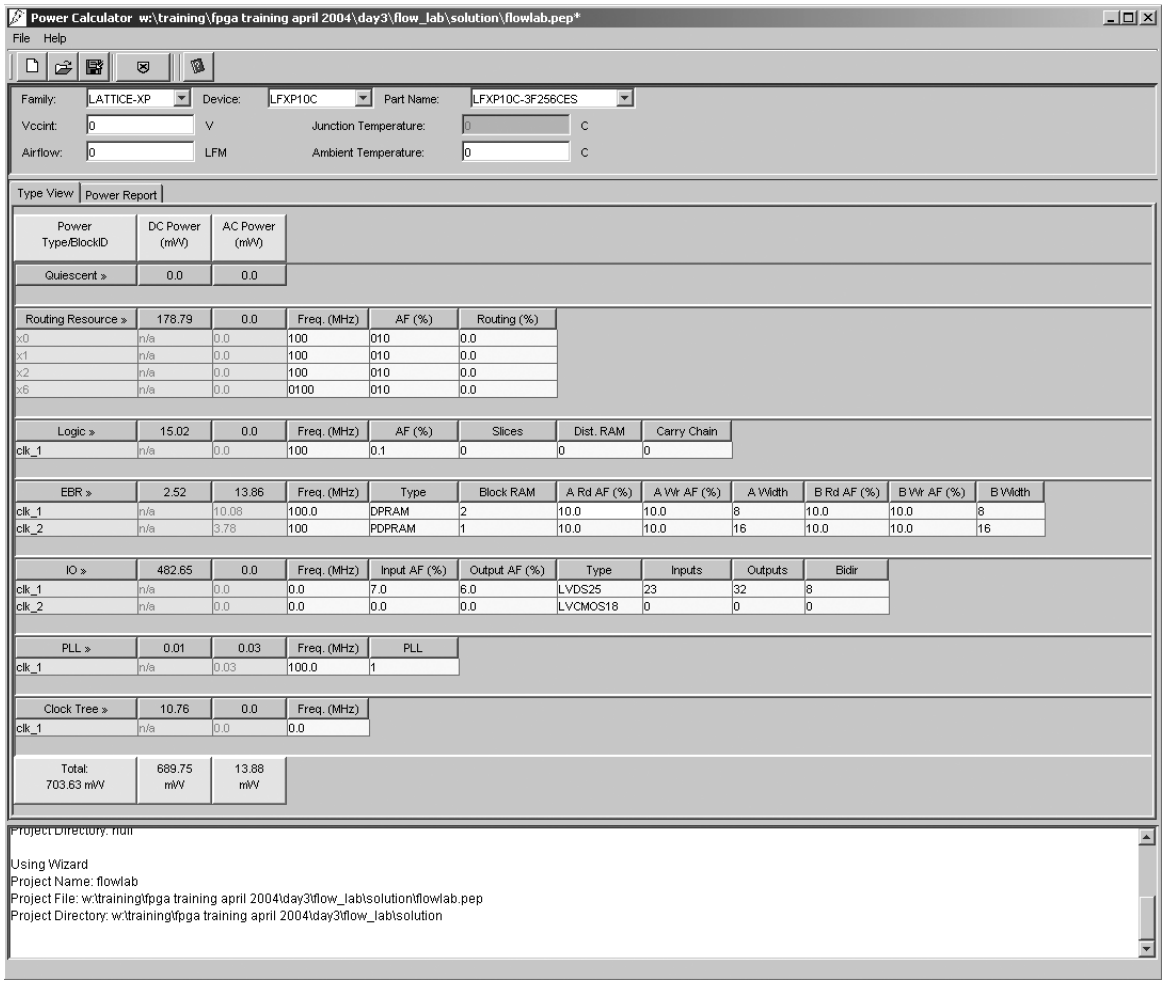


Figure 12-13. Power Calculator Wizard Mode - Main Window



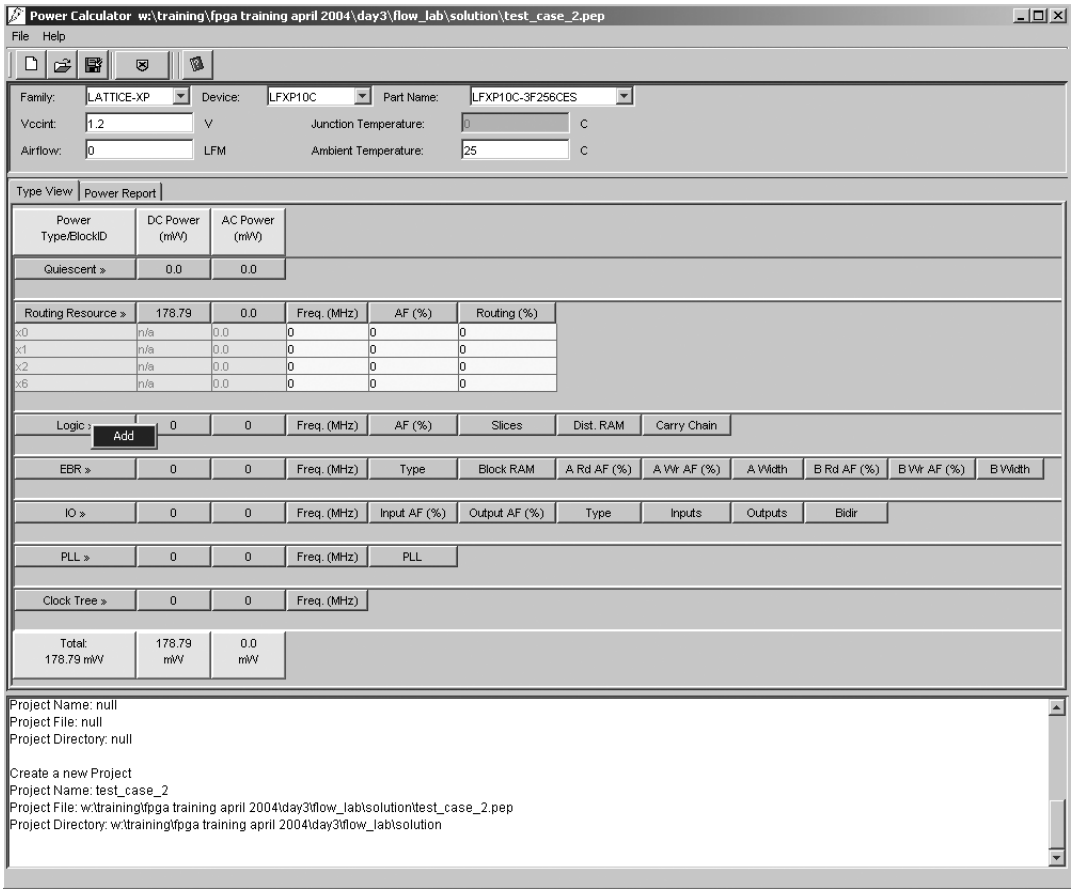
Power Calculator – Creating a New Project Without the NCD File

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



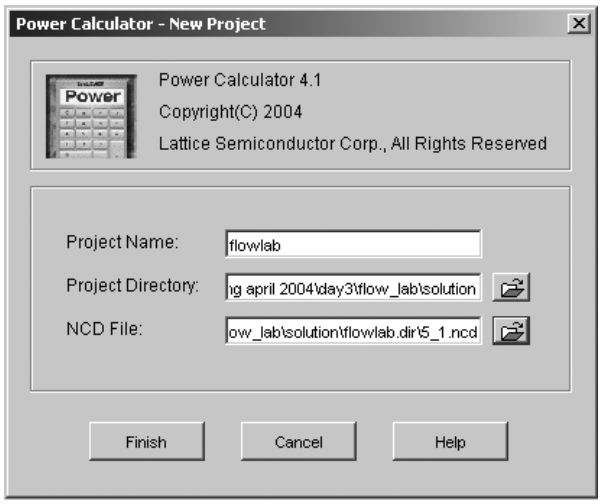
This would add a new row for the logic resource utilization with clock domain as clk_1.

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

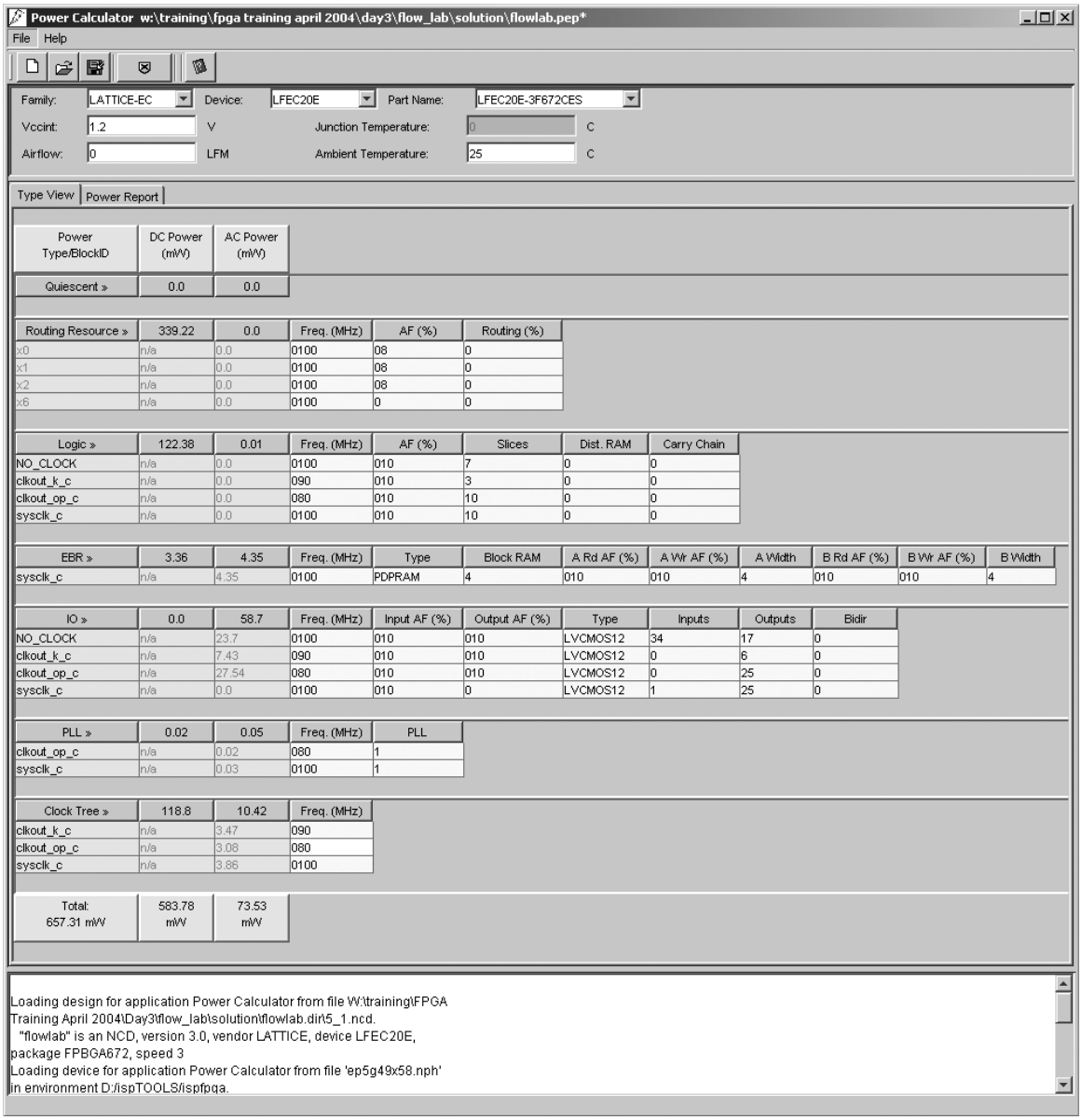
If the post place and routed NCD file is available, the Power Calculator can use it to import the accurate information about the design data and resource utilization and calculate the power. When the Power Calculator is started, the NCD file is automatically placed in the NCD File option, if available 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 Power Calculator uses the Clock names from the design, as shown in Figure 12-16.

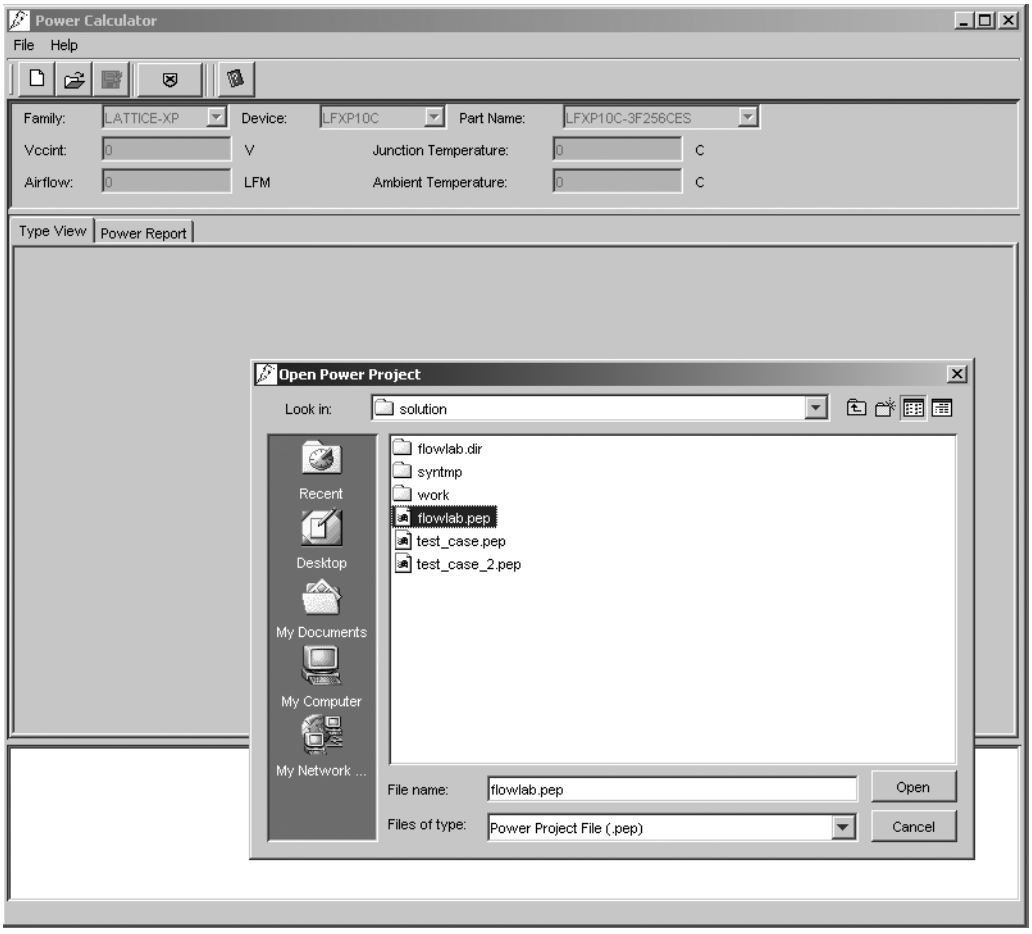
Figure 12-16. Power Calculator Main Window – Resource Utilization Picked Up From the NCD File



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

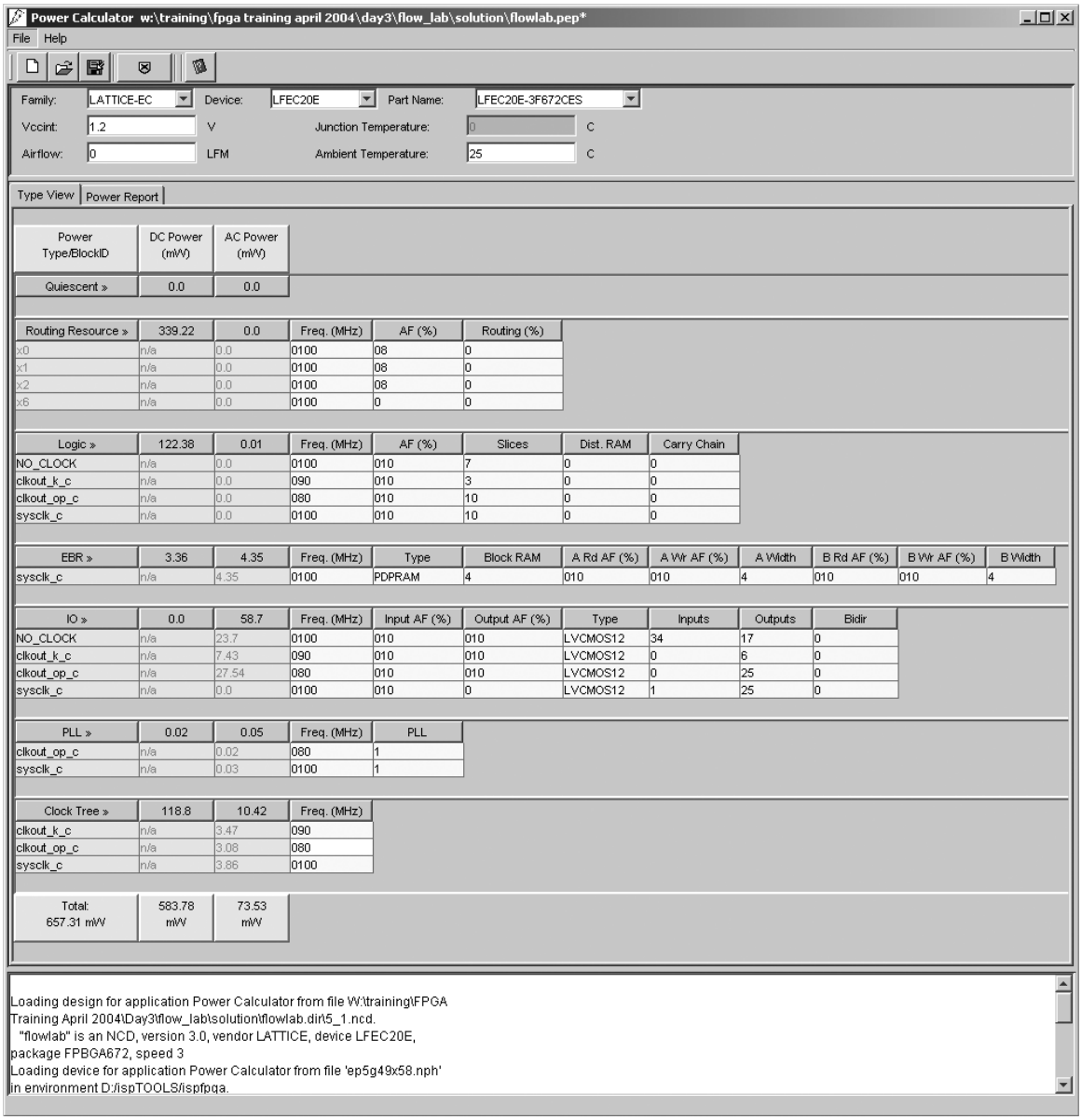


Power Calculator – Total Power

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



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 Calculation

The Activity Factor relates to how often a signal transitions. Each transition of a signal adds to the current consumption of a device. The average of all the signal Activity Factors will give you the device AF. Take for example a 16-bit counter. The LSB node of a 16-bit counter is switching every clock rising edge and its Activity Factor is 1. The

second node is switching every other clock rising edge that corresponds to an Activity Factor of 1/2. The third node is switching every fourth clock rising edge for an Activity Factor of 1/4. The series for the sequence of 16 nodes is 1 + 1/2 + 1/4 + 1/8 + 1/16 + ... + 1/215 converges to 2. Assuming a sum of 2 for the counter, divide by the total number of nodes (16) to measure the average AF per counter of 2/16 = 0.125.

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.

$$T_J = T_A + \Theta_{JA} * P \quad (1)$$
$$T_J = T_C + \Theta_{JC} * P \quad (2)$$
$$T_J = T_B + \Theta_{JB} * P \quad (3)$$

Where T_J, T_A, T_C and T_B are the junction, ambient, case (or package) and board temperatures (in °C) respectively. P is the total power dissipation of the device.

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.

Power Calculator utilizes the Ambient Temperature (°C) to calculate the junction temperature (°C) based on the Θ_{JA} for the targeted device, per Equation 1 above. Users can also provide the airflow values (in LFM) to get a more accurate value of the junction temperature.

Managing Power Consumption

There are several design techniques that FPGA designers can use to reduce overall FPGA power consumption or reduce its impact on junction temperature. Some of these are:

1. **Reducing operating voltage.** For example, operating at the low end of the voltage range allowed, given the nominal voltage chosen.
2. **Optimizing the clock frequency.** Often clock rate can be reduced, for all or some portions of the design.
3. **Reducing the span of the design across the device.** A more closely placed design utilizes less routing resources and hence less power consumption.
4. **Reducing the voltage swing of the I/Os where possible.** A double-ended LVDS has much less voltage swing as compared to single-ended CMOS.
5. **Using optimum encoding where possible.** For example, a 16-bit binary counter has, on average, only a 12% Activity Factor while a 7-bit binary counter has an average of 28% Activity Factor. On the other hand, a 7-bit LFSR counter will toggle at an Activity factor of 50%, which causes higher power consumption. A gray code counter, where only one bit changes at each clock edge, will use the least amount of power, with an Activity Factor of less than 10%.

Techniques for minimizing the operating junction temperature include:

1. Choosing packages with a higher $\Theta_{J.}$
2. Placing heat sinks and thermal planes around the device on the PCB.
3. Better airflow techniques using mechanical airflow guides and fans (both system fans and device-mounted fans).

Power Calculator Assumptions

The Power Calculator tool is based on the typical numbers. These are preliminary numbers and they are for estimation purposes only. The actual power consumption depends upon a number of factors.

The Power Calculator is used for estimating the power consumption on the core voltage supply, V_{CC} and $V_{CCIO.}$ Other voltage supplies, V_{CCAUX} and $V_{CCJ,}$ represent a much lower portion of total power. The present version of does not take into account these power supplies.

Also, the present version of the tool uses the default values of the core voltages, V_{CC}; for example, the LFEC6E-3 F484C device from the LatticeEC family, which is a 1.2V device, will use 1.2V as the V_{CC} to calculate the power consumption. Future versions of this tool will allow users to specify all these voltages for a more accurate power consumption number.

Similarly, the tool uses the default values for the I/O supplies. For example, the LVCMOS 1.2V would use the 1.2V for calculation. Future versions will allow users to enter a voltage number of their choice.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-408-826-6002 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

LatticeECP/EC sysCONFIG Usage Guide

June 2004 Technical Note TN1053

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.

- SPI3
- SPIX
- Master Serial
- Slave Serial
- Master Parallel
- Slave Parallel
- ispJTAG (1149.1 Interface)

This technical note will cover 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 dual-purpose pins are available as extra I/O pins if they are not used for configuration. A programmable option controls the dual-purpose configuration pins. This option is made 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 13-1 is provided for reference.

Table 13-1. 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	See Note 2	SERIAL/SPI
DOUT/CSON	Output	See Note 2	SERIAL/PARALLEL
CSN	Input	See Note 2	PARALLEL
CS1N	Input	See Note 2	PARALLEL
WRITEN	Input	See Note 2	PARALLEL
BUSY	Output	See Note 2	PARALLEL/SPI
D[0:7]	Input or output	See Note 2	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.
2. If used for configuration, this pin is not available as an I/O.

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 Power-On-Reset reset signal is released during Power-up, the INITN pin will be driven low to reset the configuration circuitry and the 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.

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 100 to 500 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 PROGRAMN pin being toggled. The MCCLK_FREQ Parameter controls the CCLK Master frequency. See the Master Clock Selection section of this document for more information.

Dual-Purpose sysCONFIG Pins

The following is a list of dual-purpose sysCONFIG pins. If any of these pins are used for configuration they will not be available as I/O after configuration. After configuration these pins are tristated and weakly pulled up.

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 supports an internal weak pull up. When a serial mode is selected, the DI pin can become an I/O when not used in a configuration mode.

In either SPI3 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]
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] 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 D[0], SPI memory one output will be wired to D[1], the data fed to these pins will be interleaved and then sent to the internal configuration engine. For SPIX Mode, the D[0:7] 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 D[0] will be shifted out first followed by D[1], D[2], 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], INITN 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 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.latticesemi.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 13-2 lists the Mode, CFG[0:2] state and the software CONFIG_MODE parameters. The following subsections break down each configuration mode individually.

Table 13-2. Configuration Modes for the LatticeECP/EC Devices

Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
SPI3	0	0	0	SPI3
SPIX	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 (Default)
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. When using a master clock, the master clock frequency can be set within a range as determined by the device. 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 for a slave serial device. When the bypass 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 13-2.

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. D7 of the byte wide data will be shifted out first and followed by D6, D5, 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 13-2. 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], INITN 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 CS0 and CS1, 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. The CFG[0] pin will determine whether the Master Clock is used. See the device-specific section of the CFG[0:2] descriptions.

The user can determine 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. See the *LatticeECP/EC FPGA Family Data Sheet* for MCLK_FREQ selections.

SPI3 Mode

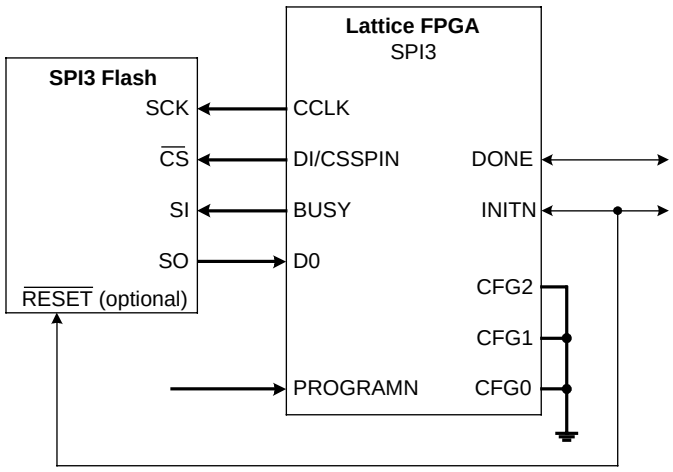
Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
SPI3	0	0	0	SPI3

The LatticeECP/EC devices offer a direct connection for memories that support the SPI3 standard. By setting the configuration pins CFG[0:2] = b'000, the LatticeECP/EC devices will configure using the SPI3 interface. The SPI3 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. The circuitry to support the any number of SPI Flash is not available to serialize out DOUT.

One FPGA, One SPI Flash
The simple SPI application is one SPI Flash serial connected to the D0 of the LatticeECP/EC devices in SPI mode, as shown in Figure 13-1.

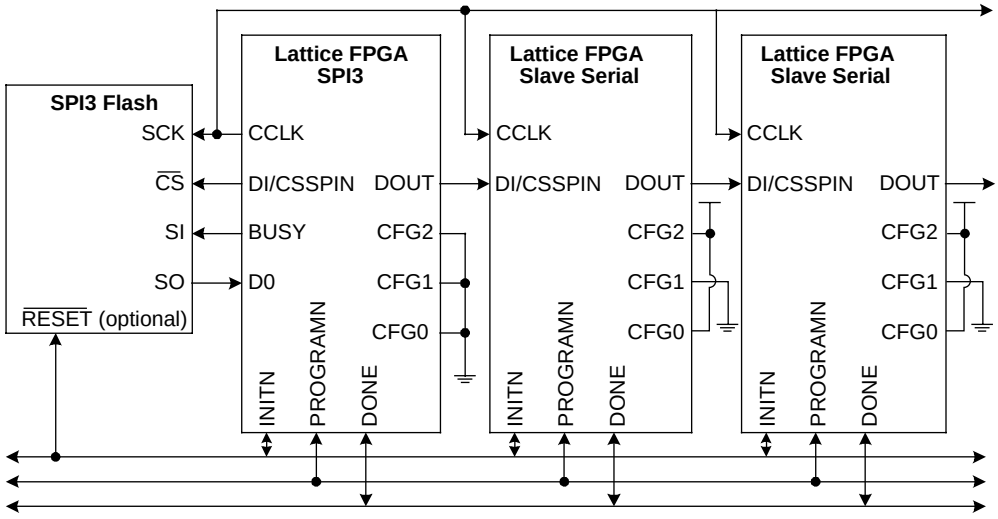
Figure 13-1. Simple Interface for FPGA Bootup in SPI3 Mode



Multiple FPGA, One SPI Flash

With a sufficiently large SPI Flash, multiple FPGAs can be configured as shown in Figure 13-2. The first FPGA is configured in SPI Mode, the following FPGAs are configured in Slave Serial Mode.

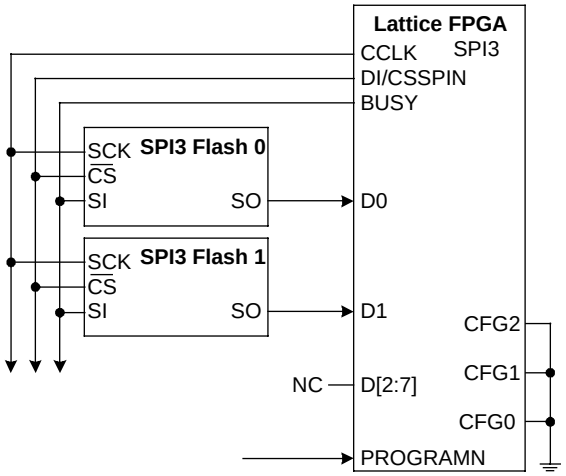
Figure 13-2. Multiple FPGA Configured by One SPI Flash



One FPGA, Two SPI Flash

The LatticeECP/EC devices support two Flash to configure a single device as shown in Figure 13-3. 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 D0 and D1 in a parallel fashion and are reassembled internally.

Figure 13-3. Two SPI Flash



SPIX Mode

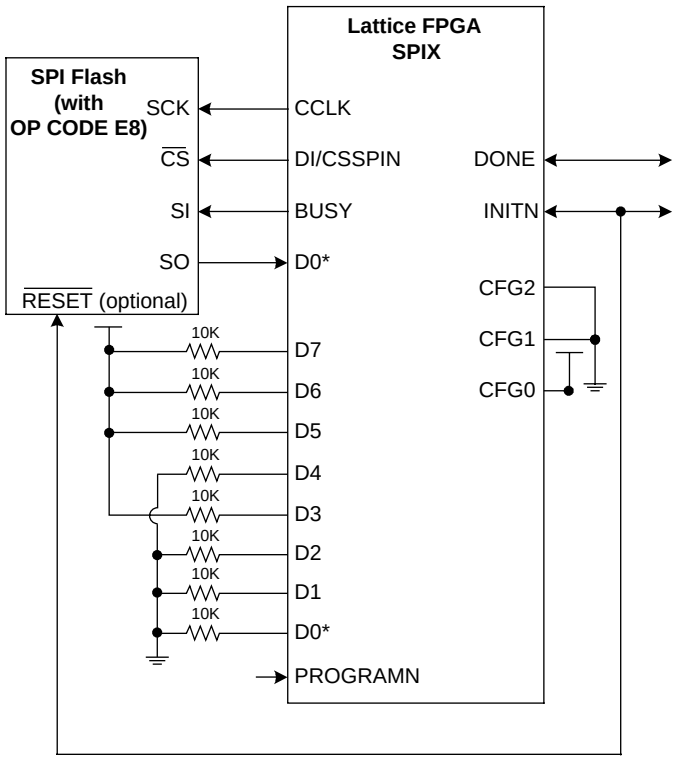
Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
SPIX	0	0	1	SPIX

Not all SPI memories are the same. A read operation code is required to be fed to the SPI Flash at the time configuration 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 D[0:7]

pins through the use of pull-ups and pull-downs as shown in Figure 13-4. When configuration begins in the SPIX mode the D[0:7] 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 SPI3 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 D[0:7] as shown in Figure 13-4.

Figure 13-4. Simple SPIX Example with OP CODE Resistors



*D0 connects to SO and resistor.

Master Serial Mode

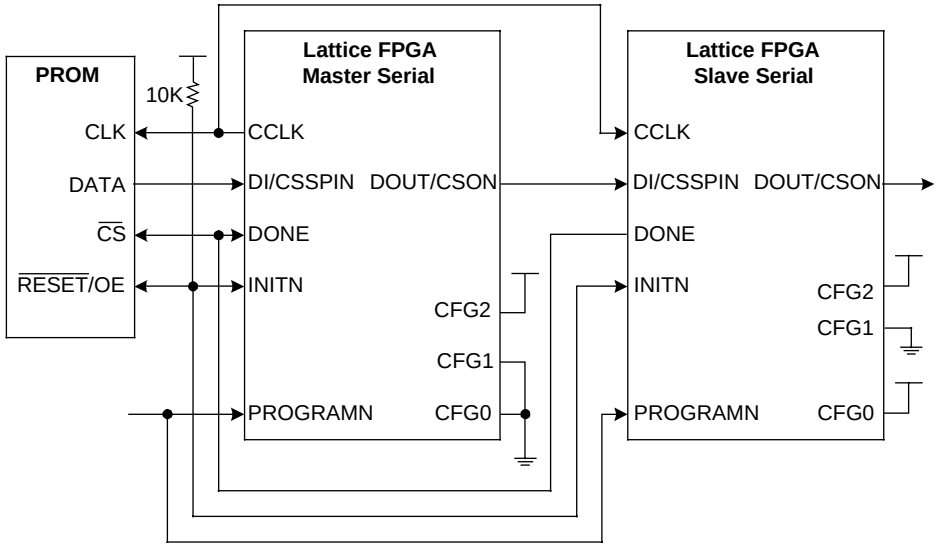
Mode	CFG[2]	CFG[1]	CFG[0]	CONFIG_MODE Parameter
Master Serial (no overflow option)	1	0	0	MASTER_SERIAL
Master Serial (Bypass ON)	1	0	0	MASTER_SERIAL_BYPASS

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 13-5 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 13-5. 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 13-5.

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

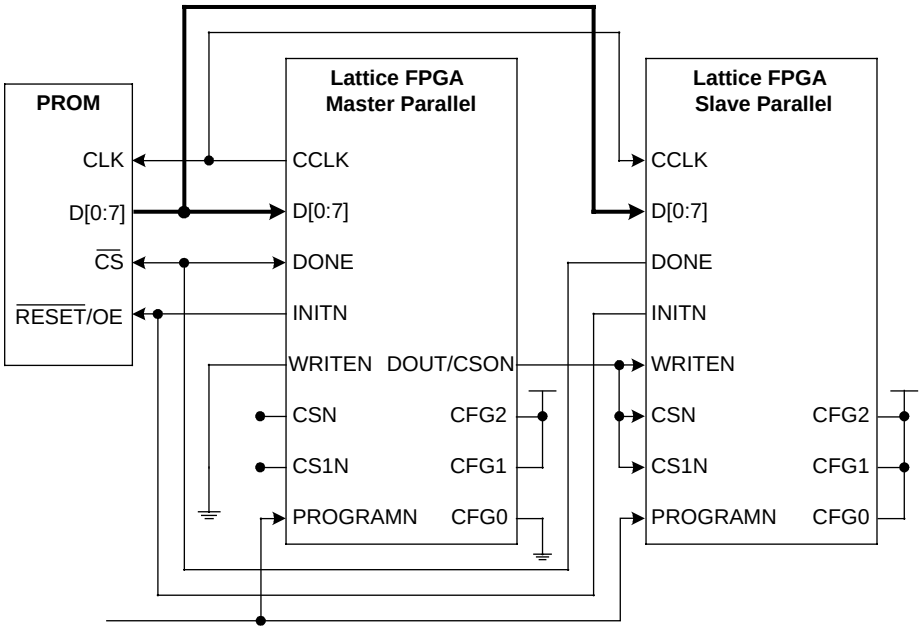
Configuration using Master Parallel Mode is used to work together with a parallel port PROM without additional external logic. When Master Parallel Mode is chosen, the device will generate CCLK as specified by the MCLK_FREQ preference. The CCLK signal is used to provide a programming clock to the PROM 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.

Master Parallel Mode can also be used for read back of the internal configuration. By driving the WRITEN pin high, the device will “listen” for the read back instructions on the D[0:7] pins. In order to support read back, the PERSISTENCE Preference must be set in the Lattice design software. See the Persistence section of this technical note for more information on the PERSISTENCE preference.

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 13-6.

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 13-6. 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 13-6.

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 13-7. Asynchronous Usage of Slave Parallel Configuration Mode

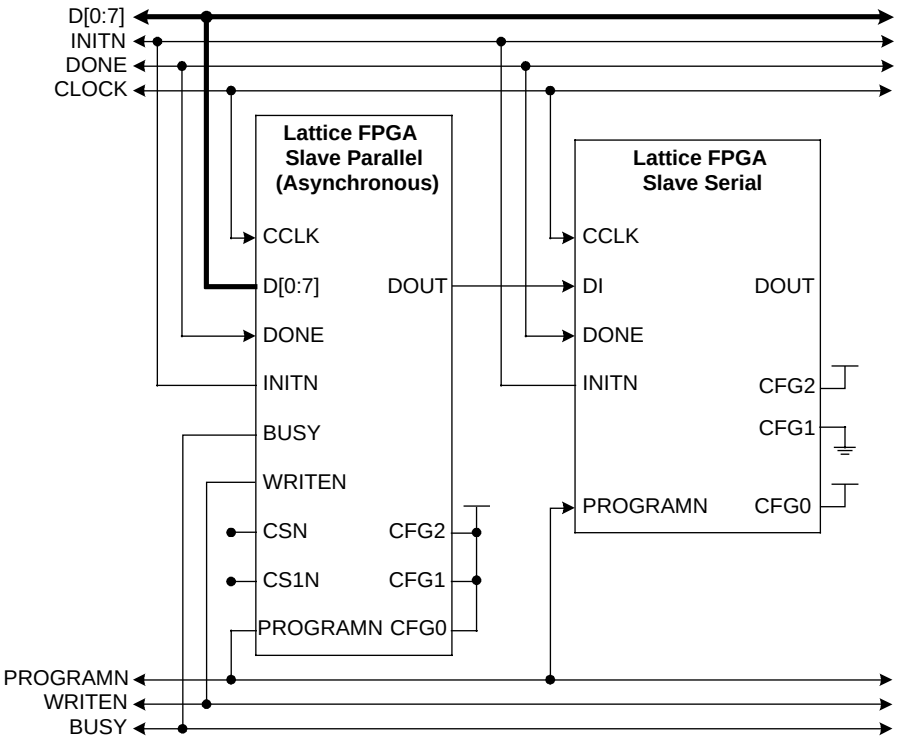


Figure 13-7 shows the Asynchronous peripheral write sequence using the Bypass option. To send configuration data to a device, the WRITEN signal has to be asserted. During the write cycle, the BUSY signal provides handshaking between the host system and the LatticeECP/EC device. When the BUSY signal is 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.

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, 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 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 PROGAMN pin low, the INITN and DONE pins will drive low and the memory will start clearing. 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, the master clock will start to be driven out CCLK. 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 and CS1N pins can be driven low 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, then 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 should 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 13-3 provides a list of the wake-up sequences supported by the devices.

Table 13-3. 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 then 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, then 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 shall select the clock source to wake up to. The clock sources are CCLK, TCK and User Clock. The default shall be either TCK or CCLK depending on the programming/configuration method. The default clock should be TCK if using ispJTAG and CCLK if using sysCONFIG. 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. Some sources use BCLK to represent the user clock. The WAKEUP_CLK shall default to CCLK or TCK.

Wake On PLL Lock
If selected, the LatticeECP/EC device will wait for a lock signal from the PLL before the wake-up sequence begins. The Wake On lock option must be set by the Wake_on_lock preference.

Read Back

Read back of the configuration memory through sysCONFIG can be done in two different ways. One is transparent and the device remains alive and functioning. The other shuts the device down and reads the configuration memory 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 pin is 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 PERSISTENCE preference to ON. CCLK becomes input only.

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 13-4 is a list of the preference, the default settings and the section more information about the preference can be found.

Table 13-4. LatticeECP/EC Device Preference List

Preference (Preference)	Default Setting (All Settings)
PERSISTENT	ON [off, on]
CONFIG_MODE	SLAVE_SERIAL (see Table 13-2)
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]
WAKE_ON_LOCK	OFF [off, on]
WAKEUP_CLK	EXTERNAL (external, user)
COMPRESS_CONFIG	OFF (off, on)

Persistence

When using the sysCONFIG port, the PERSISTENCE preference must be set to ON to reserve the dual-purpose pins for configuration. The PERSISTENCE = ON will let the software know that all the dual purpose configuration pins will NOT be available for the fitter to use.

Configuration Mode

The device knows what physical port will be used by the setting of the CFG[0:2]. But there are several options that are set by the software for configuration such as if an overflow option will be used (Bypass or Flow Through). The fitter will also need to know what pins will be available based on the selection of the Configuration mode. The software fitter cannot sample the configuration pins, so the user must select the Configuration Mode.

The CONFIG_MODE supported by LatticeECP/EC is a combination of the hardware CFG pins and CONFIG_MODE Selection. The overflow option is either Flow Through or Bypass. The overflow options default to OFF. If either overflow option is selected, then the DONE_EX and WAKE_UP selections will be set to correspond to the new options. See Table 13-5 which details Overflow Option defaults. For more information on the over flow options, see the Configuration Options section of this document.

Table 13-5. 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 13-6 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 13-6 for more information on the relationship between DONE_OD and DONE_EX.

Table 13-6. 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 security for the device is selected, NO read back operation will be supported through the sysCONFIG port or ispJTAG port of the general contents. The USERCODE area is readable and not considered securable. The CONFIG_SECURE preference will allow the user to set the security of the device. Default of the security preference is OFF. OFF indicates Read Back is enabled through any port. On will block Read Back of the configuration memory.

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 13-3. 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 13-3. 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.

Wake On Lock

The LatticeECP/EC devices support several PLLs. The WAKE_ON_LOCK preference can be set for the device to delay wake-up until the selected PLLs have phase lock. The WAKE_ON_LOCK option can be set by the PLL interface. See the PLL documentation for more information.

Wake-up Clock Selection

The wake-up sequence is synchronized to a clock source. The user selects the clock source to wake up to. The clock sources are either External (CCLK or TCK depending on if using sysCONFIG or ispJTAG) or User Clock. The Default shall be EXTERNAL, implying TCK or CCLK depending on the programming/configuration method in use. The User Clock is chosen at the time of design. The user can use any of the CLK pins of the device or a net (routing node) or the internal clock source as the User Clock source. The WAKEUP_CLK preference defaults to EXTERNAL.

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 13-7. LatticeECP/EC Configuration Memory Requirements

Family	Device	Max. Config. Bits (M)	Required Boot Memory (M)	
			w/o Comp	with Comp
LatticeECP/EC	1.5	0.6	1	512K
	3	1.1	2	1
	6	1.8	2	2
	10	3.1	4	4
	15	4.3	8	4
	20	5.3	8	4
	40	9.7	16	8

SPI3 Compatible SPI Flash Vendors

- STMicroelecctronics – M25P Serial FLASH Family
- NexFLASH – NX25P spiFLASH Family

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-408-826-6002 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com