LatticeXP Family Handbook

HB1001 Version 02.9, April 2007

LatticeXP Family Handbook Table of Contents

April 2007

Section I. LatticeXP Family Data Sheet

Section I. LatticeXP Family Data Sheet

DS1001 Version 04.9, February 2007

LatticeXP Family Data Sheet Introduction

December 2005 Data Sheet DS1001

Features

- **Non-volatile, Infinitely Reconfigurable**
  - Instant-on – powers up in microseconds
  - No external configuration memory
  - Excellent design security, no bit stream to intercept
  - Reconfigure SRAM based logic in milliseconds
  - SRAM and non-volatile memory programmable through system configuration and JTAG ports
- **Sleep Mode**
  - Allows up to 1000x static current reduction
- **TransFR™ Reconfiguration (TFR)**
  - In-field logic update while system operates
- **Extensive Density and Package Options**
  - 3.1K to 19.7K LUT4s
  - 62 to 340 I/Os
  - Density migration supported
- **Embedded and Distributed Memory**
  - 54 Kbits to 396 Kbits sysMEM™ Embedded Block RAM
  - Up to 79 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 18 Class I
    - SSTL 3/2 Class I, II
    - HSTL15 Class I, III
    - HSTL 18 Class I, II, III
    - PCI
    - LVDS, Bus-LVDS, LVPECL, RSDS
- **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
  - Onboard oscillator for configuration
  - Devices operate with 3.3V, 2.5V, 1.8V or 1.2V power supply

Table 1-1. LatticeXP Family Selection Guide

Device	LFXP3	LFXP6	LFXP10	LFXP15	LFXP20
PFU/PFF Rows	16	24	32	40	44
PFU/PFF Columns	24	30	38	48	56
PFU/PFF (Total)	384	720	1216	1932	2464
LUTs (K)	3.1	5.8	9.7	15.4	19.7
Distributed RAM (KBits)	12	23	39	61	79
EBR SRAM (KBits)	54	72	216	324	396
EBR SRAM Blocks	6	8	24	36	44
V_{CC} Voltage	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V	1.2/1.8/2.5/3.3V
PLLs	2	2	4	4	4
Max. I/O	136	188	244	300	340
Packages and I/O Combinations:					
100-pin TQFP (14 x 14 mm)	62				
144-pin TQFP (20 x 20 mm)	100	100			
208-pin PQFP (28 x 28 mm)	136	142			
256-ball fpBGA (17 x 17 mm)		188	188	188	188
388-ball fpBGA (23 x 23 mm)			244	268	268
484-ball fpBGA (23 x 23 mm)				300	340

Introduction

The LatticeXP family of FPGA devices combine logic gates, embedded memory and high performance I/Os in a single architecture that is both non-volatile and infinitely reconfigurable to support cost-effective system designs.

The re-programmable non-volatile technology used in the LatticeXP family is the next generation ispXP™ technology. With this technology, expensive external configuration memories are not required and designs are secured from unauthorized read-back. In addition, instant-on capability allows for easy interfacing in many applications.

The ispLEVER® design tool from Lattice allows large complex designs to be efficiently implemented using the LatticeXP family of FPGA devices. Synthesis library support for LatticeXP 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 LatticeXP 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 LatticeXP family. By using these IPs as standardized blocks, designers are free to concentrate on the unique aspects of their design, increasing their productivity.

LatticeXP Family Data Sheet Architecture

February 2007 Data Sheet DS1001

Architecture Overview

The LatticeXP architecture contains an array of logic blocks surrounded by Programmable I/O Cells (PIC). Interspersed between the rows of logic blocks are rows of sysMEM Embedded Block RAM (EBR) as shown in Figure 2-1.

On the left and right sides of the PFU array, there are Non-volatile Memory Blocks. In configuration mode this non-volatile memory is programmed via the IEEE 1149.1 TAP port or the sysCONFIG™ peripheral port. On power up, the configuration data is transferred from the Non-volatile Memory Blocks to the configuration SRAM. With this technology, expensive external configuration memories are not required and designs are secured from unauthorized read-back. This transfer of data from non-volatile memory to configuration SRAM via wide busses happens in microseconds, providing an "instant-on" capability that allows easy interfacing in many applications.

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.

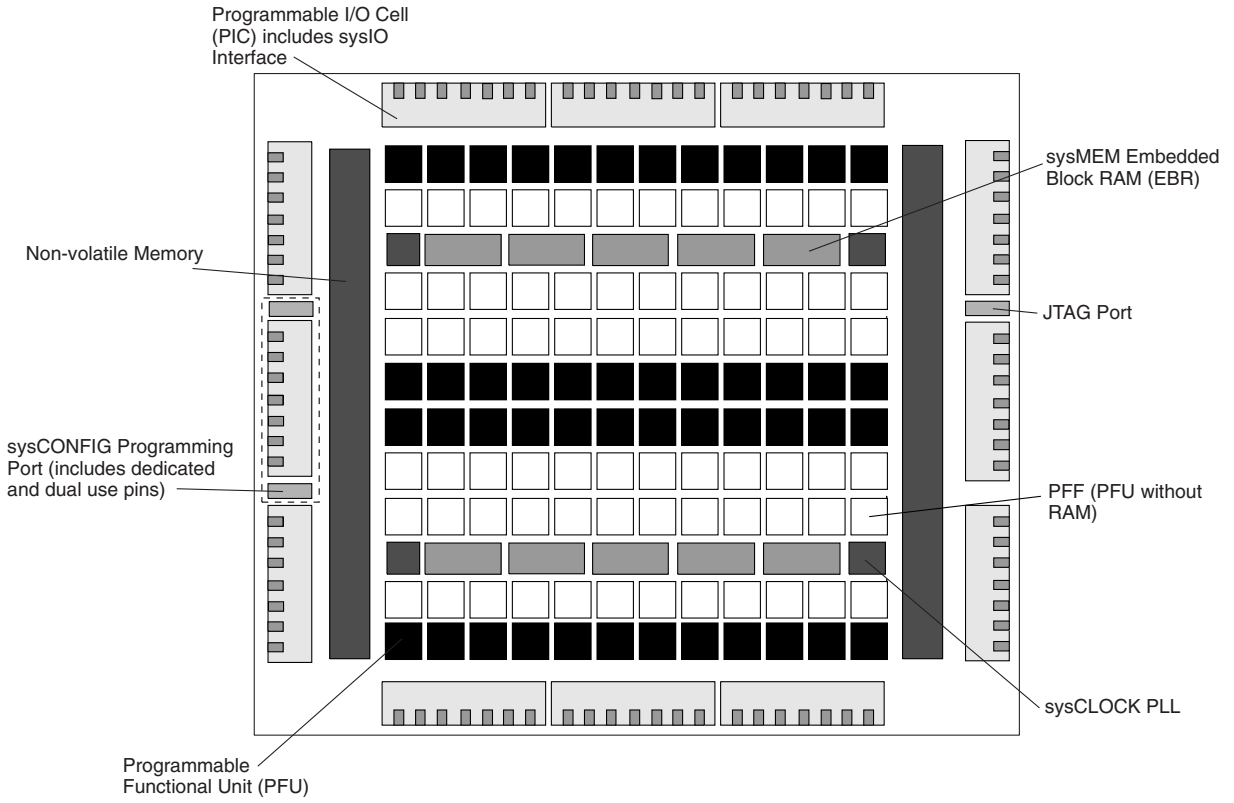
Each PIC block encompasses two PIOs (PIO pairs) with their respective sysIO interfaces. PIO pairs on the left and right edges of the device can be configured as LVDS transmit/receive pairs. sysMEM EBRs are large dedicated fast memory blocks. They can be configured as RAM or ROM.

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 LatticeXP 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 LatticeXP devices are available for operation from 3.3V, 2.5V, 1.8V and 1.2V power supplies, providing easy integration into the overall system.

Figure 2-1. LatticeXP Top Level Block Diagram

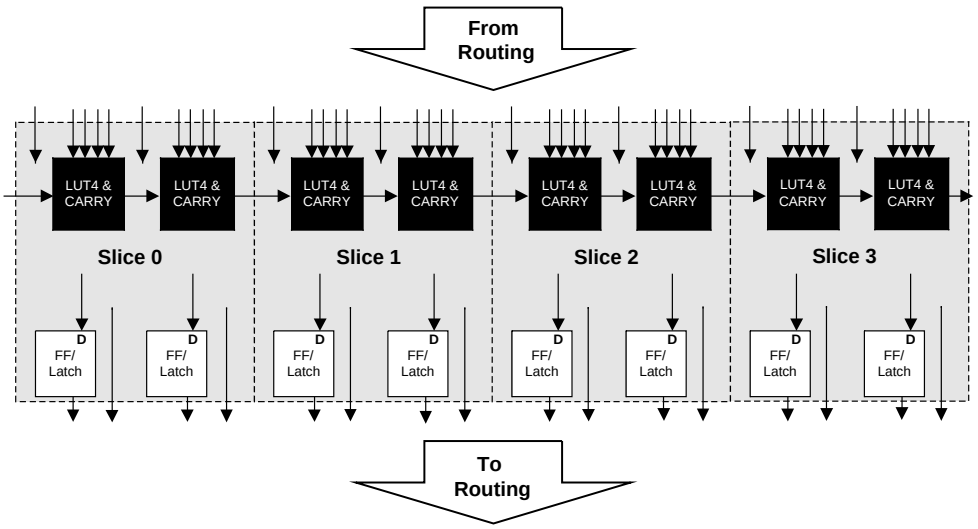


PFU and PFF Blocks

The core of the LatticeXP 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 2-2. All the interconnections to and from PFU blocks are from routing. There are 53 inputs and 25 outputs associated with each PFU block.

Figure 2-2. 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 2-3 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 2-3. Slice Diagram

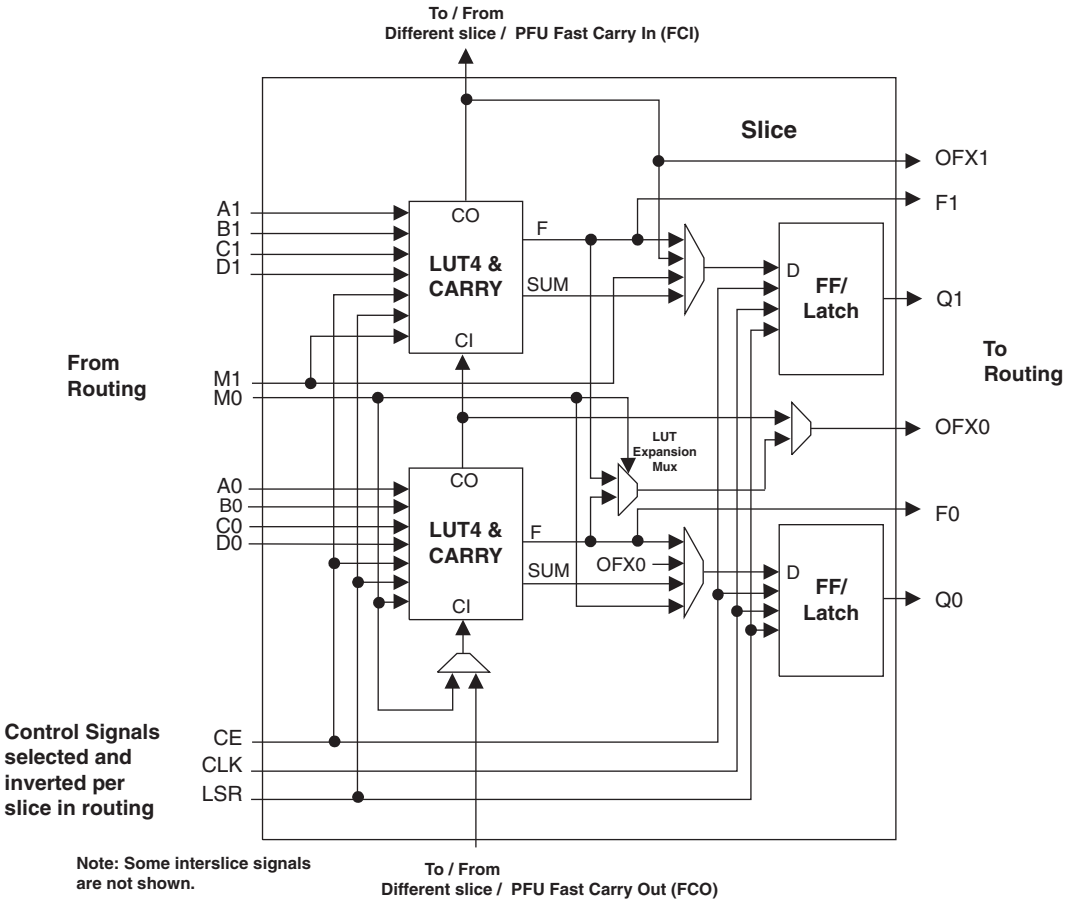


Table 2-1. Slice Signal Descriptions

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 2-2 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	SP 16x2	ROM 16x1 x 2
PFF Slice	LUT 4x2 or LUT 5x1	2-bit Arithmetic Unit	N/A	ROM 16x1 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.

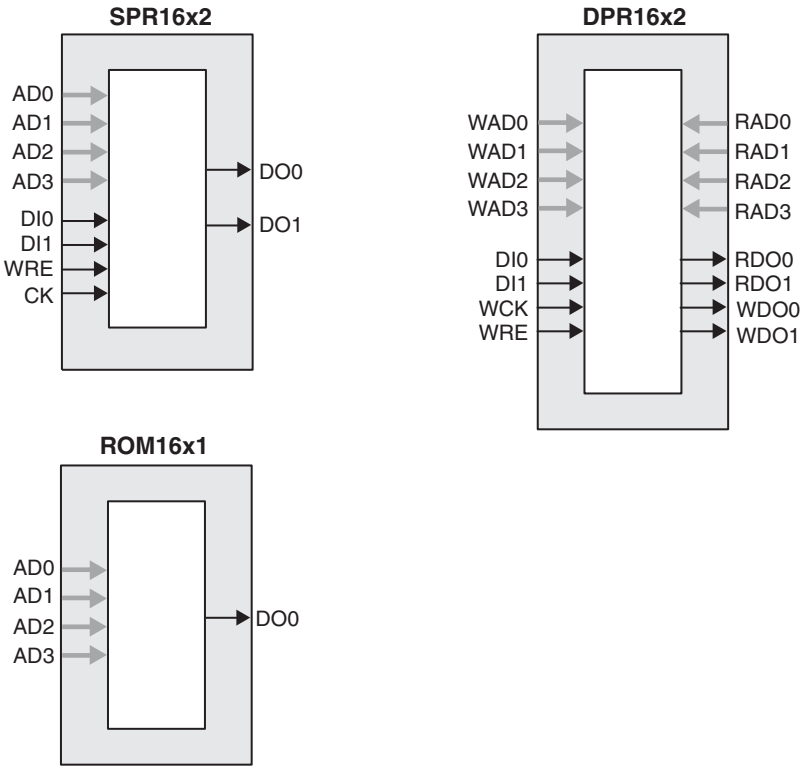
The Lattice design tools support the creation of a variety of different size memories. Where appropriate, the software will construct these using distributed memory primitives that represent the capabilities of the PFU. Table 2-3 shows the number of Slices required to implement different distributed RAM primitives. Figure 2-4 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 RAM mode in LatticeXP 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 2-4. Distributed Memory Primitives



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 LatticeXP devices to route signals individually or as buses 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, vertical and diagonal 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.

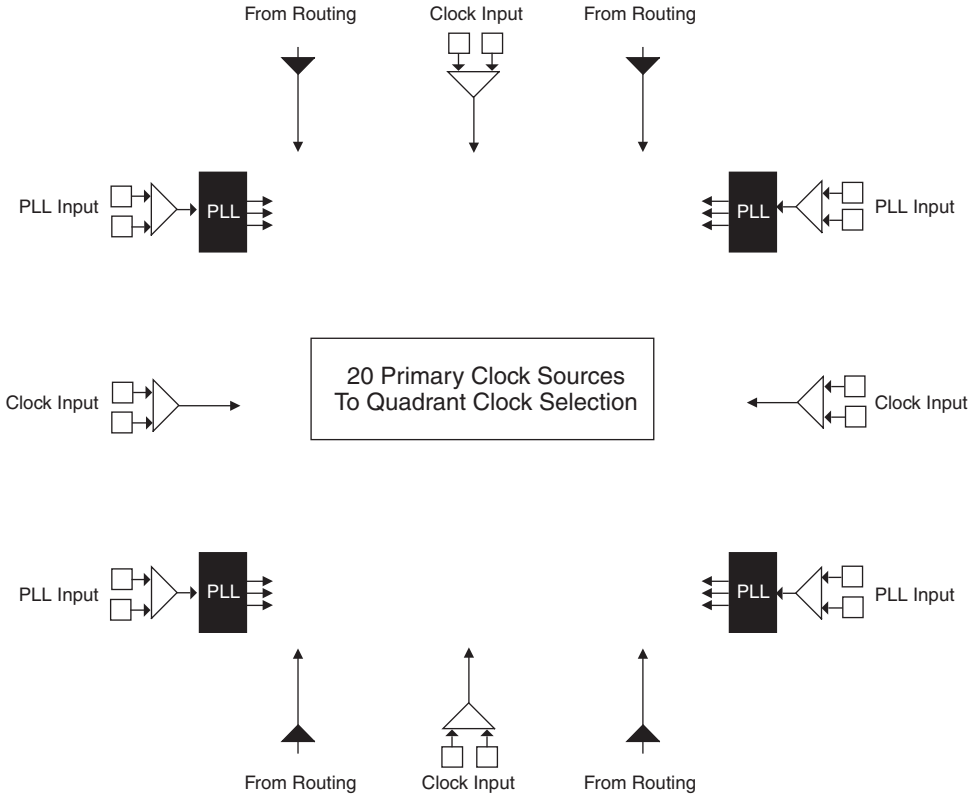
Clock Distribution Network

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

LatticeXP devices derive clocks from three primary sources: PLL outputs, dedicated clock inputs and routing. LatticeXP 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 2-5 shows the 20 primary clock sources.

Figure 2-5. Primary Clock Sources

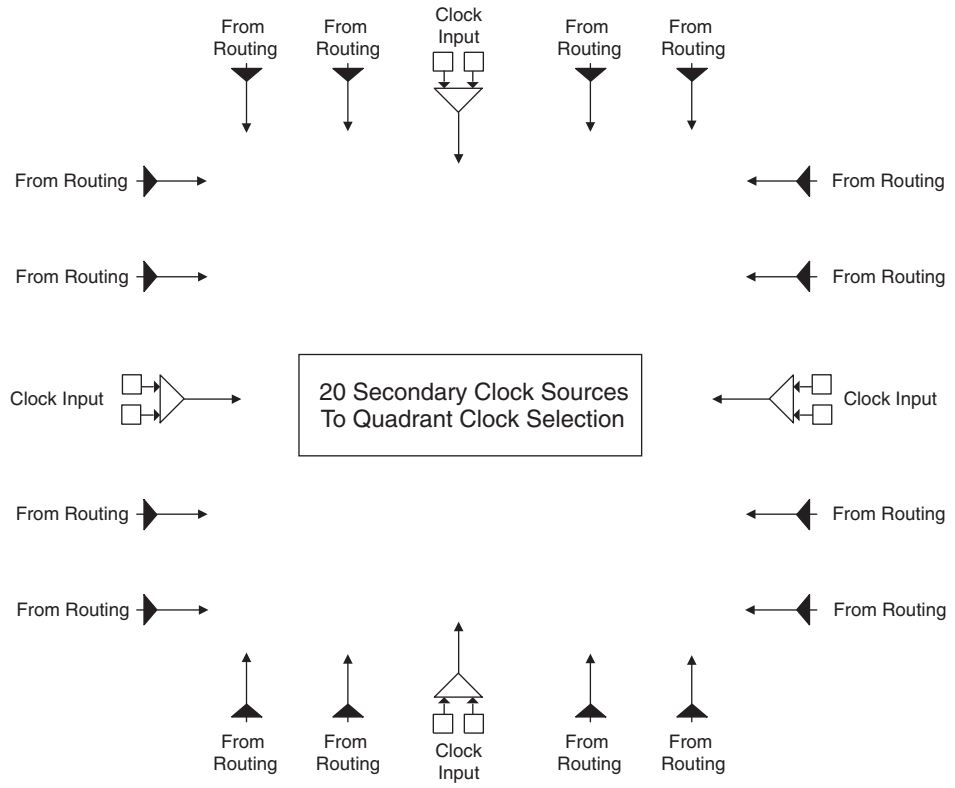


Note: Smaller devices have two PLLs.

Secondary Clock Sources

LatticeXP devices have four secondary clock resources per quadrant. The secondary clock branches are tapped at every PFU. These secondary clock networks can also be used for controls and high fanout data. These secondary clocks are derived from four clock input pads and 16 routing signals as shown in Figure 2-6.

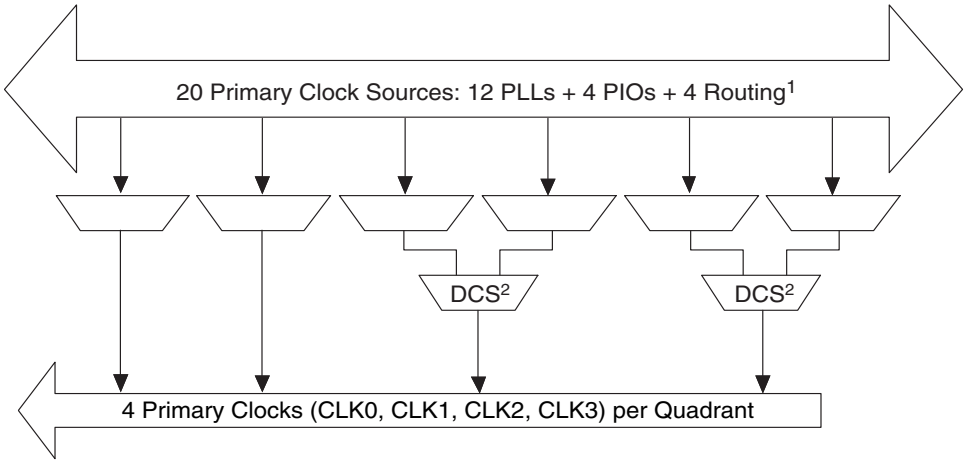
Figure 2-6. Secondary Clock Sources



Clock Routing

The clock routing structure in LatticeXP 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 four secondary clocks are generated from MUXs located in each quadrant as shown in Figure 2-8. Each slice derives its clock from the primary clock lines, secondary clock lines and routing as shown in Figure 2-9.

Figure 2-7. Per Quadrant Primary Clock Selection



1. Smaller devices have fewer PLL related lines.
2. Dynamic clock select.

Figure 2-8. Per Quadrant Secondary Clock Selection

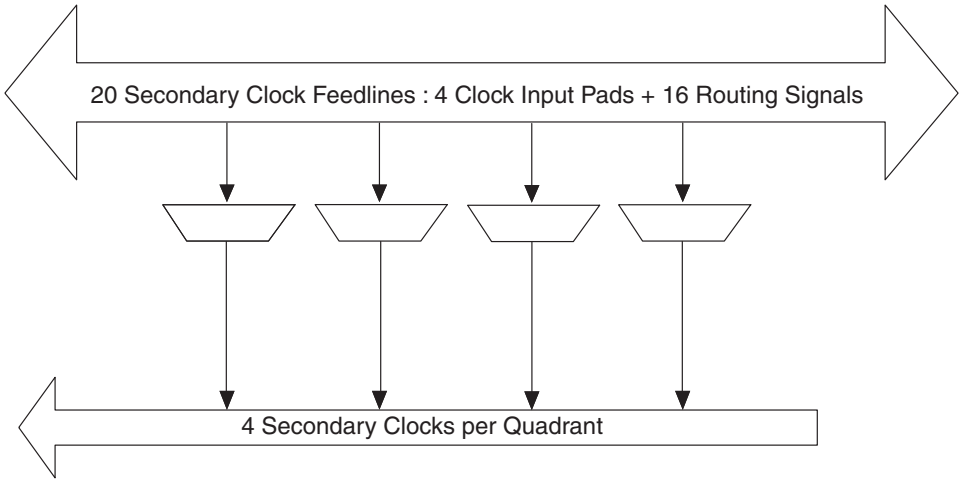
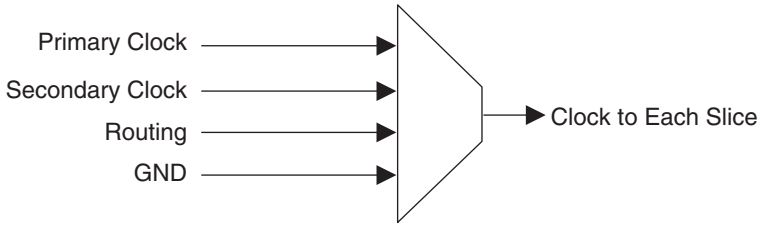


Figure 2-9. Slice Clock Selection



sysCLOCK Phase Locked Loops (PLLs)

The PLL clock input, from pin or routing, feeds into an input clock divider. There are three sources of feedback signals to the feedback divider: from CLKOP (PLL internal), from clock net (CLKOP or CLKOS) or from a user clock (PIN or logic). There is a PLL_LOCK signal to indicate that VCO has locked on to the input clock signal. Figure 2-10 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, post scalar divider and secondary clock divider. The input clock divider is used to divide the input clock signal, while the feedback divider is used to multiply the input clock signal. The post scalar divider allows the VCO to operate at higher frequencies than the clock output, thereby increasing the frequency range. The secondary divider is used to derive lower frequency outputs.

Figure 2-10. PLL Diagram

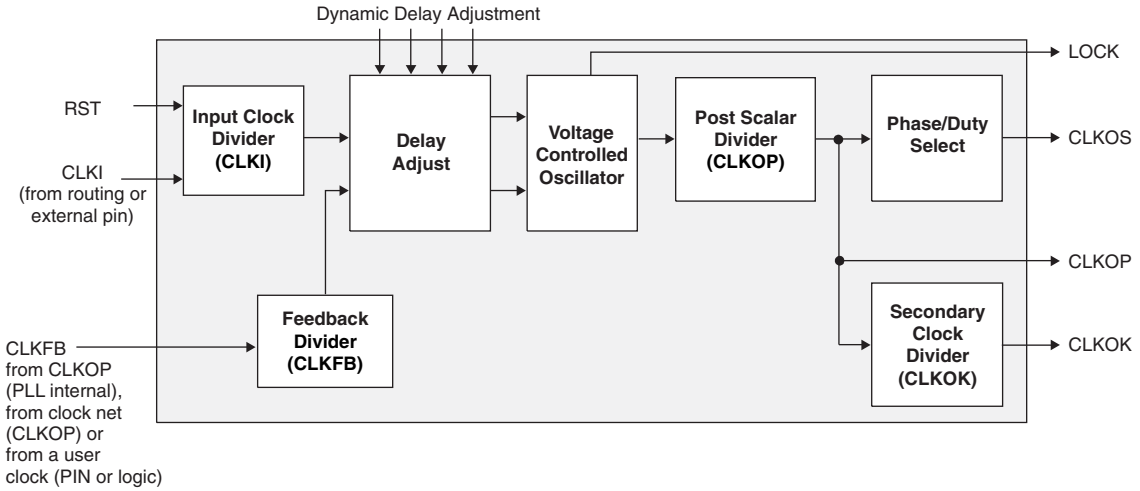


Figure 2-11 shows the available macros for the PLL. Table 2-11 provides signal description of the PLL Block.

Figure 2-11. PLL Primitive

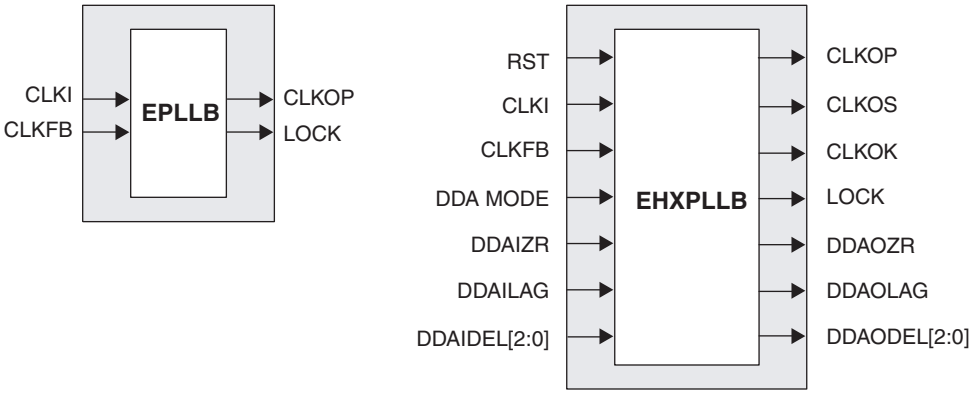


Table 2-5. PLL Signal Descriptions

Signal	I/O	Description
CLKI	I	Clock input from external pin or routing
CLKFB	I	PLL feedback input from CLKOP (PLL internal), from clock net (CLKOP) or from a user clock (PIN or logic)
RST	I	"1" to reset 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.

Dynamic Clock Select (DCS)

The DCS is a global clock buffer with smart multiplexer functions. It takes two independent input clock sources and outputs a clock signal without any glitches or runt pulses. This is achieved irrespective of where the select signal is toggled. There are eight DCS blocks per device, located in pairs at the center of each side. Figure 2-12 illustrates the DCS Block Macro.

Figure 2-12. DCS Block Primitive

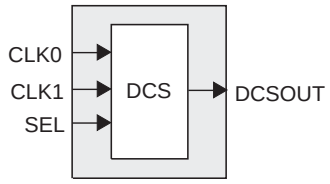
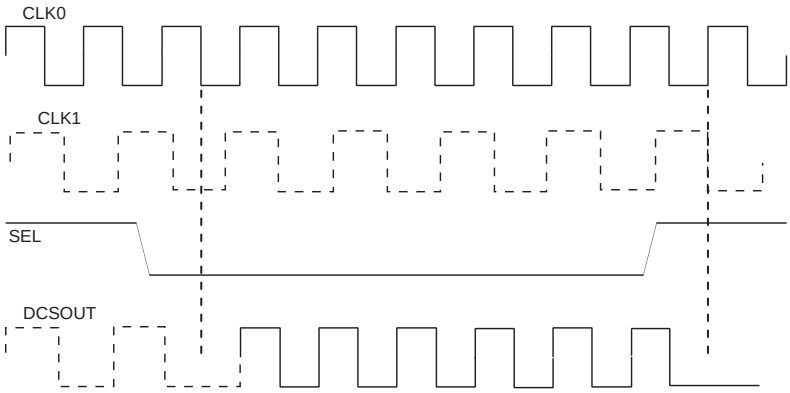


Figure 2-13 shows timing waveforms of the default DCS operating mode. The DCS block can be programmed to other modes. For more information on the DCS, please see details of additional technical documentation at the end of this data sheet.

Figure 2-13. DCS Waveforms



sysMEM Memory

The LatticeXP 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.

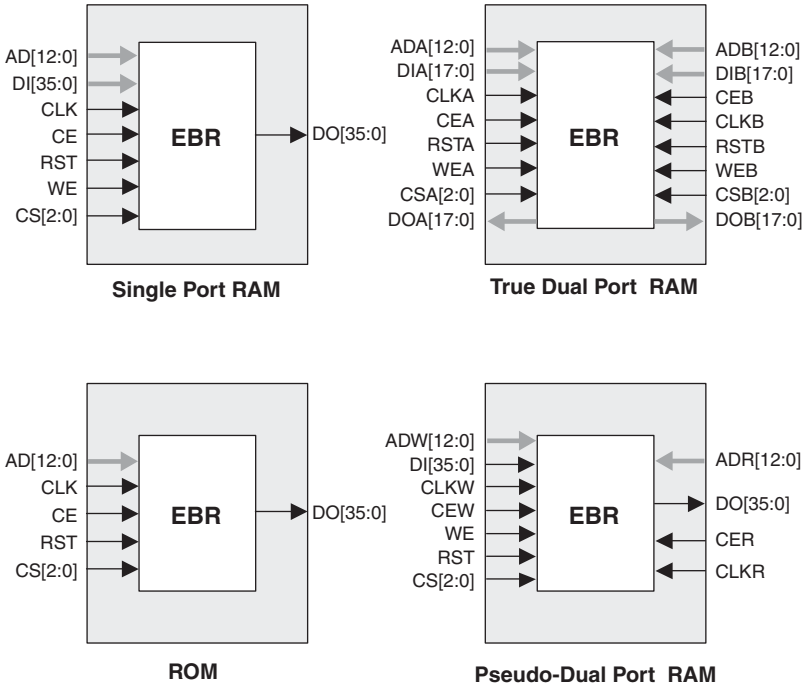
Memory Cascading

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 2-14 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 2-14. sysMEM Memory Primitives



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

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

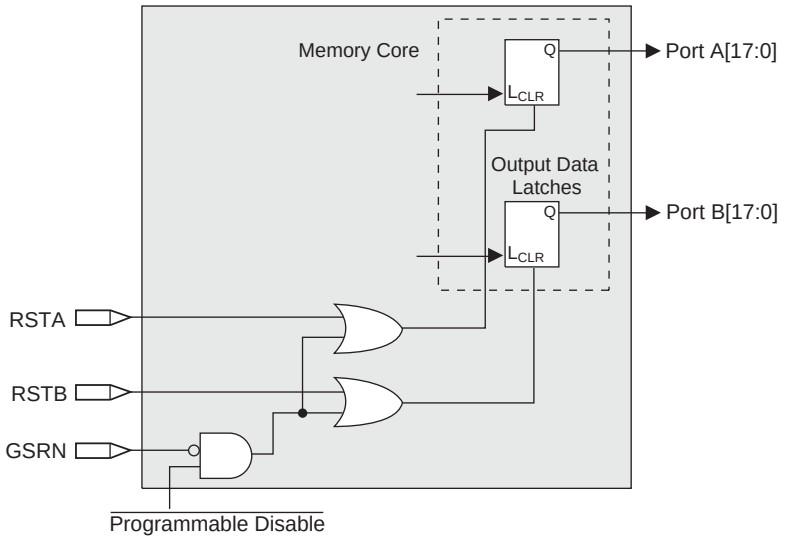
2. **Write Through** - a copy of the input data appears at the output of the same port during a write cycle. This mode is supported for all data widths.

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

Memory Core Reset

The memory array in the EBR utilizes latches at the A and B output ports. These latches can be reset asynchronously. 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 2-15.

Figure 2-15. Memory Core Reset

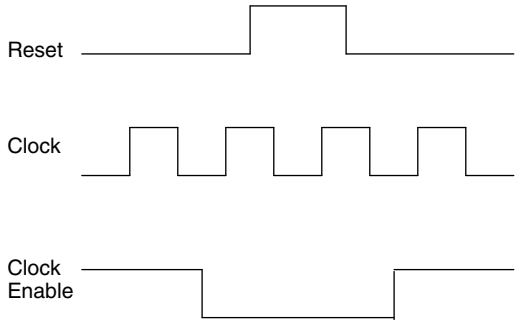


For further information on sysMEM EBR block, see the details of additional technical documentation at the end of this data sheet.

EBR Asynchronous Reset

EBR asynchronous reset or GSR (if used) can only be applied if all clock enables are low for a clock cycle before the reset is applied and released a clock cycle after the reset is released, as shown in Figure 2-16. The GSR input to the EBR is always asynchronous.

Figure 2-16. EBR Asynchronous Reset (Including GSR) Timing Diagram



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

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

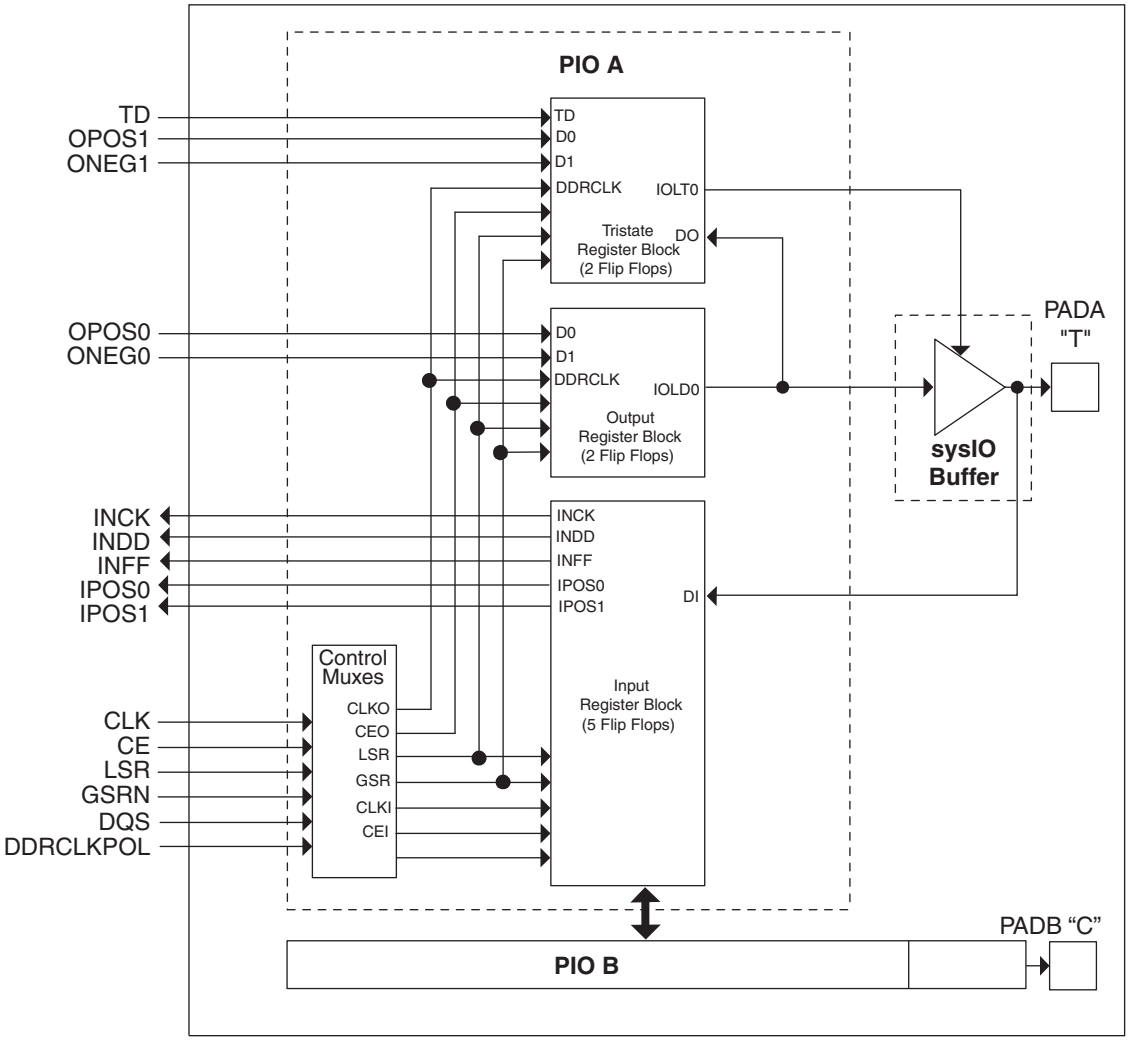
These instructions apply to all EBR RAM and ROM implementations.

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

Programmable I/O Cells (PICs)

Each PIC contains two PIOs connected to their respective sysIO Buffers which are then connected to the PADs as shown in Figure 2-17. 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 2-17. PIC Diagram



In the LatticeXP family, seven PIOs or four (3.5) PICs are grouped together to provide two LVDS differential pairs, one PIC pair and one single I/O, as shown in Figure 2-18.

Two adjacent PIOs can be joined to provide a differential I/O pair (labeled as “T” and “C”). 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 14 PIOs (a group of 8 PICs) contains a delay element to facilitate the generation of DQS signals as shown in Figure 2-19. The DQS signal feeds the DQS bus which spans the set of 13 PIOs (8 PICs). 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.

The exact DQS pins are shown in a dual function in the Logic Signal Connections table in this data sheet. Additional detail is provided in the Signal Descriptions table in this data sheet.

Figure 2-18. Group of Seven PIOs

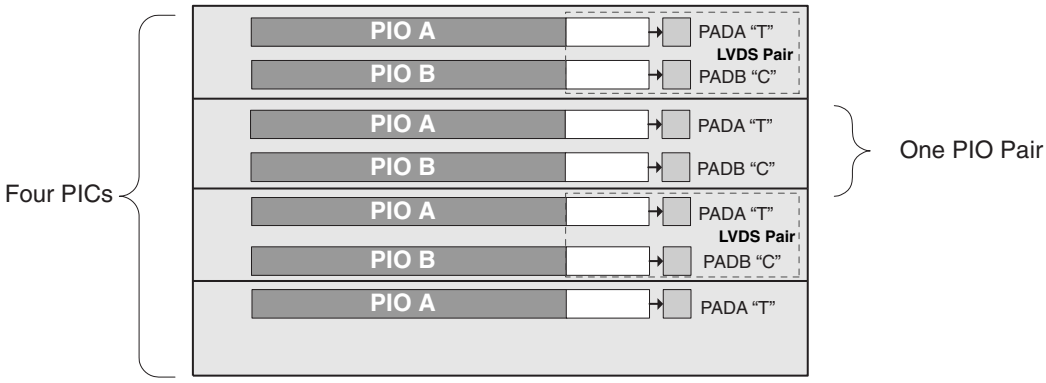
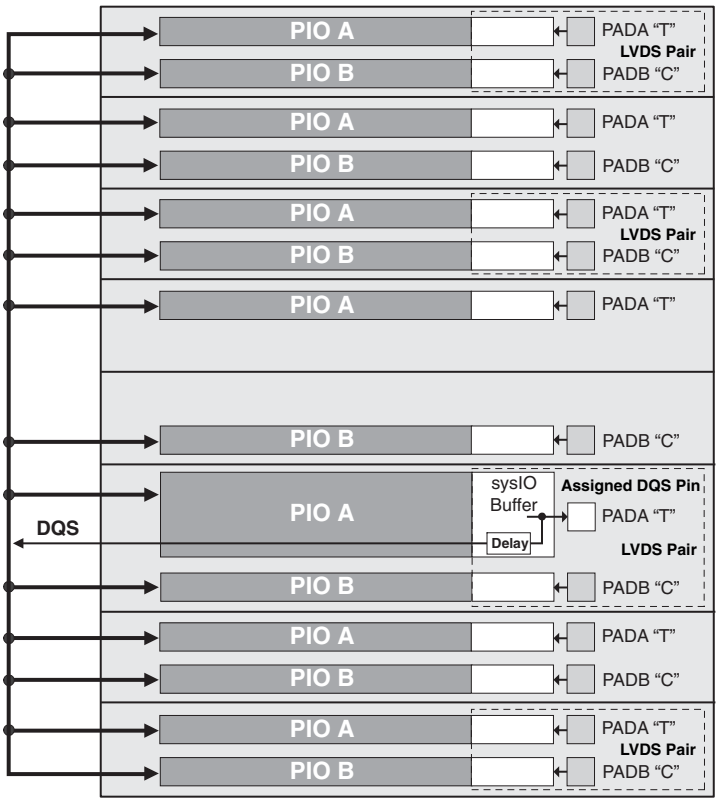


Figure 2-19. DQS Routing



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 2-20 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, ensures no positive 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 2-21 shows the input register waveforms for DDR operation and Figure 2-22 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 the system clock domain. For further discussion of this topic, see the DDR memory section of this data sheet.

Figure 2-20. Input Register Diagram

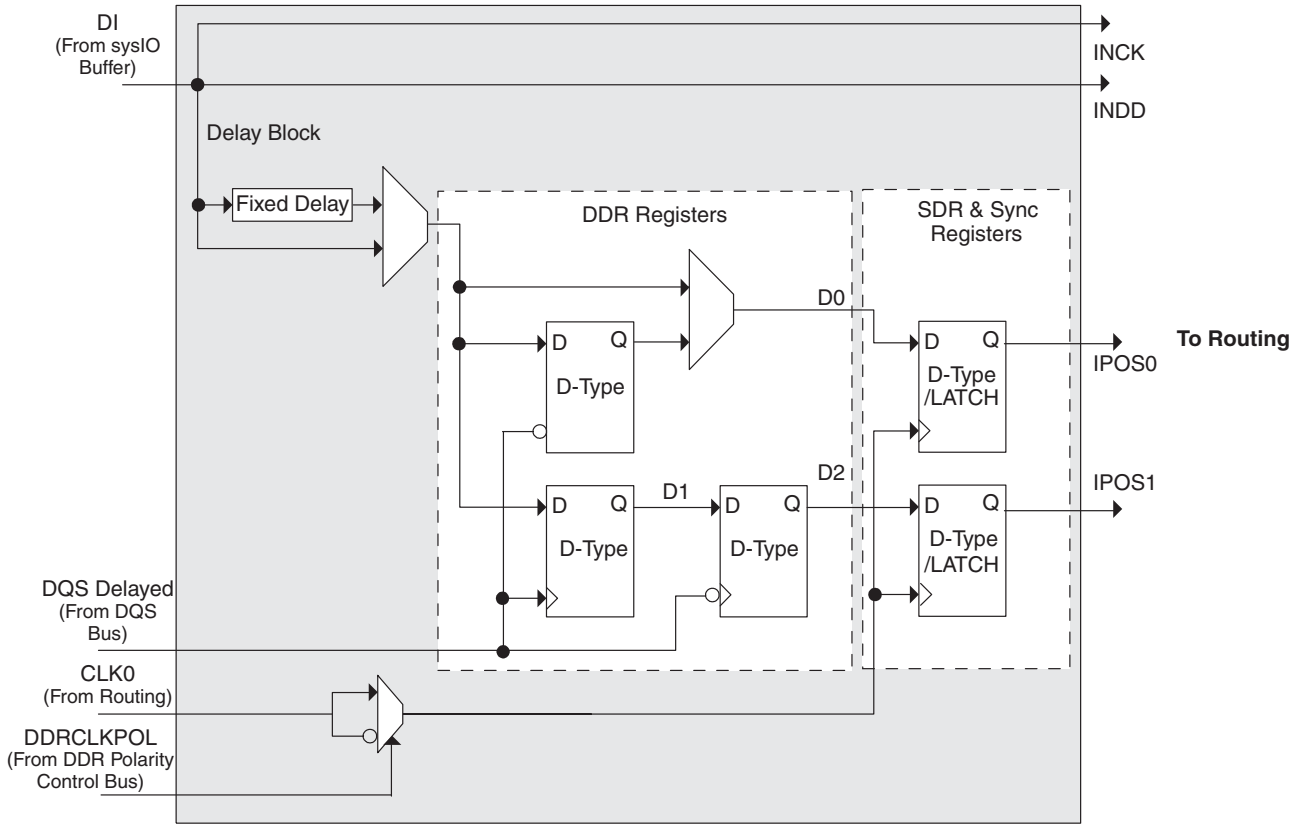


Figure 2-21. Input Register DDR Waveforms

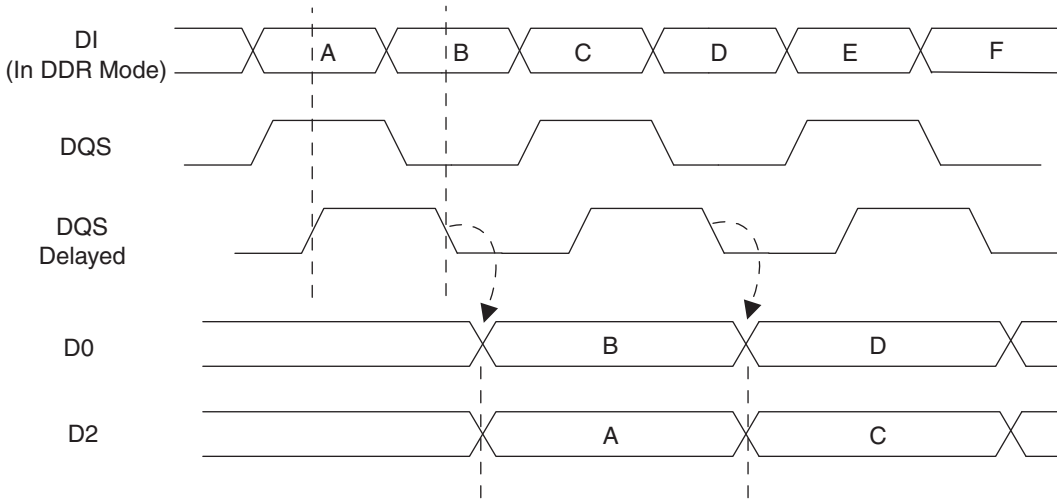
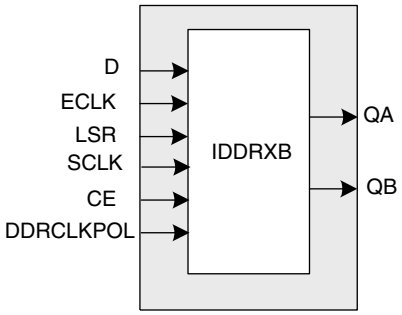


Figure 2-22. INDDRXB Primitive



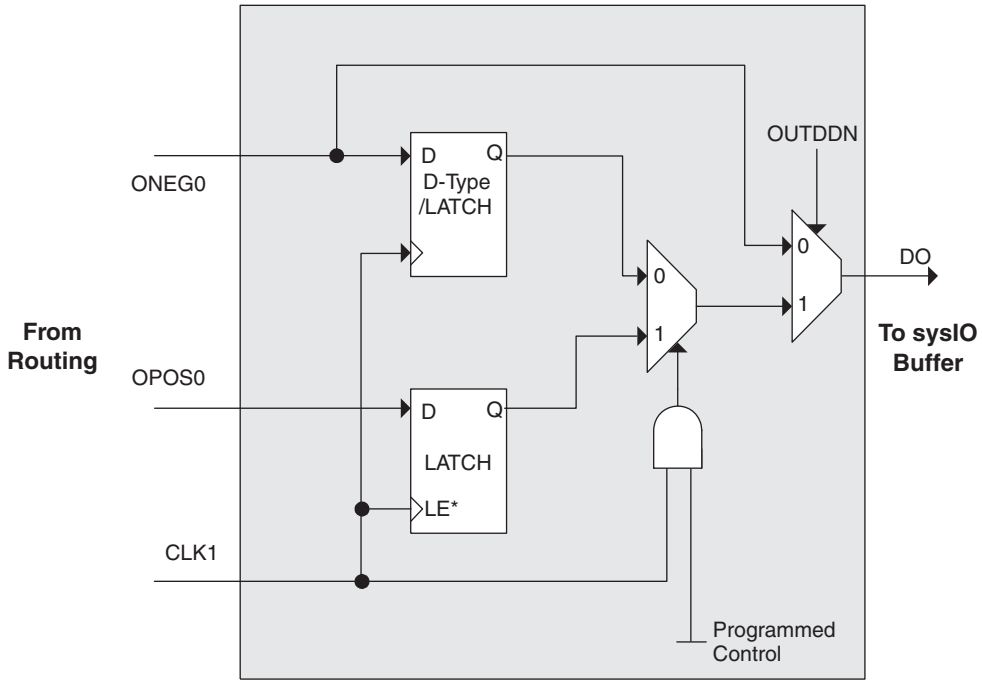
Output Register Block

The output register block provides the ability to register signals from the core of the device before they are passed to the sysIO buffers. The block contains a register for SDR operation that is combined with an additional latch for DDR operation. Figure 2-23 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 as a D-type or as a 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).

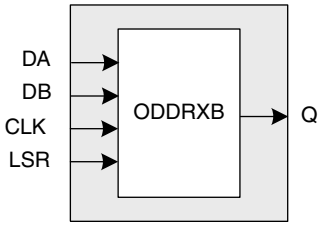
Figure 2-24 shows the design tool DDR primitives. The SDR output register has reset and clock enable available. The additional register for DDR operation does not have reset or clock enable available.

Figure 2-23. Output Register Block



*Latch is transparent when input is low.

Figure 2-24. ODDRXB Primitive

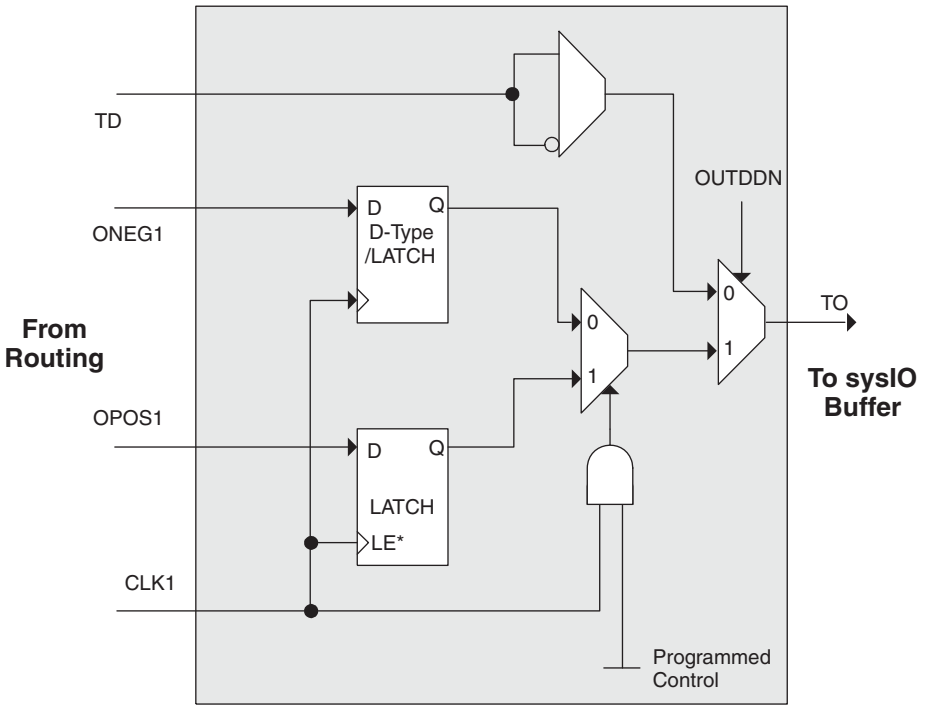


Tristate Register Block

The tristate register block provides the ability to register tri-state control signals from the core of the device before they are passed to the sysIO buffers. The block contains a register for SDR operation and an additional latch for DDR operation. Figure 2-25 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 2-25. 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 LatticeXP devices provide this capability. In addition to these registers, the LatticeXP 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 the polarity control logic which controls the polarity of the clock to the sync registers in the input register blocks. Figures 2-26 and 2-27 show how the polarity control logic 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 2-27. The DLL loop is compensated for temperature, voltage and process variations by the system clock and feedback loop.

Figure 2-26. DQS Local Bus

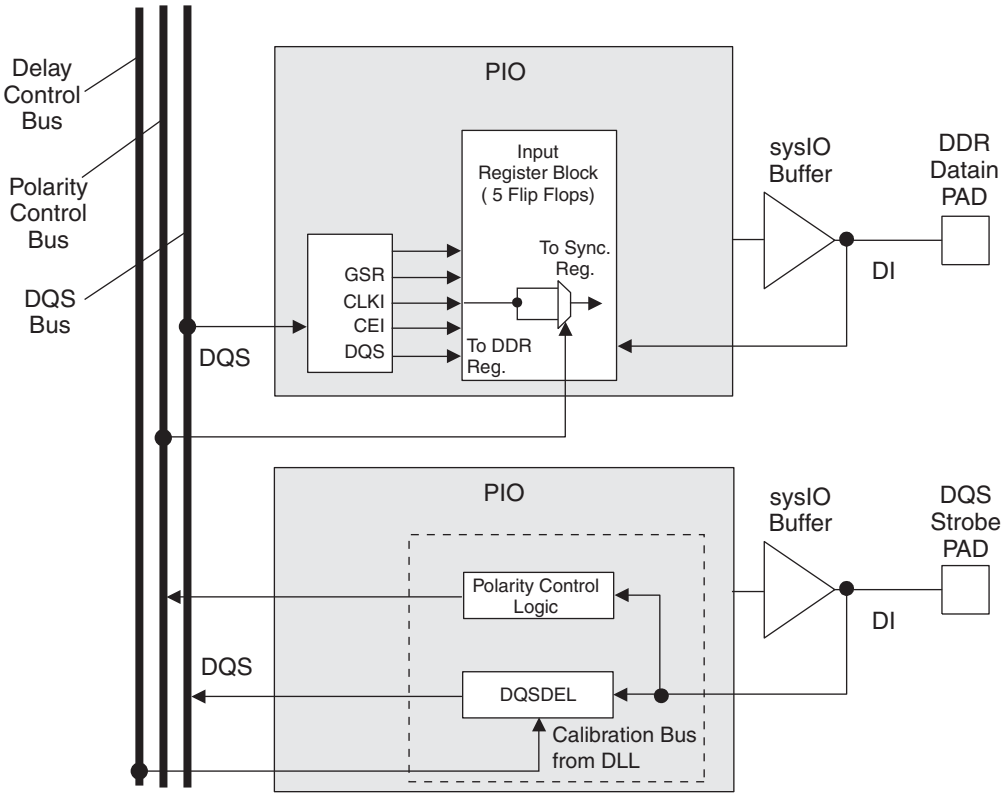
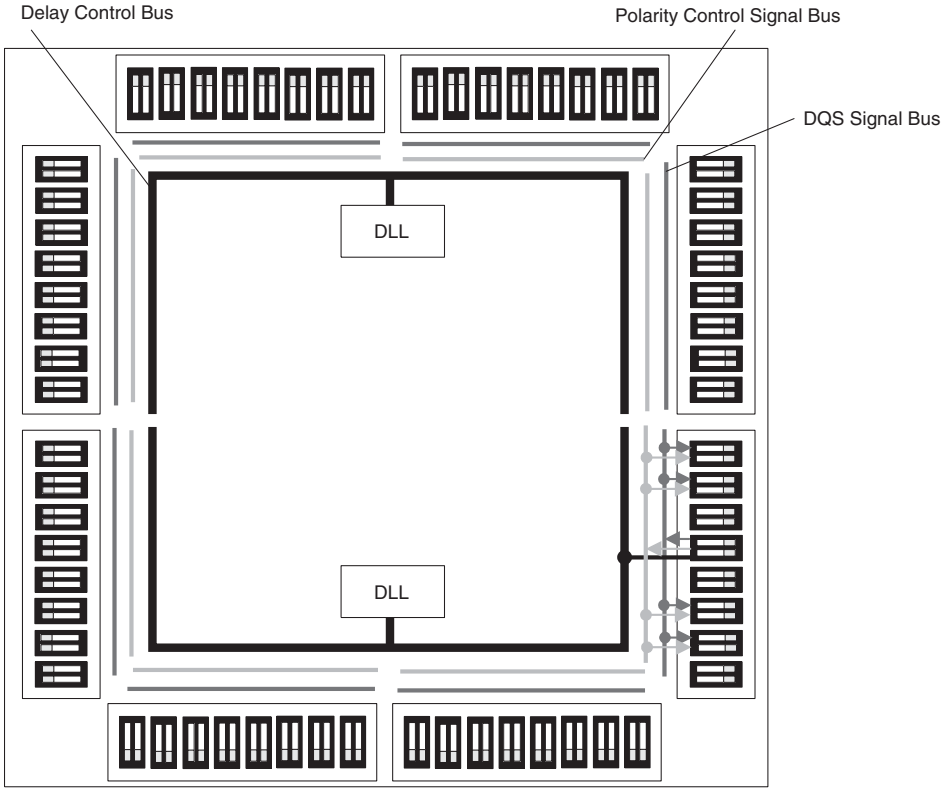


Figure 2-27. DLL Calibration Bus and DQS/DQS Transition Distribution



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 LatticeXP 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 the 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.

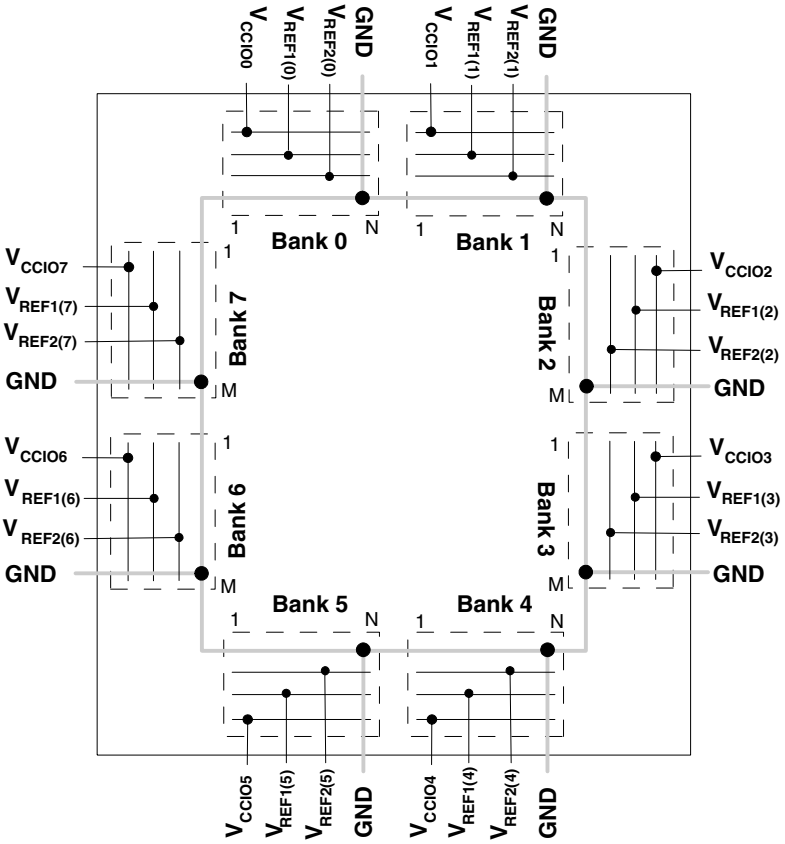
sysIO Buffer Banks

LatticeXP 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 2-28 shows the eight banks and their associated supplies.

In the LatticeXP 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 a fixed threshold input independent of $V_{CCIO.}$ In addition to the bank V_{CCIO} supplies, the LatticeXP 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 LatticeXP devices, a dedicated pin 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 2-28. LatticeXP Banks



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

LatticeXP 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 clamps. Note that the PCI clamp is enabled after $V_{CC,}$ V_{CCAUX} and V_{CCIO} are at valid operating levels and the device has been configured.

2. **Left and Right 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.

   Select I/Os in the left and right banks have LVDS differential output drivers. Refer to the Logic Signal Connections tables for more information.

Typical I/O Behavior During Power-up

The internal power-on-reset (POR) signal is deactivated when V_{CC} and V_{CCAUX} have reached satisfactory levels. After the POR signal is deactivated, the FPGA core logic becomes active. It is the user's responsibility to ensure that all other V_{CCIO} banks are active with valid input logic levels to properly control the output logic states of all the I/O banks that are critical to the application. The default configuration of the I/O pins in a blank device is tri-state with a weak pull-up to VCCIO. The I/O pins will maintain the blank configuration until VCC, VCCAUX and VCCIO have reached satisfactory levels at which time the I/Os will take on the user-configured settings.

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

Supported Standards

The LatticeXP 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 2-7 and 2-8 show the I/O standards (together with their supply and reference voltages) supported by the LatticeXP devices. For further information on utilizing the sysIO buffer to support a variety of standards please see the details of additional technical documentation at the end of this data sheet.

Table 2-7. 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 2-8. 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 LatticeXP 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, which allows easy integration with the rest of the system. These capabilities make the LatticeXP ideal for many multiple power supply and hot-swap applications.

Sleep Mode

The LatticeXP "C" devices (V_{CC} = 1.8/2.5/3.3V) have a sleep mode that allows standby current to be reduced by up to three orders of magnitude during periods of system inactivity. Entry and exit to Sleep Mode is controlled by the SLEEPN pin.

During Sleep Mode, the FPGA logic is non-operational, registers and EBR contents are not maintained and I/Os are tri-stated. Do not enter Sleep Mode during device programming or configuration operation. In Sleep Mode, power supplies can be maintained in their normal operating range, eliminating the need for external switching of power supplies. Table 2-9 compares the characteristics of Normal, Off and Sleep Modes.

Table 2-9. Characteristics of Normal, Off and Sleep Modes

Characteristic	Normal	Off	Sleep
SLEEPN Pin	High	—	Low
Static Icc	Typical <100mA	0	Typical <100uA
I/O Leakage	<10µA	<1mA	<10µA
Power Supplies VCC/VCCIO/VCCAUX	Normal Range	Off	Normal Range
Logic Operation	User Defined	Non Operational	Non operational
I/O Operation	User Defined	Tri-state	Tri-state
JTAG and Programming circuitry	Operational	Non-operational	Non-operational
EBR Contents and Registers	Maintained	Non-maintained	Non-maintained

SLEEPN Pin Characteristics

The SLEEPN pin behaves as an LVCMOS input with the voltage standard appropriate to the VCC supply for the device. This pin also has a weak pull-up typically in the order of 10µA along with a Schmidt trigger and glitch filter to prevent false triggering. An external pull-up to V_{CC} is recommended when Sleep Mode is not used to ensure the device stays in normal operation mode. Typically the device enters Sleep Mode several hundred ns after SLEEPN is held at a valid low and restarts normal operation as specified in the Sleep Mode Timing table. The AC and DC specifications portion of this data sheet show a detailed timing diagram.

Configuration and Testing

The following section describes the configuration and testing features of the LatticeXP family of devices.

IEEE 1149.1-Compliant Boundary Scan Testability

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

The non-volatile memory in the LatticeXP can be configured in three different modes:

- In sysCONFIG mode via the sysCONFIG port. Note this can also be done in background mode.
- In 1532 mode via the 1149.1 port.
- In background mode via the 1149.1 port. This allows the device to be operated while reprogramming takes place.

The SRAM configuration memory can be configured in three different ways:

- At power-up via the on-chip non-volatile memory.
- In 1532 mode via the 1149.1 port SRAM direct configuration.
- In sysCONFIG mode via the sysCONFIG port SRAM direct configuration.

Figure 2-29 provides a pictorial representation of the different programming ports and modes available in the LatticeXP devices.

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.

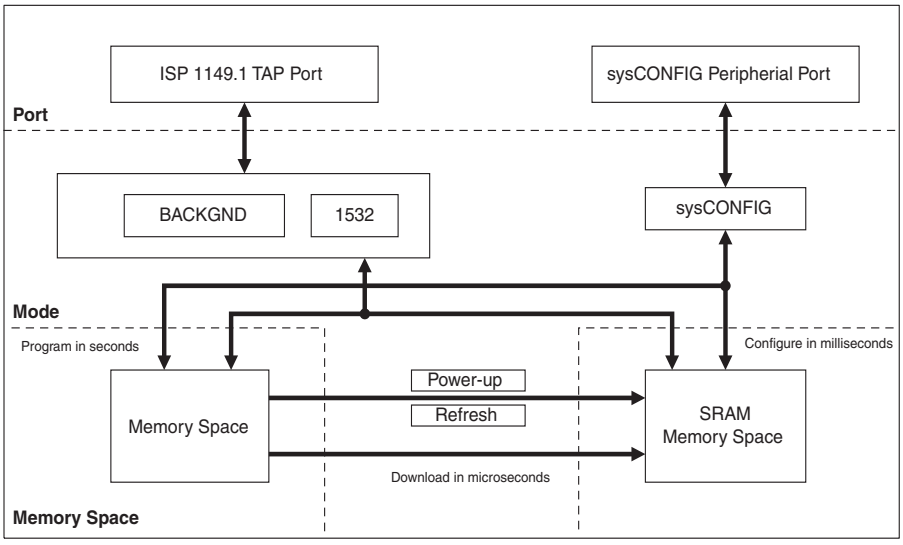
Leave Alone I/O
When using 1532 mode for non-volatile memory programming, users may specify I/Os as high, low, tristated or held at current value. This provides excellent flexibility for implementing systems where reprogramming occurs on-the-fly.

TransFR (Transparent Field Reconfiguration)
TransFR (TFR) is a unique Lattice technology that allows users to update their logic in the field without interrupting system operation using a single ispVM command. See Lattice technical note #TN1087, *Minimizing System Interruption During Configuration Using TransFR Technology,* for details.

Security
The LatticeXP devices contain security bits that, when set, prevent the readback of the SRAM configuration and non-volatile memory spaces. Once set, the only way to clear security bits is to erase the memory space.

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

Figure 2-29. ispXP Block Diagram



Internal Logic Analyzer Capability (ispTRACY)

All LatticeXP 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.

Oscillator

Every LatticeXP 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 in the configuration mode. The default value of the

master serial clock is 2.5MHz. Table 2-10 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 for configuration.
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 2-10. Selectable Master Serial Clock (CCLK) Frequencies During Configuration

CCLK (MHz)	CCLK (MHz)	CCLK (MHz)
2.5[1]	13	45
4.3	15	51
5.4	20	55
6.9	26	60
8.1	30	130
9.2	34	—
10.0	41	—

1. Default

Density Shifting

The LatticeXP 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.

LatticeXP Family Data Sheet DC and Switching Characteristics

August 2006 Data Sheet DS1001

Absolute Maximum Ratings[1, 2, 3, 4]

	XPE (1.2V)	XPC (1.8V/2.5V/3.3V)
Supply Voltage V_{CC}	-0.5 to 1.32V	-0.5 to 3.75V
Supply Voltage V_{CCP}	-0.5 to 1.32V	-0.5 to 3.75V
Supply Voltage V_{CCAUX}	-0.5 to 3.75V	-0.5 to 3.75V
Supply Voltage V_{CCJ}	-0.5 to 3.75V	-0.5 to 3.75V
Output Supply Voltage V_{CCIO}	-0.5 to 3.75V	-0.5 to 3.75V
I/O Tristate Voltage Applied[5]	-0.5 to 3.75V	-0.5 to 3.75V
Dedicated Input Voltage Applied [5]	-0.5 to 3.75V	-0.5 to 4.25V
Storage Temperature (Ambient)	-65 to 150°C	-65 to 150°C
Junction Temp. (Tj)	+125°C	+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 outside of 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. All chip grounds are connected together to a common package GND plane.
5. Overshoot and undershoot of -2V to (V_{IHMAX} + 2) volts is permitted for a duration of <20ns.

Recommended Operating Conditions[3]

Symbol	Parameter	Min.	Max.	Units
V_{CC}	Core Supply Voltage for 1.2V Devices	1.14	1.26	V
	Core Supply Voltage for 1.8V/2.5V/3.3V Devices	1.71	3.465	V
V_{CCP}	Supply Voltage for PLL for 1.2V Devices	1.14	1.26	V
	Supply Voltage for PLL for 1.8V/2.5V/3.3V Devices	1.71	3.465	V
V_{CCAUX}[4]	Auxiliary Supply Voltage	3.135	3.465	V
V_{CCIO}[1, 2]	I/O Driver Supply Voltage	1.14	3.465	V
V_{CCJ}[1]	Supply Voltage for IEEE 1149.1 Test Access Port	1.14	3.465	V
t_{JCOM}	Junction Temperature, Commercial Operation	0	85	C
t_{JIND}	Junction Temperature, Industrial Operation	-40	100	C
$t_{JFLASHCOM}$	Junction Temperature, Flash Programming, Commercial	0	85	C
$t_{JFLASHIND}$	Junction Temperature, Flash Programming, Industrial	0	85	C

1. If V_{CCIO} or V_{CCJ} is set to 3.3V, they must be connected to the same power supply as $V_{CCAUX.}$ For the XPE devices (1.2V V_{CC}), if V_{CCIO} or V_{CCJ} is set to 1.2V, they must be connected to the same power supply as V_{CC}.
2. See recommended voltages by I/O standard in subsequent table.
3. The system designer must ensure that the FPGA design stays within the specified junction temperature and package thermal capabilities of the device based on the expected operating frequency, activity factor and environment conditions of the system.
4. V_{CCAUX} ramp rate must not exceed 30mV/µs during power up when transitioning between 0V and 3.3V.

Hot Socketing Specifications[1, 2, 3, 4, 5, 6]

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) or $0 \le V_{CCAUX} \le V_{CCAUX}$ (MAX).
3. $0 \le V_{CCIO} \le V_{CCIO}$ (MAX) for top and bottom I/O banks.
4. $0.2 \le V_{CCIO} \le V_{CCIO}$ (MAX) for left and right I/O banks.
5. I_{DK} is additive to I_{PU}, I_{PW} or I_{BH}.
6. LVCMOS and LVTTL only.

DC Electrical Characteristics

Over Recommended Operating Conditions

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
I_{IL}, I_{IH}[1, 2, 4]	Input or I/O Leakage	$0 \le V_{IN} \le (V_{CCIO} - 0.2V)$	—	—	10	µA
		$(V_{CCIO} - 0.2V) < V_{IN} \le 3.6V$	—	—	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_{BHHO}	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[3]	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[3]	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

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. Not applicable to SLEEPN/TOE pin.
3. T_A 25°C, f = 1.0MHz
4. When V_{IH} is higher than V_{CCIO}, a transient current typically of 30ns in duration or less with a peak current of 6mA can be expected on the high-to-low transition.

Supply Current (Sleep Mode)[1, 2, 3]

Symbol	Parameter	Device	Typ.[4]	Max	Units
I_{CC}	Core Power Supply	LFXP3C	12	65	µA
		LFXP6C	14	75	µA
		LFXP10C	16	85	µA
		LFXP15C	18	95	µA
		LFXP20C	20	105	µA
I_{CCP}	PLL Power Supply (per PLL)	All LFXP 'C' Devices	1	5	µA
I_{CCAUX}	Auxiliary Power Supply	LFXP3C	2	90	µA
		LFXP6C	2	100	µA
		LFXP10C	2	110	µA
		LFXP15C	3	120	µA
		LFXP20C	4	130	µA
I_{CCIO}	Bank Power Supply[5]	LFXP3C	2	20	µA
		LFXP6C	2	22	µA
		LFXP10C	2	24	µA
		LFXP15C	3	27	µA
		LFXP20C	4	30	µA
I_{CCJ}	VCCJ Power Supply	All LFXP 'C' Devices	1	5	µA

1. Assumes all inputs are configured as LVCMOS and held at the VCCIO or GND.
2. Frequency 0MHz.
3. User pattern: blank.
4. T_A=25°C, power supplies at nominal voltage.
5. Per bank.

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

Over Recommended Operating Conditions

Symbol	Parameter	Device	Typ.[5]	Units
I_{CC}	Core Power Supply	LFXP3E	15	mA
		LFXP6E	20	mA
		LFXP10E	35	mA
		LFXP15E	45	mA
		LFXP20E	55	mA
		LFXP3C	35	mA
		LFXP6C	40	mA
		LFXP10C	70	mA
		LFXP15C	80	mA
		LFXP20C	90	mA
I_{CCP}	PLL Power Supply (per PLL)	All	8	mA
I_{CCAUX}	Auxiliary Power Supply V_{CCAUX} = 3.3V	LFXP3E/C	22	mA
		LFXP6E/C	22	mA
		LFXP10E/C	30	mA
		LFXP15E/C	30	mA
		LFXP20E/C	30	mA
I_{CCIO}	Bank Power Supply[6]	All	2	mA
I_{CCJ}	V_{CCJ} Power Supply	All	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 VCCIO or GND.
3. Frequency 0MHz.
4. User pattern: blank.
5. T_A=25°C, power supplies at nominal voltage.
6. Per bank.

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

Over Recommended Operating Conditions

Symbol	Parameter	Device	Typ.[7]	Units
I_{CC}	Core Power Supply	LFXP3E	40	mA
		LFXP6E	50	mA
		LFXP10E	110	mA
		LFXP15E	140	mA
		LFXP20E	250	mA
		LFXP3C	60	mA
		LFXP6C	70	mA
		LFXP10C	150	mA
		LFXP15C	180	mA
		LFXP20C	290	mA
I_{CCAUX}	Auxiliary Power Supply V_{CCAUX} = 3.3V	LFXP3E /C	50	mA
		LFXP6E /C	60	mA
		LFXP10E /C	90	mA
		LFXP15E /C	110	mA
		LFXP20E /C	130	mA
I_{CCJ}	V_{CCJ} Power Supply	All	2	mA

1. Until DONE signal is active.
2. For further information on supply current, please see details of additional technical documentation at the end of this data sheet.
3. Assumes all outputs are tristated, all inputs are configured as LVCMOS and held at the V_{CCIO} or GND.
4. Frequency 0MHz.
5. Typical user pattern.
6. Assume normal bypass capacitor/decoupling capacitor across the supply.
7. T_A=25°C, power supplies at nominal voltage.

Programming and Erase Flash Supply Current[1, 2, 3, 4, 5]

Symbol	Parameter	Device	Typ.[6]	Units
I_{CC}	Core Power Supply	LFXP3E	30	mA
		LFXP6E	40	mA
		LFXP10E	50	mA
		LFXP15E	60	mA
		LFXP20E	70	mA
		LFXP3C	50	mA
		LFXP6C	60	mA
		LFXP10C	90	mA
		LFXP15C	100	mA
		LFXP20C	110	mA
I_{CCAUX}	Auxiliary Power Supply V_{CCAUX} = 3.3V	LFXP3E /C	50	mA
		LFXP6E /C	60	mA
		LFXP10E /C	90	mA
		LFXP15E /C	110	mA
		LFXP20E /C	130	mA
I_{CCJ}	V_{CCJ} Power Supply[7]	All	2	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. Blank user pattern; typical Flash pattern.
4. Bypass or decoupling capacitor across the supply.
5. JTAG programming is at 1MHz.
6. T_A=25°C, power supplies at nominal voltage.
7. When programming via JTAG.

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	—	—	—
PCI33	3.135	3.3	3.465	—	—	—
SSTL18 Class I	1.71	1.8	1.89	0.833	0.9	0.969
SSTL2 Class I, II	2.375	2.5	2.625	1.15	1.25	1.35
SSTL3 Class I, II	3.135	3.3	3.465	1.3	1.5	1.7
HSTL15 Class I	1.425	1.5	1.575	0.68	0.75	0.9
HSTL15 Class III	1.425	1.5	1.575	—	0.9	—
HSTL 18 Class I, II	1.71	1.8	1.89	—	0.9	—
HSTL 18 Class III	1.71	1.8	1.89	—	1.08	—
LVDS	2.375	2.5	2.625	—	—	—
LVPECL[1]	3.135	3.3	3.465	—	—	—
BLVDS[1]	2.375	2.5	2.625	—	—	—

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

sysIO Single-Ended DC Electrical Characteristics

Input/Output Standard	V_{IL}		V_{IH}		V_{OL} Max. (V)	V_{OH} Min. (V)	I_{OL} (mA)	I_{OH} (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	$0.35V_{CCIO}$	$0.65V_{CCIO}$	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 ohms	—	1.38	1.60	V
V_{OL}	Output low voltage for V_{OP} or V_{OM}	R_T = 100 ohms	0.9V	1.03	—	V
V_{OD}	Output voltage differential	(V_{OP} - V_{OM}), R_T = 100 ohms	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 ohms	1.125	1.25	1.375	V
ΔV_{OS}	Change in V_{OS} between H and L		—	—	50	mV
I_{OSD}	Output short circuit current	V_{OD} = 0V Driver outputs shorted	—	—	6	mA

Differential HSTL and SSTL

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

LVDS25E

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

Figure 3-1. LVDS25E Output Termination Example

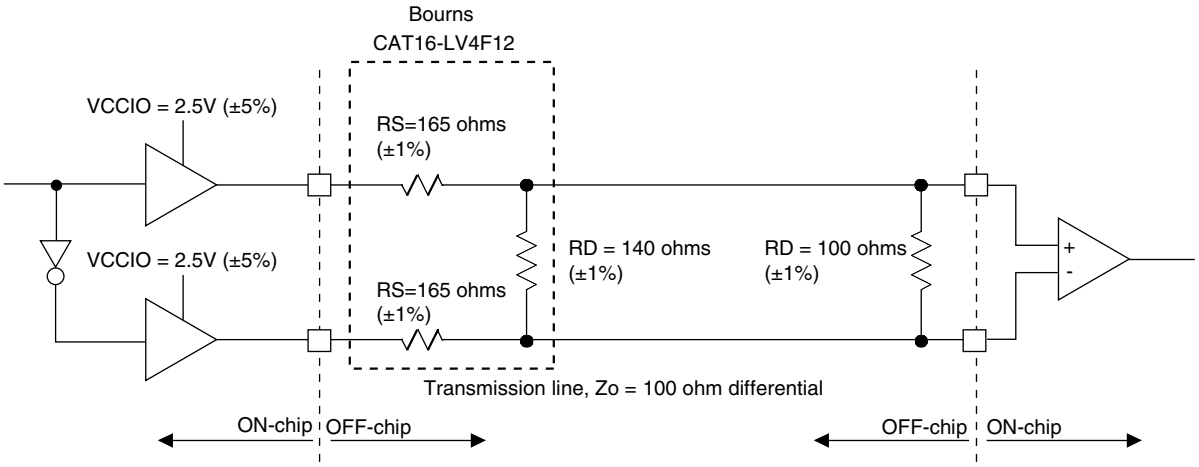


Table 3-1. LVDS25E DC Conditions

Over Recommended Operating Conditions

Parameter	Description	Typical	Units
V_{OH}	Output high voltage	1.43	V
V_{OL}	Output low voltage	1.07	V
V_{OD}	Output differential voltage	0.35	V
V_{CM}	Output common mode voltage	1.25	V
Z_{BACK}	Back impedance	100	Ω
I_{DC}	DC output current	3.66	mA

BLVDS

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

Figure 3-2. BLVDS Multi-point Output Example

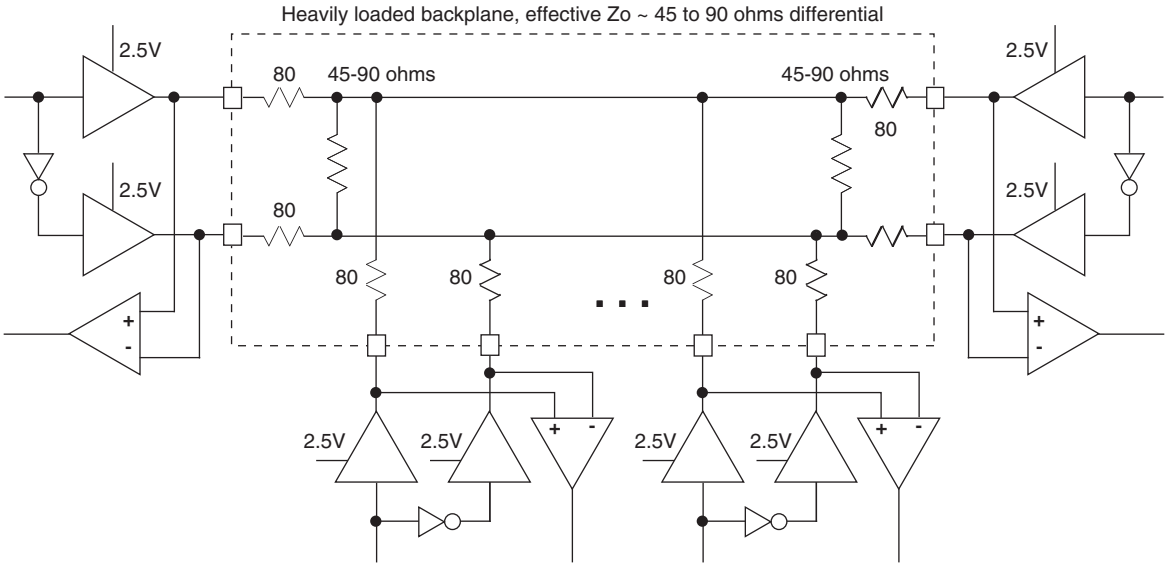


Table 3-2. BLVDS DC Conditions[1]

Over Recommended Operating Conditions

Symbol	Description	Typical		Units
		Zo = 45	Zo = 90	
Z_{OUT}	Output impedance	100	100	ohms
R_{TLEFT}	Left end termination	45	90	ohms
R_{TRIGHT}	Right end termination	45	90	ohms
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 LatticeXP devices support differential LVPECL standard. This standard is emulated using complementary LVCMOS outputs in conjunction with a parallel resistor across the driver outputs. The LVPECL input standard is supported by the LVDS differential input buffer. The scheme shown in Figure 3-3 is one possible solution for point-to-point signals.

Figure 3-3. Differential LVPECL

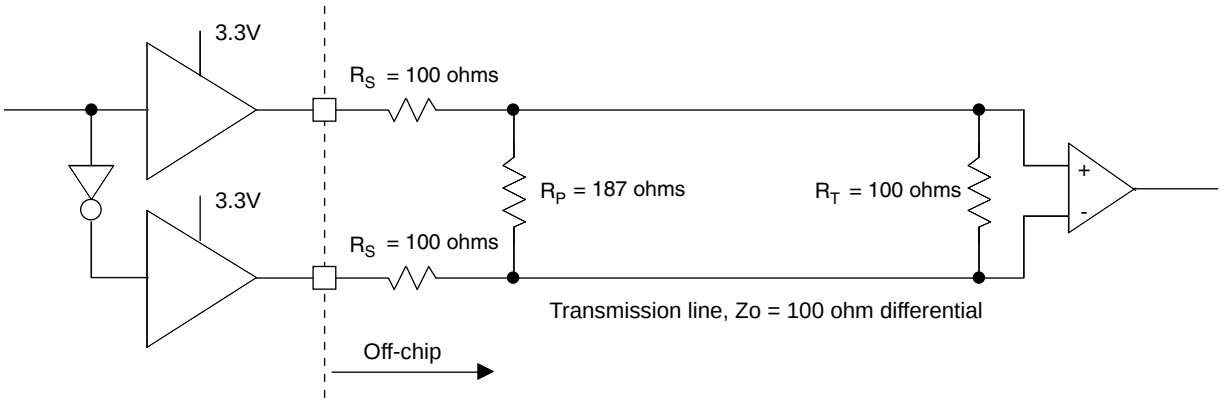


Table 3-3. LVPECL DC Conditions[1]

Over Recommended Operating Conditions

Symbol	Description	Typical	Units
Z_{OUT}	Output impedance	100	ohms
R_P	Driver parallel resistor	187	ohms
R_S	Driver series resistor	100	ohms
R_T	Receiver termination	100	ohms
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	ohms
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 documentation at the end of the data sheet.

RSDS

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

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

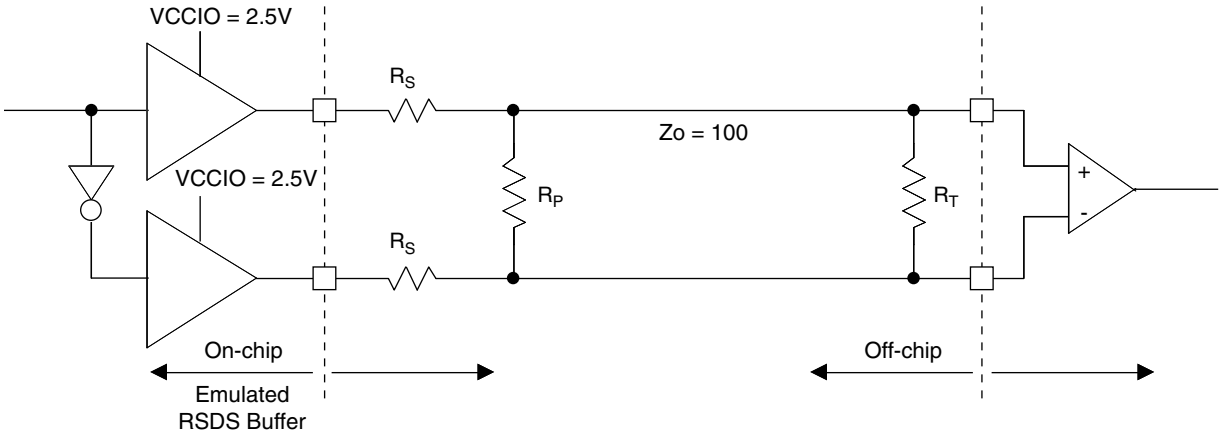


Table 3-4. RSDS DC Conditions

Parameter	Description	Typical	Units
Z_{OUT}	Output impedance	20	ohms
R_S	Driver series resistor	300	ohms
R_P	Driver parallel resistor	121	ohms
R_T	Receiver termination	100	ohms
V_{OH}	Output high voltage	1.35	V
V_{OL}	Output low voltage	1.15	V
V_{OD}	Output differential voltage	0.20	V
V_{CM}	Output common mode voltage	1.25	V
Z_{BACK}	Back impedance	101.5	ohms
I_{DC}	DC output current	3.66	mA

Typical Building Block Function Performance[1]

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

Function	-5 Timing	Units
Basic Functions		
16-bit decoder	6.1	ns
32-bit decoder	7.3	ns
64-bit decoder	8.2	ns
4:1 MUX	4.9	ns
8:1 MUX	5.3	ns
16:1 MUX	5.7	ns
32:1 MUX	6.3	ns

Register to Register Performance

Function	-5 Timing	Units
Basic Functions		
16-bit decoder	351	MHz
32-bit decoder	248	MHz
64-bit decoder	237	MHz
4:1 MUX	590	MHz
8:1 MUX	523	MHz
16:1 MUX	434	MHz
32:1 MUX	355	MHz
8-bit adder	343	MHz
16-bit adder	292	MHz
64-bit adder	130	MHz
16-bit counter	388	MHz
32-bit counter	295	MHz
64-bit counter	200	MHz
64-bit accumulator	164	MHz
Embedded Memory Functions		
Single Port RAM 256x36 bits	254	MHz
True-Dual Port RAM 512x18 bits	254	MHz
Distributed Memory Functions		
16x2 SP RAM	434	MHz
64x2 SP RAM	332	MHz
128x4 SP RAM	235	MHz
32x2 PDP RAM	322	MHz
64x4 PDP RAM	291	MHz

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

Timing v.F0.11

Derating Logic Timing

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

LatticeXP External Switching Characteristics

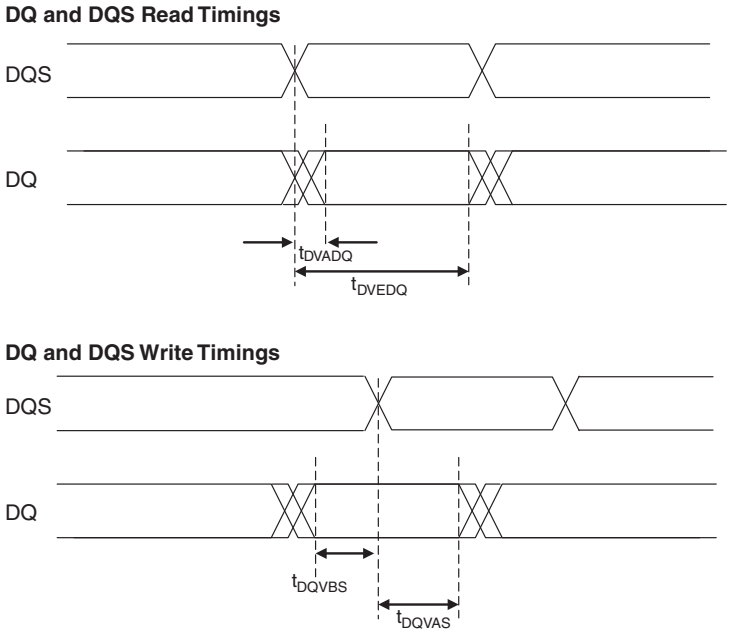
Over Recommended Operating Conditions

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	LFXP3	—	5.12	—	6.12	—	7.43	ns
		LFXP6	—	5.30	—	6.34	—	7.69	ns
		LFXP10	—	5.52	—	6.60	—	8.00	ns
		LFXP15	—	5.72	—	6.84	—	8.29	ns
		LFXP20	—	5.97	—	7.14	—	8.65	ns
t_{SU}	Clock to Data Setup - PIO Input Register	LFXP3	-0.40	—	-0.28	—	-0.16	—	ns
		LFXP6	-0.33	—	-0.32	—	-0.30	—	ns
		LFXP10	-0.61	—	-0.71	—	-0.81	—	ns
		LFXP15	-0.71	—	-0.77	—	-0.87	—	ns
		LFXP20	-0.95	—	-1.14	—	-1.35	—	ns
t_{H}	Clock to Data Hold - PIO Input Register	LFXP3	2.10	—	2.50	—	2.98	—	ns
		LFXP6	2.28	—	2.72	—	3.24	—	ns
		LFXP10	3.02	—	3.51	—	3.71	—	ns
		LFXP15	2.70	—	3.22	—	3.85	—	ns
		LFXP20	2.95	—	3.52	—	4.21	—	ns
t_{SU_DEL}	Clock to Data Setup - PIO Input Register with Input Data Delay	LFXP3	2.38	—	2.49	—	2.66	—	ns
		LFXP6	2.92	—	3.18	—	3.42	—	ns
		LFXP10	2.72	—	2.75	—	2.84	—	ns
		LFXP15	2.99	—	3.13	—	3.18	—	ns
		LFXP20	4.47	—	4.56	—	4.80	—	ns
t_{H_DEL}	Clock to Data Hold - PIO Input Register with Input Data Delay	LFXP3	-0.70	—	-0.80	—	-0.92	—	ns
		LFXP6	-0.47	—	-0.38	—	-0.31	—	ns
		LFXP10	-0.60	—	-0.47	—	-0.32	—	ns
		LFXP15	-1.05	—	-0.98	—	-1.01	—	ns
		LFXP20	-0.80	—	-0.58	—	-0.31	—	ns
f_{MAX_IO}	Clock Frequency of I/O and PFU Register	All	—	400	—	360	—	320	MHz
DDR I/O Pin Parameters[2]									
t_{DVADQ}	Data Valid After DQS (DDR Read)	All	—	0.19	—	0.19	—	0.19	UI
t_{DVEDQ}	Data Hold After DQS (DDR Read)	All	0.67	—	0.67	—	0.67	—	UI
t_{DQVBS}	Data Valid Before DQS	All	0.20	—	0.20	—	0.20	—	UI
t_{DQVAS}	Data Valid After DQS	All	0.20	—	0.20	—	0.20	—	UI
f_{MAX_DDR}	DDR Clock Frequency	All	95	166	95	133	95	100	MHz
Primary and Secondary Clocks									
f_{MAX_PRI}	Frequency for Primary Clock Tree	All	—	450	—	412	—	375	MHz
t_{W_PRI}	Clock Pulse Width for Primary Clock	All	1.19	—	1.19	—	1.19	—	ns
t_{SKEW_PRI}	Primary Clock Skew within an I/O Bank	LFXP3/6/10/15	—	250	—	300	—	350	ps
		LFXP20	—	300	—	350	—	400	ps

1. General timing numbers based on LVCMOS 2.5, 12mA.
2. DDR timing numbers based on SSTL I/O.

Timing v.F0.11

Figure 3-5. DDR Timings



LatticeXP 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)	—	0.28	—	0.34	—	0.40	ns
t_{LUT6_PFU}	LUT6 Delay (A to D Inputs to OFX Output)	—	0.44	—	0.53	—	0.63	ns
t_{LSR_PFU}	Set/Reset to Output of PFU	—	0.90	—	1.08	—	1.29	ns
t_{SUM_PFU}	Clock to Mux (M0,M1) Input Setup Time	0.13	—	0.15	—	0.19	—	ns
t_{HM_PFU}	Clock to Mux (M0,M1) Input Hold Time	-0.04	—	-0.03	—	-0.03	—	ns
t_{SUD_PFU}	Clock to D Input Setup Time	0.13	—	0.16	—	0.19	—	ns
t_{HD_PFU}	Clock to D Input Hold Time	-0.03	—	-0.02	—	-0.02	—	ns
t_{CK2Q_PFU}	Clock to Q Delay, D-type Register Configuration	—	0.40	—	0.48	—	0.58	ns
t_{LE2Q_PFU}	Clock to Q Delay Latch Configuration	—	0.53	—	0.64	—	0.76	ns
t_{LD2Q_PFU}	D to Q Throughput Delay when Latch is Enabled	—	0.55	—	0.66	—	0.79	ns
PFU Dual Port Memory Mode Timing								
t_{CORAM_PFU}	Clock to Output	—	0.40	—	0.48	—	0.58	ns
t_{SUDATA_PFU}	Data Setup Time	-0.18	—	-0.14	—	-0.11	—	ns
t_{HDATA_PFU}	Data Hold Time	0.28	—	0.34	—	0.40	—	ns
t_{SUADDR_PFU}	Address Setup Time	-0.46	—	-0.37	—	-0.30	—	ns
t_{HADDR_PFU}	Address Hold Time	0.71	—	0.85	—	1.02	—	ns
t_{SUWREN_PFU}	Write/Read Enable Setup Time	-0.22	—	-0.17	—	-0.14	—	ns
t_{HWREN_PFU}	Write/Read Enable Hold Time	0.33	—	0.40	—	0.48	—	ns
PIC Timing								
PIO Input/Output Buffer Timing								
t_{IN_PIO}	Input Buffer Delay	—	0.62	—	0.72	—	0.85	ns
t_{OUT_PIO}	Output Buffer Delay	—	2.12	—	2.54	—	3.05	ns
IOLOGIC Input/Output Timing								
t_{SUI_PIO}	Input Register Setup Time (Data Before Clock)	1.35	—	1.83	—	2.37	—	ns
t_{HI_PIO}	Input Register Hold Time (Data After Clock)	0.05	—	0.05	—	0.05	—	ns
t_{COO_PIO}	Output Register Clock to Output Delay	—	0.36	—	0.44	—	0.52	ns
t_{SUCE_PIO}	Input Register Clock Enable Setup Time	-0.09	—	-0.07	—	-0.06	—	ns
t_{HCE_PIO}	Input Register Clock Enable Hold Time	0.13	—	0.16	—	0.19	—	ns
t_{SULSR_PIO}	Set/Reset Setup Time	0.19	—	0.23	—	0.28	—	ns
t_{HLSR_PIO}	Set/Reset Hold Time	-0.14	—	-0.11	—	-0.09	—	ns
EBR Timing								
t_{CO_EBR}	Clock to Output from Address or Data	—	4.01	—	4.81	—	5.78	ns
t_{COO_EBR}	Clock to Output from EBR Output Register	—	0.81	—	0.97	—	1.17	ns
t_{SUDATA_EBR}	Setup Data to EBR Memory	-0.26	—	-0.21	—	-0.17	—	ns
t_{HDATA_EBR}	Hold Data to EBR Memory	0.41	—	0.49	—	0.59	—	ns
t_{SUADDR_EBR}	Setup Address to EBR Memory	-0.26	—	-0.21	—	-0.17	—	ns
t_{HADDR_EBR}	Hold Address to EBR Memory	0.41	—	0.49	—	0.59	—	ns
t_{SUWREN_EBR}	Setup Write/Read Enable to EBR Memory	-0.17	—	-0.13	—	-0.11	—	ns
t_{HWREN_EBR}	Hold Write/Read Enable to EBR Memory	0.26	—	0.31	—	0.37	—	ns
t_{SUCE_EBR}	Clock Enable Setup Time to EBR Output Register	0.19	—	0.23	—	0.28	—	ns
t_{HCE_EBR}	Clock Enable Hold Time to EBR Output Register	-0.13	—	-0.10	—	-0.08	—	ns

LatticeXP Internal Timing Parameters[1] (Continued)

Over Recommended Operating Conditions

Parameter	Description	-5		-4		-3		Units
		Min.	Max.	Min.	Max.	Min.	Max.	
t_{RSTO_EBR}	Reset To Output Delay Time from EBR Output Register	—	1.61	—	1.94	—	2.32	ns
PLL Parameters								
t_{RSTREC}	Reset Recovery to Rising Clock	—	1.00	—	1.00	—	1.00	ns
t_{RSTSU}	Reset Signal Setup Time	1.00	—	1.00	—	1.00	—	ns

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

Timing v.F0.11

Timing Diagrams

PFU Timing Diagrams

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

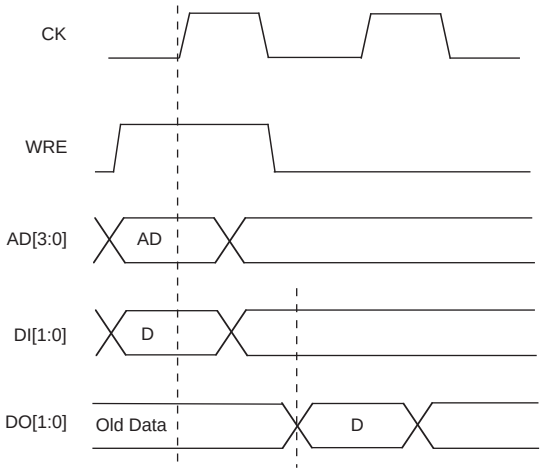
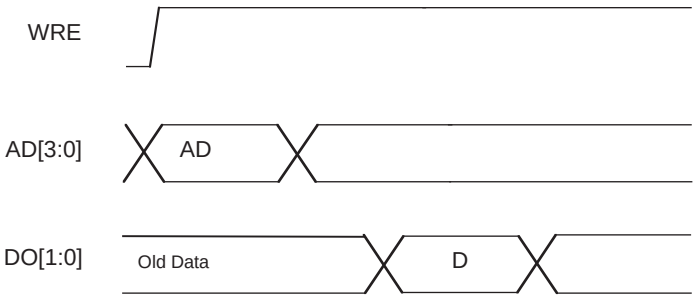
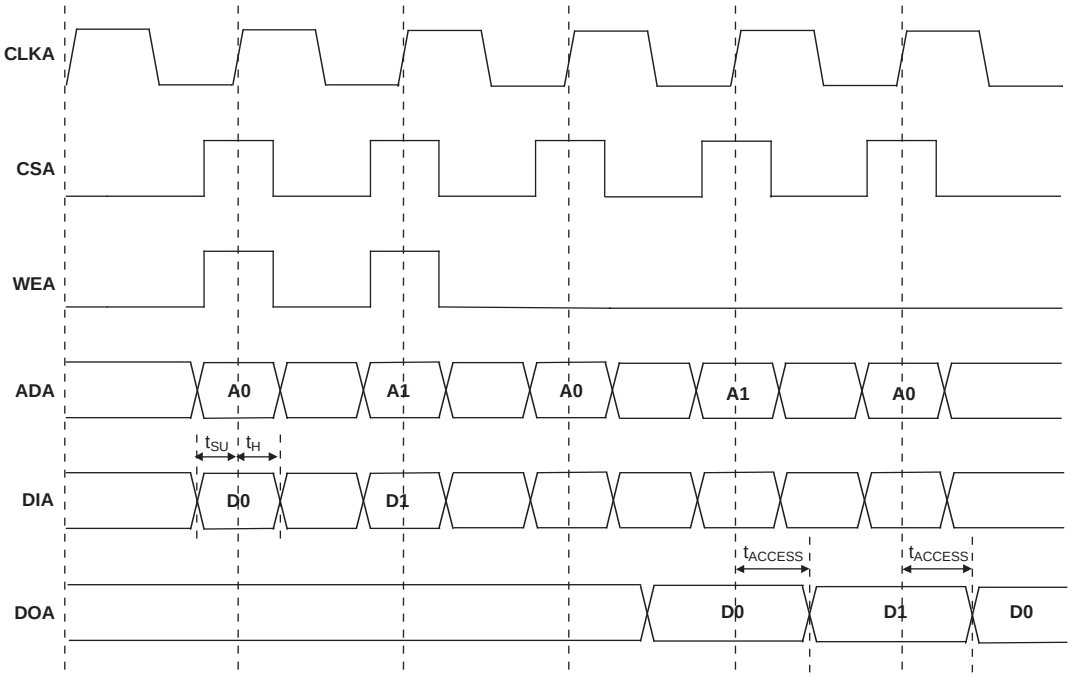


Figure 3-7. Slice Single /Dual Port Read Cycle Timing



EBR Memory Timing Diagrams

Figure 3-8. Read Mode (Normal)



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

Figure 3-9. Read Mode with Input and Output Registers

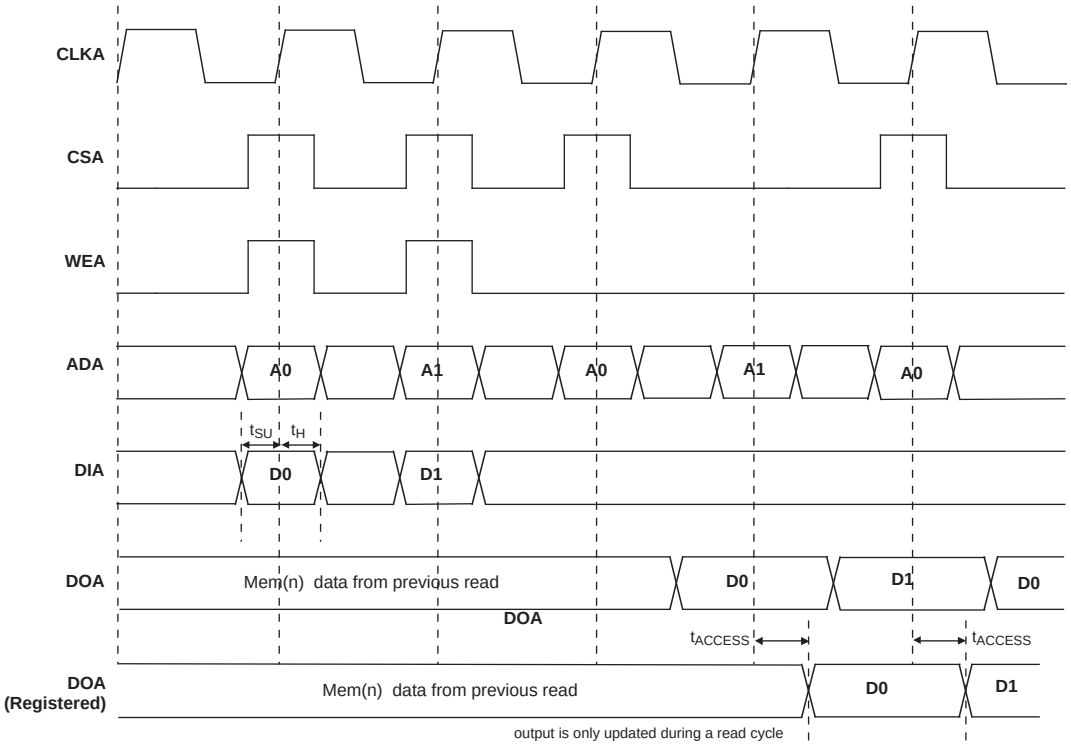
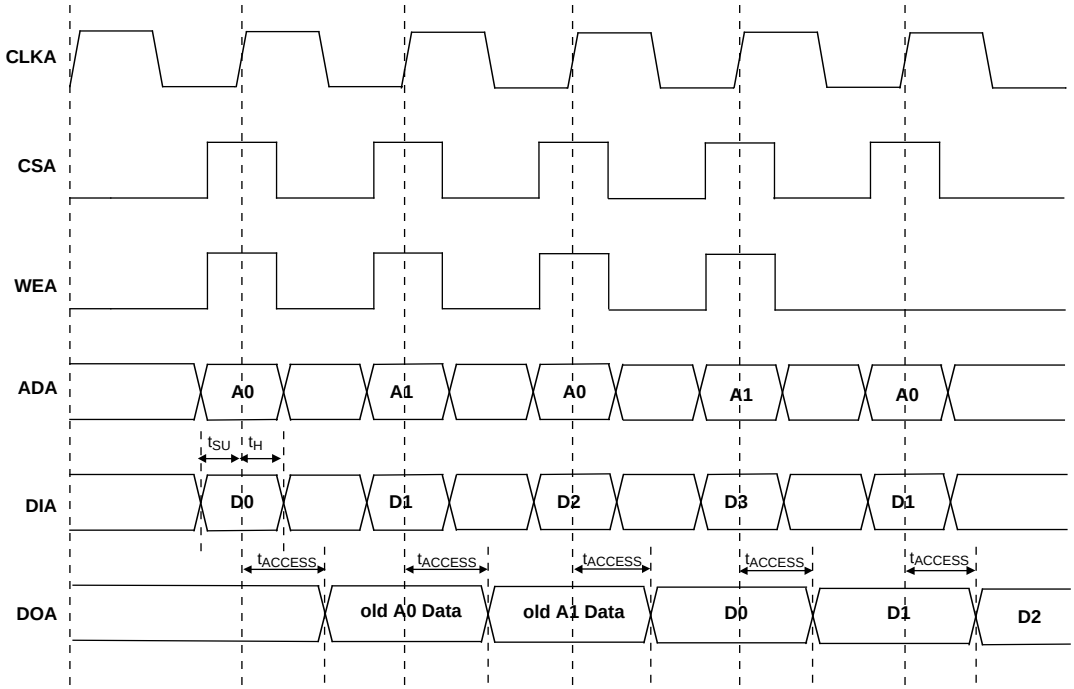
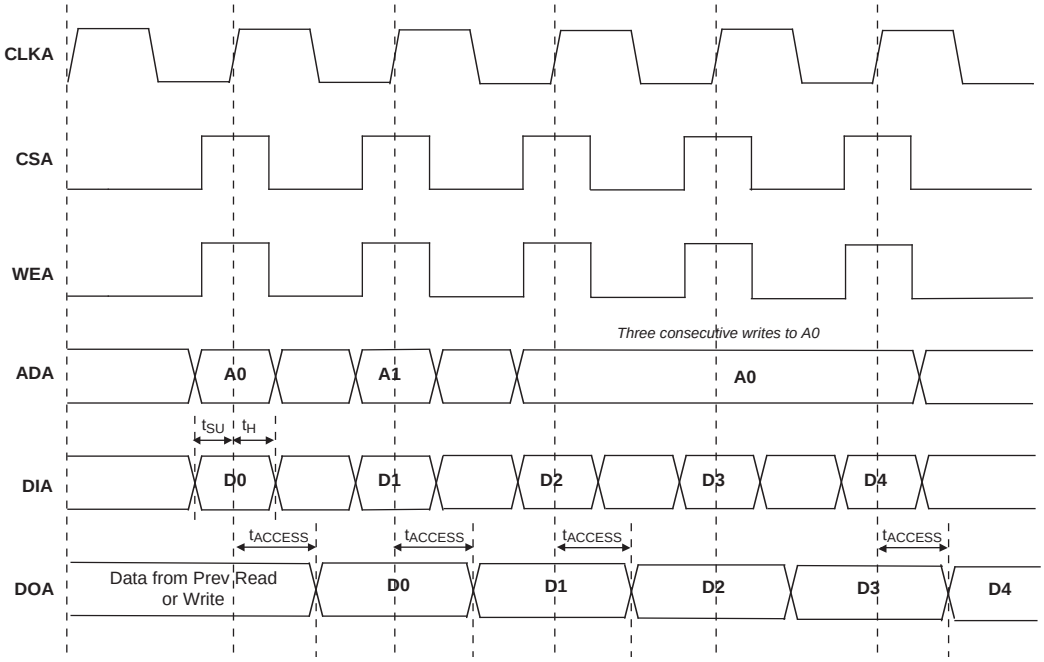


Figure 3-10. Read Before Write (SP Read/Write on Port A, Input Registers Only)



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

LatticeXP Family Timing Adders[1]

Over Recommended Operating Conditions

Buffer Type	Description	-5	-4	-3	Units
Input Adjusters					
LVDS25E	LVDS 2.5 Emulated	0.5	0.5	0.5	ns
LVDS25	LVDS	0.4	0.4	0.4	ns
BLVDS25	BLVDS	0.5	0.5	0.5	ns
LVPECL33	LVPECL	0.6	0.6	0.6	ns
HSTL18_I	HSTL_18 class I	0.4	0.4	0.4	ns
HSTL18_II	HSTL_18 class II	0.4	0.4	0.4	ns
HSTL18_III	HSTL_18 class III	0.4	0.4	0.4	ns
HSTL18D_I	Differential HSTL 18 class I	0.4	0.4	0.4	ns
HSTL18D_II	Differential HSTL 18 class II	0.4	0.4	0.4	ns
HSTL18D_III	Differential HSTL 18 class III	0.4	0.4	0.4	ns
HSTL15_I	HSTL_15 class I	0.5	0.5	0.5	ns
HSTL15_III	HSTL_15 class III	0.5	0.5	0.5	ns
HSTL15D_I	Differential HSTL 15 class I	0.5	0.5	0.5	ns
HSTL15D_III	Differential HSTL 15 class III	0.5	0.5	0.5	ns
SSTL33_I	SSTL_3 class I	0.6	0.6	0.6	ns
SSTL33_II	SSTL_3 class II	0.6	0.6	0.6	ns
SSTL33D_I	Differential SSTL_3 class I	0.6	0.6	0.6	ns
SSTL33D_II	Differential SSTL_3 class II	0.6	0.6	0.6	ns
SSTL25_I	SSTL_2 class I	0.5	0.5	0.5	ns
SSTL25_II	SSTL_2 class II	0.5	0.5	0.5	ns
SSTL25D_I	Differential SSTL_2 class I	0.5	0.5	0.5	ns
SSTL25D_II	Differential SSTL_2 class II	0.5	0.5	0.5	ns
SSTL18_I	SSTL_18 class I	0.5	0.5	0.5	ns
SSTL18D_I	Differential SSTL_18 class I	0.5	0.5	0.5	ns
LVTTL33	LVTTL	0.2	0.2	0.2	ns
LVCMOS33	LVCMOS 3.3	0.2	0.2	0.2	ns
LVCMOS25	LVCMOS 2.5	0.0	0.0	0.0	ns
LVCMOS18	LVCMOS 1.8	0.1	0.1	0.1	ns
LVCMOS15	LVCMOS 1.5	0.1	0.1	0.1	ns
LVCMOS12	LVCMOS 1.2	0.1	0.1	0.1	ns
PCI33	PCI	0.2	0.2	0.2	ns
Output Adjusters					
LVDS25E	LVDS 2.5 Emulated	0.3	0.3	0.3	ns
LVDS25	LVDS 2.5	0.3	0.3	0.3	ns
BLVDS25	BLVDS 2.5	0.3	0.3	0.3	ns
LVPECL33	LVPECL 3.3	0.1	0.1	0.1	ns
HSTL18_I	HSTL_18 class I	0.1	0.1	0.1	ns
HSTL18_II	HSTL_18 class II	0.1	0.1	0.1	ns
HSTL18_III	HSTL_18 class III	0.2	0.2	0.2	ns
HSTL18D_I	Differential HSTL 18 class I	0.1	0.1	0.1	ns
HSTL18D_II	Differential HSTL 18 class II	-0.1	-0.1	-0.1	ns
HSTL18D_III	Differential HSTL 18 class III	0.2	0.2	0.2	ns

LatticeXP Family Timing Adders[1] (Continued)

Over Recommended Operating Conditions

Buffer Type	Description	-5	-4	-3	Units
HSTL15_I	HSTL_15 class I	0.2	0.2	0.2	ns
HSTL15_III	HSTL_15 class III	0.2	0.2	0.2	ns
HSTL15D_I	Differential HSTL 15 class I	0.2	0.2	0.2	ns
HSTL15D_III	Differential HSTL 15 class III	0.2	0.2	0.2	ns
SSTL33_I	SSTL_3 class I	0.1	0.1	0.1	ns
SSTL33_II	SSTL_3 class II	0.3	0.3	0.3	ns
SSTL33D_I	Differential SSTL_3 class I	0.1	0.1	0.1	ns
SSTL33D_II	Differential SSTL_3 class II	0.3	0.3	0.3	ns
SSTL25_I	SSTL_2 class I	-0.1	-0.1	-0.1	ns
SSTL25_II	SSTL_2 class II	0.3	0.3	0.3	ns
SSTL25D_I	Differential SSTL_2 class I	-0.1	-0.1	-0.1	ns
SSTL25D_II	Differential SSTL_2 class II	0.3	0.3	0.3	ns
SSTL18_I	SSTL_1.8 class I	0.1	0.1	0.1	ns
SSTL18D_I	Differential SSTL_1.8 class I	0.1	0.1	0.1	ns
LVTTL33_4mA	LVTTL 4mA drive	0.8	0.8	0.8	ns
LVTTL33_8mA	LVTTL 8mA drive	0.5	0.5	0.5	ns
LVTTL33_12mA	LVTTL 12mA drive	0.3	0.3	0.3	ns
LVTTL33_16mA	LVTTL 16mA drive	0.4	0.4	0.4	ns
LVTTL33_20mA	LVTTL 20mA drive	0.3	0.3	0.3	ns
LVCMOS33_2mA	LVCMOS 3.3 2mA drive	0.8	0.8	0.8	ns
LVCMOS33_4mA	LVCMOS 3.3 4mA drive	0.8	0.8	0.8	ns
LVCMOS33_8mA	LVCMOS 3.3 8mA drive	0.5	0.5	0.5	ns
LVCMOS33_12mA	LVCMOS 3.3 12mA drive	0.3	0.3	0.3	ns
LVCMOS33_16mA	LVCMOS 3.3 16mA drive	0.4	0.4	0.4	ns
LVCMOS33_20mA	LVCMOS 3.3 20mA drive	0.3	0.3	0.3	ns
LVCMOS25_2mA	LVCMOS 2.5 2mA drive	0.7	0.7	0.7	ns
LVCMOS25_4mA	LVCMOS 2.5 4mA drive	0.7	0.7	0.7	ns
LVCMOS25_8mA	LVCMOS 2.5 8mA drive	0.4	0.4	0.4	ns
LVCMOS25_12mA	LVCMOS 2.5 12mA drive	0.0	0.0	0.0	ns
LVCMOS25_16mA	LVCMOS 2.5 16mA drive	0.2	0.2	0.2	ns
LVCMOS25_20mA	LVCMOS 2.5 20mA drive	0.4	0.4	0.4	ns
LVCMOS18_2mA	LVCMOS 1.8 2mA drive	0.6	0.6	0.6	ns
LVCMOS18_4mA	LVCMOS 1.8 4mA drive	0.6	0.6	0.6	ns
LVCMOS18_8mA	LVCMOS 1.8 8mA drive	0.4	0.4	0.4	ns
LVCMOS18_12mA	LVCMOS 1.8 12mA drive	0.2	0.2	0.2	ns
LVCMOS18_16mA	LVCMOS 1.8 16mA drive	0.2	0.2	0.2	ns
LVCMOS15_2mA	LVCMOS 1.5 2mA drive	0.6	0.6	0.6	ns
LVCMOS15_4mA	LVCMOS 1.5 4mA drive	0.6	0.6	0.6	ns
LVCMOS15_8mA	LVCMOS 1.5 8mA drive	0.2	0.2	0.2	ns
LVCMOS12_2mA	LVCMOS 1.2 2mA drive	0.4	0.4	0.4	ns
LVCMOS12_6mA	LVCMOS 1.2 6mA drive	0.4	0.4	0.4	ns
PCI33	PCI33	0.3	0.3	0.3	ns

1. General timing numbers based on LVCMOS 2.5, 12mA.

Timing v.F0.11

sysCLOCK PLL Timing

Over Recommended Operating Conditions

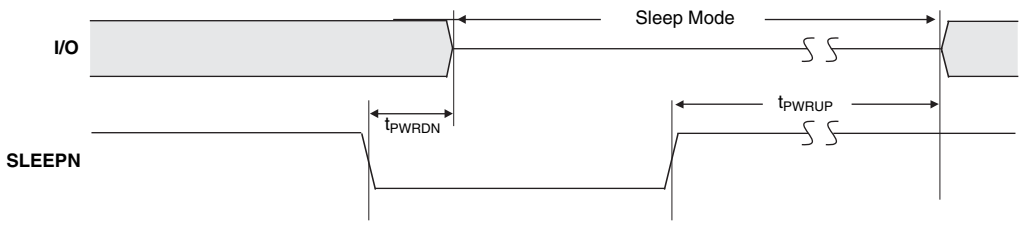
Parameter	Descriptions	Conditions	Min.	Typ.	Max.	Units
f_{IN}	Input Clock Frequency (CLKI, CLKFB)		25	—	375	MHz
f_{OUT}	Output Clock Frequency (CLKOP, CLKOS)		25	—	375	MHz
f_{OUT2}	K-Divider Output Frequency (CLKOK)		0.195	—	187.5	MHz
f_{VCO}	PLL VCO Frequency		375	—	750	MHz
f_{PFD}	Phase Detector Input Frequency		25	—	—	MHz
AC Characteristics						
t_{DT}	Output Clock Duty Cycle	Default duty cycle elected[3]	45	50	55	%
t_{PH}[4]	Output Phase Accuracy		—	—	0.05	UI
t_{OPJIT}[1]	Output Clock Period Jitter	$f_{OUT} \geq 100MHz$	—	—	+/- 125	ps
		$f_{OUT} < 100MHz$	—	—	0.02	UIPP
t_{SK}	Input Clock to Output Clock Skew	Divider ratio = integer	—	—	+/- 200	ps
t_W	Output Clock Pulse Width	At 90% or 10%[3]	1	—	—	ns
t_{LOCK}[2]	PLL Lock-in Time		—	—	150	us
t_{PA}	Programmable Delay Unit		100	250	400	ps
t_{IPJIT}	Input Clock Period Jitter		—	—	+/- 200	ps
t_{FBKDLY}	External Feedback Delay		—	—	10	ns
t_{HI}	Input Clock High Time	90% to 90%	0.5	—	—	ns
t_{LO}	Input Clock Low Time	10% to 10%	0.5	—	—	ns
t_{RST}	RST Pulse Width		10	—	—	ns

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

Timing v.F0.11

LatticeXP "C" Sleep Mode Timing

Parameter	Descriptions		Min.	Typ.	Max.	Units
t_{PWRDN}	SLEEPN Low to I/O Tristate		—	20	32	ns
t_{PWRUP}	SLEEPN High to Power Up	LFXP3	—	1.4	2.1	ms
		LFXP6	—	1.7	2.4	ms
		LFXP10	—	1.1	1.8	ms
		LFXP15	—	1.4	2.1	ms
		LFXP20	—	1.7	2.4	ms
$t_{WSLEEPN}$	SLEEPN Pulse Width to Initiate Sleep Mode		400	—	—	ns
t_{WAWAKE}	SLEEPN Pulse Rejection		—	—	120	ns



LatticeXP sysCONFIG Port Timing Specifications

Over Recommended Operating Conditions

Parameter	Description	Min.	Max.	Units
sysCONFIG Byte Data Flow				
t_{SUCBDI}	Byte D[0:7] Setup Time to CCLK	7	—	ns
t_{HCBDI}	Byte D[0:7] Hold Time to CCLK	3	—	ns
t_{CODO}	Clock to Dout in Flowthrough Mode	—	12	ns
t_{SUCS}	CS[0:1] Setup Time to CCLK	7	—	ns
t_{HCS}	CS[0:1] Hold Time to CCLK	2	—	ns
t_{SUWD}	Write Signal Setup Time to CCLK	7	—	ns
t_{HWD}	Write Signal Hold Time to CCLK	2	—	ns
t_{DCB}	CCLK to BUSY Delay Time	—	12	ns
t_{CORD}	Clock to Out for Read Data	—	12	ns
sysCONFIG Byte Slave Clocking				
t_{BSCH}	Byte Slave Clock Minimum High Pulse	6	—	ns
t_{BSCL}	Byte Slave Clock Minimum Low Pulse	8	—	ns
t_{BSCYC}	Byte Slave Clock Cycle Time	15	—	ns
sysCONFIG Serial (Bit) Data Flow				
t_{SUSCDI}	DI (Data In) Setup Time to CCLK	7	—	ns
t_{HSCDI}	DI (Data In) Hold Time to CCLK	2	—	ns
t_{CODO}	Clock to Dout in Flowthrough Mode	—	12	ns
sysCONFIG Serial Slave Clocking				
t_{SSCH}	Serial Slave Clock Minimum High Pulse	6	—	ns
t_{SSCL}	Serial Slave Clock Minimum Low Pulse	6	—	ns
sysCONFIG POR, Initialization and Wake Up				
t_{ICFG}	Minimum Vcc to INIT High	—	50	ms
t_{VMC}	Time from t_{ICFG} to Valid Master Clock	—	2	us
t_{PRGMRJ}	Program Pin Pulse Rejection	—	7	ns
t_{PRGM}[2]	PROGRAMN Low Time to Start Configuration	25	—	ns
t_{DINIT}	INIT Low Time	—	1	ms
$t_{DPPINIT}$	Delay Time from PROGRAMN Low to INIT Low	—	37	ns
t_{DINITD}	Delay Time from PROGRAMN Low to DONE Low	—	37	ns
t_{IODISS}	User I/O Disable from PROGRAMN Low	—	25	ns
t_{IOENSS}	User I/O Enabled Time from CCLK Edge During Wake-up Sequence	—	25	ns
t_{MWC}	Additional Wake Master Clock Signals after Done Pin High	120	—	cycles
Configuration Master Clock (CCLK)				
Frequency[1]		Selected Value - 30%	Selected Value + 30%	MHz
Duty Cycle		40	60	%

1. See Table 2-10 for available CCLK frequencies.
2. The threshold level for PROGRAMN, as well as for CFG[1] and CFG[0], is determined by V_{CC}, such that the threshold = V_{CC}/2.

Timing v.F0.11

Flash Download Time

Symbol	Parameter		Min.	Typ.	Max.	Units
$t_{REFRESH}$	PROGRAMN Low-to-High. Transition to Done High.	LFXP3	—	1.1	1.7	ms
		LFXP6	—	1.4	2.0	ms
		LFXP10	—	0.9	1.5	ms
		LFXP15	—	1.1	1.7	ms
		LFXP20	—	1.3	1.9	ms

JTAG Port Timing Specifications

Over Recommended Operating Conditions

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

Timing v.F0.11

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

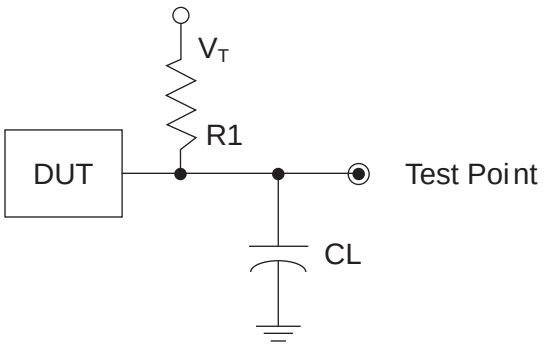


Table 3-5. Test Fixture Required Components, Non-Terminated Interfaces

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 configured to be GSRN.
NC	—	No connect.
GND	—	GND - Ground. Dedicated Pins.
V_{CC}	—	VCC - The power supply pins for core logic. Dedicated Pins.
V_{CCAUX}	—	V_{CCAUX} - The Auxiliary power supply pin. It powers all the differential and referenced input buffers. Dedicated Pins.
V_{CCP0}	—	Voltage supply pins for ULM0PLL (and LLM1PLL[1]).
V_{CCP1}	—	Voltage supply pins for URM0PLL (and LRM1PLL[1]).
GNDP0	—	Ground pins for ULM0PLL (and LLM1PLL[1]).
GNDP1	—	Ground pins for URM0PLL (and LRM1PLL[1]).
V_{CCIOx}	—	V_{CCIO} - The power supply pins for I/O bank x. Dedicated Pins.
$V_{REF1(x)}$, $V_{REF2(x)}$	—	Reference supply pins for I/O bank x. Pre-determined pins in each bank are assigned as V_{REF} inputs. When not used, they may be used as I/O pins.
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	—	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	—	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]	—	Primary Clock Pads, T = true and C = complement, n per side, indexed by bank and 0,1, 2, 3 within bank.
[LOC]DQS[num]	—	DQS input Pads: T (Top), R (Right), B (Bottom), L (Left), DQS, num = Ball function number. Any pad can be configured to be DQS output.

Signal Descriptions (Cont.)

Signal Name	I/O	Descriptions
Test and Programming (Dedicated pins. Pull-up is enabled on input pins during configuration.)		
TMS	I	Test Mode Select input, used to control the 1149.1 state machine.
TCK	I	Test Clock input pin, used to clock the 1149.1 state machine.
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).
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[1:0]	I	Mode pins used to specify configuration modes values latched on rising edge of INITN. During configuration, a pull-up is enabled.
INITN	I/O	Open Drain pin - Indicates the FPGA is ready to be configured. During configuration, a pull-up is enabled. If CFG1 and CFG0 are high (SDM) then this pin is pulled low.
PROGRAMN	I	Initiates configuration sequence when asserted low. This pin always has an active pull-up.
DONE	I/O	Open Drain pin - Indicates that the configuration sequence is complete, and the startup sequence is in progress.
CCLK	I/O	Configuration Clock for configuring an FPGA in sysCONFIG mode.
BUSY	I/O	Generally not used. After configuration it is a user-programmable I/O pin.
CSN	I	sysCONFIG chip select (Active low). During configuration, a pull-up is enabled. After configuration it is user a programmable I/O pin.
CS1N	I	sysCONFIG chip select (Active Low). During configuration, a pull-up is enabled. After configuration it is user programmable I/O pin
WRITEN	I	Write Data on Parallel port (Active low). After configuration it is a user programmable I/O pin
D[7:0]	I/O	sysCONFIG Port Data I/O. After configuration these are user programmable I/O pins.
DOUT, CSON	O	Output for serial configuration data (rising edge of CCLK) when using sysCONFIG port. After configuration, it is a user-programmable I/O pin.
DI	I	Input for serial configuration data (clocked with CCLK) when using sysCONFIG port. During configuration, a pull-up is enabled. After configuration it is a user-programmable I/O pin.
SLEEPN[2]	I	Sleep Mode pin - Active low sleep pin. When this pin is held high, the device operates normally. When driven low, the device moves into Sleep Mode after a specified time.This pin has a weak internal pull-up, but when not used an external pull-up to V_{CC} is recommended.
TOE[3]	I	Test Output Enable tri-states all I/O pins when driven low. This pin has a weak internal pull-up, but when not used an external pull-up to V_{CC} is recommended.

1. Applies to LFXP10, LFXP15 and LFXP20 only.
2. Applies to LFXP “C” devices only.
3. Applies to LFXP “E” devices only.

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

PICs Associated with DQS Strobe	PIO within PIC	Polarity	DDR Strobe (DQS) and Data (DQ) Pins
P[Edge] [n-4]	A	True	DQ
	B	Complement	DQ
P[Edge] [n-3]	A	True	DQ
	B	Complement	DQ
P[Edge] [n-2]	A	True	DQ
	B	Complement	DQ
P[Edge] [n-1]	A	True	DQ
P[Edge] [n]			
	B	Complement	DQ
P[Edge] [n+1]	A	True	[Edge]DQSn
	B	Complement	DQ
P[Edge] [n+2]	A	True	DQ
	B	Complement	DQ
P[Edge] [n+3]	A	True	DQ
	B	Complement	DQ

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

Pin Information Summary[1]

Pin Type		XP3			XP6		
		100 TQFP	144 TQFP	208 PQFP	144 TQFP	208 PQFP	256 fpBGA
Single Ended User I/O		62	100	136	100	142	188
Differential Pair User I/O[2]		19	35	56	35	58	80
Configuration	Dedicated	11	11	11	11	11	11
	Muxed	14	14	14	14	14	14
TAP		5	5	5	5	5	5
Dedicated (total without supplies)		6	6	6	6	6	6
V_{CC}		2	4	8	4	8	8
V_{CCAUX}		2	2	2	2	2	4
V_{CCPLL}		2	2	2	2	2	2
V_{CCIO}	Bank0	1	1	2	1	2	2
	Bank1	1	1	2	1	2	2
	Bank2	1	1	2	1	2	2
	Bank3	1	1	2	1	2	2
	Bank4	1	2	2	2	2	2
	Bank5	1	1	2	1	2	2
	Bank6	1	1	2	1	2	2
	Bank7	1	1	2	1	2	2
GND		10	13	24	13	24	24
GND_{PLL}		2	2	2	2	2	2
NC		0	0	6	0	0	0
Single Ended/Differential I/O per Bank[2]	Bank0	8/2	12/3	20/8	12/3	20/8	26/11
	Bank1	9/0	12/2	18/6	12/2	18/6	26/11
	Bank2	8/3	12/5	14/6	12/5	17/7	21/9
	Bank3	6/2	13/5	14/6	13/5	14/6	21/9
	Bank4	5/2	14/6	21/9	14/6	21/9	26/11
	Bank5	12/4	12/4	21/9	12/4	21/9	26/11
	Bank6	4/2	13/5	14/6	13/5	17/7	21/9
	Bank7	10/4	12/5	14/6	12/5	14/6	21/9
V_{CCJ}		1	1	1	1	1	1

1. 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.
2. The differential I/O per bank includes both dedicated LVDS and emulated LVDS pin pairs. Please see the Logic Signal Connections table for more information.

Pin Information Summary[1] (Cont.)

Pin Type		XP10		XP15			XP20		
		256 fpBGA	388 fpBGA	256 fpBGA	388 fpBGA	484 fpBGA	256 fpBGA	388 fpBGA	484 fpBGA
Single Ended User I/O		188	244	188	268	300	188	268	340
Differential Pair User I/O[2]		76	104	76	112	128	76	112	144
Configuration	Dedicated	11	11	11	11	11	11	11	11
	Muxed	14	14	14	14	14	14	14	14
TAP		5	5	5	5	5	5	5	5
Dedicated (total without supplies)		6	6	6	6	6	6	6	6
V_{CC}		8	14	8	14	28	8	14	28
V_{CCAUX}		4	4	4	4	12	4	4	12
V_{CCPLL}		2	2	2	2	2	2	2	2
V_{CCIO}	Bank0	2	5	2	5	4	2	5	4
	Bank1	2	5	2	5	4	2	5	4
	Bank2	2	4	2	4	4	2	4	4
	Bank3	2	4	2	4	4	2	4	4
	Bank4	2	5	2	5	4	2	5	4
	Bank5	2	5	2	5	4	2	5	4
	Bank6	2	4	2	4	4	2	4	4
	Bank7	2	4	2	4	4	2	4	4
GND		24	50	24	50	56	24	50	56
GND_{PLL}		2	2	2	2	2	2	2	2
NC		0	24	0	0	40	0	0	0
Single Ended/ Differential I/O per Bank[2]	Bank0	26/11	33/14	26/11	39/16	40/17	26/11	39/16	47/20
	Bank1	26/11	33/14	26/11	39/16	40/17	26/11	39/16	47/20
	Bank2	21/8	28/12	21/8	28/12	35/15	21/8	28/12	38/16
	Bank3	21/8	28/12	21/8	28/12	35/15	21/8	28/12	38/16
	Bank4	26/11	33/14	26/11	39/16	40/17	26/11	39/16	47/20
	Bank5	26/11	33/14	26/11	39/16	40/17	26/11	39/16	47/20
	Bank6	21/8	28/12	21/8	28/12	35/15	21/8	28/12	38/16
	Bank7	21/8	28/12	21/8	28/12	35/15	21/8	28/12	38/16
V_{CCJ}		1	1	1	1	1	1	1	1

1. 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.
2. The differential I/O per bank includes both dedicated LVDS and emulated LVDS pin pairs. Please see the Logic Signal Connections table for more information.

Power Supply and NC Connections

Signals	100 TQFP	144 TQFP	208 PQFP	256 fpBGA	388 fpBGA	484 fpBGA
V_{CC}	28, 77	14, 39, 73, 112	19, 35, 53, 80, 107, 151, 158, 182	D4, D13, E5, E12, M5, M12, N4, N13	H9, J8, J15, K8, K15, L8, L15, M8, M15, N8, N15, P8, P15, R9	F10, F13, G9, G10, G13, G14, H8, H15, J7, J16, K6, K7, K16, K17, N6, N7, N16, N17, P7, P16, R8, R15, T9, T10, T13, T14, U10, U13
V_{CCIO0}	94	133	189, 199	F7, F8	G8, G9, G10, G11, H8	F11, G11, H10, H11
V_{CCIO1}	82	119	167, 177	F9, F10	G12, G13, G14, G15, H15	F12, G12, H12, H13
V_{CCIO2}	65	98	140, 149	G11, H11	H16, J16, K16, L16	K15, L15, L16, L17
V_{CCIO3}	58	88	115, 125	J11, K11	M16, N16, P16, R16	M15, M16, M17, N15
V_{CCIO4}	47	61, 68	87, 97	L9, L10	R15, T12, T13, T14, T15	R12, R13, T12, U12
V_{CCIO5}	38	49	64, 74	L7, L8	R8, T8, T9, T10, T11	R10, R11, T11, U11
V_{CCIO6}	22	21	28, 41	J6, K6	M7, N7, P7, R7	M6, M7, M8, N8
V_{CCIO7}	7	8	13, 23	G6, H6	H7, J7, K7, L7	K8, L6, L7, L8
V_{CCJ}	73	108	154	D16	E20	E20
V_{CCP0}	17	19	25	H4	M2	L5
V_{CCP1}	60	91	128	J12	M21	L18
V_{CCAUX}	25, 71	36, 106	50, 152	E4, E13, M4, M13	G7, G16, T7, T16	G7, G8, G15, G16, H7, H16, R7, R16, T7, T8, T15, T16
GND[1]	10, 18, 21, 33, 43, 44, 52, 59, 68, 84, 90, 99	3, 11, 20, 28, 44, 54, 56, 64, 75, 85, 90, 101, 121, 127, 136	5, 7, 16, 26, 38, 47, 49, 59, 69, 79, 82, 92, 106, 109, 118, 121, 127, 130, 135, 143, 163, 172, 181, 184, 194, 207	A1, A16, F6, F11, G7, G8, G9, G10, H5, H7, H8, H9, H10, J7, J8, J9, J10, J13, K7, K8, K9, K10, L6, L11, T1, T16	A1, A22, H10, H11, H12, H13, H14, J9, J10, J11, J12, J13, J14, K9, K10, K11, K12, K13, K14, L9, L10, L11, L12, L13, L14, M9, M10, M11, M12, M13, M14, N1, N9, N10, N11, N12, N13, N14, N22, P9, P10, P11, P12, P13, P14, R10, R11, R12, R13, R14, AB1, AB22	A1, A2, A21, A22, B1, B22, H9, H14, J8, J9, J10, J11, J12, J13, J14, J15, K9, K10, K11, K12, K13, K14, L9, L10, L11, L12, L13, L14, M9, M10, M11, M12, M13, M14, M20, N2, N9, N10, N11, N12, N13, N14, P8, P9, P10, P11, P12, P13, P14, P15, R9, R14, AA1, AA22, AB1, AB2, AB21, AB22
NC[2]	—	—	XP3: 27, 33, 34, 129, 133, 134	—	XP10: C2, C15, C16, C17, D4, D5, D6, D7, D16, D17, E4, E19, W3, W4, W7, W17, W18, W19, W20, Y3, Y15, Y16, AA1, AA2	XP15: B21, C4, C5, C6, C18, C19, C20, C21, D6, D18, E4, E6, E18, F6, L1, L19, L20, M1, M2, M19, M21, N1, N21, N22, P1, P2, U5, U6, U17, U18, V5, V6, V17, V18, W17, W18, W19, Y3, Y4, Y5

1. All grounds must be electrically connected at the board level.
2. NC pins should not be connected to any active signals, V_{CC} or GND.

LFXP3 Logic Signal Connections: 100 TQFP

Pin Number	Pin Function	Bank	Differential	Dual Function
1	CFG1	0	-	-
2	DONE	0	-	-
3	PROGRAMN	7	-	-
4	CCLK	7	-	-
5	PL3A	7	T	LUM0_PLLT_FB_A
6	PL3B	7	C	LUM0_PLLC_FB_A
7	VCCIO7	7	-	-
8	PL5A	7	-	VREF1_7
9	PL6B	7	-	VREF2_7
10	GNDIO7	7	-	-
11	PL7A	7	T[3]	DQS
12	PL7B	7	C[3]	-
13	PL8A	7	T	LUM0_PLLT_IN_A
14	PL8B	7	C	LUM0_PLLC_IN_A
15	PL9A	7	T[3]	-
16	PL9B	7	C[3]	-
17	VCCP0	-	-	-
18	GNDP0	-	-	-
19	PL12A	6	T	PCLKT6_0
20	PL12B	6	C	PCLKC6_0
21	GNDIO6	6	-	-
22	VCCIO6	6	-	-
23	PL18A	6	T[3]	-
24	PL18B	6	C[3]	-
25	VCCAUX	-	-	-
26	SLEEPN[1]/TOE[2]	-	-	-
27	INITN	5	-	-
28	VCC	-	-	-
29	PB2B	5	-	VREF1_5
30	PB5B	5	-	VREF2_5
31	PB8A	5	T	-
32	PB8B	5	C	-
33	GNDIO5	5	-	-
34	PB9A	5	-	-
35	PB10B	5	-	-
36	PB11A	5	T	DQS
37	PB11B	5	C	-
38	VCCIO5	5	-	-
39	PB12A	5	T	-
40	PB12B	5	C	-
41	PB13A	5	T	-
42	PB13B	5	C	-
43	GND	-	-	-

LFXP3 Logic Signal Connections: 100 TQFP (Cont.)

Pin Number	Pin Function	Bank	Differential	Dual Function
44	GNDIO4	4	-	-
45	PB15A	4	T	PCLKT4_0
46	PB15B	4	C	PCLKC4_0
47	VCCIO4	4	-	-
48	PB19A	4	T	DQS
49	PB19B	4	C	VREF1_4
50	PB24A	4	-	VREF2_4
51	PR18B	3	C[3]	-
52	GNDIO3	3	-	-
53	PR18A	3	T[3]	-
54	PR15B	3	-	VREF1_3
55	PR14A	3	-	VREF2_3
56	PR13B	3	C	-
57	PR13A	3	T	-
58	VCCIO3	3	-	-
59	GNDP1	-	-	-
60	VCCP1	-	-	-
61	PR9B	2	C	PCLKC2_0
62	PR9A	2	T	PCLKT2_0
63	PR8B	2	C	RUM0_PLLC_IN_A
64	PR8A	2	T	RUM0_PLLT_IN_A
65	VCCIO2	2	-	-
66	PR6B	2	-	VREF1_2
67	PR5A	2	-	VREF2_2
68	GNDIO2	2	-	-
69	PR3B	2	C	RUM0_PLLC_FB_A
70	PR3A	2	T	RUM0_PLLT_FB_A
71	VCCAUX	-	-	-
72	TDO	-	-	-
73	VCCJ	-	-	-
74	TDI	-	-	-
75	TMS	-	-	-
76	TCK	-	-	-
77	VCC	-	-	-
78	PT24A	1	-	-
79	PT23A	1	-	D0
80	PT22B	1	-	D1
81	PT21A	1	-	D2
82	VCCIO1	1	-	-
83	PT20B	1	-	D3
84	GNDIO1	1	-	-
85	PT17A	1	-	D4
86	PT16A	1	-	D5
87	PT15B	1	-	D6

LFXP3 Logic Signal Connections: 100 TQFP (Cont.)

Pin Number	Pin Function	Bank	Differential	Dual Function
88	PT14B	1	-	D7
89	PT13B	0	C	BUSY
90	GNDIO0	0	-	-
91	PT13A	0	T	CS1N
92	PT12B	0	C	PCLKC0_0
93	PT12A	0	T	PCLKT0_0
94	VCCIO0	0	-	-
95	PT9A	0	-	DOUT
96	PT8A	0	-	WRITEN
97	PT6A	0	-	DI
98	PT5A	0	-	CSN
99	GND	-	-	-
100	CFG0	0	-	-

1. Applies to LFXP “C” only.
2. Applies to LFXP “E” only.
3. Supports dedicated LVDS outputs.

LFXP3 & LFXP6 Logic Signal Connections: 144 TQFP

Pin Number	LFXP3				LFXP6			
	Pin Function	Bank	Differential	Dual Function	Pin Function	Bank	Differential	Dual Function
1	PROGRAMN	7	-	-	PROGRAMN	7	-	-
2	CCLK	7	-	-	CCLK	7	-	-
3	GND	-	-	-	GND	-	-	-
4	PL2A	7	T[3]	-	PL2A	7	T[3]	-
5	PL2B	7	C[3]	-	PL2B	7	C[3]	-
6	PL3A	7	T	LUM0_PLLT_FB_A	PL3A	7	T	LUM0_PLLT_FB_A
7	PL3B	7	C	LUM0_PLLC_FB_A	PL3B	7	C	LUM0_PLLC_FB_A
8	VCCIO7	7	-	-	VCCIO7	7	-	-
9	PL5A	7	-	VREF1_7	PL5A	7	-	VREF1_7
10	PL6B	7	-	VREF2_7	PL6B	7	-	VREF2_7
11	GNDIO7	7	-	-	GNDIO7	7	-	-
12	PL7A	7	T[3]	DQS	PL7A	7	T[3]	DQS
13	PL7B	7	C[3]	-	PL7B	7	C[3]	-
14	VCC	-	-	-	VCC	-	-	-
15	PL8A	7	T	LUM0_PLLT_IN_A	PL8A	7	T	LUM0_PLLT_IN_A
16	PL8B	7	C	LUM0_PLLC_IN_A	PL8B	7	C	LUM0_PLLC_IN_A
17	PL9A	7	T[3]	-	PL9A	7	T[3]	-
18	PL9B	7	C[3]	-	PL9B	7	C[3]	-
19	VCCP0	-	-	-	VCCP0	-	-	-
20	GNDP0	-	-	-	GNDP0	-	-	-
21	VCCIO6	6	-	-	VCCIO6	6	-	-
22	PL11A	6	T[3]	-	PL16A	6	T[3]	-
23	PL11B	6	C[3]	-	PL16B	6	C[3]	-
24	PL12A	6	T	PCLKT6_0	PL17A	6	T	PCLKT6_0
25	PL12B	6	C	PCLKC6_0	PL17B	6	C	PCLKC6_0
26	PL13A	6	T[3]	-	PL18A	6	T[3]	-
27	PL13B	6	C[3]	-	PL18B	6	C[3]	-
28	GNDIO6	6	-	-	GNDIO6	6	-	-
29	PL14A	6	-	VREF1_6	PL22A	6	-	VREF1_6
30	PL15B	6	-	VREF2_6	PL23B	6	-	VREF2_6
31	PL16A	6	T[3]	DQS	PL24A	6	T[3]	DQS
32	PL16B	6	C[3]	-	PL24B	6	C[3]	-
33	PL17A	6	-	-	PL25A	6	-	-
34	PL18A	6	T[3]	-	PL26A	6	T[3]	-
35	PL18B	6	C[3]	-	PL26B	6	C[3]	-
36	VCCAUX	-	-	-	VCCAUX	-	-	-
37	SLEEPN[1]/TOE[2]	-	-	-	SLEEPN[1]/TOE[2]	-	-	-
38	INITN	5	-	-	INITN	5	-	-
39	VCC	-	-	-	VCC	-	-	-
40	PB2B	5	-	VREF1_5	PB5B	5	-	VREF1_5
41	PB5B	5	-	VREF2_5	PB8B	5	-	VREF2_5
42	PB7A	5	T	-	PB10A	5	T	-
43	PB7B	5	C	-	PB10B	5	C	-
44	GNDIO5	5	-	-	GNDIO5	5	-	-
45	PB9A	5	-	-	PB12A	5	-	-
46	PB10B	5	-	-	PB13B	5	-	-

LFXP3 & LFXP6 Logic Signal Connections: 144 TQFP (Cont.)

Pin Number	LFXP3				LFXP6			
	Pin Function	Bank	Differential	Dual Function	Pin Function	Bank	Differential	Dual Function
47	PB11A	5	T	DQS	PB14A	5	T	DQS
48	PB11B	5	C	-	PB14B	5	C	-
49	VCCIO5	5	-	-	VCCIO5	5	-	-
50	PB12A	5	T	-	PB15A	5	T	-
51	PB12B	5	C	-	PB15B	5	C	-
52	PB13A	5	T	-	PB16A	5	T	-
53	PB13B	5	C	-	PB16B	5	C	-
54	GND	-	-	-	GND	-	-	-
55	PB14A	4	T	-	PB17A	4	T	-
56	GNDIO4	4	-	-	GNDIO4	4	-	-
57	PB14B	4	C	-	PB17B	4	C	-
58	PB15A	4	T	PCLKT4_0	PB18A	4	T	PCLKT4_0
59	PB15B	4	C	PCLKC4_0	PB18B	4	C	PCLKC4_0
60	PB16A	4	T	-	PB19A	4	T	-
61	VCCIO4	4	-	-	VCCIO4	4	-	-
62	PB16B	4	C	-	PB19B	4	C	-
63	PB19A	4	T	DQS	PB22A	4	T	DQS
64	GNDIO4	4	-	-	GNDIO4	4	-	-
65	PB19B	4	C	VREF1_4	PB22B	4	C	VREF1_4
66	PB20A	4	T	-	PB23A	4	T	-
67	PB20B	4	C	-	PB23B	4	C	-
68	VCCIO4	4	-	-	VCCIO4	4	-	-
69	PB22A	4	-	-	PB25A	4	-	-
70	PB24A	4	T	VREF2_4	PB27A	4	T	VREF2_4
71	PB24B	4	C	-	PB27B	4	C	-
72	PB25A	4	-	-	PB28A	4	-	-
73	VCC	-	-	-	VCC	-	-	-
74	PR18B	3	C[3]	-	PR26B	3	C[3]	-
75	GNDIO3	3	-	-	GNDIO3	3	-	-
76	PR18A	3	T[3]	-	PR26A	3	T[3]	-
77	PR17B	3	C	-	PR25B	3	C	-
78	PR17A	3	T	-	PR25A	3	T	-
79	PR16B	3	C[3]	-	PR24B	3	C[3]	-
80	PR16A	3	T[3]	DQS	PR24A	3	T[3]	DQS
81	PR15B	3	-	VREF1_3	PR23B	3	-	VREF1_3
82	PR14A	3	-	VREF2_3	PR22A	3	-	VREF2_3
83	PR13B	3	C	-	PR21B	3	C[3]	-
84	PR13A	3	T	-	PR21A	3	T[3]	-
85	GND	-	-	-	GND	-	-	-
86	PR12A	3	-	-	PR20A	3	-	-
87	PR11B	3	C	-	PR19B	3	C[3]	-
88	VCCIO3	3	-	-	VCCIO3	3	-	-
89	PR11A	3	T	-	PR19A	3	T[3]	-
90	GNDP1	-	-	-	GNDP1	-	-	-
91	VCCP1	-	-	-	VCCP1	-	-	-
92	PR9B	2	C	PCLKC2_0	PR12B	2	C	PCLKC2_0

LFXP3 & LFXP6 Logic Signal Connections: 144 TQFP (Cont.)

Pin Number	LFXP3				LFXP6			
	Pin Function	Bank	Differential	Dual Function	Pin Function	Bank	Differential	Dual Function
93	PR9A	2	T	PCLKT2_0	PR12A	2	T	PCLKT2_0
94	PR8B	2	C	RUM0_PLLC_IN_A	PR8B	2	C	RUM0_PLLC_IN_A
95	PR8A	2	T	RUM0_PLLT_IN_A	PR8A	2	T	RUM0_PLLT_IN_A
96	PR7B	2	C[3]	-	PR7B	2	C[3]	-
97	PR7A	2	T[3]	DQS	PR7A	2	T[3]	DQS
98	VCCIO2	2	-	-	VCCIO2	2	-	-
99	PR6B	2	-	VREF1_2	PR6B	2	-	VREF1_2
100	PR5A	2	-	VREF2_2	PR5A	2	-	VREF2_2
101	GNDIO2	2	-	-	GNDIO2	2	-	-
102	PR3B	2	C	RUM0_PLLC_FB_A	PR3B	2	C	RUM0_PLLC_FB_A
103	PR3A	2	T	RUM0_PLLT_FB_A	PR3A	2	T	RUM0_PLLT_FB_A
104	PR2B	2	C[3]	-	PR2B	2	C[3]	-
105	PR2A	2	T[3]	-	PR2A	2	T[3]	-
106	VCCAUX	-	-	-	VCCAUX	-	-	-
107	TDO	-	-	-	TDO	-	-	-
108	VCCJ	-	-	-	VCCJ	-	-	-
109	TDI	-	-	-	TDI	-	-	-
110	TMS	-	-	-	TMS	-	-	-
111	TCK	-	-	-	TCK	-	-	-
112	VCC	-	-	-	VCC	-	-	-
113	PT25A	1	-	VREF1_1	PT28A	1	-	VREF1_1
114	PT24A	1	-	-	PT27A	1	-	-
115	PT23A	1	-	D0	PT26A	1	-	D0
116	PT22B	1	C	D1	PT25B	1	C	D1
117	PT22A	1	T	VREF2_1	PT25A	1	T	VREF2_1
118	PT21A	1	-	D2	PT24A	1	-	D2
119	VCCIO1	1	-	-	VCCIO1	1	-	-
120	PT20B	1	-	D3	PT23B	1	-	D3
121	GNDIO1	1	-	-	GNDIO1	1	-	-
122	PT17A	1	-	D4	PT20A	1	-	D4
123	PT16A	1	-	D5	PT19A	1	-	D5
124	PT15B	1	C	D6	PT18B	1	C	D6
125	PT15A	1	T	-	PT18A	1	T	-
126	PT14B	1	-	D7	PT17B	1	-	D7
127	GND	-	-	-	GND	-	-	-
128	PT13B	0	C	BUSY	PT16B	0	C	BUSY
129	PT13A	0	T	CS1N	PT16A	0	T	CS1N
130	PT12B	0	C	PCLKC0_0	PT15B	0	C	PCLKC0_0
131	PT12A	0	T	PCLKT0_0	PT15A	0	T	PCLKT0_0
132	PT11B	0	C	-	PT14B	0	C	-
133	VCCIO0	0	-	-	VCCIO0	0	-	-
134	PT11A	0	T	DQS	PT14A	0	T	DQS
135	PT9A	0	-	DOUT	PT12A	0	-	DOUT
136	GNDIO0	0	-	-	GNDIO0	0	-	-
137	PT8A	0	-	WRITEN	PT11A	0	-	WRITEN
138	PT7A	0	-	VREF1_0	PT10A	0	-	VREF1_0

LFXP3 & LFXP6 Logic Signal Connections: 144 TQFP (Cont.)

Pin Number	LFXP3				LFXP6			
	Pin Function	Bank	Differential	Dual Function	Pin Function	Bank	Differential	Dual Function
139	PT6A	0	-	DI	PT9A	0	-	DI
140	PT5A	0	-	CSN	PT8A	0	-	CSN
141	PT3B	0	-	VREF2_0	PT6B	0	-	VREF2_0
142	CFG0	0	-	-	CFG0	0	-	-
143	CFG1	0	-	-	CFG1	0	-	-
144	DONE	0	-	-	DONE	0	-	-

1. Applies to LFXP “C” only.
2. Applies to LFXP “E” only.
3. Supports dedicated LVDS outputs.

LFXP3 & LFXP6 Logic Signal Connections: 208 PQFP

Pin Number	LFXP3				LFXP6			
	Pin Function	Bank	Differential	Dual Function	Pin Function	Bank	Differential	Dual Function
1	CFG1	0	-	-	CFG1	0	-	-
2	DONE	0	-	-	DONE	0	-	-
3	PROGRAMN	7	-	-	PROGRAMN	7	-	-
4	CCLK	7	-	-	CCLK	7	-	-
5	GND	-	-	-	GND	-	-	-
6	PL2A	7	T[3]	-	PL2A	7	T[3]	-
7	GNDIO7	7	-	-	GNDIO7	7	-	-
8	PL2B	7	C[3]	-	PL2B	7	C[3]	-
9	PL3A	7	T	LUM0_PLLT_FB_A	PL3A	7	T	LUM0_PLLT_FB_A
10	PL3B	7	C	LUM0_PLLC_FB_A	PL3B	7	C	LUM0_PLLC_FB_A
11	PL4A	7	T[3]	-	PL4A	7	T[3]	-
12	PL4B	7	C[3]	-	PL4B	7	C[3]	-
13	VCCIO7	7	-	-	VCCIO7	7	-	-
14	PL5A	7	-	VREF1_7	PL5A	7	-	VREF1_7
15	PL6B	7	-	VREF2_7	PL6B	7	-	VREF2_7
16	GNDIO7	7	-	-	GNDIO7	7	-	-
17	PL7A	7	T[3]	DQS	PL7A	7	T[3]	DQS
18	PL7B	7	C[3]	-	PL7B	7	C[3]	-
19	VCC	-	-	-	VCC	-	-	-
20	PL8A	7	T	LUM0_PLLT_IN_A	PL8A	7	T	LUM0_PLLT_IN_A
21	PL8B	7	C	LUM0_PLLC_IN_A	PL8B	7	C	LUM0_PLLC_IN_A
22	PL9A	7	T[3]	-	PL9A	7	T[3]	-
23	VCCIO7	7	-	-	VCCIO7	7	-	-
24	PL9B	7	C[3]	-	PL9B	7	C[3]	-
25	VCCP0	-	-	-	VCCP0	-	-	-
26	GNDP0	-	-	-	GNDP0	-	-	-
27	NC	-	-	-	PL15B	6	-	-
28	VCCIO6	6	-	-	VCCIO6	6	-	-
29	PL11A	6	T[3]	-	PL16A	6	T[3]	-
30	PL11B	6	C[3]	-	PL16B	6	C[3]	-
31	PL12A	6	T	PCLKT6_0	PL17A	6	T	PCLKT6_0
32	PL12B	6	C	PCLKC6_0	PL17B	6	C	PCLKC6_0
33	NC	-	-	-	PL18A	6	T[3]	-
34	NC	-	-	-	PL18B	6	C[3]	-
35	VCC	-	-	-	VCC	-	-	-
36	PL13A	6	T[3]	-	PL21A	6	T[3]	-
37	PL13B	6	C[3]	-	PL21B	6	C[3]	-
38	GNDIO6	6	-	-	GNDIO6	6	-	-
39	PL14A	6	-	VREF1_6	PL22A	6	-	VREF1_6
40	PL15B	6	-	VREF2_6	PL23B	6	-	VREF2_6
41	VCCIO6	6	-	-	VCCIO6	6	-	-
42	PL16A	6	T[3]	DQS	PL24A	6	T[3]	DQS
43	PL16B	6	C[3]	-	PL24B	6	C[3]	-
44	PL17A	6	T	-	PL25A	6	T	-
45	PL17B	6	C	-	PL25B	6	C	-
46	PL18A	6	T[3]	-	PL26A	6	T[3]	-

LFXP3 & LFXP6 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFXP3				LFXP6			
	Pin Function	Bank	Differential	Dual Function	Pin Function	Bank	Differential	Dual Function
47	GNDIO6	6	-	-	GNDIO6	6	-	-
48	PL18B	6	C[3]	-	PL26B	6	C[3]	-
49	GND	-	-	-	GND	-	-	-
50	VCCAUX	-	-	-	VCCAUX	-	-	-
51	SLEEPN[1]/TOE[2]	-	-	-	SLEEPN[1]/TOE[2]	-	-	-
52	INITN	5	-	-	INITN	5	-	-
53	VCC	-	-	-	VCC	-	-	-
54	PB2B	5	-	VREF1_5	PB5B	5	-	VREF1_5
55	PB3A	5	T	-	PB6A	5	T	DQS
56	PB3B	5	C	-	PB6B	5	C	-
57	PB4A	5	T	-	PB7A	5	T	-
58	PB4B	5	C	-	PB7B	5	C	-
59	GNDIO5	5	-	-	GNDIO5	5	-	-
60	PB5A	5	T	-	PB8A	5	T	-
61	PB5B	5	C	VREF2_5	PB8B	5	C	VREF2_5
62	PB6A	5	T	-	PB9A	5	T	-
63	PB6B	5	C	-	PB9B	5	C	-
64	VCCIO5	5	-	-	VCCIO5	5	-	-
65	PB7A	5	T	-	PB10A	5	T	-
66	PB7B	5	C	-	PB10B	5	C	-
67	PB8A	5	T	-	PB11A	5	T	-
68	PB8B	5	C	-	PB11B	5	C	-
69	GNDIO5	5	-	-	GNDIO5	5	-	-
70	PB9A	5	-	-	PB12A	5	-	-
71	PB10B	5	-	-	PB13B	5	-	-
72	PB11A	5	T	DQS	PB14A	5	T	DQS
73	PB11B	5	C	-	PB14B	5	C	-
74	VCCIO5	5	-	-	VCCIO5	5	-	-
75	PB12A	5	T	-	PB15A	5	T	-
76	PB12B	5	C	-	PB15B	5	C	-
77	PB13A	5	T	-	PB16A	5	T	-
78	PB13B	5	C	-	PB16B	5	C	-
79	GND	-	-	-	GND	-	-	-
80	VCC	-	-	-	VCC	-	-	-
81	PB14A	4	T	-	PB17A	4	T	-
82	GNDIO4	4	-	-	GNDIO4	4	-	-
83	PB14B	4	C	-	PB17B	4	C	-
84	PB15A	4	T	PCLKT4_0	PB18A	4	T	PCLKT4_0
85	PB15B	4	C	PCLKC4_0	PB18B	4	C	PCLKC4_0
86	PB16A	4	T	-	PB19A	4	T	-
87	VCCIO4	4	-	-	VCCIO4	4	-	-
88	PB16B	4	C	-	PB19B	4	C	-
89	PB17A	4	-	-	PB20A	4	-	-
90	PB18B	4	-	-	PB21B	4	-	-
91	PB19A	4	T	DQS	PB22A	4	T	DQS
92	GNDIO4	4	-	-	GNDIO4	4	-	-

LFXP3 & LFXP6 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFXP3				LFXP6			
	Pin Function	Bank	Differential	Dual Function	Pin Function	Bank	Differential	Dual Function
93	PB19B	4	C	VREF1_4	PB22B	4	C	VREF1_4
94	PB20A	4	T	-	PB23A	4	T	-
95	PB20B	4	C	-	PB23B	4	C	-
96	PB21A	4	T	-	PB24A	4	T	-
97	VCCIO4	4	-	-	VCCIO4	4	-	-
98	PB21B	4	C	-	PB24B	4	C	-
99	PB22A	4	T	-	PB25A	4	T	-
100	PB22B	4	C	-	PB25B	4	C	-
101	PB23A	4	T	-	PB26A	4	T	-
102	PB23B	4	C	-	PB26B	4	C	-
103	PB24A	4	T	VREF2_4	PB27A	4	-	VREF2_4
104	PB24B	4	C	-	PB30A	4	T	DQS
105	PB25A	4	-	-	PB30B	4	C	-
106	GND	-	-	-	GND	-	-	-
107	VCC	-	-	-	VCC	-	-	-
108	PR18B	3	C[3]	-	PR26B	3	C[3]	-
109	GNDIO3	3	-	-	GNDIO3	3	-	-
110	PR18A	3	T[3]	-	PR26A	3	T[3]	-
111	PR17B	3	C	-	PR25B	3	C	-
112	PR17A	3	T	-	PR25A	3	T	-
113	PR16B	3	C[3]	-	PR24B	3	C[3]	-
114	PR16A	3	T[3]	DQS	PR24A	3	T[3]	DQS
115	VCCIO3	3	-	-	VCCIO3	3	-	-
116	PR15B	3	-	VREF1_3	PR23B	3	-	VREF1_3
117	PR14A	3	-	VREF2_3	PR22A	3	-	VREF2_3
118	GNDIO3	3	-	-	GNDIO3	3	-	-
119	PR13B	3	C	-	PR21B	3	C[3]	-
120	PR13A	3	T	-	PR21A	3	T[3]	-
121	GND	-	-	-	GND	-	-	-
122	PR12B	3	C	-	PR20B	3	C	-
123	PR12A	3	T	-	PR20A	3	T	-
124	PR11B	3	C	-	PR19B	3	C[3]	-
125	VCCIO3	3	-	-	VCCIO3	3	-	-
126	PR11A	3	T	-	PR19A	3	T[3]	-
127	GNDP1	-	-	-	GNDP1	-	-	-
128	VCCP1	-	-	-	VCCP1	-	-	-
129	NC	-	-	-	PR13A	2	-	-
130	GND	-	-	-	GND	-	-	-
131	PR9B	2	C	PCLKC2_0	PR12B	2	C	PCLKC2_0
132	PR9A	2	T	PCLKT2_0	PR12A	2	T	PCLKT2_0
133	NC	-	-	-	PR11B	2	C[3]	-
134	NC	-	-	-	PR11A	2	T[3]	-
135	GNDIO2	2	-	-	GNDIO2	2	-	-
136	PR8B	2	C	RUM0_PLLC_IN_A	PR8B	2	C	RUM0_PLLC_IN_A
137	PR8A	2	T	RUM0_PLLT_IN_A	PR8A	2	T	RUM0_PLLT_IN_A
138	PR7B	2	C[3]	-	PR7B	2	C[3]	-

LFXP3 & LFXP6 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFXP3				LFXP6			
	Pin Function	Bank	Differential	Dual Function	Pin Function	Bank	Differential	Dual Function
139	PR7A	2	T[3]	DQS	PR7A	2	T[3]	DQS
140	VCCIO2	2	-	-	VCCIO2	2	-	-
141	PR6B	2	-	VREF1_2	PR6B	2	-	VREF1_2
142	PR5A	2	-	VREF2_2	PR5A	2	-	VREF2_2
143	GNDIO2	2	-	-	GNDIO2	2	-	-
144	PR4B	2	C[3]	-	PR4B	2	C[3]	-
145	PR4A	2	T[3]	-	PR4A	2	T[3]	-
146	PR3B	2	C	RUM0_PLLC_FB_A	PR3B	2	C	RUM0_PLLC_FB_A
147	PR3A	2	T	RUM0_PLLT_FB_A	PR3A	2	T	RUM0_PLLT_FB_A
148	PR2B	2	C[3]	-	PR2B	2	C[3]	-
149	VCCIO2	2	-	-	VCCIO2	2	-	-
150	PR2A	2	T[3]	-	PR2A	2	T[3]	-
151	VCC	-	-	-	VCC	-	-	-
152	VCCAUX	-	-	-	VCCAUX	-	-	-
153	TDO	-	-	-	TDO	-	-	-
154	VCCJ	-	-	-	VCCJ	-	-	-
155	TDI	-	-	-	TDI	-	-	-
156	TMS	-	-	-	TMS	-	-	-
157	TCK	-	-	-	TCK	-	-	-
158	VCC	-	-	-	VCC	-	-	-
159	PT25A	1	-	VREF1_1	PT28A	1	-	VREF1_1
160	PT24B	1	C	-	PT27B	1	C	-
161	PT24A	1	T	-	PT27A	1	T	-
162	PT23A	1	-	D0	PT26A	1	-	D0
163	GNDIO1	1	-	-	GNDIO1	1	-	-
164	PT22B	1	C	D1	PT25B	1	C	D1
165	PT22A	1	T	VREF2_1	PT25A	1	T	VREF2_1
166	PT21A	1	-	D2	PT24A	1	-	D2
167	VCCIO1	1	-	-	VCCIO1	1	-	-
168	PT20B	1	C	D3	PT23B	1	C	D3
169	PT20A	1	T	-	PT23A	1	T	-
170	PT19B	1	C	-	PT22B	1	C	-
171	PT19A	1	T	DQS	PT22A	1	T	DQS
172	GNDIO1	1	-	-	GNDIO1	1	-	-
173	PT18B	1	-	-	PT21B	1	-	-
174	PT17A	1	-	D4	PT20A	1	-	D4
175	PT16B	1	C	-	PT19B	1	C	-
176	PT16A	1	T	D5	PT19A	1	T	D5
177	VCCIO1	1	-	-	VCCIO1	1	-	-
178	PT15B	1	C	D6	PT18B	1	C	D6
179	PT15A	1	T	-	PT18A	1	T	-
180	PT14B	1	-	D7	PT17B	1	-	D7
181	GND	-	-	-	GND	-	-	-
182	VCC	-	-	-	VCC	-	-	-
183	PT13B	0	C	BUSY	PT16B	0	C	BUSY
184	GNDIO0	0	-	-	GNDIO0	0	-	-

LFXP3 & LFXP6 Logic Signal Connections: 208 PQFP (Cont.)

Pin Number	LFXP3				LFXP6			
	Pin Function	Bank	Differential	Dual Function	Pin Function	Bank	Differential	Dual Function
185	PT13A	0	T	CS1N	PT16A	0	T	CS1N
186	PT12B	0	C	PCLKC0_0	PT15B	0	C	PCLKC0_0
187	PT12A	0	T	PCLKT0_0	PT15A	0	T	PCLKT0_0
188	PT11B	0	C	-	PT14B	0	C	-
189	VCCIO0	0	-	-	VCCIO0	0	-	-
190	PT11A	0	T	DQS	PT14A	0	T	DQS
191	PT10B	0	-	-	PT13B	0	-	-
192	PT9A	0	-	DOUT	PT12A	0	-	DOUT
193	PT8B	0	C	-	PT11B	0	C	-
194	GNDIO0	0	-	-	GNDIO0	0	-	-
195	PT8A	0	T	WRITEN	PT11A	0	T	WRITEN
196	PT7B	0	C	-	PT10B	0	C	-
197	PT7A	0	T	VREF1_0	PT10A	0	T	VREF1_0
198	PT6B	0	C	-	PT9B	0	C	-
199	VCCIO0	0	-	-	VCCIO0	0	-	-
200	PT6A	0	T	DI	PT9A	0	T	DI
201	PT5B	0	C	-	PT8B	0	C	-
202	PT5A	0	T	CSN	PT8A	0	T	CSN
203	PT4B	0	C	-	PT7B	0	C	-
204	PT4A	0	T	-	PT7A	0	T	-
205	PT3B	0	-	VREF2_0	PT6B	0	-	VREF2_0
206	PT2B	0	-	-	PT5B	0	-	-
207	GND	-	-	-	GND	-	-	-
208	CFG0	0	-	-	CFG0	0	-	-

1. Applies to LFXP “C” only.
2. Applies to LFXP “E” only.
3. Supports dedicated LVDS outputs.

LFXP6 & LFXP10 Logic Signal Connections: 256 fpBGA

Ball Number	LFXP6				LFXP10			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
C2	PROGRAMN	7	-	-	PROGRAMN	7	-	-
C1	CCLK	7	-	-	CCLK	7	-	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
D2	PL3A	7	T	LUM0_PLLT_FB_A	PL3A	7	T	LUM0_PLLT_FB_A
D3	PL3B	7	C	LUM0_PLLC_FB_A	PL3B	7	C	LUM0_PLLC_FB_A
D1	PL2A	7	T[3]	-	PL5A	7	-	-
E2	PL5A	7	-	VREF1_7	PL6B	7	-	VREF1_7
-	GNDIO7	7	-	-	GNDIO7	7	-	-
E1	PL7A	7	T[3]	DQS	PL7A	7	T[3]	DQS
F1	PL7B	7	C[3]	-	PL7B	7	C[3]	-
E3	PL12A	7	T	-	PL8A	7	T	-
F4	PL12B	7	C	-	PL8B	7	C	-
F3	PL4A	7	T[3]	-	PL9A	7	T[3]	-
F2	PL4B	7	C[3]	-	PL9B	7	C[3]	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
G1	PL2B	7	C[3]	-	PL11B	7	-	-
G3	PL8A	7	T	LUM0_PLLT_IN_A	PL12A	7	T	LUM0_PLLT_IN_A
G2	PL8B	7	C	LUM0_PLLC_IN_A	PL12B	7	C	LUM0_PLLC_IN_A
H1	PL9A	7	T[3]	-	PL13A	7	T[3]	-
H2	PL9B	7	C[3]	-	PL13B	7	C[3]	-
G4	PL6B	7	-	VREF2_7	PL14A	7	-	VREF2_7
G5	PL14A	7	-	-	PL15B	7	-	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
J1	PL11A	7	T[3]	-	PL16A	7	T[3]	DQS
J2	PL11B	7	C[3]	-	PL16B	7	C[3]	-
H3	PL13A	7	T[3]	-	PL18A	7	T[3]	-
J3	PL13B	7	C[3]	-	PL18B	7	C[3]	-
H4	VCCP0	-	-	-	VCCP0	-	-	-
H5	GNDP0	-	-	-	GNDP0	-	-	-
K1	PL17A	6	T	PCLKT6_0	PL20A	6	T	PCLKT6_0
K2	PL17B	6	C	PCLKC6_0	PL20B	6	C	PCLKC6_0
-	GNDIO6	6	-	-	GNDIO6	6	-	-
J4	PL15B	6	-	-	PL22A	6	-	-
J5	PL22A	6	-	VREF1_6	PL23B	6	-	VREF1_6
L1	PL16A	6	T[3]	-	PL24A	6	T[3]	DQS
L2	PL16B	6	C[3]	-	PL24B	6	C[3]	-
M1	PL18A	6	T[3]	-	PL25A	6	T	LLM0_PLLT_IN_A
M2	PL18B	6	C[3]	-	PL25B	6	C	LLM0_PLLC_IN_A
K3	PL19A	6	T[3]	-	PL26A	6	T[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
L3	PL19B	6	C[3]	-	PL26B	6	C[3]	-
L4	PL21A	6	T[3]	-	PL28A	6	-	-

LFXP6 & LFXP10 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP6				LFXP10			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
K4	PL20A	6	T	-	PL29A	6	T	-
K5	PL20B	6	C	-	PL29B	6	C	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
N1	PL23B	6	-	VREF2_6	PL31A	6	-	VREF2_6
N2	PL21B	6	C[3]	-	PL32B	6	-	-
P1	PL24A	6	T[3]	DQS	PL33A	6	T[3]	DQS
P2	PL24B	6	C[3]	-	PL33B	6	C[3]	-
L5	PL25A	6	T	-	PL34A	6	T	LLM0_PLLT_FB_A
M6	PL25B	6	C	-	PL34B	6	C	LLM0_PLLC_FB_A
M3	PL26A	6	T[3]	-	PL35A	6	T[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
N3	PL26B	6	C[3]	-	PL35B	6	C[3]	-
P4	SLEEPN[1]/TOE[2]	-	-	-	SLEEPN[1]/TOE[2]	-	-	-
P3	INITN	5	-	-	INITN	5	-	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
R4	PB2A	5	T	-	PB6A	5	T	-
N5	PB2B	5	C	-	PB6B	5	C	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
P5	PB5B	5	-	VREF1_5	PB7A	5	T	VREF1_5
R1	PB3B	5	C	-	PB7B	5	C	-
N6	PB4A	5	-	-	PB8A	5	-	-
M7	PB3A	5	T	-	PB9B	5	-	-
R2	PB6A	5	T	DQS	PB10A	5	T	DQS
T2	PB6B	5	C	-	PB10B	5	C	-
R3	PB7A	5	T	-	PB11A	5	T	-
T3	PB7B	5	C	-	PB11B	5	C	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
T4	PB8A	5	T	-	PB12A	5	T	-
R5	PB8B	5	C	VREF2_5	PB12B	5	C	VREF2_5
N7	PB9A	5	T	-	PB13A	5	T	-
M8	PB9B	5	C	-	PB13B	5	C	-
T5	PB10A	5	T	-	PB14A	5	T	-
P6	PB10B	5	C	-	PB14B	5	C	-
T6	PB11A	5	T	-	PB15A	5	T	-
R6	PB11B	5	C	-	PB15B	5	C	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
P7	PB12A	5	-	-	PB16A	5	-	-
N8	PB13B	5	-	-	PB17B	5	-	-
R7	PB14A	5	T	DQS	PB18A	5	T	DQS
T7	PB14B	5	C	-	PB18B	5	C	-
P8	PB15A	5	T	-	PB19A	5	T	-
T8	PB15B	5	C	-	PB19B	5	C	-

LFXP6 & LFXP10 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP6				LFXP10			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
R8	PB16A	5	T	-	PB20A	5	T	-
T9	PB16B	5	C	-	PB20B	5	C	-
R9	PB17A	4	T	-	PB21A	4	T	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
P9	PB17B	4	C	-	PB21B	4	C	-
T10	PB18A	4	T	PCLKT4_0	PB22A	4	T	PCLKT4_0
T11	PB18B	4	C	PCLKC4_0	PB22B	4	C	PCLKC4_0
R10	PB19A	4	T	-	PB23A	4	T	-
P10	PB19B	4	C	-	PB23B	4	C	-
N9	PB20A	4	-	-	PB24A	4	-	-
M9	PB21B	4	-	-	PB25B	4	-	-
R12	PB22A	4	T	DQS	PB26A	4	T	DQS
-	GNDIO4	4	-	-	GNDIO4	4	-	-
T12	PB22B	4	C	VREF1_4	PB26B	4	C	VREF1_4
P13	PB23A	4	T	-	PB27A	4	T	-
R13	PB23B	4	C	-	PB27B	4	C	-
M11	PB24A	4	T	-	PB28A	4	T	-
N11	PB24B	4	C	-	PB28B	4	C	-
N10	PB25A	4	T	-	PB29A	4	T	-
M10	PB25B	4	C	-	PB29B	4	C	-
T13	PB26A	4	T	-	PB30A	4	T	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
P14	PB26B	4	C	-	PB30B	4	C	-
R11	PB27A	4	T	VREF2_4	PB31A	4	T	VREF2_4
P12	PB27B	4	C	-	PB31B	4	C	-
T14	PB28A	4	-	-	PB32A	4	-	-
R14	PB29B	4	-	-	PB33B	4	-	-
P11	PB30A	4	T	DQS	PB34A	4	T	DQS
N12	PB30B	4	C	-	PB34B	4	C	-
T15	PB31A	4	T	-	PB35A	4	T	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
R15	PB31B	4	C	-	PB35B	4	C	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-
P15	PR26B	3	C[3]	-	PR34B	3	C	RLM0_PLLC_FB_A
N15	PR26A	3	T[3]	-	PR34A	3	T	RLM0_PLLT_FB_A
P16	PR24B	3	C[3]	-	PR33B	3	C[3]	-
R16	PR24A	3	T[3]	DQS	PR33A	3	T[3]	DQS
M15	PR15B	3	-	-	PR32B	3	-	-
N14	PR23B	3	-	VREF1_3	PR31A	3	-	VREF1_3
-	GNDIO3	3	-	-	GNDIO3	3	-	-
M14	PR25B	3	C	-	PR29B	3	C	-
L13	PR25A	3	T	-	PR29A	3	T	-

LFXP6 & LFXP10 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP6				LFXP10			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
L15	PR21B	3	C[3]	-	PR28B	3	C[3]	-
L14	PR21A	3	T[3]	-	PR28A	3	T[3]	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-
L12	PR17B	3	C	-	PR26A	3	-	-
M16	PR20B	3	C	-	PR25B	3	C	RLM0_PLLC_IN_A
N16	PR20A	3	T	-	PR25A	3	T	RLM0_PLLT_IN_A
K14	PR19B	3	C[3]	-	PR24B	3	C[3]	-
K15	PR19A	3	T[3]	-	PR24A	3	T[3]	DQS
K12	PR17A	3	T	-	PR23B	3	-	-
K13	PR22A	3	-	VREF2_3	PR22A	3	-	VREF2_3
-	GNDIO3	3	-	-	GNDIO3	3	-	-
L16	PR18B	3	C[3]	-	PR21B	3	C[3]	-
K16	PR18A	3	T[3]	-	PR21A	3	T[3]	-
J15	PR16B	3	C[3]	-	PR19B	3	C[3]	-
J14	PR16A	3	T[3]	-	PR19A	3	T[3]	-
J13	GNDP1	-	-	-	GNDP1	-	-	-
J12	VCCP1	-	-	-	VCCP1	-	-	-
-	GNDIO2	2	-	-	GNDIO2	2	-	-
J16	PR12B	2	C	PCLKC2_0	PR17B	2	C	PCLKC2_0
H16	PR12A	2	T	PCLKT2_0	PR17A	2	T	PCLKT2_0
H13	PR13B	2	C[3]	-	PR16B	2	C[3]	-
H12	PR13A	2	T[3]	-	PR16A	2	T[3]	DQS
H15	PR2B	2	C[3]	-	PR15B	2	-	-
H14	PR6B	2	-	VREF1_2	PR14A	2	-	VREF1_2
-	GNDIO2	2	-	-	GNDIO2	2	-	-
G15	PR11B	2	C[3]	-	PR13B	2	C[3]	-
G14	PR11A	2	T[3]	-	PR13A	2	T[3]	-
G16	PR8B	2	C	RUM0_PLLC_IN_A	PR12B	2	C	RUM0_PLLC_IN_A
F16	PR8A	2	T	RUM0_PLLT_IN_A	PR12A	2	T	RUM0_PLLT_IN_A
G13	PR2A	2	T[3]	-	PR11B	2	-	-
-	GNDIO2	2	-	-	GNDIO2	2	-	-
G12	PR9B	2	C[3]	-	PR8B	2	C	-
F13	PR9A	2	T[3]	-	PR8A	2	T	-
B16	PR7B	2	C[3]	-	PR7B	2	C[3]	-
C16	PR7A	2	T[3]	DQS	PR7A	2	T[3]	DQS
F15	PR14A	2	-	-	PR6B	2	-	-
E15	PR5A	2	-	VREF2_2	PR5A	2	-	VREF2_2
-	GNDIO2	2	-	-	GNDIO2	2	-	-
F14	PR4B	2	C[3]	-	PR4B	2	C[3]	-
E14	PR4A	2	T[3]	-	PR4A	2	T[3]	-
D15	PR3B	2	C	RUM0_PLLC_FB_A	PR3B	2	C	RUM0_PLLC_FB_A
C15	PR3A	2	T	RUM0_PLLT_FB_A	PR3A	2	T	RUM0_PLLT_FB_A

LFXP6 & LFXP10 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP6				LFXP10			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
E16	TDO	-	-	-	TDO	-	-	-
D16	VCCJ	-	-	-	VCCJ	-	-	-
D14	TDI	-	-	-	TDI	-	-	-
C14	TMS	-	-	-	TMS	-	-	-
B14	TCK	-	-	-	TCK	-	-	-
-	GNDIO1	1	-	-	GNDIO1	1	-	-
A15	PT31B	1	C	-	PT35B	1	C	-
B15	PT31A	1	T	-	PT35A	1	T	-
-	GNDIO1	1	-	-	GNDIO1	1	-	-
D12	PT28A	1	-	VREF1_1	PT34B	1	C	VREF1_1
C11	PT30A	1	T	DQS	PT34A	1	T	DQS
A14	PT29B	1	-	-	PT33B	1	-	-
B13	PT30B	1	C	-	PT32A	1	-	-
F12	PT27B	1	C	-	PT31B	1	C	-
E11	PT27A	1	T	-	PT31A	1	T	-
A13	PT26B	1	C	-	PT30B	1	C	-
C13	PT26A	1	T	D0	PT30A	1	T	D0
-	GNDIO1	1	-	-	GNDIO1	1	-	-
C10	PT25B	1	C	D1	PT29B	1	C	D1
E10	PT25A	1	T	VREF2_1	PT29A	1	T	VREF2_1
A12	PT24B	1	C	-	PT28B	1	C	-
B12	PT24A	1	T	D2	PT28A	1	T	D2
C12	PT23B	1	C	D3	PT27B	1	C	D3
A11	PT23A	1	T	-	PT27A	1	T	-
B11	PT22B	1	C	-	PT26B	1	C	-
D11	PT22A	1	T	DQS	PT26A	1	T	DQS
-	GNDIO1	1	-	-	GNDIO1	1	-	-
B9	PT21B	1	-	-	PT25B	1	-	-
D9	PT20A	1	-	D4	PT24A	1	-	D4
A10	PT19B	1	C	-	PT23B	1	C	-
B10	PT19A	1	T	D5	PT23A	1	T	D5
D10	PT18B	1	C	D6	PT22B	1	C	D6
A9	PT18A	1	T	-	PT22A	1	T	-
C9	PT17B	1	C	D7	PT21B	1	C	D7
C8	PT17A	1	T	-	PT21A	1	T	-
E9	PT16B	0	C	BUSY	PT20B	0	C	BUSY
-	GNDIO0	0	-	-	GNDIO0	0	-	-
B8	PT16A	0	T	CS1N	PT20A	0	T	CS1N
A8	PT15B	0	C	PCLKC0_0	PT19B	0	C	PCLKC0_0
A7	PT15A	0	T	PCLKT0_0	PT19A	0	T	PCLKT0_0
B7	PT14B	0	C	-	PT18B	0	C	-
C7	PT14A	0	T	DQS	PT18A	0	T	DQS

LFXP6 & LFXP10 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP6				LFXP10			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
E8	PT13B	0	-	-	PT17B	0	-	-
D8	PT12A	0	-	DOUT	PT16A	0	-	DOUT
A6	PT11B	0	C	-	PT15B	0	C	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
C6	PT11A	0	T	WRITEN	PT15A	0	T	WRITEN
E7	PT10B	0	C	-	PT14B	0	C	-
D7	PT10A	0	T	VREF1_0	PT14A	0	T	VREF1_0
A5	PT9B	0	C	-	PT13B	0	C	-
B5	PT9A	0	T	DI	PT13A	0	T	DI
A4	PT8B	0	C	-	PT12B	0	C	-
B6	PT8A	0	T	CSN	PT12A	0	T	CSN
E6	PT7B	0	C	-	PT11B	0	C	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
D6	PT7A	0	T	-	PT11A	0	T	-
D5	PT6B	0	C	VREF2_0	PT10B	0	C	VREF2_0
A3	PT6A	0	T	DQS	PT10A	0	T	DQS
B3	PT5B	0	-	-	PT9B	0	-	-
B2	PT4A	0	-	-	PT8A	0	-	-
A2	PT3B	0	C	-	PT7B	0	C	-
B1	PT3A	0	T	-	PT7A	0	T	-
F5	PT2B	0	C	-	PT6B	0	C	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
C5	PT2A	0	T	-	PT6A	0	T	-
C4	CFG0	0	-	-	CFG0	0	-	-
B4	CFG1	0	-	-	CFG1	0	-	-
C3	DONE	0	-	-	DONE	0	-	-
A1	GND	-	-	-	GND	-	-	-
A16	GND	-	-	-	GND	-	-	-
F11	GND	-	-	-	GND	-	-	-
F6	GND	-	-	-	GND	-	-	-
G10	GND	-	-	-	GND	-	-	-
G7	GND	-	-	-	GND	-	-	-
G8	GND	-	-	-	GND	-	-	-
G9	GND	-	-	-	GND	-	-	-
H10	GND	-	-	-	GND	-	-	-
H7	GND	-	-	-	GND	-	-	-
H8	GND	-	-	-	GND	-	-	-
H9	GND	-	-	-	GND	-	-	-
J10	GND	-	-	-	GND	-	-	-
J7	GND	-	-	-	GND	-	-	-
J8	GND	-	-	-	GND	-	-	-
J9	GND	-	-	-	GND	-	-	-

LFXP6 & LFXP10 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP6				LFXP10			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
K10	GND	-	-	-	GND	-	-	-
K7	GND	-	-	-	GND	-	-	-
K8	GND	-	-	-	GND	-	-	-
K9	GND	-	-	-	GND	-	-	-
L11	GND	-	-	-	GND	-	-	-
L6	GND	-	-	-	GND	-	-	-
T1	GND	-	-	-	GND	-	-	-
T16	GND	-	-	-	GND	-	-	-
D13	VCC	-	-	-	VCC	-	-	-
D4	VCC	-	-	-	VCC	-	-	-
E12	VCC	-	-	-	VCC	-	-	-
E5	VCC	-	-	-	VCC	-	-	-
M12	VCC	-	-	-	VCC	-	-	-
M5	VCC	-	-	-	VCC	-	-	-
N13	VCC	-	-	-	VCC	-	-	-
N4	VCC	-	-	-	VCC	-	-	-
E13	VCCAUX	-	-	-	VCCAUX	-	-	-
E4	VCCAUX	-	-	-	VCCAUX	-	-	-
M13	VCCAUX	-	-	-	VCCAUX	-	-	-
M4	VCCAUX	-	-	-	VCCAUX	-	-	-
F7	VCCIO0	0	-	-	VCCIO0	0	-	-
F8	VCCIO0	0	-	-	VCCIO0	0	-	-
F10	VCCIO1	1	-	-	VCCIO1	1	-	-
F9	VCCIO1	1	-	-	VCCIO1	1	-	-
G11	VCCIO2	2	-	-	VCCIO2	2	-	-
H11	VCCIO2	2	-	-	VCCIO2	2	-	-
J11	VCCIO3	3	-	-	VCCIO3	3	-	-
K11	VCCIO3	3	-	-	VCCIO3	3	-	-
L10	VCCIO4	4	-	-	VCCIO4	4	-	-
L9	VCCIO4	4	-	-	VCCIO4	4	-	-
L7	VCCIO5	5	-	-	VCCIO5	5	-	-
L8	VCCIO5	5	-	-	VCCIO5	5	-	-
J6	VCCIO6	6	-	-	VCCIO6	6	-	-
K6	VCCIO6	6	-	-	VCCIO6	6	-	-
G6	VCCIO7	7	-	-	VCCIO7	7	-	-
H6	VCCIO7	7	-	-	VCCIO7	7	-	-

1. Applies to LFXP “C” only.
2. Applies to LFXP “E” only.
3. Supports dedicated LVDS outputs.

LFXP15 & LFXP20 Logic Signal Connections: 256 fpBGA

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
C2	PROGRAMN	7	-	-	PROGRAMN	7	-	-
C1	CCLK	7	-	-	CCLK	7	-	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
D2	PL7A	7	T	LUM0_PLLT_FB_A	PL7A	7	T	LUM0_PLLT_FB_A
D3	PL7B	7	C	LUM0_PLLC_FB_A	PL7B	7	C	LUM0_PLLC_FB_A
D1	PL9A	7	-	-	PL9A	7	-	-
E2	PL10B	7	-	VREF1_7	PL10B	7	-	VREF1_7
E1	PL11A	7	T[3]	DQS	PL11A	7	T[3]	DQS
F1	PL11B	7	C[3]	-	PL11B	7	C[3]	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
E3	PL12A	7	T	-	PL12A	7	T	-
F4	PL12B	7	C	-	PL12B	7	C	-
F3	PL13A	7	T[3]	-	PL13A	7	T[3]	-
F2	PL13B	7	C[3]	-	PL13B	7	C[3]	-
G1	PL15B	7	-	-	PL15B	7	-	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
G3	PL16A	7	T	LUM0_PLLT_IN_A	PL16A	7	T	LUM0_PLLT_IN_A
G2	PL16B	7	C	LUM0_PLLC_IN_A	PL16B	7	C	LUM0_PLLC_IN_A
H1	PL17A	7	T[3]	-	PL17A	7	T[3]	-
H2	PL17B	7	C[3]	-	PL17B	7	C[3]	-
G4	PL18A	7	-	VREF2_7	PL18A	7	-	VREF2_7
G5	PL19B	7	-	-	PL19B	7	-	-
J1	PL20A	7	T[3]	DQS	PL20A	7	T[3]	DQS
-	GNDIO7	7	-	-	GNDIO7	7	-	-
J2	PL20B	7	C[3]	-	PL20B	7	C[3]	-
H3	PL22A	7	T[3]	-	PL22A	7	T[3]	-
J3	PL22B	7	C[3]	-	PL22B	7	C[3]	-
H4	VCCP0	-	-	-	VCCP0	-	-	-
H5	GNDP0	-	-	-	GNDP0	-	-	-
K1	PL24A	6	T	PCLKT6_0	PL28A	6	T	PCLKT6_0
-	GNDIO6	6	-	-	GNDIO6	6	-	-
K2	PL24B	6	C	PCLKC6_0	PL28B	6	C	PCLKC6_0
J4	PL26A	6	-	-	PL30A	6	-	-
J5	PL27B	6	-	VREF1_6	PL31B	6	-	VREF1_6
L1	PL28A	6	T[3]	DQS	PL32A	6	T[3]	DQS
L2	PL28B	6	C[3]	-	PL32B	6	C[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
M1	PL29A	6	T	LLM0_PLLT_IN_A	PL33A	6	T	LLM0_PLLT_IN_A
M2	PL29B	6	C	LLM0_PLLC_IN_A	PL33B	6	C	LLM0_PLLC_IN_A
K3	PL30A	6	T[3]	-	PL34A	6	T[3]	-
L3	PL30B	6	C[3]	-	PL34B	6	C[3]	-

LFXP15 & LFXP20 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
L4	PL32A	6	-	-	PL36A	6	-	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
K4	PL33A	6	T	-	PL37A	6	T	-
K5	PL33B	6	C	-	PL37B	6	C	-
N1	PL35A	6	-	VREF2_6	PL39A	6	-	VREF2_6
N2	PL36B	6	-	-	PL40B	6	-	-
P1	PL37A	6	T[3]	DQS	PL41A	6	T[3]	DQS
P2	PL37B	6	C[3]	-	PL41B	6	C[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
L5	PL38A	6	T	LLM0_PLLT_FB_A	PL42A	6	T	LLM0_PLLT_FB_A
M6	PL38B	6	C	LLM0_PLLC_FB_A	PL42B	6	C	LLM0_PLLC_FB_A
M3	PL39A	6	T[3]	-	PL43A	6	T[3]	-
N3	PL39B	6	C[3]	-	PL43B	6	C[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
P4	SLEEPN[1]/TOE[2]	-	-	-	SLEEPN[1]/TOE[2]	-	-	-
P3	INITN	5	-	-	INITN	5	-	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
R4	PB11A	5	T	-	PB15A	5	T	-
N5	PB11B	5	C	-	PB15B	5	C	-
P5	PB12A	5	T	VREF1_5	PB16A	5	T	VREF1_5
-	GNDIO5	5	-	-	GNDIO5	5	-	-
R1	PB12B	5	C	-	PB16B	5	C	-
N6	PB13A	5	-	-	PB17A	5	-	-
M7	PB14B	5	-	-	PB18B	5	-	-
R2	PB15A	5	T	DQS	PB19A	5	T	DQS
T2	PB15B	5	C	-	PB19B	5	C	-
R3	PB16A	5	T	-	PB20A	5	T	-
T3	PB16B	5	C	-	PB20B	5	C	-
T4	PB17A	5	T	-	PB21A	5	T	-
R5	PB17B	5	C	VREF2_5	PB21B	5	C	VREF2_5
N7	PB18A	5	T	-	PB22A	5	T	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
M8	PB18B	5	C	-	PB22B	5	C	-
T5	PB19A	5	T	-	PB23A	5	T	-
P6	PB19B	5	C	-	PB23B	5	C	-
T6	PB20A	5	T	-	PB24A	5	T	-
R6	PB20B	5	C	-	PB24B	5	C	-
P7	PB21A	5	-	-	PB25A	5	-	-
N8	PB22B	5	-	-	PB26B	5	-	-
R7	PB23A	5	T	DQS	PB27A	5	T	DQS

LFXP15 & LFXP20 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
T7	PB23B	5	C	-	PB27B	5	C	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
P8	PB24A	5	T	-	PB28A	5	T	-
T8	PB24B	5	C	-	PB28B	5	C	-
R8	PB25A	5	T	-	PB29A	5	T	-
T9	PB25B	5	C	-	PB29B	5	C	-
R9	PB26A	4	T	-	PB30A	4	T	-
P9	PB26B	4	C	-	PB30B	4	C	-
T10	PB27A	4	T	PCLKT4_0	PB31A	4	T	PCLKT4_0
T11	PB27B	4	C	PCLKC4_0	PB31B	4	C	PCLKC4_0
-	GNDIO4	4	-	-	GNDIO4	4	-	-
R10	PB28A	4	T	-	PB32A	4	T	-
P10	PB28B	4	C	-	PB32B	4	C	-
N9	PB29A	4	-	-	PB33A	4	-	-
M9	PB30B	4	-	-	PB34B	4	-	-
R12	PB31A	4	T	DQS	PB35A	4	T	DQS
T12	PB31B	4	C	VREF1_4	PB35B	4	C	VREF1_4
P13	PB32A	4	T	-	PB36A	4	T	-
R13	PB32B	4	C	-	PB36B	4	C	-
M11	PB33A	4	T	-	PB37A	4	T	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
N11	PB33B	4	C	-	PB37B	4	C	-
N10	PB34A	4	T	-	PB38A	4	T	-
M10	PB34B	4	C	-	PB38B	4	C	-
T13	PB35A	4	T	-	PB39A	4	T	-
P14	PB35B	4	C	-	PB39B	4	C	-
R11	PB36A	4	T	VREF2_4	PB40A	4	T	VREF2_4
P12	PB36B	4	C	-	PB40B	4	C	-
T14	PB37A	4	-	-	PB41A	4	-	-
R14	PB38B	4	-	-	PB42B	4	-	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
P11	PB39A	4	T	DQS	PB43A	4	T	DQS
N12	PB39B	4	C	-	PB43B	4	C	-
T15	PB40A	4	T	-	PB44A	4	T	-
R15	PB40B	4	C	-	PB44B	4	C	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-
P15	PR38B	3	C	RLM0_PLLC_FB_A	PR42B	3	C	RLM0_PLLC_FB_A
N15	PR38A	3	T	RLM0_PLLT_FB_A	PR42A	3	T	RLM0_PLLT_FB_A

LFXP15 & LFXP20 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
P16	PR37B	3	C[3]	-	PR41B	3	C[3]	-
R16	PR37A	3	T[3]	DQS	PR41A	3	T[3]	DQS
M15	PR36B	3	-	-	PR40B	3	-	-
N14	PR35A	3	-	VREF1_3	PR39A	3	-	VREF1_3
-	GNDIO3	3	-	-	GNDIO3	3	-	-
M14	PR33B	3	C	-	PR37B	3	C	-
L13	PR33A	3	T	-	PR37A	3	T	-
L15	PR32B	3	C[3]	-	PR36B	3	C[3]	-
L14	PR32A	3	T[3]	-	PR36A	3	T[3]	-
L12	PR30A	3	-	-	PR34A	3	-	-
M16	PR29B	3	C	RLM0_PLLC_IN_A	PR33B	3	C	RLM0_PLLC_IN_A
N16	PR29A	3	T	RLM0_PLLT_IN_A	PR33A	3	T	RLM0_PLLT_IN_A
-	GNDIO3	3	-	-	GNDIO3	3	-	-
K14	PR28B	3	C[3]	-	PR32B	3	C[3]	-
K15	PR28A	3	T[3]	DQS	PR32A	3	T[3]	DQS
K12	PR27B	3	-	-	PR31B	3	-	-
K13	PR26A	3	-	VREF2_3	PR30A	3	-	VREF2_3
L16	PR25B	3	C[3]	-	PR29B	3	C[3]	-
K16	PR25A	3	T[3]	-	PR29A	3	T[3]	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-
J15	PR23B	3	C[3]	-	PR27B	3	C[3]	-
J14	PR23A	3	T[3]	-	PR27A	3	T[3]	-
J13	GNDP1	-	-	-	GNDP1	-	-	-
J12	VCCP1	-	-	-	VCCP1	-	-	-
-	GNDIO2	2	-	-	GNDIO2	2	-	-
J16	PR21B	2	C	PCLKC2_0	PR21B	2	C	PCLKC2_0
H16	PR21A	2	T	PCLKT2_0	PR21A	2	T	PCLKT2_0
H13	PR20B	2	C[3]	-	PR20B	2	C[3]	-
H12	PR20A	2	T[3]	DQS	PR20A	2	T[3]	DQS
H15	PR19B	2	-	-	PR19B	2	-	-
H14	PR18A	2	-	VREF1_2	PR18A	2	-	VREF1_2
-	GNDIO2	2	-	-	GNDIO2	2	-	-
G15	PR17B	2	C[3]	-	PR17B	2	C[3]	-
G14	PR17A	2	T[3]	-	PR17A	2	T[3]	-
G16	PR16B	2	C	RUM0_PLLC_IN_A	PR16B	2	C	RUM0_PLLC_IN_A
F16	PR16A	2	T	RUM0_PLLT_IN_A	PR16A	2	T	RUM0_PLLT_IN_A
G13	PR15B	2	-	-	PR15B	2	-	-
-	GNDIO2	2	-	-	GNDIO2	2	-	-
G12	PR12B	2	C	-	PR12B	2	C	-
F13	PR12A	2	T	-	PR12A	2	T	-
B16	PR11B	2	C[3]	-	PR11B	2	C[3]	-
C16	PR11A	2	T[3]	DQS	PR11A	2	T[3]	DQS

LFXP15 & LFXP20 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
-	GNDIO2	2	-	-	GNDIO2	2	-	-
F15	PR10B	2	-	-	PR10B	2	-	-
E15	PR9A	2	-	VREF2_2	PR9A	2	-	VREF2_2
F14	PR8B	2	C[3]	-	PR8B	2	C[3]	-
E14	PR8A	2	T[3]	-	PR8A	2	T[3]	-
D15	PR7B	2	C	RUM0_PLLC_FB_A	PR7B	2	C	RUM0_PLLC_FB_A
C15	PR7A	2	T	RUM0_PLLT_FB_A	PR7A	2	T	RUM0_PLLT_FB_A
-	GNDIO2	2	-	-	GNDIO2	2	-	-
E16	TDO	-	-	-	TDO	-	-	-
D16	VCCJ	-	-	-	VCCJ	-	-	-
D14	TDI	-	-	-	TDI	-	-	-
C14	TMS	-	-	-	TMS	-	-	-
B14	TCK	-	-	-	TCK	-	-	-
-	GNDIO1	1	-	-	GNDIO1	1	-	-
-	GNDIO1	1	-	-	GNDIO1	1	-	-
-	GNDIO1	1	-	-	GNDIO1	1	-	-
A15	PT40B	1	C	-	PT44B	1	C	-
B15	PT40A	1	T	-	PT44A	1	T	-
D12	PT39B	1	C	VREF1_1	PT43B	1	C	VREF1_1
-	GNDIO1	1	-	-	GNDIO1	1	-	-
C11	PT39A	1	T	DQS	PT43A	1	T	DQS
A14	PT38B	1	-	-	PT42B	1	-	-
B13	PT37A	1	-	-	PT41A	1	-	-
F12	PT36B	1	C	-	PT40B	1	C	-
E11	PT36A	1	T	-	PT40A	1	T	-
A13	PT35B	1	C	-	PT39B	1	C	-
C13	PT35A	1	T	D0	PT39A	1	T	D0
C10	PT34B	1	C	D1	PT38B	1	C	D1
E10	PT34A	1	T	VREF2_1	PT38A	1	T	VREF2_1
A12	PT33B	1	C	-	PT37B	1	C	-
B12	PT33A	1	T	D2	PT37A	1	T	D2
-	GNDIO1	1	-	-	GNDIO1	1	-	-
C12	PT32B	1	C	D3	PT36B	1	C	D3
A11	PT32A	1	T	-	PT36A	1	T	-
B11	PT31B	1	C	-	PT35B	1	C	-
D11	PT31A	1	T	DQS	PT35A	1	T	DQS
B9	PT30B	1	-	-	PT34B	1	-	-
D9	PT29A	1	-	D4	PT33A	1	-	D4
A10	PT28B	1	C	-	PT32B	1	C	-
B10	PT28A	1	T	D5	PT32A	1	T	D5
-	GNDIO1	1	-	-	GNDIO1	1	-	-
D10	PT27B	1	C	D6	PT31B	1	C	D6

LFXP15 & LFXP20 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
A9	PT27A	1	T	-	PT31A	1	T	-
C9	PT26B	1	C	D7	PT30B	1	C	D7
C8	PT26A	1	T	-	PT30A	1	T	-
E9	PT25B	0	C	BUSY	PT29B	0	C	BUSY
-	GNDIO0	0	-	-	GNDIO0	0	-	-
B8	PT25A	0	T	CS1N	PT29A	0	T	CS1N
A8	PT24B	0	C	PCLKC0_0	PT28B	0	C	PCLKC0_0
A7	PT24A	0	T	PCLKT0_0	PT28A	0	T	PCLKT0_0
B7	PT23B	0	C	-	PT27B	0	C	-
C7	PT23A	0	T	DQS	PT27A	0	T	DQS
E8	PT22B	0	-	-	PT26B	0	-	-
D8	PT21A	0	-	DOUT	PT25A	0	-	DOUT
A6	PT20B	0	C	-	PT24B	0	C	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
C6	PT20A	0	T	WRITEN	PT24A	0	T	WRITEN
E7	PT19B	0	C	-	PT23B	0	C	-
D7	PT19A	0	T	VREF1_0	PT23A	0	T	VREF1_0
A5	PT18B	0	C	-	PT22B	0	C	-
B5	PT18A	0	T	DI	PT22A	0	T	DI
A4	PT17B	0	C	-	PT21B	0	C	-
B6	PT17A	0	T	CSN	PT21A	0	T	CSN
E6	PT16B	0	C	-	PT20B	0	C	-
D6	PT16A	0	T	-	PT20A	0	T	-
D5	PT15B	0	C	VREF2_0	PT19B	0	C	VREF2_0
A3	PT15A	0	T	DQS	PT19A	0	T	DQS
B3	PT14B	0	-	-	PT18B	0	-	-
B2	PT13A	0	-	-	PT17A	0	-	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
A2	PT12B	0	C	-	PT16B	0	C	-
B1	PT12A	0	T	-	PT16A	0	T	-
F5	PT11B	0	C	-	PT15B	0	C	-
C5	PT11A	0	T	-	PT15A	0	T	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
C4	CFG0	0	-	-	CFG0	0	-	-
B4	CFG1	0	-	-	CFG1	0	-	-
C3	DONE	0	-	-	DONE	0	-	-
A1	GND	-	-	-	GND	-	-	-
A16	GND	-	-	-	GND	-	-	-
F11	GND	-	-	-	GND	-	-	-
F6	GND	-	-	-	GND	-	-	-

LFXP15 & LFXP20 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
G10	GND	-	-	-	GND	-	-	-
G7	GND	-	-	-	GND	-	-	-
G8	GND	-	-	-	GND	-	-	-
G9	GND	-	-	-	GND	-	-	-
H10	GND	-	-	-	GND	-	-	-
H7	GND	-	-	-	GND	-	-	-
H8	GND	-	-	-	GND	-	-	-
H9	GND	-	-	-	GND	-	-	-
J10	GND	-	-	-	GND	-	-	-
J7	GND	-	-	-	GND	-	-	-
J8	GND	-	-	-	GND	-	-	-
J9	GND	-	-	-	GND	-	-	-
K10	GND	-	-	-	GND	-	-	-
K7	GND	-	-	-	GND	-	-	-
K8	GND	-	-	-	GND	-	-	-
K9	GND	-	-	-	GND	-	-	-
L11	GND	-	-	-	GND	-	-	-
L6	GND	-	-	-	GND	-	-	-
T1	GND	-	-	-	GND	-	-	-
T16	GND	-	-	-	GND	-	-	-
D13	VCC	-	-	-	VCC	-	-	-
D4	VCC	-	-	-	VCC	-	-	-
E12	VCC	-	-	-	VCC	-	-	-
E5	VCC	-	-	-	VCC	-	-	-
M12	VCC	-	-	-	VCC	-	-	-
M5	VCC	-	-	-	VCC	-	-	-
N13	VCC	-	-	-	VCC	-	-	-
N4	VCC	-	-	-	VCC	-	-	-
E13	VCCAUX	-	-	-	VCCAUX	-	-	-
E4	VCCAUX	-	-	-	VCCAUX	-	-	-
M13	VCCAUX	-	-	-	VCCAUX	-	-	-
M4	VCCAUX	-	-	-	VCCAUX	-	-	-
F7	VCCIO0	0	-	-	VCCIO0	0	-	-
F8	VCCIO0	0	-	-	VCCIO0	0	-	-
F10	VCCIO1	1	-	-	VCCIO1	1	-	-
F9	VCCIO1	1	-	-	VCCIO1	1	-	-
G11	VCCIO2	2	-	-	VCCIO2	2	-	-
H11	VCCIO2	2	-	-	VCCIO2	2	-	-
J11	VCCIO3	3	-	-	VCCIO3	3	-	-
K11	VCCIO3	3	-	-	VCCIO3	3	-	-
L10	VCCIO4	4	-	-	VCCIO4	4	-	-
L9	VCCIO4	4	-	-	VCCIO4	4	-	-

LFXP15 & LFXP20 Logic Signal Connections: 256 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
L7	VCCIO5	5	-	-	VCCIO5	5	-	-
L8	VCCIO5	5	-	-	VCCIO5	5	-	-
J6	VCCIO6	6	-	-	VCCIO6	6	-	-
K6	VCCIO6	6	-	-	VCCIO6	6	-	-
G6	VCCIO7	7	-	-	VCCIO7	7	-	-
H6	VCCIO7	7	-	-	VCCIO7	7	-	-

1. Applies to LFXP “C” only.
2. Applies to LFXP “E” only.
3. Supports dedicated LVDS outputs.

LFXP10, LFXP15 & LFXP20 Logic Signal Connections: 388 fpBGA

Ball Number	LFXP10				LFXP15				LFXP20			
	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function
F4	PROGRAMN	7	-	-	PROGRAMN	7	-	-	PROGRAMN	7	-	-
G4	CCLK	7	-	-	CCLK	7	-	-	CCLK	7	-	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-	GNDIO7	7	-	-
D2	PL2A	7	T[3]	-	PL6A	7	T[3]	-	PL6A	7	T[3]	-
D1	PL2B	7	C[3]	-	PL6B	7	C[3]	-	PL6B	7	C[3]	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-	GNDIO7	7	-	-
E2	PL3A	7	T	LUM0_PLLT_FB_A	PL7A	7	T	LUM0_PLLT_FB_A	PL7A	7	T	LUM0_PLLT_FB_A
E3	PL3B	7	C	LUM0_PLLC_FB_A	PL7B	7	C	LUM0_PLLC_FB_A	PL7B	7	C	LUM0_PLLC_FB_A
F3	PL4A	7	T[3]	-	PL8A	7	T[3]	-	PL8A	7	T[3]	-
F2	PL4B	7	C[3]	-	PL8B	7	C[3]	-	PL8B	7	C[3]	-
H4	PL5A	7	-	-	PL9A	7	-	-	PL9A	7	-	-
H3	PL6B	7	-	VREF1_7	PL10B	7	-	VREF1_7	PL10B	7	-	VREF1_7
G3	PL7A	7	T[3]	DQS	PL11A	7	T[3]	DQS	PL11A	7	T[3]	DQS
G2	PL7B	7	C[3]	-	PL11B	7	C[3]	-	PL11B	7	C[3]	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-	GNDIO7	7	-	-
F1	PL8A	7	T	-	PL12A	7	T	-	PL12A	7	T	-
E1	PL8B	7	C	-	PL12B	7	C	-	PL12B	7	C	-
J4	PL9A	7	T[3]	-	PL13A	7	T[3]	-	PL13A	7	T[3]	-
K4	PL9B	7	C[3]	-	PL13B	7	C[3]	-	PL13B	7	C[3]	-
G1	PL11A	7	T[3]	-	PL15A	7	T[3]	-	PL15A	7	T[3]	-
H2	PL11B	7	C[3]	-	PL15B	7	C[3]	-	PL15B	7	C[3]	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-	GNDIO7	7	-	-
J2	PL12A	7	T	LUM0_PLLT_IN_A	PL16A	7	T	LUM0_PLLT_IN_A	PL16A	7	T	LUM0_PLLT_IN_A
H1	PL12B	7	C	LUM0_PLLC_IN_A	PL16B	7	C	LUM0_PLLC_IN_A	PL16B	7	C	LUM0_PLLC_IN_A
J1	PL13A	7	T[3]	-	PL17A	7	T[3]	-	PL17A	7	T[3]	-
K2	PL13B	7	C[3]	-	PL17B	7	C[3]	-	PL17B	7	C[3]	-
K3	PL14A	7	-	VREF2_7	PL18A	7	-	VREF2_7	PL18A	7	-	VREF2_7
J3	PL15B	7	-	-	PL19B	7	-	-	PL19B	7	-	-
K1	PL16A	7	T[3]	DQS	PL20A	7	T[3]	DQS	PL20A	7	T[3]	DQS
-	GNDIO7	7	-	-	GNDIO7	7	-	-	GNDIO7	7	-	-
L2	PL16B	7	C[3]	-	PL20B	7	C[3]	-	PL20B	7	C[3]	-
L3	PL17A	7	T	-	PL21A	7	T	-	PL21A	7	T	-
L4	PL17B	7	C	-	PL21B	7	C	-	PL21B	7	C	-
L1	PL18A	7	T[3]	-	PL22A	7	T[3]	-	PL22A	7	T[3]	-
M1	PL18B	7	C[3]	-	PL22B	7	C[3]	-	PL22B	7	C[3]	-
M2	VCCP0	-	-	-	VCCP0	-	-	-	VCCP0	-	-	-
N1	GNDP0	-	-	-	GNDP0	-	-	-	GNDP0	-	-	-
M3	PL19A	6	T[3]	-	PL23A	6	T[3]	-	PL27A	6	T[3]	-
M4	PL19B	6	C[3]	-	PL23B	6	C[3]	-	PL27B	6	C[3]	-
P1	PL20A	6	T	PCLKT6_0	PL24A	6	T	PCLKT6_0	PL28A	6	T	PCLKT6_0
-	GNDIO6	6	-	-	GNDIO6	6	-	-	GNDIO6	6	-	-
N2	PL20B	6	C	PCLKC6_0	PL24B	6	C	PCLKC6_0	PL28B	6	C	PCLKC6_0
R1	PL21A	6	T[3]	-	PL25A	6	T[3]	-	PL29A	6	T[3]	-
P2	PL21B	6	C[3]	-	PL25B	6	C[3]	-	PL29B	6	C[3]	-
N3	PL22A	6	-	-	PL26A	6	-	-	PL30A	6	-	-
N4	PL23B	6	-	VREF1_6	PL27B	6	-	VREF1_6	PL31B	6	-	VREF1_6
T1	PL24A	6	T[3]	DQS	PL28A	6	T[3]	DQS	PL32A	6	T[3]	DQS
R2	PL24B	6	C[3]	-	PL28B	6	C[3]	-	PL32B	6	C[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-	GNDIO6	6	-	-

LFXP10, LFXP15 & LFXP20 Logic Signal Connections: 388 fpBGA (Cont.)

Ball Number	LFXP10				LFXP15				LFXP20			
	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function
U1	PL25A	6	T	LLM0_PLLT_IN_A	PL29A	6	T	LLM0_PLLT_IN_A	PL33A	6	T	LLM0_PLLT_IN_A
T2	PL25B	6	C	LLM0_PLLC_IN_A	PL29B	6	C	LLM0_PLLC_IN_A	PL33B	6	C	LLM0_PLLC_IN_A
V1	PL26A	6	T[3]	-	PL30A	6	T[3]	-	PL34A	6	T[3]	-
U2	PL26B	6	C[3]	-	PL30B	6	C[3]	-	PL34B	6	C[3]	-
W1	PL28A	6	T[3]	-	PL32A	6	T[3]	-	PL36A	6	T[3]	-
V2	PL28B	6	C[3]	-	PL32B	6	C[3]	-	PL36B	6	C[3]	-
-	GNDIO6	6	-	-	GNDIO6	-	-	-	GNDIO6	6	-	-
P3	PL29A	6	T	-	PL33A	6	T	-	PL37A	6	T	-
P4	PL29B	6	C	-	PL33B	6	C	-	PL37B	6	C	-
Y1	PL30A	6	T[3]	-	PL34A	6	T[3]	-	PL38A	6	T[3]	-
W2	PL30B	6	C[3]	-	PL34B	6	C[3]	-	PL38B	6	C[3]	-
R3	PL31A	6	-	VREF2_6	PL35A	6	-	VREF2_6	PL39A	6	-	VREF2_6
R4	PL32B	6	-	-	PL36B	6	-	-	PL40B	6	-	-
T3	PL33A	6	T[3]	DQS	PL37A	6	T[3]	DQS	PL41A	6	T[3]	DQS
T4	PL33B	6	C[3]	-	PL37B	6	C[3]	-	PL41B	6	C[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-	GNDIO6	6	-	-
V4	PL34A	6	T	LLM0_PLLT_FB_A	PL38A	6	T	LLM0_PLLT_FB_A	PL42A	6	T	LLM0_PLLT_FB_A
V3	PL34B	6	C	LLM0_PLLC_FB_A	PL38B	6	C	LLM0_PLLC_FB_A	PL42B	6	C	LLM0_PLLC_FB_A
U4	PL35A	6	T[3]	-	PL39A	6	T[3]	-	PL43A	6	T[3]	-
U3	PL35B	6	C[3]	-	PL39B	6	C[3]	-	PL43B	6	C[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-	GNDIO6	6	-	-
W5	SLEEPN[1]/ TOE[2]	-	-	-	SLEEPN[1]/ TOE[2]	-	-	-	SLEEPN[1]/ TOE[2]	-	-	-
Y2	INITN	5	-	-	INITN	5	-	-	INITN	5	-	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-	GNDIO5	5	-	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-	GNDIO5	5	-	-
Y3	-	-	-	-	PB3B	5	-	-	PB7B	5	-	-
W3	-	-	-	-	PB4A	5	T	-	PB8A	5	T	-
W4	-	-	-	-	PB4B	5	C	-	PB8B	5	C	-
AA2	-	-	-	-	PB5A	5	-	-	PB9A	5	-	-
AA1	-	-	-	-	PB6B	5	-	-	PB10B	5	-	-
W6	PB2A	5	-	-	PB7A	5	T	DQS	PB11A	5	T	DQS
W7	-	-	-	-	PB7B	5	C	-	PB11B	5	C	-
Y4	PB3A	5	T	-	PB8A	5	T	-	PB12A	5	T	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-	GNDIO5	5	-	-
Y5	PB3B	5	C	-	PB8B	5	C	-	PB12B	5	C	-
AB2	PB4A	5	T	-	PB9A	5	T	-	PB13A	5	T	-
AA3	PB4B	5	C	-	PB9B	5	C	-	PB13B	5	C	-
AB3	PB5A	5	T	-	PB10A	5	T	-	PB14A	5	T	-
AA4	PB5B	5	C	-	PB10B	5	C	-	PB14B	5	C	-
W8	PB6A	5	T	-	PB11A	5	T	-	PB15A	5	T	-
W9	PB6B	5	C	-	PB11B	5	C	-	PB15B	5	C	-
AB4	PB7A	5	T	VREF1_5	PB12A	5	T	VREF1_5	PB16A	5	T	VREF1_5
-	GNDIO5	5	-	-	GNDIO5	5	-	-	GNDIO5	5	-	-
AA5	PB7B	5	C	-	PB12B	5	C	-	PB16B	5	C	-
AB5	PB8A	5	-	-	PB13A	5	-	-	PB17A	5	-	-
Y6	PB9B	5	-	-	PB14B	5	-	-	PB18B	5	-	-
AA6	PB10A	5	T	DQS	PB15A	5	T	DQS	PB19A	5	T	DQS
AB6	PB10B	5	C	-	PB15B	5	C	-	PB19B	5	C	-
Y9	PB11A	5	T	-	PB16A	5	T	-	PB20A	5	T	-

LFXP10, LFXP15 & LFXP20 Logic Signal Connections: 388 fpBGA (Cont.)

Ball Number	LFXP10				LFXP15				LFXP20			
	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function
Y10	PB11B	5	C	-	PB16B	5	C	-	PB20B	5	C	-
AA7	PB12A	5	T	-	PB17A	5	T	-	PB21A	5	T	-
AB7	PB12B	5	C	VREF2_5	PB17B	5	C	VREF2_5	PB21B	5	C	VREF2_5
Y7	PB13A	5	T	-	PB18A	5	T	-	PB22A	5	T	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-	GNDIO5	5	-	-
AA8	PB13B	5	C	-	PB18B	5	C	-	PB22B	5	C	-
AB8	PB14A	5	T	-	PB19A	5	T	-	PB23A	5	T	-
Y8	PB14B	5	C	-	PB19B	5	C	-	PB23B	5	C	-
AB9	PB15A	5	T	-	PB20A	5	T	-	PB24A	5	T	-
AA9	PB15B	5	C	-	PB20B	5	C	-	PB24B	5	C	-
W10	PB16A	5	-	-	PB21A	5	-	-	PB25A	5	-	-
W11	PB17B	5	-	-	PB22B	5	-	-	PB26B	5	-	-
AB10	PB18A	5	T	DQS	PB23A	5	T	DQS	PB27A	5	T	DQS
AA10	PB18B	5	C	-	PB23B	5	C	-	PB27B	5	C	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-	GNDIO5	5	-	-
AA11	PB19A	5	T	-	PB24A	5	T	-	PB28A	5	T	-
AB11	PB19B	5	C	-	PB24B	5	C	-	PB28B	5	C	-
Y11	PB20A	5	T	-	PB25A	5	T	-	PB29A	5	T	-
Y12	PB20B	5	C	-	PB25B	5	C	-	PB29B	5	C	-
AB12	PB21A	4	T	-	PB26A	4	T	-	PB30A	4	T	-
AA12	PB21B	4	C	-	PB26B	4	C	-	PB30B	4	C	-
AB13	PB22A	4	T	PCLKT4_0	PB27A	4	T	PCLKT4_0	PB31A	4	T	PCLKT4_0
AA13	PB22B	4	C	PCLKC4_0	PB27B	4	C	PCLKC4_0	PB31B	4	C	PCLKC4_0
-	GNDIO4	4	-	-	GNDIO4	4	-	-	GNDIO4	4	-	-
AA14	PB23A	4	T	-	PB28A	4	T	-	PB32A	4	T	-
AB14	PB23B	4	C	-	PB28B	4	C	-	PB32B	4	C	-
W12	PB24A	4	-	-	PB29A	4	-	-	PB33A	4	-	-
W13	PB25B	4	-	-	PB30B	4	-	-	PB34B	4	-	-
AA15	PB26A	4	T	DQS	PB31A	4	T	DQS	PB35A	4	T	DQS
AB15	PB26B	4	C	VREF1_4	PB31B	4	C	VREF1_4	PB35B	4	C	VREF1_4
AA16	PB27A	4	T	-	PB32A	4	T	-	PB36A	4	T	-
AB16	PB27B	4	C	-	PB32B	4	C	-	PB36B	4	C	-
Y17	PB28A	4	T	-	PB33A	4	T	-	PB37A	4	T	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-	GNDIO4	4	-	-
AA17	PB28B	4	C	-	PB33B	4	C	-	PB37B	4	C	-
Y13	PB29A	4	T	-	PB34A	4	T	-	PB38A	4	T	-
Y14	PB29B	4	C	-	PB34B	4	C	-	PB38B	4	C	-
AB17	PB30A	4	T	-	PB35A	4	T	-	PB39A	4	T	-
Y18	PB30B	4	C	-	PB35B	4	C	-	PB39B	4	C	-
AA18	PB31A	4	T	VREF2_4	PB36A	4	T	VREF2_4	PB40A	4	T	VREF2_4
AB18	PB31B	4	C	-	PB36B	4	C	-	PB40B	4	C	-
Y19	PB32A	4	-	-	PB37A	4	-	-	PB41A	4	-	-
AB19	PB33B	4	-	-	PB38B	4	-	-	PB42B	4	-	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-	GNDIO4	4	-	-
AA19	PB34A	4	T	DQS	PB39A	4	T	DQS	PB43A	4	T	DQS
Y20	PB34B	4	C	-	PB39B	4	C	-	PB43B	4	C	-
W14	PB35A	4	T	-	PB40A	4	T	-	PB44A	4	T	-
W15	PB35B	4	C	-	PB40B	4	C	-	PB44B	4	C	-
AB20	PB36A	4	T	-	PB41A	4	T	-	PB45A	4	T	-

LFXP10, LFXP15 & LFXP20 Logic Signal Connections: 388 fpBGA (Cont.)

Ball Number	LFXP10				LFXP15				LFXP20			
	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function
AA20	PB36B	4	C	-	PB41B	4	C	-	PB45B	4	C	-
AB21	PB37A	4	T	-	PB42A	4	T	-	PB46A	4	T	-
AA21	PB37B	4	C	-	PB42B	4	C	-	PB46B	4	C	-
AA22	PB38A	4	T	-	PB43A	4	T	-	PB47A	4	T	-
Y21	PB38B	4	C	-	PB43B	4	C	-	PB47B	4	C	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-	GNDIO4	4	-	-
W16	PB39A	4	-	-	PB44A	4	T	-	PB48A	4	T	-
W17	-	-	-	-	PB44B	4	C	-	PB48B	4	C	-
Y15	-	-	-	-	PB45A	4	-	-	PB49A	4	-	-
Y16	-	-	-	-	PB46B	4	-	-	PB50B	4	-	-
W19	-	-	-	-	PB47A	4	T	DQS	PB51A	4	T	DQS
W18	-	-	-	-	PB47B	4	C	-	PB51B	4	C	-
W20	-	-	-	-	PB48A	4	-	-	PB52A	4	-	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-	GNDIO4	4	-	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-	GNDIO4	4	-	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-	GNDIO3	3	-	-
T20	PR35B	3	C[3]	-	PR39B	3	C[3]	-	PR43B	3	C[3]	-
T19	PR35A	3	T[3]	-	PR39A	3	T[3]	-	PR43A	3	T[3]	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-	GNDIO3	3	-	-
U19	PR34B	3	C	RLM0_PLLC_FB_A	PR38B	3	C	RLM0_PLLC_FB_A	PR42B	3	C	RLM0_PLLC_FB_A
U20	PR34A	3	T	RLM0_PLLT_FB_A	PR38A	3	T	RLM0_PLLT_FB_A	PR42A	3	T	RLM0_PLLT_FB_A
V19	PR33B	3	C[3]	-	PR37B	3	C[3]	-	PR41B	3	C[3]	-
V20	PR33A	3	T[3]	DQS	PR37A	3	T[3]	DQS	PR41A	3	T[3]	DQS
R19	PR32B	3	-	-	PR36B	3	-	-	PR40B	3	-	-
R20	PR31A	3	-	VREF1_3	PR35A	3	-	VREF1_3	PR39A	3	-	VREF1_3
W21	PR30B	3	C[3]	-	PR34B	3	C[3]	-	PR38B	3	C[3]	-
Y22	PR30A	3	T[3]	-	PR34A	3	T[3]	-	PR38A	3	T[3]	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-	GNDIO3	3	-	-
P19	PR29B	3	C	-	PR33B	3	C	-	PR37B	3	C	-
P20	PR29A	3	T	-	PR33A	3	T	-	PR37A	3	T	-
V21	PR28B	3	C[3]	-	PR32B	3	C[3]	-	PR36B	3	C[3]	-
W22	PR28A	3	T[3]	-	PR32A	3	T[3]	-	PR36A	3	T[3]	-
U21	PR26B	3	C[3]	-	PR30B	3	C[3]	-	PR34B	3	C[3]	-
V22	PR26A	3	T[3]	-	PR30A	3	T[3]	-	PR34A	3	T[3]	-
T21	PR25B	3	C	RLM0_PLLC_IN_A	PR29B	3	C	RLM0_PLLC_IN_A	PR33B	3	C	RLM0_PLLC_IN_A
U22	PR25A	3	T	RLM0_PLLT_IN_A	PR29A	3	T	RLM0_PLLT_IN_A	PR33A	3	T	RLM0_PLLT_IN_A
-	GNDIO3	3	-	-	GNDIO3	3	-	-	GNDIO3	3	-	-
R21	PR24B	3	C[3]	-	PR28B	3	C[3]	-	PR32B	3	C[3]	-
T22	PR24A	3	T[3]	DQS	PR28A	3	T[3]	DQS	PR32A	3	T[3]	DQS
N19	PR23B	3	-	-	PR27B	3	-	-	PR31B	3	-	-
N20	PR22A	3	-	VREF2_3	PR26A	3	-	VREF2_3	PR30A	3	-	VREF2_3
R22	PR21B	3	C[3]	-	PR25B	3	C[3]	-	PR29B	3	C[3]	-
P22	PR21A	3	T[3]	-	PR25A	3	T[3]	-	PR29A	3	T[3]	-
P21	PR20B	3	C	-	PR24B	3	C	-	PR28B	3	C	-
N21	PR20A	3	T	-	PR24A	3	T	-	PR28A	3	T	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-	GNDIO3	3	-	-
M20	PR19B	3	C[3]	-	PR23B	3	C[3]	-	PR27B	3	C[3]	-
M19	PR19A	3	T[3]	-	PR23A	3	T[3]	-	PR27A	3	T[3]	-
N22	GNDP1	-	-	-	GNDP1	-	-	-	GNDP1	-	-	-

LFXP10, LFXP15 & LFXP20 Logic Signal Connections: 388 fpBGA (Cont.)

Ball Number	LFXP10				LFXP15				LFXP20			
	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function
M21	VCCP1	-	-	-	VCCP1	-	-	-	VCCP1	-	-	-
-	GNDIO2	2	-	-	GNDIO2	2	-	-	GNDIO2	2	-	-
M22	PR18B	2	C[3]	-	PR22B	2	C[3]	-	PR22B	2	C[3]	-
L22	PR18A	2	T[3]	-	PR22A	2	T[3]	-	PR22A	2	T[3]	-
K22	PR17B	2	C	PCLKC2_0	PR21B	2	C	PCLKC2_0	PR21B	2	C	PCLKC2_0
K21	PR17A	2	T	PCLKT2_0	PR21A	2	T	PCLKT2_0	PR21A	2	T	PCLKT2_0
L19	PR16B	2	C[3]	-	PR20B	2	C[3]	-	PR20B	2	C[3]	-
K20	PR16A	2	T[3]	DQS	PR20A	2	T[3]	DQS	PR20A	2	T[3]	DQS
L20	PR15B	2	-	-	PR19B	2	-	-	PR19B	2	-	-
L21	PR14A	2	-	VREF1_2	PR18A	2	-	VREF1_2	PR18A	2	-	VREF1_2
-	GNDIO2	2	-	-	GNDIO2	2	-	-	GNDIO2	2	-	-
J22	PR13B	2	C[3]	-	PR17B	2	C[3]	-	PR17B	2	C[3]	-
J21	PR13A	2	T[3]	-	PR17A	2	T[3]	-	PR17A	2	T[3]	-
H22	PR12B	2	C	RUM0_PLLC_IN_A	PR16B	2	C	RUM0_PLLC_IN_A	PR16B	2	C	RUM0_PLLC_IN_A
H21	PR12A	2	T	RUM0_PLLT_IN_A	PR16A	2	T	RUM0_PLLT_IN_A	PR16A	2	T	RUM0_PLLT_IN_A
K19	PR11B	2	C[3]	-	PR15B	2	C[3]	-	PR15B	2	C[3]	-
J19	PR11A	2	T[3]	-	PR15A	2	T[3]	-	PR15A	2	T[3]	-
-	GNDIO2	2	-	-	GNDIO2	2	-	-	GNDIO2	2	-	-
J20	PR9B	2	C[3]	-	PR13B	2	C[3]	-	PR13B	2	C[3]	-
H20	PR9A	2	T[3]	-	PR13A	2	T[3]	-	PR13A	2	T[3]	-
H19	PR8B	2	C	-	PR12B	2	C	-	PR12B	2	C	-
G19	PR8A	2	T	-	PR12A	2	T	-	PR12A	2	T	-
G22	PR7B	2	C[3]	-	PR11B	2	C[3]	-	PR11B	2	C[3]	-
G21	PR7A	2	T[3]	DQS	PR11A	2	T[3]	DQS	PR11A	2	T[3]	DQS
-	GNDIO2	2	-	-	GNDIO2	2	-	-	GNDIO2	2	-	-
F20	PR6B	2	-	-	PR10B	2	-	-	PR10B	2	-	-
G20	PR5A	2	-	VREF2_2	PR9A	2	-	VREF2_2	PR9A	2	-	VREF2_2
F22	PR4B	2	C[3]	-	PR8B	2	C[3]	-	PR8B	2	C[3]	-
F21	PR4A	2	T[3]	-	PR8A	2	T[3]	-	PR8A	2	T[3]	-
E22	PR3B	2	C	RUM0_PLLC_FB_A	PR7B	2	C	RUM0_PLLC_FB_A	PR7B	2	C	RUM0_PLLC_FB_A
E21	PR3A	2	T	RUM0_PLLT_FB_A	PR7A	2	T	RUM0_PLLT_FB_A	PR7A	2	T	RUM0_PLLT_FB_A
D22	PR2B	2	C[3]	-	PR6B	2	C[3]	-	PR6B	2	C[3]	-
D21	PR2A	2	T[3]	-	PR6A	2	T[3]	-	PR6A	2	T[3]	-
-	GNDIO2	2	-	-	GNDIO2	2	-	-	GNDIO2	2	-	-
F19	TDO	-	-	-	TDO	-	-	-	TDO	-	-	-
E20	VCCJ	-	-	-	VCCJ	-	-	-	VCCJ	-	-	-
D20	TDI	-	-	-	TDI	-	-	-	TDI	-	-	-
D19	TMS	-	-	-	TMS	-	-	-	TMS	-	-	-
D18	TCK	-	-	-	TCK	-	-	-	TCK	-	-	-
-	GNDIO1	1	-	-	GNDIO1	1	-	-	GNDIO1	1	-	-
E19	-	-	-	-	PT48A	1	-	-	PT52A	1	-	-
D17	-	-	-	-	PT47B	1	C	-	PT51B	1	C	-
D16	-	-	-	-	PT47A	1	T	DQS	PT51A	1	T	DQS
C16	-	-	-	-	PT46B	1	-	-	PT50B	1	-	-
C15	-	-	-	-	PT45A	1	-	-	PT49A	1	-	-
C17	-	-	-	-	PT44B	1	C	-	PT48B	1	C	-
C18	PT39A	1	-	-	PT44A	1	T	-	PT48A	1	T	-
C19	PT38B	1	C	-	PT43B	1	C	-	PT47B	1	C	-
-	GNDIO1	1	-	-	GNDIO1	1	-	-	GNDIO1	1	-	-

LFXP10, LFXP15 & LFXP20 Logic Signal Connections: 388 fpBGA (Cont.)

Ball Number	LFXP10				LFXP15				LFXP20			
	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function
C20	PT38A	1	T	-	PT43A	1	T	-	PT47A	1	T	-
C21	PT37B	1	C	-	PT42B	1	C	-	PT46B	1	C	-
C22	PT37A	1	T	-	PT42A	1	T	-	PT46A	1	T	-
B22	PT36B	1	C	-	PT41B	1	C	-	PT45B	1	C	-
A21	PT36A	1	T	-	PT41A	1	T	-	PT45A	1	T	-
D15	PT35B	1	C	-	PT40B	1	C	-	PT44B	1	C	-
D14	PT35A	1	T	-	PT40A	1	T	-	PT44A	1	T	-
B21	PT34B	1	C	VREF1_1	PT39B	1	C	VREF1_1	PT43B	1	C	VREF1_1
-	GNDIO1	1	-	-	GNDIO1	1	-	-	GNDIO1	1	-	-
A20	PT34A	1	T	DQS	PT39A	1	T	DQS	PT43A	1	T	DQS
B20	PT33B	1	-	-	PT38B	1	-	-	PT42B	1	-	-
A19	PT32A	1	-	-	PT37A	1	-	-	PT41A	1	-	-
B19	PT31B	1	C	-	PT36B	1	C	-	PT40B	1	C	-
A18	PT31A	1	T	-	PT36A	1	T	-	PT40A	1	T	-
C14	PT30B	1	C	-	PT35B	1	C	-	PT39B	1	C	-
C13	PT30A	1	T	D0	PT35A	1	T	D0	PT39A	1	T	D0
B18	PT29B	1	C	D1	PT34B	1	C	D1	PT38B	1	C	D1
A17	PT29A	1	T	VREF2_1	PT34A	1	T	VREF2_1	PT38A	1	T	VREF2_1
B17	PT28B	1	C	-	PT33B	1	C	-	PT37B	1	C	-
A16	PT28A	1	T	D2	PT33A	1	T	D2	PT37A	1	T	D2
-	GNDIO1	1	-	-	GNDIO1	1	-	-	GNDIO1	1	-	-
B16	PT27B	1	C	D3	PT32B	1	C	D3	PT36B	1	C	D3
A15	PT27A	1	T	-	PT32A	1	T	-	PT36A	1	T	-
B15	PT26B	1	C	-	PT31B	1	C	-	PT35B	1	C	-
A14	PT26A	1	T	DQS	PT31A	1	T	DQS	PT35A	1	T	DQS
D13	PT25B	1	-	-	PT30B	1	-	-	PT34B	1	-	-
D12	PT24A	1	-	D4	PT29A	1	-	D4	PT33A	1	-	D4
B14	PT23B	1	C	-	PT28B	1	C	-	PT32B	1	C	-
A13	PT23A	1	T	D5	PT28A	1	T	D5	PT32A	1	T	D5
-	GNDIO1	1	-	-	GNDIO1	1	-	-	GNDIO1	1	-	-
B13	PT22B	1	C	D6	PT27B	1	C	D6	PT31B	1	C	D6
A12	PT22A	1	T	-	PT27A	1	T	-	PT31A	1	T	-
B12	PT21B	1	C	D7	PT26B	1	C	D7	PT30B	1	C	D7
C12	PT21A	1	T	-	PT26A	1	T	-	PT30A	1	T	-
C11	PT20B	0	C	BUSY	PT25B	0	C	BUSY	PT29B	0	C	BUSY
-	GNDIO0	0	-	-	GNDIO0	0	-	-	GNDIO0	0	-	-
B11	PT20A	0	T	CS1N	PT25A	0	T	CS1N	PT29A	0	T	CS1N
A11	PT19B	0	C	PCLKC0_0	PT24B	0	C	PCLKC0_0	PT28B	0	C	PCLKC0_0
A10	PT19A	0	T	PCLKT0_0	PT24A	0	T	PCLKT0_0	PT28A	0	T	PCLKT0_0
B10	PT18B	0	C	-	PT23B	0	C	-	PT27B	0	C	-
B9	PT18A	0	T	DQS	PT23A	0	T	DQS	PT27A	0	T	DQS
D11	PT17B	0	-	-	PT22B	0	-	-	PT26B	0	-	-
D10	PT16A	0	-	DOUT	PT21A	0	-	DOUT	PT25A	0	-	DOUT
A9	PT15B	0	C	-	PT20B	0	C	-	PT24B	0	C	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-	GNDIO0	0	-	-
C8	PT15A	0	T	WRITEN	PT20A	0	T	WRITEN	PT24A	0	T	WRITEN
B8	PT14B	0	C	-	PT19B	0	C	-	PT23B	0	C	-
A8	PT14A	0	T	VREF1_0	PT19A	0	T	VREF1_0	PT23A	0	T	VREF1_0
C7	PT13B	0	C	-	PT18B	0	C	-	PT22B	0	C	-

LFXP10, LFXP15 & LFXP20 Logic Signal Connections: 388 fpBGA (Cont.)

Ball Number	LFXP10				LFXP15				LFXP20			
	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function
A7	PT13A	0	T	DI	PT18A	0	T	DI	PT22A	0	T	DI
B7	PT12B	0	C	-	PT17B	0	C	-	PT21B	0	C	-
C6	PT12A	0	T	CSN	PT17A	0	T	CSN	PT21A	0	T	CSN
C10	PT11B	0	C	-	PT16B	0	C	-	PT20B	0	C	-
C9	PT11A	0	T	-	PT16A	0	T	-	PT20A	0	T	-
A6	PT10B	0	C	VREF2_0	PT15B	0	C	VREF2_0	PT19B	0	C	VREF2_0
B6	PT10A	0	T	DQS	PT15A	0	T	DQS	PT19A	0	T	DQS
A5	PT9B	0	-	-	PT14B	0	-	-	PT18B	0	-	-
B5	PT8A	0	-	-	PT13A	0	-	-	PT17A	0	-	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-	GNDIO0	0	-	-
C5	PT7B	0	C	-	PT12B	0	C	-	PT16B	0	C	-
A4	PT7A	0	T	-	PT12A	0	T	-	PT16A	0	T	-
D9	PT6B	0	C	-	PT11B	0	C	-	PT15B	0	C	-
D8	PT6A	0	T	-	PT11A	0	T	-	PT15A	0	T	-
B4	PT5B	0	C	-	PT10B	0	C	-	PT14B	0	C	-
A2	PT5A	0	T	-	PT10A	0	T	-	PT14A	0	T	-
A3	PT4B	0	C	-	PT9B	0	C	-	PT13B	0	C	-
B3	PT4A	0	T	-	PT9A	0	T	-	PT13A	0	T	-
C4	PT3B	0	C	-	PT8B	0	C	-	PT12B	0	C	-
C3	PT3A	0	T	-	PT8A	0	T	-	PT12A	0	T	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-	GNDIO0	0	-	-
C2	-	-	-	-	PT7B	0	C	-	PT11B	0	C	-
D3	PT2A	0	-	-	PT7A	0	T	DQS	PT11A	0	T	DQS
D7	-	-	-	-	PT6B	0	-	-	PT10B	0	-	-
D6	-	-	-	-	PT5A	0	-	-	PT9A	0	-	-
E4	-	-	-	-	PT4B	0	C	-	PT8B	0	C	-
D4	-	-	-	-	PT4A	0	T	-	PT8A	0	T	-
D5	-	-	-	-	PT3B	0	-	-	PT7B	0	-	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-	GNDIO0	0	-	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-	GNDIO0	0	-	-
C1	CFG0	0	-	-	CFG0	0	-	-	CFG0	0	-	-
B2	CFG1	0	-	-	CFG1	0	-	-	CFG1	0	-	-
B1	DONE	0	-	-	DONE	0	-	-	DONE	0	-	-
A1	GND	-	-	-	GND	-	-	-	GND	-	-	-
A22	GND	-	-	-	GND	-	-	-	GND	-	-	-
AB1	GND	-	-	-	GND	-	-	-	GND	-	-	-
AB22	GND	-	-	-	GND	-	-	-	GND	-	-	-
H10	GND	-	-	-	GND	-	-	-	GND	-	-	-
H11	GND	-	-	-	GND	-	-	-	GND	-	-	-
H12	GND	-	-	-	GND	-	-	-	GND	-	-	-
H13	GND	-	-	-	GND	-	-	-	GND	-	-	-
H14	GND	-	-	-	GND	-	-	-	GND	-	-	-
J10	GND	-	-	-	GND	-	-	-	GND	-	-	-
J11	GND	-	-	-	GND	-	-	-	GND	-	-	-
J12	GND	-	-	-	GND	-	-	-	GND	-	-	-
J13	GND	-	-	-	GND	-	-	-	GND	-	-	-
J14	GND	-	-	-	GND	-	-	-	GND	-	-	-
J9	GND	-	-	-	GND	-	-	-	GND	-	-	-
K10	GND	-	-	-	GND	-	-	-	GND	-	-	-

LFXP10, LFXP15 & LFXP20 Logic Signal Connections: 388 fpBGA (Cont.)

Ball Number	LFXP10				LFXP15				LFXP20			
	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function
K11	GND	-	-	-	GND	-	-	-	GND	-	-	-
K12	GND	-	-	-	GND	-	-	-	GND	-	-	-
K13	GND	-	-	-	GND	-	-	-	GND	-	-	-
K14	GND	-	-	-	GND	-	-	-	GND	-	-	-
K9	GND	-	-	-	GND	-	-	-	GND	-	-	-
L10	GND	-	-	-	GND	-	-	-	GND	-	-	-
L11	GND	-	-	-	GND	-	-	-	GND	-	-	-
L12	GND	-	-	-	GND	-	-	-	GND	-	-	-
L13	GND	-	-	-	GND	-	-	-	GND	-	-	-
L14	GND	-	-	-	GND	-	-	-	GND	-	-	-
L9	GND	-	-	-	GND	-	-	-	GND	-	-	-
M10	GND	-	-	-	GND	-	-	-	GND	-	-	-
M11	GND	-	-	-	GND	-	-	-	GND	-	-	-
M12	GND	-	-	-	GND	-	-	-	GND	-	-	-
M13	GND	-	-	-	GND	-	-	-	GND	-	-	-
M14	GND	-	-	-	GND	-	-	-	GND	-	-	-
M9	GND	-	-	-	GND	-	-	-	GND	-	-	-
N10	GND	-	-	-	GND	-	-	-	GND	-	-	-
N11	GND	-	-	-	GND	-	-	-	GND	-	-	-
N12	GND	-	-	-	GND	-	-	-	GND	-	-	-
N13	GND	-	-	-	GND	-	-	-	GND	-	-	-
N14	GND	-	-	-	GND	-	-	-	GND	-	-	-
N9	GND	-	-	-	GND	-	-	-	GND	-	-	-
P10	GND	-	-	-	GND	-	-	-	GND	-	-	-
P11	GND	-	-	-	GND	-	-	-	GND	-	-	-
P12	GND	-	-	-	GND	-	-	-	GND	-	-	-
P13	GND	-	-	-	GND	-	-	-	GND	-	-	-
P14	GND	-	-	-	GND	-	-	-	GND	-	-	-
P9	GND	-	-	-	GND	-	-	-	GND	-	-	-
R10	GND	-	-	-	GND	-	-	-	GND	-	-	-
R11	GND	-	-	-	GND	-	-	-	GND	-	-	-
R12	GND	-	-	-	GND	-	-	-	GND	-	-	-
R13	GND	-	-	-	GND	-	-	-	GND	-	-	-
R14	GND	-	-	-	GND	-	-	-	GND	-	-	-
H9	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
J15	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
J8	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
K15	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
K8	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
L15	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
L8	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
M15	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
M8	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
N15	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
N8	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
P15	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
P8	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
R9	VCC	-	-	-	VCC	-	-	-	VCC	-	-	-
G16	VCCAUX	-	-	-	VCCAUX	-	-	-	VCCAUX	-	-	-

LFXP10, LFXP15 & LFXP20 Logic Signal Connections: 388 fpBGA (Cont.)

Ball Number	LFXP10				LFXP15				LFXP20			
	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function	Ball Function	Bank	Diff.	Dual Function
G7	VCCAUX	-	-	-	VCCAUX	-	-	-	VCCAUX	-	-	-
T16	VCCAUX	-	-	-	VCCAUX	-	-	-	VCCAUX	-	-	-
T7	VCCAUX	-	-	-	VCCAUX	-	-	-	VCCAUX	-	-	-
G10	VCCIO0	0	-	-	VCCIO0	0	-	-	VCCIO0	0	-	-
G11	VCCIO0	0	-	-	VCCIO0	0	-	-	VCCIO0	0	-	-
G8	VCCIO0	0	-	-	VCCIO0	0	-	-	VCCIO0	0	-	-
G9	VCCIO0	0	-	-	VCCIO0	0	-	-	VCCIO0	0	-	-
H8	VCCIO0	0	-	-	VCCIO0	0	-	-	VCCIO0	0	-	-
G12	VCCIO1	1	-	-	VCCIO1	1	-	-	VCCIO1	1	-	-
G13	VCCIO1	1	-	-	VCCIO1	1	-	-	VCCIO1	1	-	-
G14	VCCIO1	1	-	-	VCCIO1	1	-	-	VCCIO1	1	-	-
G15	VCCIO1	1	-	-	VCCIO1	1	-	-	VCCIO1	1	-	-
H15	VCCIO1	1	-	-	VCCIO1	1	-	-	VCCIO1	1	-	-
H16	VCCIO2	2	-	-	VCCIO2	2	-	-	VCCIO2	2	-	-
J16	VCCIO2	2	-	-	VCCIO2	2	-	-	VCCIO2	2	-	-
K16	VCCIO2	2	-	-	VCCIO2	2	-	-	VCCIO2	2	-	-
L16	VCCIO2	2	-	-	VCCIO2	2	-	-	VCCIO2	2	-	-
M16	VCCIO3	3	-	-	VCCIO3	3	-	-	VCCIO3	3	-	-
N16	VCCIO3	3	-	-	VCCIO3	3	-	-	VCCIO3	3	-	-
P16	VCCIO3	3	-	-	VCCIO3	3	-	-	VCCIO3	3	-	-
R16	VCCIO3	3	-	-	VCCIO3	3	-	-	VCCIO3	3	-	-
R15	VCCIO4	4	-	-	VCCIO4	4	-	-	VCCIO4	4	-	-
T12	VCCIO4	4	-	-	VCCIO4	4	-	-	VCCIO4	4	-	-
T13	VCCIO4	4	-	-	VCCIO4	4	-	-	VCCIO4	4	-	-
T14	VCCIO4	4	-	-	VCCIO4	4	-	-	VCCIO4	4	-	-
T15	VCCIO4	4	-	-	VCCIO4	4	-	-	VCCIO4	4	-	-
R8	VCCIO5	5	-	-	VCCIO5	5	-	-	VCCIO5	5	-	-
T10	VCCIO5	5	-	-	VCCIO5	5	-	-	VCCIO5	5	-	-
T11	VCCIO5	5	-	-	VCCIO5	5	-	-	VCCIO5	5	-	-
T8	VCCIO5	5	-	-	VCCIO5	5	-	-	VCCIO5	5	-	-
T9	VCCIO5	5	-	-	VCCIO5	5	-	-	VCCIO5	5	-	-
M7	VCCIO6	6	-	-	VCCIO6	6	-	-	VCCIO6	6	-	-
N7	VCCIO6	6	-	-	VCCIO6	6	-	-	VCCIO6	6	-	-
P7	VCCIO6	6	-	-	VCCIO6	6	-	-	VCCIO6	6	-	-
R7	VCCIO6	6	-	-	VCCIO6	6	-	-	VCCIO6	6	-	-
H7	VCCIO7	7	-	-	VCCIO7	7	-	-	VCCIO7	7	-	-
J7	VCCIO7	7	-	-	VCCIO7	7	-	-	VCCIO7	7	-	-
K7	VCCIO7	7	-	-	VCCIO7	7	-	-	VCCIO7	7	-	-
L7	VCCIO7	7	-	-	VCCIO7	7	-	-	VCCIO7	7	-	-

1. Applies to LFXP “C” only.
2. Applies to LFXP “E” only.
3. Supports dedicated LVDS outputs.

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
F5	PROGRAMN	7	-	-	PROGRAMN	7	-	-
E3	CCLK	7	-	-	CCLK	7	-	-
C1	PL2B	7	-	-	PL2B	7	-	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
G5	PL3A	7	T[3]	-	PL3A	7	T[3]	-
G6	PL3B	7	C[3]	-	PL3B	7	C[3]	-
F4	PL4A	7	T	-	PL4A	7	T	-
F3	PL4B	7	C	-	PL4B	7	C	-
G4	PL5A	7	T[3]	-	PL5A	7	T[3]	-
G3	PL5B	7	C[3]	-	PL5B	7	C[3]	-
D1	PL6A	7	T[3]	-	PL6A	7	T[3]	-
D2	PL6B	7	C[3]	-	PL6B	7	C[3]	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
E1	PL7A	7	T	LUM0_PLLT_FB_A	PL7A	7	T	LUM0_PLLT_FB_A
E2	PL7B	7	C	LUM0_PLLC_FB_A	PL7B	7	C	LUM0_PLLC_FB_A
H5	PL8A	7	T[3]	-	PL8A	7	T[3]	-
H6	PL8B	7	C[3]	-	PL8B	7	C[3]	-
H4	PL9A	7	-	-	PL9A	7	-	-
H3	PL10B	7	-	VREF1_7	PL10B	7	-	VREF1_7
F1	PL11A	7	T[3]	DQS	PL11A	7	T[3]	DQS
F2	PL11B	7	C[3]	-	PL11B	7	C[3]	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
J5	PL12A	7	T	-	PL12A	7	T	-
J6	PL12B	7	C	-	PL12B	7	C	-
G1	PL13A	7	T[3]	-	PL13A	7	T[3]	-
G2	PL13B	7	C[3]	-	PL13B	7	C[3]	-
J4	PL15A	7	T[3]	-	PL15A	7	T[3]	-
J3	PL15B	7	C[3]	-	PL15B	7	C[3]	-
-	GNDIO7	7	-	-	GNDIO7	7	-	-
H1	PL16A	7	T	LUM0_PLLT_IN_A	PL16A	7	T	LUM0_PLLT_IN_A
H2	PL16B	7	C	LUM0_PLLC_IN_A	PL16B	7	C	LUM0_PLLC_IN_A
J1	PL17A	7	T[3]	-	PL17A	7	T[3]	-
J2	PL17B	7	C[3]	-	PL17B	7	C[3]	-
K3	PL18A	7	-	VREF2_7	PL18A	7	-	VREF2_7
K2	PL19B	7	-	-	PL19B	7	-	-
K4	PL20A	7	T[3]	DQS	PL20A	7	T[3]	DQS
-	GNDIO7	7	-	-	GNDIO7	7	-	-
K5	PL20B	7	C[3]	-	PL20B	7	C[3]	-
K1	PL21A	7	T	-	PL21A	7	T	-
L2	PL21B	7	C	-	PL21B	7	C	-
L4	PL22A	7	T[3]	-	PL22A	7	T[3]	-
L3	PL22B	7	C[3]	-	PL22B	7	C[3]	-

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
L1	-	-	-	-	PL23A	7	T[3]	-
M1	-	-	-	-	PL23B	7	C[3]	-
M2	-	-	-	-	PL24A	7	-	-
L5	VCCP0	-	-	-	VCCP0	-	-	-
N2	GNDP0	-	-	-	GNDP0	-	-	-
N1	-	-	-	-	PL25B	6	-	-
P2	-	-	-	-	PL26A	6	T[3]	-
P1	-	-	-	-	PL26B	6	C[3]	-
M4	PL23A	6	T[3]	-	PL27A	6	T[3]	-
M3	PL23B	6	C[3]	-	PL27B	6	C[3]	-
R2	PL24A	6	T	PCLKT6_0	PL28A	6	T	PCLKT6_0
-	GNDIO6	6	-	-	GNDIO6	6	-	-
R1	PL24B	6	C	PCLKC6_0	PL28B	6	C	PCLKC6_0
N3	PL25A	6	T[3]	-	PL29A	6	T[3]	-
N4	PL25B	6	C[3]	-	PL29B	6	C[3]	-
M5	PL26A	6	-	-	PL30A	6	-	-
N5	PL27B	6	-	VREF1_6	PL31B	6	-	VREF1_6
T2	PL28A	6	T[3]	DQS	PL32A	6	T[3]	DQS
T1	PL28B	6	C[3]	-	PL32B	6	C[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
U2	PL29A	6	T	LLM0_PLLT_IN_A	PL33A	6	T	LLM0_PLLT_IN_A
U1	PL29B	6	C	LLM0_PLLC_IN_A	PL33B	6	C	LLM0_PLLC_IN_A
P3	PL30A	6	T[3]	-	PL34A	6	T[3]	-
P4	PL30B	6	C[3]	-	PL34B	6	C[3]	-
P6	PL32A	6	T[3]	-	PL36A	6	T[3]	-
P5	PL32B	6	C[3]	-	PL36B	6	C[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
V2	PL33A	6	T	-	PL37A	6	T	-
V1	PL33B	6	C	-	PL37B	6	C	-
W2	PL34A	6	T[3]	-	PL38A	6	T[3]	-
W1	PL34B	6	C[3]	-	PL38B	6	C[3]	-
R3	PL35A	6	-	VREF2_6	PL39A	6	-	VREF2_6
R4	PL36B	6	-	-	PL40B	6	-	-
R6	PL37A	6	T[3]	DQS	PL41A	6	T[3]	DQS
R5	PL37B	6	C[3]	-	PL41B	6	C[3]	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
Y2	PL38A	6	T	LLM0_PLLT_FB_A	PL42A	6	T	LLM0_PLLT_FB_A
Y1	PL38B	6	C	LLM0_PLLC_FB_A	PL42B	6	C	LLM0_PLLC_FB_A
T3	PL39A	6	T[3]	-	PL43A	6	T[3]	-
T4	PL39B	6	C[3]	-	PL43B	6	C[3]	-
W3	PL40A	6	T[3]	-	PL44A	6	T[3]	-
V3	PL40B	6	C[3]	-	PL44B	6	C[3]	-

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
T6	PL41A	6	T	-	PL45A	6	T	-
T5	PL41B	6	C	-	PL45B	6	C	-
-	GNDIO6	6	-	-	GNDIO6	6	-	-
U3	PL42A	6	T[3]	-	PL46A	6	T[3]	-
U4	PL42B	6	C[3]	-	PL46B	6	C[3]	-
V4	PL43A	6	-	-	PL47A	6	-	-
W4	SLEEPN[1]/ TOE[2]	-	-	-	SLEEPN[1]/ TOE[2]	-	-	-
W5	INITN	5	-	-	INITN	5	-	-
Y3	-	-	-	-	PB3B	5	-	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
U5	-	-	-	-	PB4A	5	T	-
V5	-	-	-	-	PB4B	5	C	-
Y4	-	-	-	-	PB5A	5	T	-
Y5	-	-	-	-	PB5B	5	C	-
V6	-	-	-	-	PB6A	5	T	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
U6	-	-	-	-	PB6B	5	C	-
W6	PB3A	5	T	-	PB7A	5	T	-
Y6	PB3B	5	C	-	PB7B	5	C	-
AA2	PB4A	5	T	-	PB8A	5	T	-
AA3	PB4B	5	C	-	PB8B	5	C	-
V7	PB5A	5	-	-	PB9A	5	-	-
U7	PB6B	5	-	-	PB10B	5	-	-
Y7	PB7A	5	T	DQS	PB11A	5	T	DQS
W7	PB7B	5	C	-	PB11B	5	C	-
AA4	PB8A	5	T	-	PB12A	5	T	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
AA5	PB8B	5	C	-	PB12B	5	C	-
AB3	PB9A	5	T	-	PB13A	5	T	-
AB4	PB9B	5	C	-	PB13B	5	C	-
AA6	PB10A	5	T	-	PB14A	5	T	-
AA7	PB10B	5	C	-	PB14B	5	C	-
U8	PB11A	5	T	-	PB15A	5	T	-
V8	PB11B	5	C	-	PB15B	5	C	-
Y8	PB12A	5	T	VREF1_5	PB16A	5	T	VREF1_5
-	GNDIO5	5	-	-	GNDIO5	5	-	-
W8	PB12B	5	C	-	PB16B	5	C	-
V9	PB13A	5	-	-	PB17A	5	-	-
U9	PB14B	5	-	-	PB18B	5	-	-
Y9	PB15A	5	T	DQS	PB19A	5	T	DQS
W9	PB15B	5	C	-	PB19B	5	C	-

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
AB5	PB16A	5	T	-	PB20A	5	T	-
AB6	PB16B	5	C	-	PB20B	5	C	-
AA8	PB17A	5	T	-	PB21A	5	T	-
AA9	PB17B	5	C	VREF2_5	PB21B	5	C	VREF2_5
W10	PB18A	5	T	-	PB22A	5	T	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
V10	PB18B	5	C	-	PB22B	5	C	-
AB7	PB19A	5	T	-	PB23A	5	T	-
AB8	PB19B	5	C	-	PB23B	5	C	-
AB9	PB20A	5	T	-	PB24A	5	T	-
AB10	PB20B	5	C	-	PB24B	5	C	-
Y10	PB21A	5	-	-	PB25A	5	-	-
AA10	PB22B	5	-	-	PB26B	5	-	-
W11	PB23A	5	T	DQS	PB27A	5	T	DQS
V11	PB23B	5	C	-	PB27B	5	C	-
-	GNDIO5	5	-	-	GNDIO5	5	-	-
Y11	PB24A	5	T	-	PB28A	5	T	-
AA11	PB24B	5	C	-	PB28B	5	C	-
AB11	PB25A	5	T	-	PB29A	5	T	-
AB12	PB25B	5	C	-	PB29B	5	C	-
Y12	PB26A	4	T	-	PB30A	4	T	-
AA12	PB26B	4	C	-	PB30B	4	C	-
W12	PB27A	4	T	PCLKT4_0	PB31A	4	T	PCLKT4_0
V12	PB27B	4	C	PCLKC4_0	PB31B	4	C	PCLKC4_0
-	GNDIO4	4	-	-	GNDIO4	4	-	-
AB13	PB28A	4	T	-	PB32A	4	T	-
AB14	PB28B	4	C	-	PB32B	4	C	-
AA13	PB29A	4	-	-	PB33A	4	-	-
Y13	PB30B	4	-	-	PB34B	4	-	-
AB15	PB31A	4	T	DQS	PB35A	4	T	DQS
AB16	PB31B	4	C	VREF1_4	PB35B	4	C	VREF1_4
V13	PB32A	4	T	-	PB36A	4	T	-
W13	PB32B	4	C	-	PB36B	4	C	-
AA14	PB33A	4	T	-	PB37A	4	T	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
AA15	PB33B	4	C	-	PB37B	4	C	-
AB17	PB34A	4	T	-	PB38A	4	T	-
AB18	PB34B	4	C	-	PB38B	4	C	-
W14	PB35A	4	T	-	PB39A	4	T	-
Y14	PB35B	4	C	-	PB39B	4	C	-
U14	PB36A	4	T	VREF2_4	PB40A	4	T	VREF2_4
V14	PB36B	4	C	-	PB40B	4	C	-

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
AB19	PB37A	4	-	-	PB41A	4	-	-
AB20	PB38B	4	-	-	PB42B	4	-	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
V15	PB39A	4	T	DQS	PB43A	4	T	DQS
U15	PB39B	4	C	-	PB43B	4	C	-
Y15	PB40A	4	T	-	PB44A	4	T	-
W15	PB40B	4	C	-	PB44B	4	C	-
AA16	PB41A	4	T	-	PB45A	4	T	-
AA17	PB41B	4	C	-	PB45B	4	C	-
AA18	PB42A	4	T	-	PB46A	4	T	-
AA19	PB42B	4	C	-	PB46B	4	C	-
Y16	PB43A	4	T	-	PB47A	4	T	-
W16	PB43B	4	C	-	PB47B	4	C	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
AA20	PB44A	4	T	-	PB48A	4	T	-
AA21	PB44B	4	C	-	PB48B	4	C	-
Y17	PB45A	4	-	-	PB49A	4	-	-
Y18	PB46B	4	-	-	PB50B	4	-	-
Y19	PB47A	4	T	DQS	PB51A	4	T	DQS
Y20	PB47B	4	C	-	PB51B	4	C	-
V16	PB48A	4	T	-	PB52A	4	T	-
U16	PB48B	4	C	-	PB52B	4	C	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
U18	-	-	-	-	PB53A	4	T	-
V18	-	-	-	-	PB53B	4	C	-
W19	-	-	-	-	PB54A	4	T	-
W18	-	-	-	-	PB54B	4	C	-
U17	-	-	-	-	PB55A	4	T	-
V17	-	-	-	-	PB55B	4	C	-
-	GNDIO4	4	-	-	GNDIO4	4	-	-
W17	-	-	-	-	PB56A	4	-	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-
V19	PR43A	3	-	-	PR47A	3	-	-
U20	PR42B	3	C[3]	-	PR46B	3	C[3]	-
U19	PR42A	3	T[3]	-	PR46A	3	T[3]	-
V20	PR41B	3	C	-	PR45B	3	C	-
W20	PR41A	3	T	-	PR45A	3	T	-
T17	PR40B	3	C[3]	-	PR44B	3	C[3]	-
T18	PR40A	3	T[3]	-	PR44A	3	T[3]	-
T19	PR39B	3	C[3]	-	PR43B	3	C[3]	-
T20	PR39A	3	T[3]	-	PR43A	3	T[3]	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
R18	PR38B	3	C	RLM0_PLLC_FB_A	PR42B	3	C	RLM0_PLLC_FB_A
R17	PR38A	3	T	RLM0_PLLT_FB_A	PR42A	3	T	RLM0_PLLT_FB_A
Y22	PR37B	3	C[3]	-	PR41B	3	C[3]	-
Y21	PR37A	3	T[3]	DQS	PR41A	3	T[3]	DQS
W22	PR36B	3	-	-	PR40B	3	-	-
W21	PR35A	3	-	VREF1_3	PR39A	3	-	VREF1_3
P17	PR34B	3	C[3]	-	PR38B	3	C[3]	-
P18	PR34A	3	T[3]	-	PR38A	3	T[3]	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-
R19	PR33B	3	C	-	PR37B	3	C	-
R20	PR33A	3	T	-	PR37A	3	T	-
V22	PR32B	3	C[3]	-	PR36B	3	C[3]	-
V21	PR32A	3	T[3]	-	PR36A	3	T[3]	-
U22	PR30B	3	C[3]	-	PR34B	3	C[3]	-
U21	PR30A	3	T[3]	-	PR34A	3	T[3]	-
P19	PR29B	3	C	RLM0_PLLC_IN_A	PR33B	3	C	RLM0_PLLC_IN_A
P20	PR29A	3	T	RLM0_PLLT_IN_A	PR33A	3	T	RLM0_PLLT_IN_A
-	GNDIO3	3	-	-	GNDIO3	3	-	-
T22	PR28B	3	C[3]	-	PR32B	3	C[3]	-
T21	PR28A	3	T[3]	DQS	PR32A	3	T[3]	DQS
R22	PR27B	3	-	-	PR31B	3	-	-
R21	PR26A	3	-	VREF2_3	PR30A	3	-	VREF2_3
N19	PR25B	3	C[3]	-	PR29B	3	C[3]	-
N20	PR25A	3	T[3]	-	PR29A	3	T[3]	-
N18	PR24B	3	C	-	PR28B	3	C	-
M18	PR24A	3	T	-	PR28A	3	T	-
-	GNDIO3	3	-	-	GNDIO3	3	-	-
P22	PR23B	3	C[3]	-	PR27B	3	C[3]	-
P21	PR23A	3	T[3]	-	PR27A	3	T[3]	-
N22	-	-	-	-	PR26B	3	C[3]	-
N21	-	-	-	-	PR26A	3	T[3]	-
M19	-	-	-	-	PR25B	3	-	-
M20	GNDP1	-	-	-	GNDP1	-	-	-
L18	VCCP1	-	-	-	VCCP1	-	-	-
M21	-	-	-	-	PR24A	2	-	-
M22	PR22B	2	C[3]	-	PR23B	2	C[3]	-
L22	PR22A	2	T[3]	-	PR23A	2	T[3]	-
-	GNDIO2	2	-	-	GNDIO2	2	-	-
L19	-	-	-	-	PR22B	2	C[3]	-
L20	-	-	-	-	PR22A	2	T[3]	-
L21	PR21B	2	C	PCLKC2_0	PR21B	2	C	PCLKC2_0
K22	PR21A	2	T	PCLKT2_0	PR21A	2	T	PCLKT2_0

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
J21	PR20B	2	C[3]	-	PR20B	2	C[3]	-
J22	PR20A	2	T[3]	DQS	PR20A	2	T[3]	DQS
K18	PR19B	2	-	-	PR19B	2	-	-
K19	PR18A	2	-	VREF1_2	PR18A	2	-	VREF1_2
-	GNDIO2	2	-	-	GNDIO2	2	-	-
K21	PR17B	2	C[3]	-	PR17B	2	C[3]	-
K20	PR17A	2	T[3]	-	PR17A	2	T[3]	-
H21	PR16B	2	C	RUM0_PLLC_IN_A	PR16B	2	C	RUM0_PLLC_IN_A
H22	PR16A	2	T	RUM0_PLLT_IN_A	PR16A	2	T	RUM0_PLLT_IN_A
J20	PR15B	2	C[3]	-	PR15B	2	C[3]	-
J19	PR15A	2	T[3]	-	PR15A	2	T[3]	-
-	GNDIO2	2	-	-	GNDIO2	2	-	-
J17	PR13B	2	C[3]	-	PR13B	2	C[3]	-
J18	PR13A	2	T[3]	-	PR13A	2	T[3]	-
G21	PR12B	2	C	-	PR12B	2	C	-
G22	PR12A	2	T	-	PR12A	2	T	-
F21	PR11B	2	C[3]	-	PR11B	2	C[3]	-
F22	PR11A	2	T[3]	DQS	PR11A	2	T[3]	DQS
-	GNDIO2	2	-	-	GNDIO2	2	-	-
H20	PR10B	2	-	-	PR10B	2	-	-
H19	PR9A	2	-	VREF2_2	PR9A	2	-	VREF2_2
H17	PR8B	2	C[3]	-	PR8B	2	C[3]	-
H18	PR8A	2	T[3]	-	PR8A	2	T[3]	-
E21	PR7B	2	C	RUM0_PLLC_FB_A	PR7B	2	C	RUM0_PLLC_FB_A
E22	PR7A	2	T	RUM0_PLLT_FB_A	PR7A	2	T	RUM0_PLLT_FB_A
D21	PR6B	2	C[3]	-	PR6B	2	C[3]	-
D22	PR6A	2	T[3]	-	PR6A	2	T[3]	-
G20	PR5B	2	C[3]	-	PR5B	2	C[3]	-
G19	PR5A	2	T[3]	-	PR5A	2	T[3]	-
G17	PR4B	2	C	-	PR4B	2	C	-
G18	PR4A	2	T	-	PR4A	2	T	-
-	GNDIO2	2	-	-	GNDIO2	2	-	-
F18	PR3B	2	C[3]	-	PR3B	2	C[3]	-
F19	PR3A	2	T[3]	-	PR3A	2	T[3]	-
C22	PR2B	2	-	-	PR2B	2	-	-
F20	TDO	-	-	-	TDO	-	-	-
E20	VCCJ	-	-	-	VCCJ	-	-	-
D19	TDI	-	-	-	TDI	-	-	-
E19	TMS	-	-	-	TMS	-	-	-
D20	TCK	-	-	-	TCK	-	-	-
C20	-	-	-	-	PT56A	1	-	-
-	GNDIO1	1	-	-	GNDIO1	1	-	-

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
D18	-	-	-	-	PT55B	1	C	-
E18	-	-	-	-	PT55A	1	T	-
C19	-	-	-	-	PT54B	1	C	-
C18	-	-	-	-	PT54A	1	T	-
C21	-	-	-	-	PT53B	1	C	-
-	GNDIO1	1	-	-	GNDIO1	1	-	-
B21	-	-	-	-	PT53A	1	T	-
E17	PT48B	1	C	-	PT52B	1	C	-
E16	PT48A	1	T	-	PT52A	1	T	-
C17	PT47B	1	C	-	PT51B	1	C	-
D17	PT47A	1	T	DQS	PT51A	1	T	DQS
F17	PT46B	1	-	-	PT50B	1	-	-
F16	PT45A	1	-	-	PT49A	1	-	-
C16	PT44B	1	C	-	PT48B	1	C	-
D16	PT44A	1	T	-	PT48A	1	T	-
A20	PT43B	1	C	-	PT47B	1	C	-
-	GNDIO1	1	-	-	GNDIO1	1	-	-
B20	PT43A	1	T	-	PT47A	1	T	-
A19	PT42B	1	C	-	PT46B	1	C	-
B19	PT42A	1	T	-	PT46A	1	T	-
C15	PT41B	1	C	-	PT45B	1	C	-
D15	PT41A	1	T	-	PT45A	1	T	-
A18	PT40B	1	C	-	PT44B	1	C	-
B18	PT40A	1	T	-	PT44A	1	T	-
F15	PT39B	1	C	VREF1_1	PT43B	1	C	VREF1_1
-	GNDIO1	1	-	-	GNDIO1	1	-	-
E15	PT39A	1	T	DQS	PT43A	1	T	DQS
A17	PT38B	1	-	-	PT42B	1	-	-
B17	PT37A	1	-	-	PT41A	1	-	-
E14	PT36B	1	C	-	PT40B	1	C	-
F14	PT36A	1	T	-	PT40A	1	T	-
D14	PT35B	1	C	-	PT39B	1	C	-
C14	PT35A	1	T	D0	PT39A	1	T	D0
A16	PT34B	1	C	D1	PT38B	1	C	D1
B16	PT34A	1	T	VREF2_1	PT38A	1	T	VREF2_1
A15	PT33B	1	C	-	PT37B	1	C	-
B15	PT33A	1	T	D2	PT37A	1	T	D2
-	GNDIO1	1	-	-	GNDIO1	1	-	-
E13	PT32B	1	C	D3	PT36B	1	C	D3
D13	PT32A	1	T	-	PT36A	1	T	-
C13	PT31B	1	C	-	PT35B	1	C	-
B13	PT31A	1	T	DQS	PT35A	1	T	DQS

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
A14	PT30B	1	-	-	PT34B	1	-	-
B14	PT29A	1	-	D4	PT33A	1	-	D4
C12	PT28B	1	C	-	PT32B	1	C	-
B12	PT28A	1	T	D5	PT32A	1	T	D5
-	GNDIO1	1	-	-	GNDIO1	1	-	-
D12	PT27B	1	C	D6	PT31B	1	C	D6
E12	PT27A	1	T	-	PT31A	1	T	-
A13	PT26B	1	C	D7	PT30B	1	C	D7
A12	PT26A	1	T	-	PT30A	1	T	-
A11	PT25B	0	C	BUSY	PT29B	0	C	BUSY
-	GNDIO0	0	-	-	GNDIO0	0	-	-
A10	PT25A	0	T	CS1N	PT29A	0	T	CS1N
D11	PT24B	0	C	PCLKC0_0	PT28B	0	C	PCLKC0_0
E11	PT24A	0	T	PCLKT0_0	PT28A	0	T	PCLKT0_0
B11	PT23B	0	C	-	PT27B	0	C	-
C11	PT23A	0	T	DQS	PT27A	0	T	DQS
B9	PT22B	0	-	-	PT26B	0	-	-
A9	PT21A	0	-	DOUT	PT25A	0	-	DOUT
B8	PT20B	0	C	-	PT24B	0	C	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
A8	PT20A	0	T	WRITEN	PT24A	0	T	WRITEN
E10	PT19B	0	C	-	PT23B	0	C	-
D10	PT19A	0	T	VREF1_0	PT23A	0	T	VREF1_0
C10	PT18B	0	C	-	PT22B	0	C	-
B10	PT18A	0	T	DI	PT22A	0	T	DI
B7	PT17B	0	C	-	PT21B	0	C	-
A7	PT17A	0	T	CSN	PT21A	0	T	CSN
C9	PT16B	0	C	-	PT20B	0	C	-
D9	PT16A	0	T	-	PT20A	0	T	-
B6	PT15B	0	C	VREF2_0	PT19B	0	C	VREF2_0
A6	PT15A	0	T	DQS	PT19A	0	T	DQS
F9	PT14B	0	-	-	PT18B	0	-	-
E9	PT13A	0	-	-	PT17A	0	-	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
B5	PT12B	0	C	-	PT16B	0	C	-
A5	PT12A	0	T	-	PT16A	0	T	-
C8	PT11B	0	C	-	PT15B	0	C	-
D8	PT11A	0	T	-	PT15A	0	T	-
B4	PT10B	0	C	-	PT14B	0	C	-
A4	PT10A	0	T	-	PT14A	0	T	-
F8	PT9B	0	C	-	PT13B	0	C	-
E8	PT9A	0	T	-	PT13A	0	T	-

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
B3	PT8B	0	C	-	PT12B	0	C	-
A3	PT8A	0	T	-	PT12A	0	T	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
D7	PT7B	0	C	-	PT11B	0	C	-
C7	PT7A	0	T	DQS	PT11A	0	T	DQS
B2	PT6B	0	-	-	PT10B	0	-	-
C2	PT5A	0	-	-	PT9A	0	-	-
C3	PT4B	0	C	-	PT8B	0	C	-
D3	PT4A	0	T	-	PT8A	0	T	-
F7	PT3B	0	C	-	PT7B	0	C	-
E7	PT3A	0	T	-	PT7A	0	T	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
C6	-	-	-	-	PT6B	0	C	-
D6	-	-	-	-	PT6A	0	T	-
C5	-	-	-	-	PT5B	0	C	-
C4	-	-	-	-	PT5A	0	T	-
F6	-	-	-	-	PT4B	0	C	-
E6	-	-	-	-	PT4A	0	T	-
-	GNDIO0	0	-	-	GNDIO0	0	-	-
E4	-	-	-	-	PT3B	0	-	-
E5	CFG0	0	-	-	CFG0	0	-	-
D4	CFG1	0	-	-	CFG1	0	-	-
D5	DONE	0	-	-	DONE	0	-	-
A1	GND	-	-	-	GND	-	-	-
A2	GND	-	-	-	GND	-	-	-
A21	GND	-	-	-	GND	-	-	-
A22	GND	-	-	-	GND	-	-	-
AA1	GND	-	-	-	GND	-	-	-
AA22	GND	-	-	-	GND	-	-	-
AB1	GND	-	-	-	GND	-	-	-
AB2	GND	-	-	-	GND	-	-	-
AB21	GND	-	-	-	GND	-	-	-
AB22	GND	-	-	-	GND	-	-	-
B1	GND	-	-	-	GND	-	-	-
B22	GND	-	-	-	GND	-	-	-
H14	GND	-	-	-	GND	-	-	-
H9	GND	-	-	-	GND	-	-	-
J10	GND	-	-	-	GND	-	-	-
J11	GND	-	-	-	GND	-	-	-
J12	GND	-	-	-	GND	-	-	-
J13	GND	-	-	-	GND	-	-	-
J14	GND	-	-	-	GND	-	-	-

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
J15	GND	-	-	-	GND	-	-	-
J8	GND	-	-	-	GND	-	-	-
J9	GND	-	-	-	GND	-	-	-
K10	GND	-	-	-	GND	-	-	-
K11	GND	-	-	-	GND	-	-	-
K12	GND	-	-	-	GND	-	-	-
K13	GND	-	-	-	GND	-	-	-
K14	GND	-	-	-	GND	-	-	-
K9	GND	-	-	-	GND	-	-	-
L10	GND	-	-	-	GND	-	-	-
L11	GND	-	-	-	GND	-	-	-
L12	GND	-	-	-	GND	-	-	-
L13	GND	-	-	-	GND	-	-	-
L14	GND	-	-	-	GND	-	-	-
L9	GND	-	-	-	GND	-	-	-
M10	GND	-	-	-	GND	-	-	-
M11	GND	-	-	-	GND	-	-	-
M12	GND	-	-	-	GND	-	-	-
M13	GND	-	-	-	GND	-	-	-
M14	GND	-	-	-	GND	-	-	-
M9	GND	-	-	-	GND	-	-	-
N10	GND	-	-	-	GND	-	-	-
N11	GND	-	-	-	GND	-	-	-
N12	GND	-	-	-	GND	-	-	-
N13	GND	-	-	-	GND	-	-	-
N14	GND	-	-	-	GND	-	-	-
N9	GND	-	-	-	GND	-	-	-
P10	GND	-	-	-	GND	-	-	-
P11	GND	-	-	-	GND	-	-	-
P12	GND	-	-	-	GND	-	-	-
P13	GND	-	-	-	GND	-	-	-
P14	GND	-	-	-	GND	-	-	-
P15	GND	-	-	-	GND	-	-	-
P8	GND	-	-	-	GND	-	-	-
P9	GND	-	-	-	GND	-	-	-
R14	GND	-	-	-	GND	-	-	-
R9	GND	-	-	-	GND	-	-	-
F10	VCC	-	-	-	VCC	-	-	-
F13	VCC	-	-	-	VCC	-	-	-
G10	VCC	-	-	-	VCC	-	-	-
G13	VCC	-	-	-	VCC	-	-	-
G14	VCC	-	-	-	VCC	-	-	-

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
G9	VCC	-	-	-	VCC	-	-	-
H15	VCC	-	-	-	VCC	-	-	-
H8	VCC	-	-	-	VCC	-	-	-
J16	VCC	-	-	-	VCC	-	-	-
J7	VCC	-	-	-	VCC	-	-	-
K16	VCC	-	-	-	VCC	-	-	-
K17	VCC	-	-	-	VCC	-	-	-
K6	VCC	-	-	-	VCC	-	-	-
K7	VCC	-	-	-	VCC	-	-	-
N16	VCC	-	-	-	VCC	-	-	-
N17	VCC	-	-	-	VCC	-	-	-
N6	VCC	-	-	-	VCC	-	-	-
N7	VCC	-	-	-	VCC	-	-	-
P16	VCC	-	-	-	VCC	-	-	-
P7	VCC	-	-	-	VCC	-	-	-
R15	VCC	-	-	-	VCC	-	-	-
R8	VCC	-	-	-	VCC	-	-	-
T10	VCC	-	-	-	VCC	-	-	-
T13	VCC	-	-	-	VCC	-	-	-
T14	VCC	-	-	-	VCC	-	-	-
T9	VCC	-	-	-	VCC	-	-	-
U10	VCC	-	-	-	VCC	-	-	-
U13	VCC	-	-	-	VCC	-	-	-
G15	VCCAUX	-	-	-	VCCAUX	-	-	-
G16	VCCAUX	-	-	-	VCCAUX	-	-	-
G7	VCCAUX	-	-	-	VCCAUX	-	-	-
G8	VCCAUX	-	-	-	VCCAUX	-	-	-
H16	VCCAUX	-	-	-	VCCAUX	-	-	-
H7	VCCAUX	-	-	-	VCCAUX	-	-	-
R16	VCCAUX	-	-	-	VCCAUX	-	-	-
R7	VCCAUX	-	-	-	VCCAUX	-	-	-
T15	VCCAUX	-	-	-	VCCAUX	-	-	-
T16	VCCAUX	-	-	-	VCCAUX	-	-	-
T7	VCCAUX	-	-	-	VCCAUX	-	-	-
T8	VCCAUX	-	-	-	VCCAUX	-	-	-
F11	VCCIO0	0	-	-	VCCIO0	0	-	-
G11	VCCIO0	0	-	-	VCCIO0	0	-	-
H10	VCCIO0	0	-	-	VCCIO0	0	-	-
H11	VCCIO0	0	-	-	VCCIO0	0	-	-
F12	VCCIO1	1	-	-	VCCIO1	1	-	-
G12	VCCIO1	1	-	-	VCCIO1	1	-	-
H12	VCCIO1	1	-	-	VCCIO1	1	-	-

LFXP15 & LFXP20 Logic Signal Connections: 484 fpBGA (Cont.)

Ball Number	LFXP15				LFXP20			
	Ball Function	Bank	Differential	Dual Function	Ball Function	Bank	Differential	Dual Function
H13	VCCIO1	1	-	-	VCCIO1	1	-	-
K15	VCCIO2	2	-	-	VCCIO2	2	-	-
L15	VCCIO2	2	-	-	VCCIO2	2	-	-
L16	VCCIO2	2	-	-	VCCIO2	2	-	-
L17	VCCIO2	2	-	-	VCCIO2	2	-	-
M15	VCCIO3	3	-	-	VCCIO3	3	-	-
M16	VCCIO3	3	-	-	VCCIO3	3	-	-
M17	VCCIO3	3	-	-	VCCIO3	3	-	-
N15	VCCIO3	3	-	-	VCCIO3	3	-	-
R12	VCCIO4	4	-	-	VCCIO4	4	-	-
R13	VCCIO4	4	-	-	VCCIO4	4	-	-
T12	VCCIO4	4	-	-	VCCIO4	4	-	-
U12	VCCIO4	4	-	-	VCCIO4	4	-	-
R10	VCCIO5	5	-	-	VCCIO5	5	-	-
R11	VCCIO5	5	-	-	VCCIO5	5	-	-
T11	VCCIO5	5	-	-	VCCIO5	5	-	-
U11	VCCIO5	5	-	-	VCCIO5	5	-	-
M6	VCCIO6	6	-	-	VCCIO6	6	-	-
M7	VCCIO6	6	-	-	VCCIO6	6	-	-
M8	VCCIO6	6	-	-	VCCIO6	6	-	-
N8	VCCIO6	6	-	-	VCCIO6	6	-	-
K8	VCCIO7	7	-	-	VCCIO7	7	-	-
L6	VCCIO7	7	-	-	VCCIO7	7	-	-
L7	VCCIO7	7	-	-	VCCIO7	7	-	-
L8	VCCIO7	7	-	-	VCCIO7	7	-	-

1. Applies to LFXP “C” only.
2. Applies to LFXP “E” only.
3. Supports dedicated LVDS outputs.

LatticeXP Family Data Sheet Ordering Information

December 2005 Data Sheet DS1001

Part Number Description

LFXP XX X – X XXXXXX X

Device Family
LatticeXP FPGA

Logic Capacity
3K LUTs = 3
6K LUTs = 6
10K LUTs = 10
15K LUTs = 15
20K LUTs = 20

Supply Voltage
C = 1.8V/2.5V/3.3V
E = 1.2V

Note: Parts dual marked per table below.

Grade
C = Commercial
I = Industrial

Package
T100 = 100-pin TQFP
T144 = 144-pin TQFP
Q208 = 208-pin PQFP
F256 = 256-ball fpBGA
F388 = 388-ball fpBGA
F484 = 484-ball fpBGA

TN100 = 100-pin Lead-free TQFP
TN144 = 144-pin Lead-free TQFP
QN208 = 208-pin Lead-free PQFP
FN256 = 256-ball Lead-free fpBGA
FN388 = 388-ball Lead-free fpBGA
FN484 = 484-ball Lead-free fpBGA

Speed
3 = Slowest
4
5 = Fastest

Ordering Information (Contact Factory for Specific Device Availability)

Note: LatticeXP devices are dual marked. For example, the commercial speed grade LFXP10E-4F256C is also marked with industrial grade -3I (LFXP10E-3F256I). The commercial grade is one speed grade faster than the associated dual mark industrial grade. The slowest commercial speed grade does not have industrial markings. The markings appear as follows:

Conventional Packaging

Commercial

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP3C-3Q208C	136	1.8/2.5/3.3V	-3	PQFP	208	COM	3.1K
LFXP3C-4Q208C	136	1.8/2.5/3.3V	-4	PQFP	208	COM	3.1K
LFXP3C-5Q208C	136	1.8/2.5/3.3V	-5	PQFP	208	COM	3.1K
LFXP3C-3T144C	100	1.8/2.5/3.3V	-3	TQFP	144	COM	3.1K
LFXP3C-4T144C	100	1.8/2.5/3.3V	-4	TQFP	144	COM	3.1K
LFXP3C-5T144C	100	1.8/2.5/3.3V	-5	TQFP	144	COM	3.1K
LFXP3C-3T100C	62	1.8/2.5/3.3V	-3	TQFP	100	COM	3.1K
LFXP3C-4T100C	62	1.8/2.5/3.3V	-4	TQFP	100	COM	3.1K
LFXP3C-5T100C	62	1.8/2.5/3.3V	-5	TQFP	100	COM	3.1K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP6C-3F256C	188	1.8/2.5/3.3V	-3	fpBGA	256	COM	5.8K
LFXP6C-4F256C	188	1.8/2.5/3.3V	-4	fpBGA	256	COM	5.8K
LFXP6C-5F256C	188	1.8/2.5/3.3V	-5	fpBGA	256	COM	5.8K
LFXP6C-3Q208C	142	1.8/2.5/3.3V	-3	PQFP	208	COM	5.8K
LFXP6C-4Q208C	142	1.8/2.5/3.3V	-4	PQFP	208	COM	5.8K
LFXP6C-5Q208C	142	1.8/2.5/3.3V	-5	PQFP	208	COM	5.8K
LFXP6C-3T144C	100	1.8/2.5/3.3V	-3	TQFP	144	COM	5.8K
LFXP6C-4T144C	100	1.8/2.5/3.3V	-4	TQFP	144	COM	5.8K
LFXP6C-5T144C	100	1.8/2.5/3.3V	-5	TQFP	144	COM	5.8K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP10C-3F388C	244	1.8/2.5/3.3V	-3	fpBGA	388	COM	9.7K
LFXP10C-4F388C	244	1.8/2.5/3.3V	-4	fpBGA	388	COM	9.7K
LFXP10C-5F388C	244	1.8/2.5/3.3V	-5	fpBGA	388	COM	9.7K
LFXP10C-3F256C	188	1.8/2.5/3.3V	-3	fpBGA	256	COM	9.7K
LFXP10C-4F256C	188	1.8/2.5/3.3V	-4	fpBGA	256	COM	9.7K
LFXP10C-5F256C	188	1.8/2.5/3.3V	-5	fpBGA	256	COM	9.7K

Commercial (Cont.)

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP15C-3F484C	300	1.8/2.5/3.3V	-3	fpBGA	484	COM	15.5K
LFXP15C-4F484C	300	1.8/2.5/3.3V	-4	fpBGA	484	COM	15.5K
LFXP15C-5F484C	300	1.8/2.5/3.3V	-5	fpBGA	484	COM	15.5K
LFXP15C-3F388C	268	1.8/2.5/3.3V	-3	fpBGA	388	COM	15.5K
LFXP15C-4F388C	268	1.8/2.5/3.3V	-4	fpBGA	388	COM	15.5K
LFXP15C-5F388C	268	1.8/2.5/3.3V	-5	fpBGA	388	COM	15.5K
LFXP15C-3F256C	188	1.8/2.5/3.3V	-3	fpBGA	256	COM	15.5K
LFXP15C-4F256C	188	1.8/2.5/3.3V	-4	fpBGA	256	COM	15.5K
LFXP15C-5F256C	188	1.8/2.5/3.3V	-5	fpBGA	256	COM	15.5K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP20C-3F484C	340	1.8/2.5/3.3V	-3	fpBGA	484	COM	19.7K
LFXP20C-4F484C	340	1.8/2.5/3.3V	-4	fpBGA	484	COM	19.7K
LFXP20C-5F484C	340	1.8/2.5/3.3V	-5	fpBGA	484	COM	19.7K
LFXP20C-3F388C	268	1.8/2.5/3.3V	-3	fpBGA	388	COM	19.7K
LFXP20C-4F388C	268	1.8/2.5/3.3V	-4	fpBGA	388	COM	19.7K
LFXP20C-5F388C	268	1.8/2.5/3.3V	-5	fpBGA	388	COM	19.7K
LFXP20C-3F256C	188	1.8/2.5/3.3V	-3	fpBGA	256	COM	19.7K
LFXP20C-4F256C	188	1.8/2.5/3.3V	-4	fpBGA	256	COM	19.7K
LFXP20C-5F256C	188	1.8/2.5/3.3V	-5	fpBGA	256	COM	19.7K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP3E-3Q208C	136	1.2V	-3	PQFP	208	COM	3.1K
LFXP3E-4Q208C	136	1.2V	-4	PQFP	208	COM	3.1K
LFXP3E-5Q208C	136	1.2V	-5	PQFP	208	COM	3.1K
LFXP3E-3T144C	100	1.2V	-3	TQFP	144	COM	3.1K
LFXP3E-4T144C	100	1.2V	-4	TQFP	144	COM	3.1K
LFXP3E-5T144C	100	1.2V	-5	TQFP	144	COM	3.1K
LFXP3E-3T100C	62	1.2V	-3	TQFP	100	COM	3.1K
LFXP3E-4T100C	62	1.2V	-4	TQFP	100	COM	3.1K
LFXP3E-5T100C	62	1.2V	-5	TQFP	100	COM	3.1K

Commercial (Cont.)

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP6E-3F256C	188	1.2V	-3	fpBGA	256	COM	5.8K
LFXP6E-4F256C	188	1.2V	-4	fpBGA	256	COM	5.8K
LFXP6E-5F256C	188	1.2V	-5	fpBGA	256	COM	5.8K
LFXP6E-3Q208C	142	1.2V	-3	PQFP	208	COM	5.8K
LFXP6E-4Q208C	142	1.2V	-4	PQFP	208	COM	5.8K
LFXP6E-5Q208C	142	1.2V	-5	PQFP	208	COM	5.8K
LFXP6E-3T144C	100	1.2V	-3	TQFP	144	COM	5.8K
LFXP6E-4T144C	100	1.2V	-4	TQFP	144	COM	5.8K
LFXP6E-5T144C	100	1.2V	-5	TQFP	144	COM	5.8K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP10E-3F388C	244	1.2V	-3	fpBGA	388	COM	9.7K
LFXP10E-4F388C	244	1.2V	-4	fpBGA	388	COM	9.7K
LFXP10E-5F388C	244	1.2V	-5	fpBGA	388	COM	9.7K
LFXP10E-3F256C	188	1.2V	-3	fpBGA	256	COM	9.7K
LFXP10E-4F256C	188	1.2V	-4	fpBGA	256	COM	9.7K
LFXP10E-5F256C	188	1.2V	-5	fpBGA	256	COM	9.7K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP15E-3F484C	300	1.2V	-3	fpBGA	484	COM	15.5K
LFXP15E-4F484C	300	1.2V	-4	fpBGA	484	COM	15.5K
LFXP15E-5F484C	300	1.2V	-5	fpBGA	484	COM	15.5K
LFXP15E-3F388C	268	1.2V	-3	fpBGA	388	COM	15.5K
LFXP15E-4F388C	268	1.2V	-4	fpBGA	388	COM	15.5K
LFXP15E-5F388C	268	1.2V	-5	fpBGA	388	COM	15.5K
LFXP15E-3F256C	188	1.2V	-3	fpBGA	256	COM	15.5K
LFXP15E-4F256C	188	1.2V	-4	fpBGA	256	COM	15.5K
LFXP15E-5F256C	188	1.2V	-5	fpBGA	256	COM	15.5K

Commercial (Cont.)

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP20E-3F484C	340	1.2V	-3	fpBGA	484	COM	19.7K
LFXP20E-4F484C	340	1.2V	-4	fpBGA	484	COM	19.7K
LFXP20E-5F484C	340	1.2V	-5	fpBGA	484	COM	19.7K
LFXP20E-3F388C	268	1.2V	-3	fpBGA	388	COM	19.7K
LFXP20E-4F388C	268	1.2V	-4	fpBGA	388	COM	19.7K
LFXP20E-5F388C	268	1.2V	-5	fpBGA	388	COM	19.7K
LFXP20E-3F256C	188	1.2V	-3	fpBGA	256	COM	19.7K
LFXP20E-4F256C	188	1.2V	-4	fpBGA	256	COM	19.7K
LFXP20E-5F256C	188	1.2V	-5	fpBGA	256	COM	19.7K

Industrial

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP3C-3Q208I	136	1.8/2.5/3.3V	-3	PQFP	208	IND	3.1K
LFXP3C-4Q208I	136	1.8/2.5/3.3V	-4	PQFP	208	IND	3.1K
LFXP3C-3T144I	100	1.8/2.5/3.3V	-3	TQFP	144	IND	3.1K
LFXP3C-4T144I	100	1.8/2.5/3.3V	-4	TQFP	144	IND	3.1K
LFXP3C-3T100I	62	1.8/2.5/3.3V	-3	TQFP	100	IND	3.1K
LFXP3C-4T100I	62	1.8/2.5/3.3V	-4	TQFP	100	IND	3.1K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP6C-3F256I	188	1.8/2.5/3.3V	-3	fpBGA	256	IND	5.8K
LFXP6C-4F256I	188	1.8/2.5/3.3V	-4	fpBGA	256	IND	5.8K
LFXP6C-3Q208I	142	1.8/2.5/3.3V	-3	PQFP	208	IND	5.8K
LFXP6C-4Q208I	142	1.8/2.5/3.3V	-4	PQFP	208	IND	5.8K
LFXP6C-3T144I	100	1.8/2.5/3.3V	-3	TQFP	144	IND	5.8K
LFXP6C-4T144I	100	1.8/2.5/3.3V	-4	TQFP	144	IND	5.8K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP10C-3F388I	244	1.8/2.5/3.3V	-3	fpBGA	388	IND	9.7K
LFXP10C-4F388I	244	1.8/2.5/3.3V	-4	fpBGA	388	IND	9.7K
LFXP10C-3F256I	188	1.8/2.5/3.3V	-3	fpBGA	256	IND	9.7K
LFXP10C-4F256I	188	1.8/2.5/3.3V	-4	fpBGA	256	IND	9.7K

Industrial (Cont.)

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP15C-3F484I	300	1.8/2.5/3.3V	-3	fpBGA	484	IND	15.5K
LFXP15C-4F484I	300	1.8/2.5/3.3V	-4	fpBGA	484	IND	15.5K
LFXP15C-3F388I	268	1.8/2.5/3.3V	-3	fpBGA	388	IND	15.5K
LFXP15C-4F388I	268	1.8/2.5/3.3V	-4	fpBGA	388	IND	15.5K
LFXP15C-3F256I	188	1.8/2.5/3.3V	-3	fpBGA	256	IND	15.5K
LFXP15C-4F256I	188	1.8/2.5/3.3V	-4	fpBGA	256	IND	15.5K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP20C-3F484I	340	1.8/2.5/3.3V	-3	fpBGA	484	IND	19.7K
LFXP20C-4F484I	340	1.8/2.5/3.3V	-4	fpBGA	484	IND	19.7K
LFXP20C-3F388I	268	1.8/2.5/3.3V	-3	fpBGA	388	IND	19.7K
LFXP20C-4F388I	268	1.8/2.5/3.3V	-4	fpBGA	388	IND	19.7K
LFXP20C-3F256I	188	1.8/2.5/3.3V	-3	fpBGA	256	IND	19.7K
LFXP20C-4F256I	188	1.8/2.5/3.3V	-4	fpBGA	256	IND	19.7K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP3E-3Q208I	136	1.2V	-3	PQFP	208	IND	3.1K
LFXP3E-4Q208I	136	1.2V	-4	PQFP	208	IND	3.1K
LFXP3E-3T144I	100	1.2V	-3	TQFP	144	IND	3.1K
LFXP3E-4T144I	100	1.2V	-4	TQFP	144	IND	3.1K
LFXP3E-3T100I	62	1.2V	-3	TQFP	100	IND	3.1K
LFXP3E-4T100I	62	1.2V	-4	TQFP	100	IND	3.1K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP6E-3F256I	188	1.2V	-3	fpBGA	256	IND	5.8K
LFXP6E-4F256I	188	1.2V	-4	fpBGA	256	IND	5.8K
LFXP6E-3Q208I	142	1.2V	-3	PQFP	208	IND	5.8K
LFXP6E-4Q208I	142	1.2V	-4	PQFP	208	IND	5.8K
LFXP6E-3T144I	100	1.2V	-3	TQFP	144	IND	5.8K
LFXP6E-4T144I	100	1.2V	-4	TQFP	144	IND	5.8K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP10E-3F388I	244	1.2V	-3	fpBGA	388	IND	9.7K
LFXP10E-4F388I	244	1.2V	-4	fpBGA	388	IND	9.7K
LFXP10E-3F256I	188	1.2V	-3	fpBGA	256	IND	9.7K
LFXP10E-4F256I	188	1.2V	-4	fpBGA	256	IND	9.7K

Industrial (Cont.)

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP15E-3F484I	300	1.2V	-3	fpBGA	484	IND	15.5K
LFXP15E-4F484I	300	1.2V	-4	fpBGA	484	IND	15.5K
LFXP15E-3F388I	268	1.2V	-3	fpBGA	388	IND	15.5K
LFXP15E-4F388I	268	1.2V	-4	fpBGA	388	IND	15.5K
LFXP15E-3F256I	188	1.2V	-3	fpBGA	256	IND	15.5K
LFXP15E-4F256I	188	1.2V	-4	fpBGA	256	IND	15.5K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP20E-3F484I	340	1.2V	-3	fpBGA	484	IND	19.7K
LFXP20E-4F484I	340	1.2V	-4	fpBGA	484	IND	19.7K
LFXP20E-3F388I	268	1.2V	-3	fpBGA	388	IND	19.7K
LFXP20E-4F388I	268	1.2V	-4	fpBGA	388	IND	19.7K
LFXP20E-3F256I	188	1.2V	-3	fpBGA	256	IND	19.7K
LFXP20E-4F256I	188	1.2V	-4	fpBGA	256	IND	19.7K

Lead-free Packaging

Commercial

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP3C-3QN208C	136	1.8/2.5/3.3V	-3	PQFP	208	COM	3.1K
LFXP3C-4QN208C	136	1.8/2.5/3.3V	-4	PQFP	208	COM	3.1K
LFXP3C-5QN208C	136	1.8/2.5/3.3V	-5	PQFP	208	COM	3.1K
LFXP3C-3TN144C	100	1.8/2.5/3.3V	-3	TQFP	144	COM	3.1K
LFXP3C-4TN144C	100	1.8/2.5/3.3V	-4	TQFP	144	COM	3.1K
LFXP3C-5TN144C	100	1.8/2.5/3.3V	-5	TQFP	144	COM	3.1K
LFXP3C-3TN100C	62	1.8/2.5/3.3V	-3	TQFP	100	COM	3.1K
LFXP3C-4TN100C	62	1.8/2.5/3.3V	-4	TQFP	100	COM	3.1K
LFXP3C-5TN100C	62	1.8/2.5/3.3V	-5	TQFP	100	COM	3.1K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP6C-3FN256C	188	1.8/2.5/3.3V	-3	fpBGA	256	COM	5.8K
LFXP6C-4FN256C	188	1.8/2.5/3.3V	-4	fpBGA	256	COM	5.8K
LFXP6C-5FN256C	188	1.8/2.5/3.3V	-5	fpBGA	256	COM	5.8K
LFXP6C-3QN208C	142	1.8/2.5/3.3V	-3	PQFP	208	COM	5.8K
LFXP6C-4QN208C	142	1.8/2.5/3.3V	-4	PQFP	208	COM	5.8K
LFXP6C-5QN208C	142	1.8/2.5/3.3V	-5	PQFP	208	COM	5.8K
LFXP6C-3TN144C	100	1.8/2.5/3.3V	-3	TQFP	144	COM	5.8K
LFXP6C-4TN144C	100	1.8/2.5/3.3V	-4	TQFP	144	COM	5.8K
LFXP6C-5TN144C	100	1.8/2.5/3.3V	-5	TQFP	144	COM	5.8K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP10C-3FN388C	244	1.8/2.5/3.3V	-3	fpBGA	388	COM	9.7K
LFXP10C-4FN388C	244	1.8/2.5/3.3V	-4	fpBGA	388	COM	9.7K
LFXP10C-5FN388C	244	1.8/2.5/3.3V	-5	fpBGA	388	COM	9.7K
LFXP10C-3FN256C	188	1.8/2.5/3.3V	-3	fpBGA	256	COM	9.7K
LFXP10C-4FN256C	188	1.8/2.5/3.3V	-4	fpBGA	256	COM	9.7K
LFXP10C-5FN256C	188	1.8/2.5/3.3V	-5	fpBGA	256	COM	9.7K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP15C-3FN484C	300	1.8/2.5/3.3V	-3	fpBGA	484	COM	15.5K
LFXP15C-4FN484C	300	1.8/2.5/3.3V	-4	fpBGA	484	COM	15.5K
LFXP15C-5FN484C	300	1.8/2.5/3.3V	-5	fpBGA	484	COM	15.5K
LFXP15C-3FN388C	268	1.8/2.5/3.3V	-3	fpBGA	388	COM	15.5K
LFXP15C-4FN388C	268	1.8/2.5/3.3V	-4	fpBGA	388	COM	15.5K
LFXP15C-5FN388C	268	1.8/2.5/3.3V	-5	fpBGA	388	COM	15.5K
LFXP15C-3FN256C	188	1.8/2.5/3.3V	-3	fpBGA	256	COM	15.5K
LFXP15C-4FN256C	188	1.8/2.5/3.3V	-4	fpBGA	256	COM	15.5K
LFXP15C-5FN256C	188	1.8/2.5/3.3V	-5	fpBGA	256	COM	15.5K

Commercial (Cont.)

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP20C-3FN484C	340	1.8/2.5/3.3V	-3	fpBGA	484	COM	19.7K
LFXP20C-4FN484C	340	1.8/2.5/3.3V	-4	fpBGA	484	COM	19.7K
LFXP20C-5FN484C	340	1.8/2.5/3.3V	-5	fpBGA	484	COM	19.7K
LFXP20C-3FN388C	268	1.8/2.5/3.3V	-3	fpBGA	388	COM	19.7K
LFXP20C-4FN388C	268	1.8/2.5/3.3V	-4	fpBGA	388	COM	19.7K
LFXP20C-5FN388C	268	1.8/2.5/3.3V	-5	fpBGA	388	COM	19.7K
LFXP20C-3FN256C	188	1.8/2.5/3.3V	-3	fpBGA	256	COM	19.7K
LFXP20C-4FN256C	188	1.8/2.5/3.3V	-4	fpBGA	256	COM	19.7K
LFXP20C-5FN256C	188	1.8/2.5/3.3V	-5	fpBGA	256	COM	19.7K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP3E-3QN208C	136	1.2V	-3	PQFP	208	COM	3.1K
LFXP3E-4QN208C	136	1.2V	-4	PQFP	208	COM	3.1K
LFXP3E-5QN208C	136	1.2V	-5	PQFP	208	COM	3.1K
LFXP3E-3TN144C	100	1.2V	-3	TQFP	144	COM	3.1K
LFXP3E-4TN144C	100	1.2V	-4	TQFP	144	COM	3.1K
LFXP3E-5TN144C	100	1.2V	-5	TQFP	144	COM	3.1K
LFXP3E-3TN100C	62	1.2V	-3	TQFP	100	COM	3.1K
LFXP3E-4TN100C	62	1.2V	-4	TQFP	100	COM	3.1K
LFXP3E-5TN100C	62	1.2V	-5	TQFP	100	COM	3.1K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP6E-3FN256C	188	1.2V	-3	fpBGA	256	COM	5.8K
LFXP6E-4FN256C	188	1.2V	-4	fpBGA	256	COM	5.8K
LFXP6E-5FN256C	188	1.2V	-5	fpBGA	256	COM	5.8K
LFXP6E-3QN208C	142	1.2V	-3	PQFP	208	COM	5.8K
LFXP6E-4QN208C	142	1.2V	-4	PQFP	208	COM	5.8K
LFXP6E-5QN208C	142	1.2V	-5	PQFP	208	COM	5.8K
LFXP6E-3TN144C	100	1.2V	-3	TQFP	144	COM	5.8K
LFXP6E-4TN144C	100	1.2V	-4	TQFP	144	COM	5.8K
LFXP6E-5TN144C	100	1.2V	-5	TQFP	144	COM	5.8K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP10E-3FN388C	244	1.2V	-3	fpBGA	388	COM	9.7K
LFXP10E-4FN388C	244	1.2V	-4	fpBGA	388	COM	9.7K
LFXP10E-5FN388C	244	1.2V	-5	fpBGA	388	COM	9.7K
LFXP10E-3FN256C	188	1.2V	-3	fpBGA	256	COM	9.7K
LFXP10E-4FN256C	188	1.2V	-4	fpBGA	256	COM	9.7K
LFXP10E-5FN256C	188	1.2V	-5	fpBGA	256	COM	9.7K

Commercial (Cont.)

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP15E-3FN484C	300	1.2V	-3	fpBGA	484	COM	15.5K
LFXP15E-4FN484C	300	1.2V	-4	fpBGA	484	COM	15.5K
LFXP15E-5FN484C	300	1.2V	-5	fpBGA	484	COM	15.5K
LFXP15E-3FN388C	268	1.2V	-3	fpBGA	388	COM	15.5K
LFXP15E-4FN388C	268	1.2V	-4	fpBGA	388	COM	15.5K
LFXP15E-5FN388C	268	1.2V	-5	fpBGA	388	COM	15.5K
LFXP15E-3FN256C	188	1.2V	-3	fpBGA	256	COM	15.5K
LFXP15E-4FN256C	188	1.2V	-4	fpBGA	256	COM	15.5K
LFXP15E-5FN256C	188	1.2V	-5	fpBGA	256	COM	15.5K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP20E-3FN484C	340	1.2V	-3	fpBGA	484	COM	19.7K
LFXP20E-4FN484C	340	1.2V	-4	fpBGA	484	COM	19.7K
LFXP20E-5FN484C	340	1.2V	-5	fpBGA	484	COM	19.7K
LFXP20E-3FN388C	268	1.2V	-3	fpBGA	388	COM	19.7K
LFXP20E-4FN388C	268	1.2V	-4	fpBGA	388	COM	19.7K
LFXP20E-5FN388C	268	1.2V	-5	fpBGA	388	COM	19.7K
LFXP20E-3FN256C	188	1.2V	-3	fpBGA	256	COM	19.7K
LFXP20E-4FN256C	188	1.2V	-4	fpBGA	256	COM	19.7K
LFXP20E-5FN256C	188	1.2V	-5	fpBGA	256	COM	19.7K

Industrial

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP3C-3QN208I	136	1.8/2.5/3.3V	-3	PQFP	208	IND	3.1K
LFXP3C-4QN208I	136	1.8/2.5/3.3V	-4	PQFP	208	IND	3.1K
LFXP3C-3TN144I	100	1.8/2.5/3.3V	-3	TQFP	144	IND	3.1K
LFXP3C-4TN144I	100	1.8/2.5/3.3V	-4	TQFP	144	IND	3.1K
LFXP3C-3TN100I	62	1.8/2.5/3.3V	-3	TQFP	100	IND	3.1K
LFXP3C-4TN100I	62	1.8/2.5/3.3V	-4	TQFP	100	IND	3.1K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP6C-3FN256I	188	1.8/2.5/3.3V	-3	fpBGA	256	IND	5.8K
LFXP6C-4FN256I	188	1.8/2.5/3.3V	-4	fpBGA	256	IND	5.8K
LFXP6C-3QN208I	142	1.8/2.5/3.3V	-3	PQFP	208	IND	5.8K
LFXP6C-4QN208I	142	1.8/2.5/3.3V	-4	PQFP	208	IND	5.8K
LFXP6C-3TN144I	100	1.8/2.5/3.3V	-3	TQFP	144	IND	5.8K
LFXP6C-4TN144I	100	1.8/2.5/3.3V	-4	TQFP	144	IND	5.8K

Industrial (Cont.)

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP10C-3FN388I	244	1.8/2.5/3.3V	-3	fpBGA	388	IND	9.7K
LFXP10C-4FN388I	244	1.8/2.5/3.3V	-4	fpBGA	388	IND	9.7K
LFXP10C-3FN256I	188	1.8/2.5/3.3V	-3	fpBGA	256	IND	9.7K
LFXP10C-4FN256I	188	1.8/2.5/3.3V	-4	fpBGA	256	IND	9.7K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP15C-3FN484I	300	1.8/2.5/3.3V	-3	fpBGA	484	IND	15.5K
LFXP15C-4FN484I	300	1.8/2.5/3.3V	-4	fpBGA	484	IND	15.5K
LFXP15C-3FN388I	268	1.8/2.5/3.3V	-3	fpBGA	388	IND	15.5K
LFXP15C-4FN388I	268	1.8/2.5/3.3V	-4	fpBGA	388	IND	15.5K
LFXP15C-3FN256I	188	1.8/2.5/3.3V	-3	fpBGA	256	IND	15.5K
LFXP15C-4FN256I	188	1.8/2.5/3.3V	-4	fpBGA	256	IND	15.5K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP20C-3FN484I	340	1.8/2.5/3.3V	-3	fpBGA	484	IND	19.7K
LFXP20C-4FN484I	340	1.8/2.5/3.3V	-4	fpBGA	484	IND	19.7K
LFXP20C-3FN388I	268	1.8/2.5/3.3V	-3	fpBGA	388	IND	19.7K
LFXP20C-4FN388I	268	1.8/2.5/3.3V	-4	fpBGA	388	IND	19.7K
LFXP20C-3FN256I	188	1.8/2.5/3.3V	-3	fpBGA	256	IND	19.7K
LFXP20C-4FN256I	188	1.8/2.5/3.3V	-4	fpBGA	256	IND	19.7K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP3E-3QN208I	136	1.2V	-3	PQFP	208	IND	3.1K
LFXP3E-4QN208I	136	1.2V	-4	PQFP	208	IND	3.1K
LFXP3E-3TN144I	100	1.2V	-3	TQFP	144	IND	3.1K
LFXP3E-4TN144I	100	1.2V	-4	TQFP	144	IND	3.1K
LFXP3E-3TN100I	62	1.2V	-3	TQFP	100	IND	3.1K
LFXP3E-4TN100I	62	1.2V	-4	TQFP	100	IND	3.1K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP6E-3FN256I	188	1.2V	-3	fpBGA	256	IND	5.8K
LFXP6E-4FN256I	188	1.2V	-4	fpBGA	256	IND	5.8K
LFXP6E-3QN208I	142	1.2V	-3	PQFP	208	IND	5.8K
LFXP6E-4QN208I	142	1.2V	-4	PQFP	208	IND	5.8K
LFXP6E-3TN144I	100	1.2V	-3	TQFP	144	IND	5.8K
LFXP6E-4TN144I	100	1.2V	-4	TQFP	144	IND	5.8K

Industrial (Cont.)

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP10E-3FN388I	244	1.2V	-3	fpBGA	388	IND	9.7K
LFXP10E-4FN388I	244	1.2V	-4	fpBGA	388	IND	9.7K
LFXP10E-3FN256I	188	1.2V	-3	fpBGA	256	IND	9.7K
LFXP10E-4FN256I	188	1.2V	-4	fpBGA	256	IND	9.7K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP15E-3FN484I	300	1.2V	-3	fpBGA	484	IND	15.5K
LFXP15E-4FN484I	300	1.2V	-4	fpBGA	484	IND	15.5K
LFXP15E-3FN388I	268	1.2V	-3	fpBGA	388	IND	15.5K
LFXP15E-4FN388I	268	1.2V	-4	fpBGA	388	IND	15.5K
LFXP15E-3FN256I	188	1.2V	-3	fpBGA	256	IND	15.5K
LFXP15E-4FN256I	188	1.2V	-4	fpBGA	256	IND	15.5K

Part Number	I/Os	Voltage	Grade	Package	Pins	Temp.	LUTs
LFXP20E-3FN484I	340	1.2V	-3	fpBGA	484	IND	19.7K
LFXP20E-4FN484I	340	1.2V	-4	fpBGA	484	IND	19.7K
LFXP20E-3FN388I	268	1.2V	-3	fpBGA	388	IND	19.7K
LFXP20E-4FN388I	268	1.2V	-4	fpBGA	388	IND	19.7K
LFXP20E-3FN256I	188	1.2V	-3	fpBGA	256	IND	19.7K
LFXP20E-4FN256I	188	1.2V	-4	fpBGA	256	IND	19.7K

LatticeXP Family Data Sheet Supplemental Information

December 2005 **Data Sheet DS1001**

For Further Information

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

- *LatticeECP/EC and LatticeXP sysIO Usage Guide* (TN1056)
- *Lattice ispTRACY Usage Guide* (TN1054)
- *LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide* (TN1049)
- *Memory Usage Guide for LatticeECP/EC and LatticeXP Devices* (TN1051)
- *LatticeECP/EC and XP DDR Usage Guide* (TN1050)
- *Estimating Power Using Power Calculator for LatticeECP/EC and LatticeXP Devices* (TN1052)
- *LatticeXP sysCONFIG Usage Guide* (TN1082)

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

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

LatticeXP Family Data Sheet Revision History

February 2007 **Data Sheet DS1001**

Revision History

Date	Version	Section	Change Summary
February 2005	01.0	—	Initial release.
April 2005	01.1	Architecture	EBR memory support section updated with clarification.
May 2005	01.2	Introduction	Added TransFR Reconfiguration to Features section.
		Architecture	Added TransFR section.
June 2005	01.3	Pinout Information	Added pinout information for LFXP3, LFXP6, LFXP15 and LFXP20.
July 2005	02.0	Introduction	Updated XP6, XP15 and XP20 EBR SRAM Bits and Block numbers.
		Architecture	Updated Per Quadrant Primary Clock Selection figure.
			Added Typical I/O Behavior During Power-up section.
			Updated Device Configuration section under Configuration and Testing.
		DC and Switching Characteristics	Clarified Hot Socketing Specification
			Updated Supply Current (Standby) Table
			Updated Initialization Supply Current Table
			Added Programming and Erase Flash Supply Current table
			Added LVDS Emulation section. Updated LVDS25E Output Termination Example figure and LVDS25E DC Conditions table.
			Updated Differential LVPECL diagram and LVPECL DC Conditions table.
			Deleted 5V Tolerant Input Buffer section. Updated RSDS figure and RSDS DC Conditions table.
			Updated sysCONFIG Port Timing Specifications
			Updated JTAG Port Timing Specifications. Added Flash Download Time table.
		Pinout Information	Updated Signal Descriptions table.
			Updated Logic Signal Connections Dual Function column.
		Ordering Information	Added lead-free ordering part numbers.
July 2005	02.1	DC and Switching Characteristics	Clarification of Flash Programming Junction Temperature
August 2005	02.2	Introduction	Added Sleep Mode feature.
		Architecture	Added Sleep Mode section.
		DC and Switching Characteristics	Added Sleep Mode Supply Current Table
			Added Sleep Mode Timing section
		Pinout Information	Added SLEEPN and TOE signal names, descriptions and footnotes.
			Added SLEEPN and TOE to pinout information and footnotes.
			Added footnote 3 to Logic Signal Connections tables for clarification on emulated LVDS output.
September 2005	03.0	Architecture	Added clarification of PCI clamp.
			Added clarification to SLEEPN Pin Characteristics section.
		DC and Switching Characteristics	DC Characteristics, added footnote 4 for clarification. Updated Supply Current (Sleep Mode), Supply Current (Standby), Initialization Supply Current, and Programming and Erase Flash Supply Current typical numbers.

Date	Version	Section	Change Summary
September 2005 (cont.)	03.0 (cont.)	DC and Switching Characteristics (cont.)	Updated Typical Building Block Function Performance timing numbers.
			Updated External Switching Characteristics timing numbers.
			Updated Internal Timing Parameters.
			Updated LatticeXP Family timing adders.
			Updated LatticeXP "C" Sleep Mode timing numbers.
			Updated JTAG Port Timing numbers.
		Pinout Information	Added clarification to SLEEPN and TOE description.
			Clarification of dedicated LVDS outputs.
		Supplemental Information	Updated list of technical notes.
September 2005	03.1	Pinout Information	Power Supply and NC Connections table corrected VCCP1 pin number for 208 PQFP.
December 2005	04.0	Introduction	Moved data sheet from Advance to Final.
		Architecture	Added clarification to Typical I/O Behavior During Power-up section.
		DC and Switching Characteristics	Added clarification to Recommended Operating Conditions.
			Updated timing numbers.
		Pinout Information	Updated Signal Descriptions table.
			Added clarification to Differential I/O Per Bank.
			Updated Differential dedicated LVDS output support.
		Ordering Information	Added 208 PQFP lead-free package and ordering part numbers.
February 2006	04.1	Pinout Information	Corrected description of Signal Names VREF1(x) and VREF2(x).
March 2006	04.2	DC and Switching Characteristics	Corrected condition for IIL and IIH.
March 2006	04.3	DC and Switching Characteristics	Added clarification to Recommended Operating Conditions for VCCAUX.
April 2006	04.4	Pinout Information	Removed Bank designator "5" from SLEEPN/TOE ball function.
May 2006	04.5	DC and Switching Characteristics	Added footnote 2 regarding threshold level for PROGRAMN to sysCONFIG Port Timing Specifications table.
June 2006	04.6	DC and Switching Characteristics	Corrected LVDS25E Output Termination Example.
August 2006	04.7	Architecture	Added clarification to Typical I/O Behavior During Power-Up section.
			Added clarification to Left and Right sysIO Buffer Pair section.
		DC and Switching Characteristics	Changes to LVDS25E Output Termination Example diagram.
December 2006	04.8	Architecture	EBR Asynchronous Reset section added.
February 2007	04.9	Architecture	Updated EBR Asynchronous Reset section.

Section II. LatticeXP Family Technical Notes

LatticeECP/EC and LatticeXP sysIO Usage Guide

March 2006 Technical Note TN1056

Introduction

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

sysIO Buffer Overview

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

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

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

Supported sysIO Standards

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

Table 8-1. Supported sysIO Standards

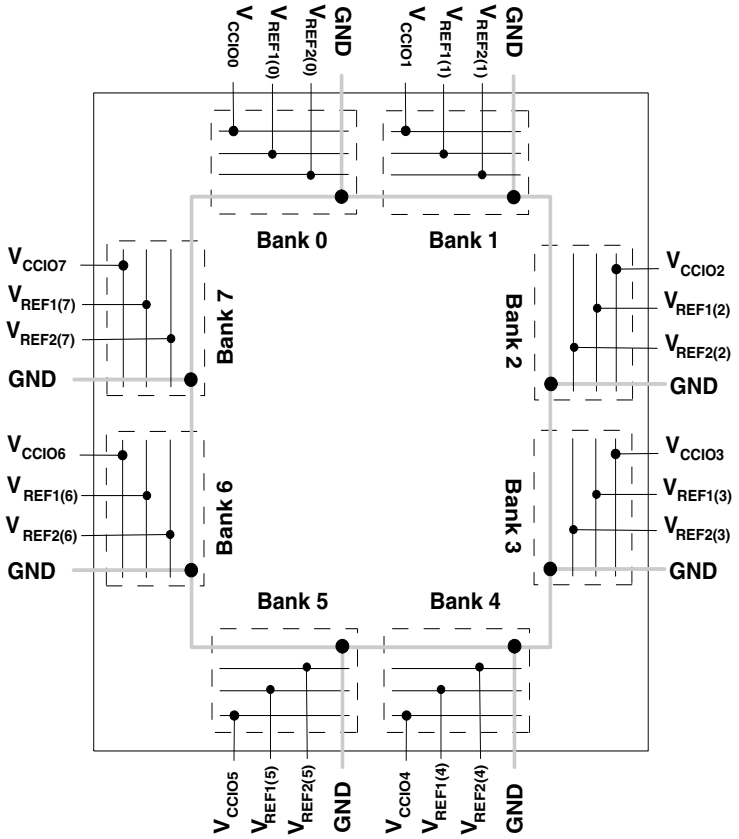
Standard	V_{CCIO}			V_{REF} (V)		
	Min.	Typ.	Max.	Min.	Typ.	Max.
LVCMOS 3.3	3.135	3.3	3.465	—	—	—
LVCMOS 2.5	2.375	2.5	2.625	—	—	—
LVCMOS 1.8	1.71	1.8	1.89	—	—	—
LVCMOS 1.5	1.425	1.5	1.575	—	—	—
LVCMOS 1.2	1.14	1.2	1.26	—	—	—
LVTTL	3.135	3.3	3.465	—	—	—
PCI	3.135	3.3	3.465	—	—	—
SSTL18 Class I	1.71	2.5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

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

sysIO Banking Scheme

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

Figure 8-1. sysIO Banking



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

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

V_{CCAUX} (3.3V)

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

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

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

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

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

V_{REF1} for DDR Memory Interface

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

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

Mixed Voltage Support in a Bank

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

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

Table 8-2. Mixed Voltage Support

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

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

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

LVCMOS Buffer Configurations

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

Programmable Pull-up/Pull-Down/Buskeeper

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

Programmable Drive

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

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

Table 8-4 shows the drive current setting required to match 50Ω transmission line with 50Ω and 200Ω terminations.

Table 8-4. Impedance Matching Using Programmable Drive Strength

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

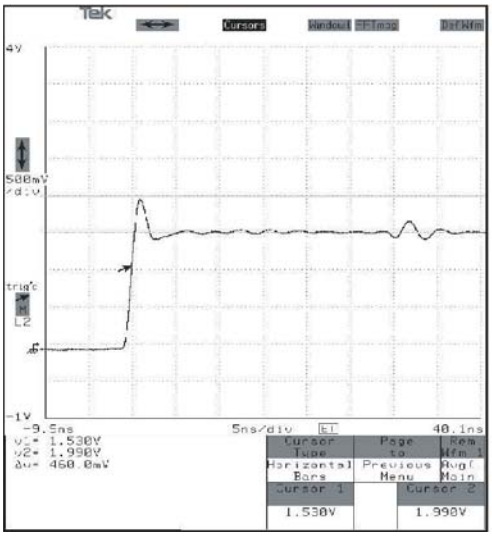
The actual impedance matching may vary on the transmission line design and the load. To find the best matching, it is recommended to drive the transmission line with different combinations of I/O standards and drive strengths that best match the line impedance. Lattice provides IBIS buffer models for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

The figure below shows how this impedance matching is done for a 50Ω transmission line with 200Ω termination using LVCMOS18 I/O buffers programmed to drive 16mA, 12mA, 8mA and 4mA. From this experiment it is empirical that the best matching is achieved with the 8mA drive setting.

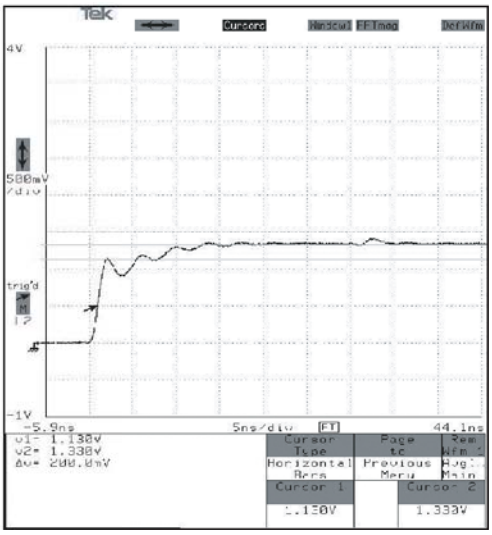
Figure 8-2. Impedance Matching for a 50Ω Transmission Line with 200Ω Termination

LVCMOS18 16mA **LVCMOS18 12mA**

Figure 7-2. Impedance Matching for a 50Ω Transmission Line with 200Ω Termination (Cont.)



LVCMOS18 8mA



LVCMOS18 4mA

Programmable Slew Rate

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

Open Drain Control

All LVCMOS and LVTTL output buffers can be configured to function as open drain outputs. The user can implement an open drain output by turning on the OPENDRAIN attribute in the software.

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

Differential SSTL and HSTL Support

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

PCI Support with Programmable PCICLAMP

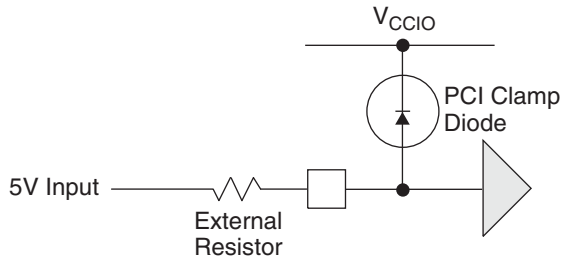
Each sysIO buffer can be configured to support PCI33. The buffers on the top and bottom of the device have an optional PCI clamp diode that may optionally be specified in the ispLEVER® design tool.

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

5V Interface with PCI Clamp Diode

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

Figure 8-3. 5V Tolerant Input Buffer



The value of this external resistor will depend on the PCI clamp diode characteristics. You can find the voltage vs. current data across this diode in the device IBIS model.

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

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

- Maximum voltage at input pin, V_{INMAX} = 3.75V (see device data sheet for more details)
- Bank V_{CCIO} = 2.75V
- Maximum voltage drop across clamp diode, $V_D = V_{INMAX} - V_{CCIO} = 3.75 - 2.75 = 1V$
- The current across the clamp diode at V_D can be found in the power clamp data of the IBIS file. Below is the power clamp portion of the IBIS file for a LVCMOS3.3 input model with PCI Clamp turned on. When V_D is 1V, the clamp diode current is I_D = 27.4mA.

Table 8-5. Power Clamp Data from IBIS Model

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

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

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

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

Programmable Input Delay

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

Software sysIO Attributes

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

IO_TYPE

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

Table 8-6. I/O_TYPE Attribute Values

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

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

OPENDRAIN

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

Values: ON, OFF
Default: OFF

DRIVE

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

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

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

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

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

PULLMODE

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

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

PCICLAMP

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

Values: ON, OFF
Default: OFF

SLEWRATE

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

Values: FAST, SLOW
Default: FAST

FIXEDDELAY

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

Values: TRUE, FALSE
Default: FALSE

DIN/DOUT

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

LOC

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

Design Considerations and Usage

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

Banking Rules

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

Differential I/O Rules

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

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

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

Preference Syntax

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

Example of VREF Groups

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

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

LOCATE VREF “ref1” SITE PR29C;
LOCATE VREF “ref2” SITE PR48B;
```

or

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

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

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

Differential I/O Implementation

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

LVDS

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

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

BLVDS

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

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

RSDS

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

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

LVPECL

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

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

Differential SSTL and HSTL

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

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

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

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

VHDL Synplify/Precision RTL Synthesis

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

Syntax

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

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

Examples

IO_TYPE

--*Attribute Declaration*****

ATTRIBUTE IO_TYPE: string;

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

ATTRIBUTE IO_TYPE OF portA: SIGNAL IS “PCI33”;

ATTRIBUTE IO_TYPE OF portB: SIGNAL IS “LVCMOS33”;

ATTRIBUTE IO_TYPE OF portC: SIGNAL IS “LVDS25”;

OPENDRAIN

--*Attribute Declaration***

ATTRIBUTE OPENDRAIN: string;

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

ATTRIBUTE OPENDRAIN OF portB: SIGNAL IS "ON";

DRIVE
--*Attribute Declaration***

ATTRIBUTE DRIVE: string;

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

ATTRIBUTE DRIVE OF portB: SIGNAL IS “20”;

PULLMODE

--*Attribute Declaration***

ATTRIBUTE PULLMODE : string;

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

ATTRIBUTE PULLMODE OF portA: SIGNAL IS "DOWN";

ATTRIBUTE PULLMODE OF portB: SIGNAL IS "UP";

PCICLAMP

--***Attribute Declaration***

ATTRIBUTE PCICLAMP: string;

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

ATTRIBUTE PCICLAMP OF portA: SIGNAL IS "ON";

SLEWRATE
--*Attribute Declaration***

ATTRIBUTE SLEWRATE : string;

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

ATTRIBUTE SLEWRATE OF portB: SIGNAL IS “FAST”;

FIXEDDELAY
--*Attribute Declaration***

ATTRIBUTE FIXEDDELAY: string;

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

ATTRIBUTE FIXEDDELAY OF portB: SIGNAL IS “TRUE”;

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

ATTRIBUTE din : string;

ATTRIBUTE dout : string;

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

ATTRIBUTE din OF input_vector: SIGNAL IS " ";

ATTRIBUTE dout OF output_vector: SIGNAL IS " ";

LOC
--*Attribute Declaration*****

ATTRIBUTE LOC : string;

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

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

Verilog for Synplify

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

Syntax

Table 8-9. Verilog Synplify Attribute Syntax

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

Examples

//IO_TYPE, PULLMODE, SLEWRATE and DRIVE assignment

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

DRIVE =”20”*/;

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

//OPENDRAIN

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

//PCICLAMP

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

// Fixeddelay

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

// Place the flip-flops near the load input

input load /* synthesis din=”” */;

// Place the flip-flops near the outload output

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

//I/O pin location

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

//Register pin location

```
reg data_in_ch1_buf_reg3 /* synthesis loc=”R40C47” */;
```

//Vectored internal bus

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

Verilog for Precision RTL Synthesis

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

Syntax

Table 8-10. Verilog Precision RTL Synthesis Attribute Syntax

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

Example

//**IO_TYPE ***

//pragma attribute portA IO_TYPE PCI33

//pragma attribute portB IO_TYPE LVCMOS33

//pragma attribute portC IO_TYPE SSTL25_II

//* Opendrain ***

//pragma attribute portB OPENDRAIN ON

//pragma attribute portD OPENDRAIN OFF

//* Drive ***

//pragma attribute portB DRIVE 20

//pragma attribute portD DRIVE 8

//* Pullmode***

//pragma attribute portB PULLMODE UP

//* PCIClamp***

//pragma attribute portB PCICLAMP ON

//* Slewrate ***

//pragma attribute portB SLEWRATE FAST

//pragma attribute portD SLEWRATE SLOW

// ***Fixeddelay***

// pragma attribute load FIXEDDELAY TRUE

//*LOC*****

//pragma attribute portB loc E3

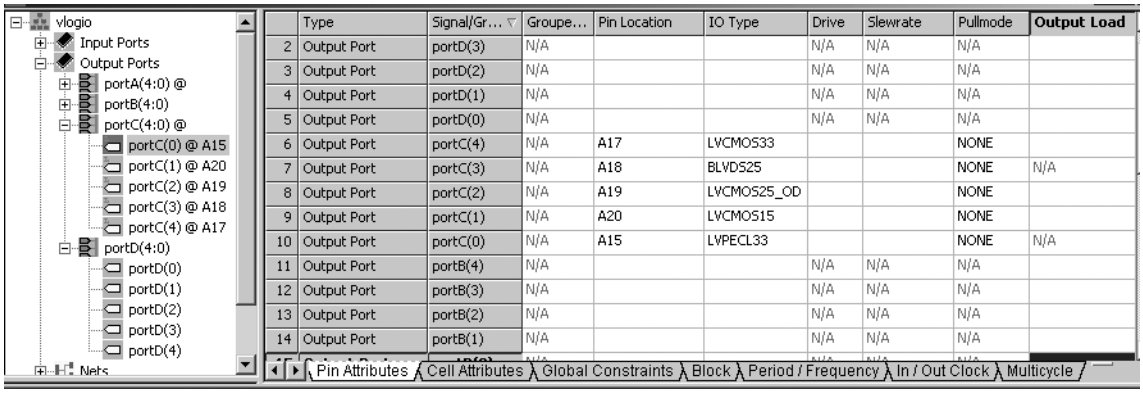
Appendix B. sysIO Attributes Using Preference Editor User Interface

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

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

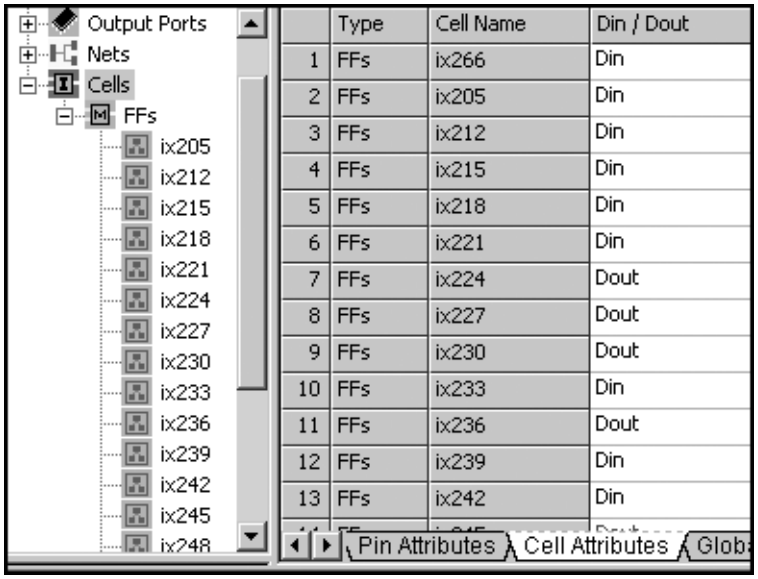
Figure 8-4 and Figure 8-5 show the Pin Attribute Sheet and the Cell Attribute Sheet views of the Preference Editor. For further information on how to use the Preference Editor, refer to the ispLEVER Help documentation located in the Help menu option of the software.

Figure 8-4. Pin Attributes Tab



	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 8-5. 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, OPENDRAIN, DRIVE, PULLMODE, PCICLAMP, SLEWRATE.

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

PCICLAMP =OFF SLEWRATE=FAST;

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

IOBUF GROUP "bank1" IO_TYPE=SSTL18_II;

LOCATE

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

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

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

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

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

<bus_name> := string

<number> := integer

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

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

LOCATE COMP “Clk0” SITE “A4”;

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

LOCATE COMP “PFU1” SITE “R1C7”;

This command places bus1 on ROW 3 and bus2 on COL4

LOCATE BUS “bus1” ROW 3;

LOCATE BUS “bus2” COL 4;

USE DIN CELL

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

Syntax
USE DIN CELL <cell_name>;

where:

<cell_name> := string

Example

USE DIN CELL “din0”;

USE DOUT CELL

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

Syntax
USE DOUT CELL <cell_name>;

where:

<cell_name> := string

Examples
USE DOUT CELL “dout1”;

PGROUP VREF

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

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

LOCATE PGROUP <pgrp_name> BANK <bank_num>;

LOCATE VREF <vref_name> SITE <site_name>;

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

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

LOCATE VREF “ref1” SITE PR29C;

LOCATE VREF “ref2” SITE PR48B;

or

LOCATE PGROUP “ vref_pg1” BANK 2;

LOCATE PGROUP “ vref_pg2” BANK 2;

Memory Usage Guide for LatticeECP/EC and LatticeXP Devices

October 2006 Technical Note TN1051

Introduction

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

The architecture of the LatticeECP/EC and LatticeXP devices provides a large amount of resources for memory intensive applications. The sysMEM™ Embedded Block RAM (EBR) complements its distributed PFU-based memory. Single-Port RAM, Dual-Port RAM, Pseudo Dual-Port RAM and ROM memories can be constructed using the EBR. LUTs and PFU can implement Distributed Single-Port RAM, Dual-Port RAM and ROM. The internal logic of the device can be used to configure the memory elements as FIFO and other storage types.

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

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

Memories in LatticeECP/EC and LatticeXP Devices

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

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

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

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

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

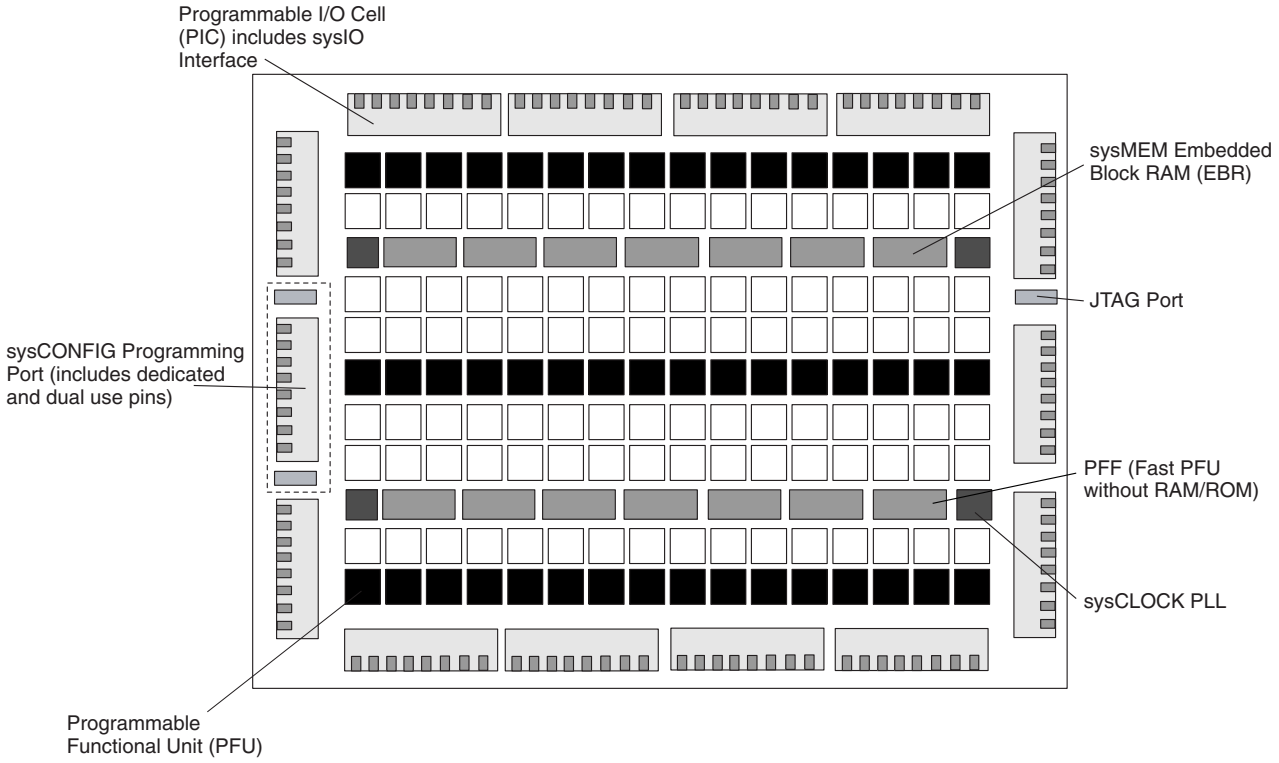


Figure 9-2. Simplified Block Diagram, LatticeECP Device (Top Level)

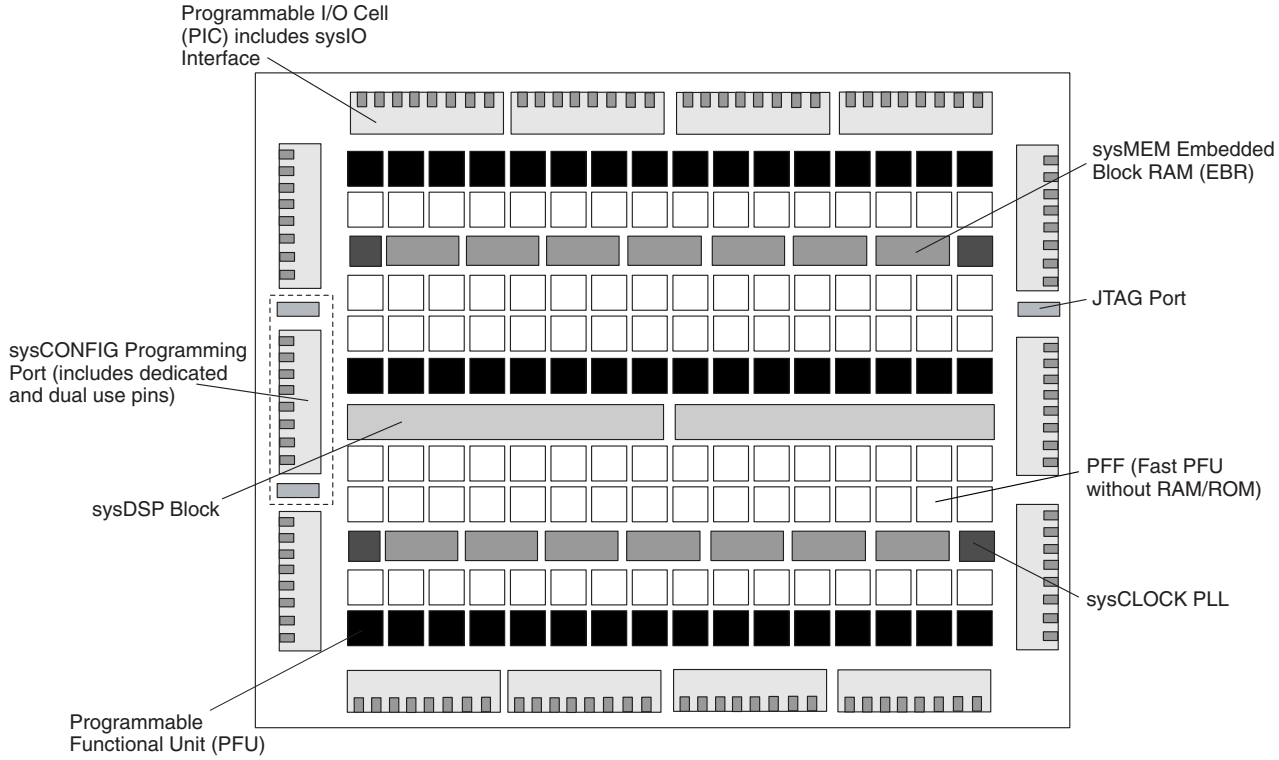
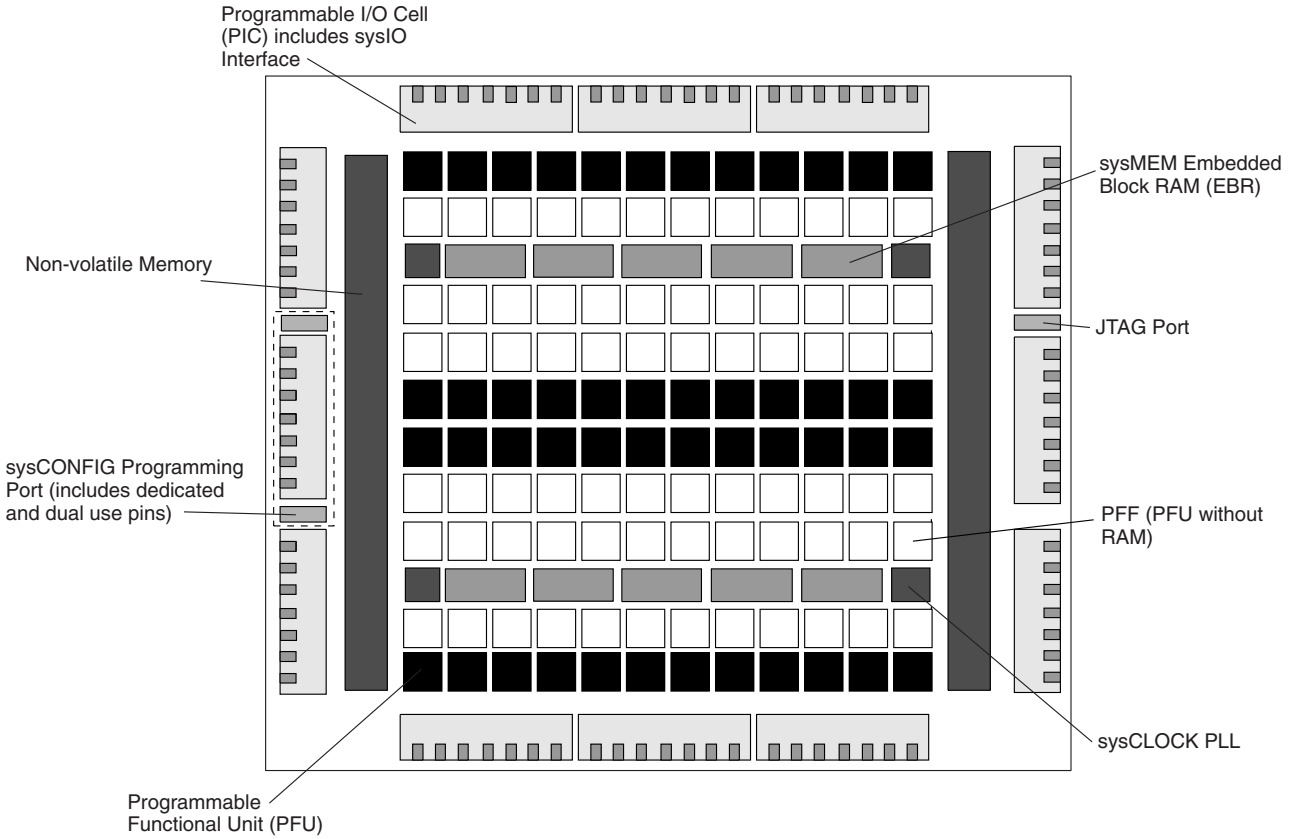


Figure 9-3. Simplified Block Diagram, LatticeXP Device (Top Level)



Utilizing IPexpress

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

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

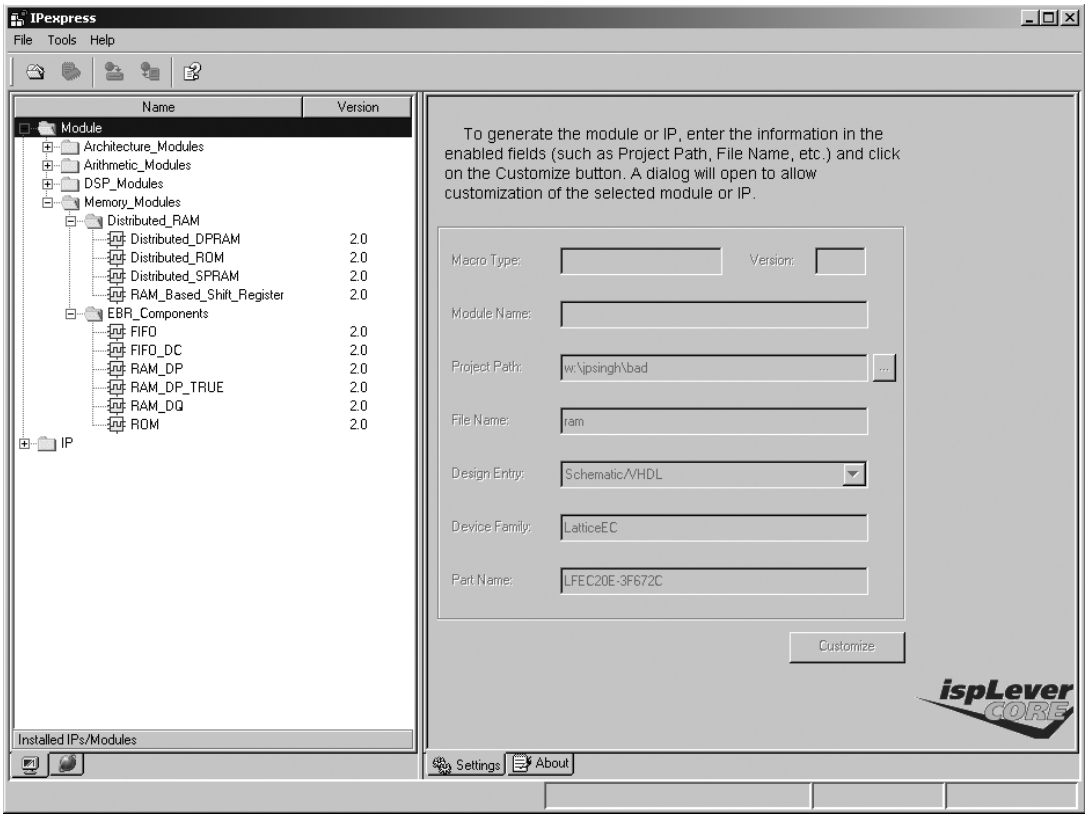
IPexpress Flow

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

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

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

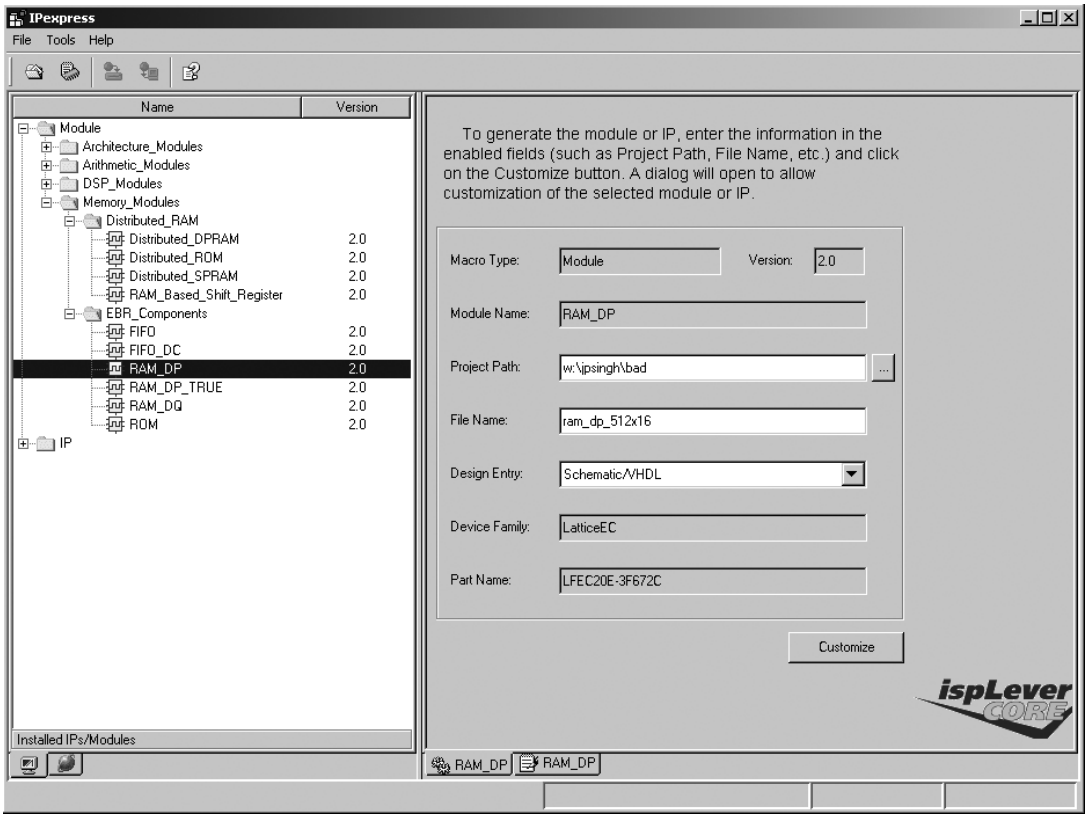
Figure 9-4. IPexpress - Main Window



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

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

Figure 9-5. Example Generating Pseudo Dual Port RAM (RAM_DP) Using IPexpress



In the right-hand pane, options like **Macro Type**, **Version**, and **Module_Name** are device and selected module dependent. These cannot be changed in IPexpress.

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

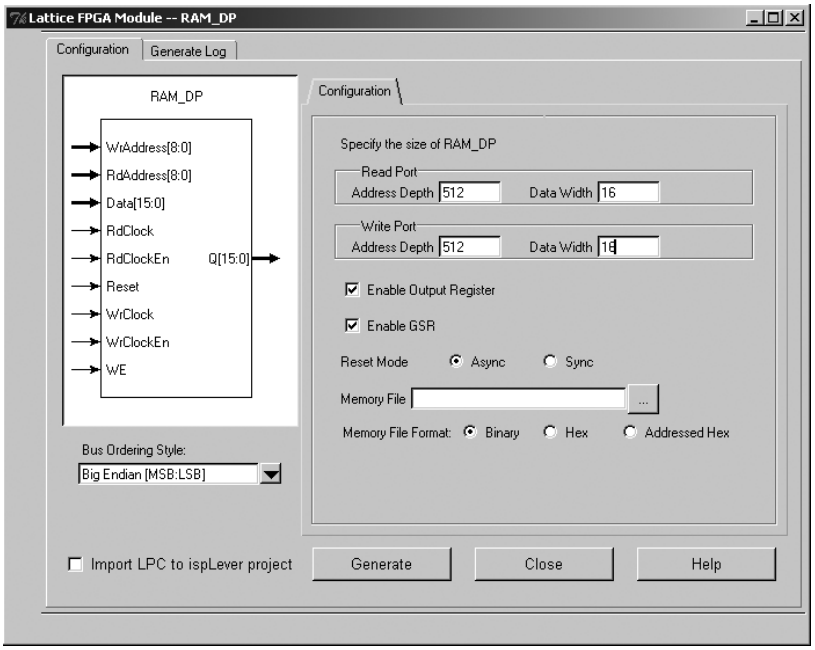
The **File Name** text box allows users to specify the entity and file name for the module they are about to generate. Users must provide this name.

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

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

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

Figure 9-6. Generating Pseudo Dual Port RAM (RAM_DP) Module Customization – Configuration Tab



Users can specify the Address Depth and Data width for the **Read Port** and the **Write Port** in the text boxes provided. In this example we are generating a Pseudo Dual Port RAM of size 512 x 16. Users can also create RAMs of different port widths in the case of Pseudo Dual Port and True Dual Port RAMs.

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

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

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

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

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

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

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

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

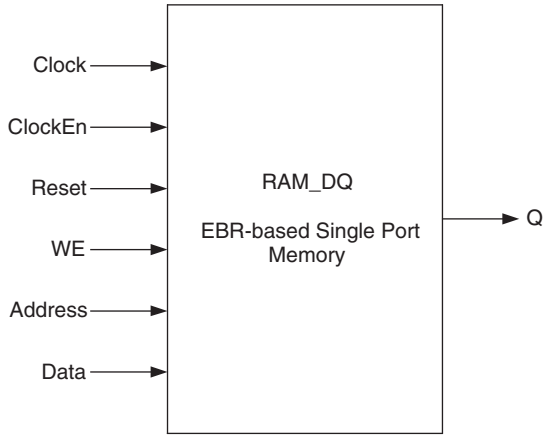
Memory Modules

Single Port RAM (RAM_DQ) – EBR Based

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

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

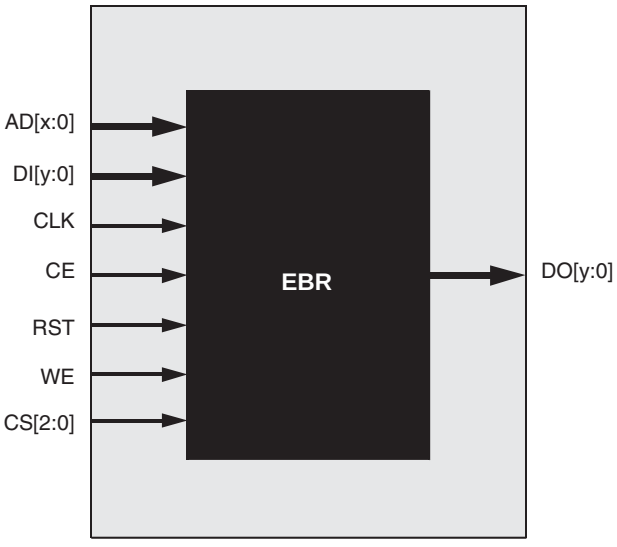
Figure 9-7. Single Port Memory Module generated by IPexpress



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

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

Figure 9-8. Single Port RAM Primitive or RAM_DQ for LatticeECP/EC and LatticeXP Devices



In Single Port RAM mode the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered.

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

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

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

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

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

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

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

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

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

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

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

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

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

Figure 9-9. Single Port RAM Timing Waveform – NORMAL Mode, without Output Registers

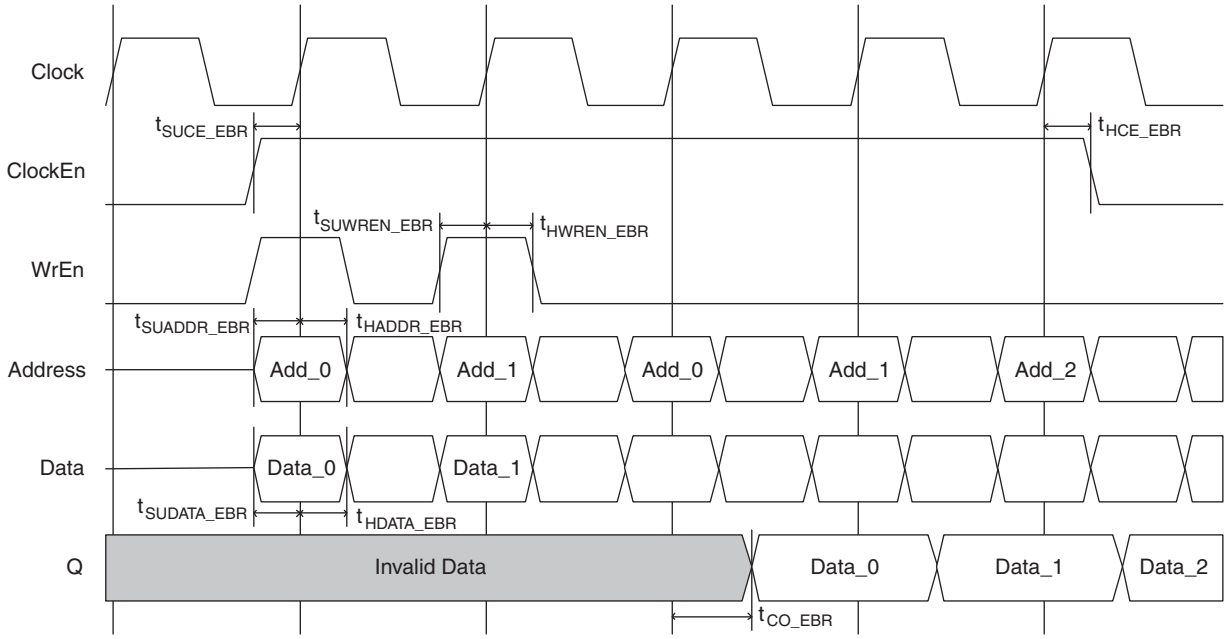


Figure 9-10. Single Port RAM Timing Waveform – NORMAL Mode, with Output Registers

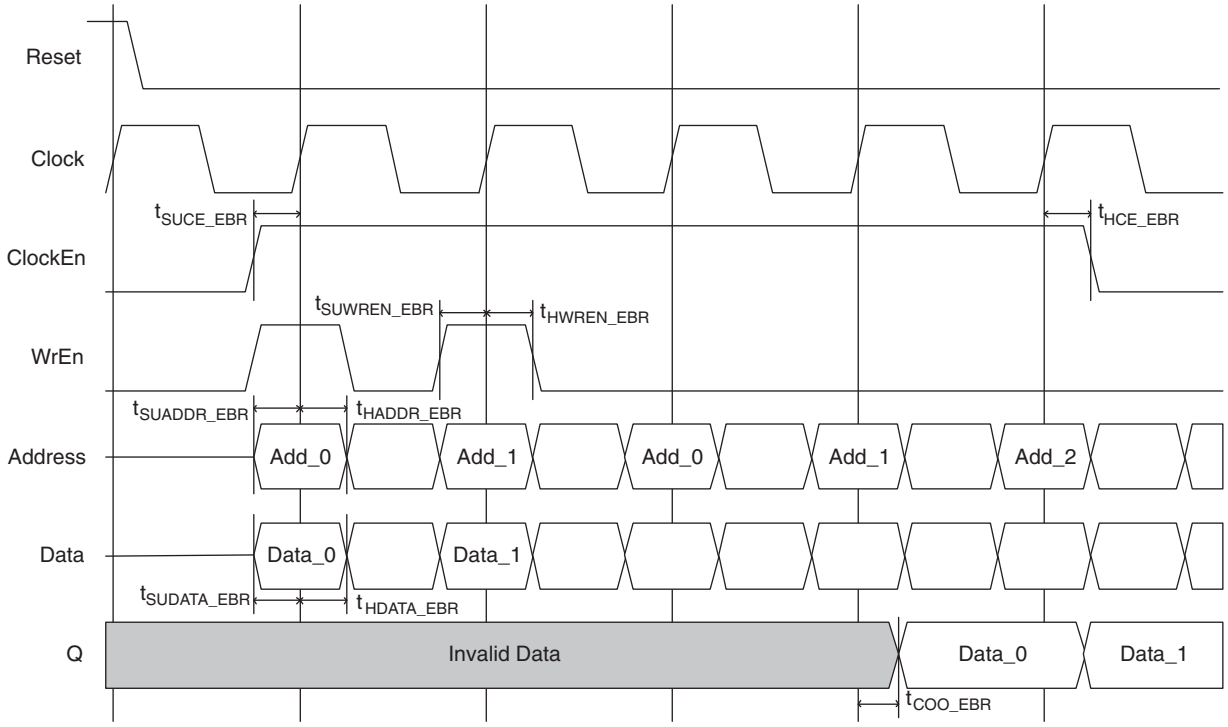


Figure 9-11. Single Port RAM Timing Waveform – READ BEFORE WRITE Mode, without Output Registers

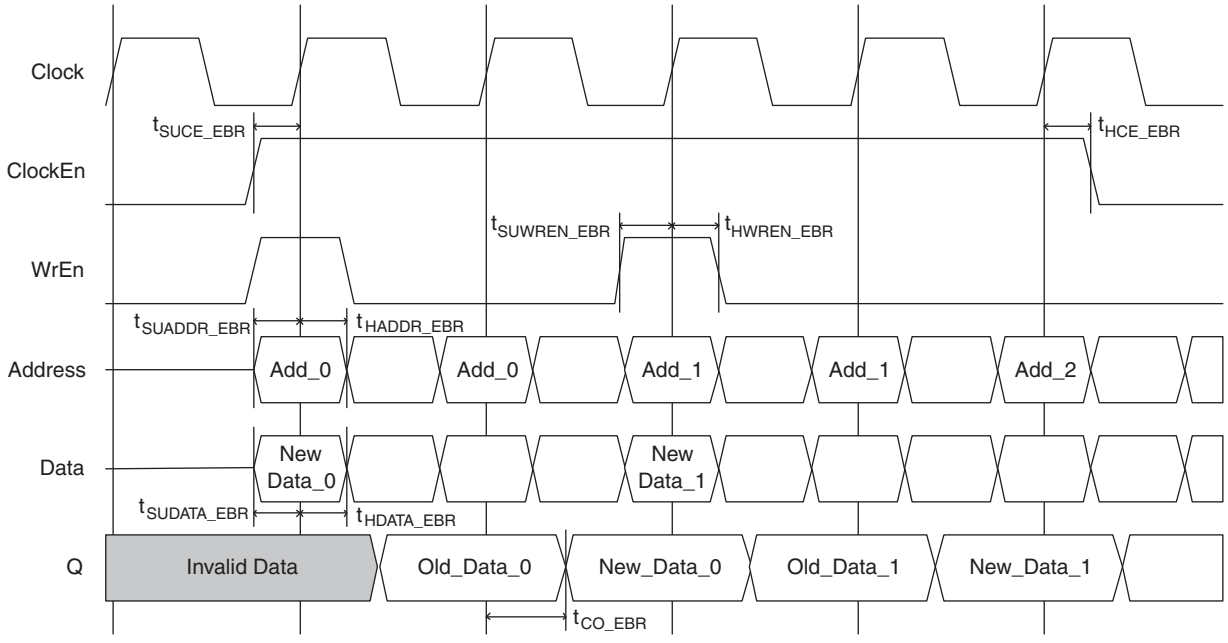


Figure 9-12. Single Port RAM Timing Waveform – READ BEFORE WRITE Mode, with Output Registers

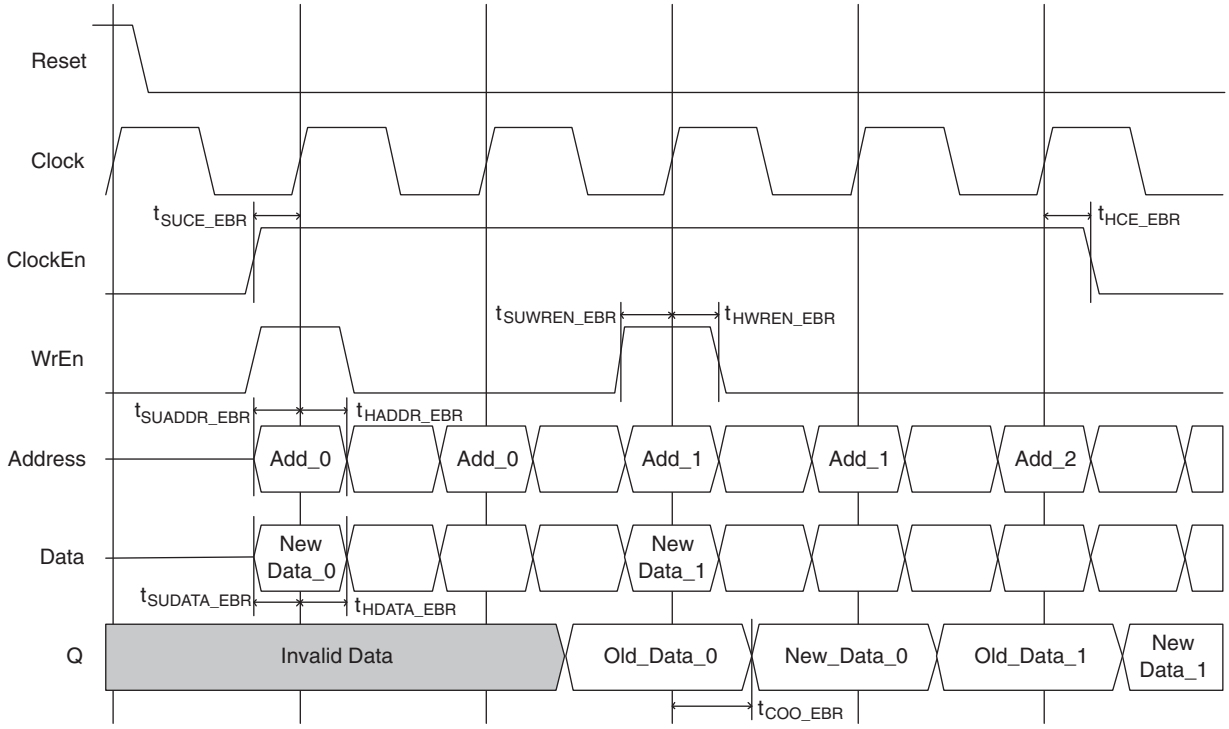


Figure 9-13. Single Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

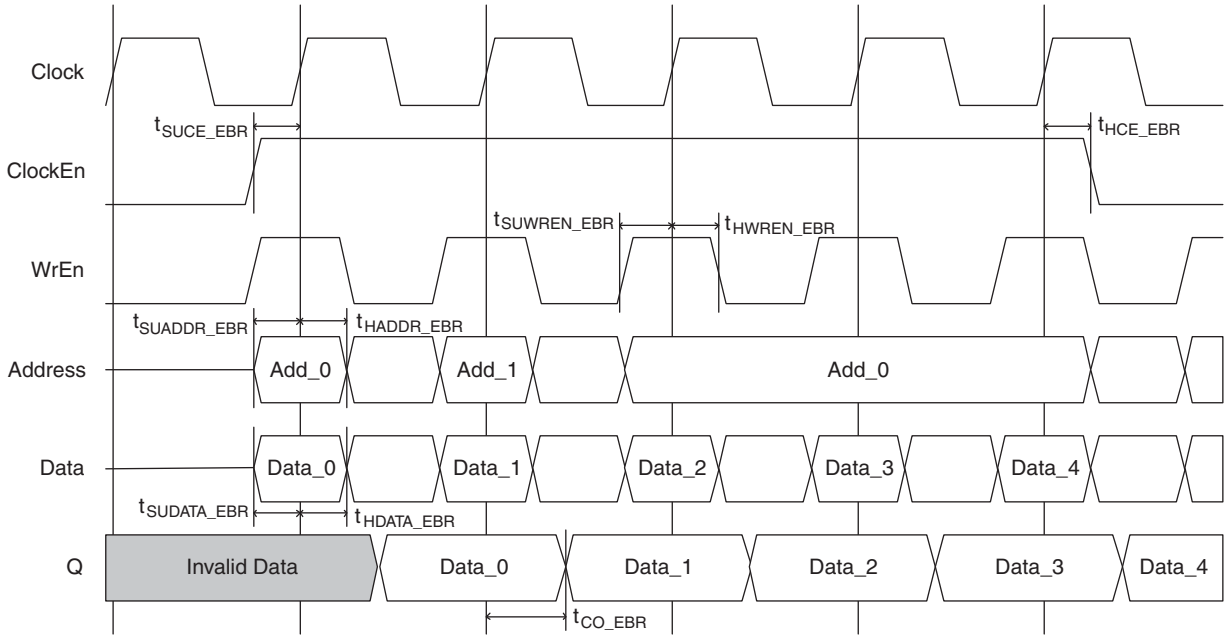
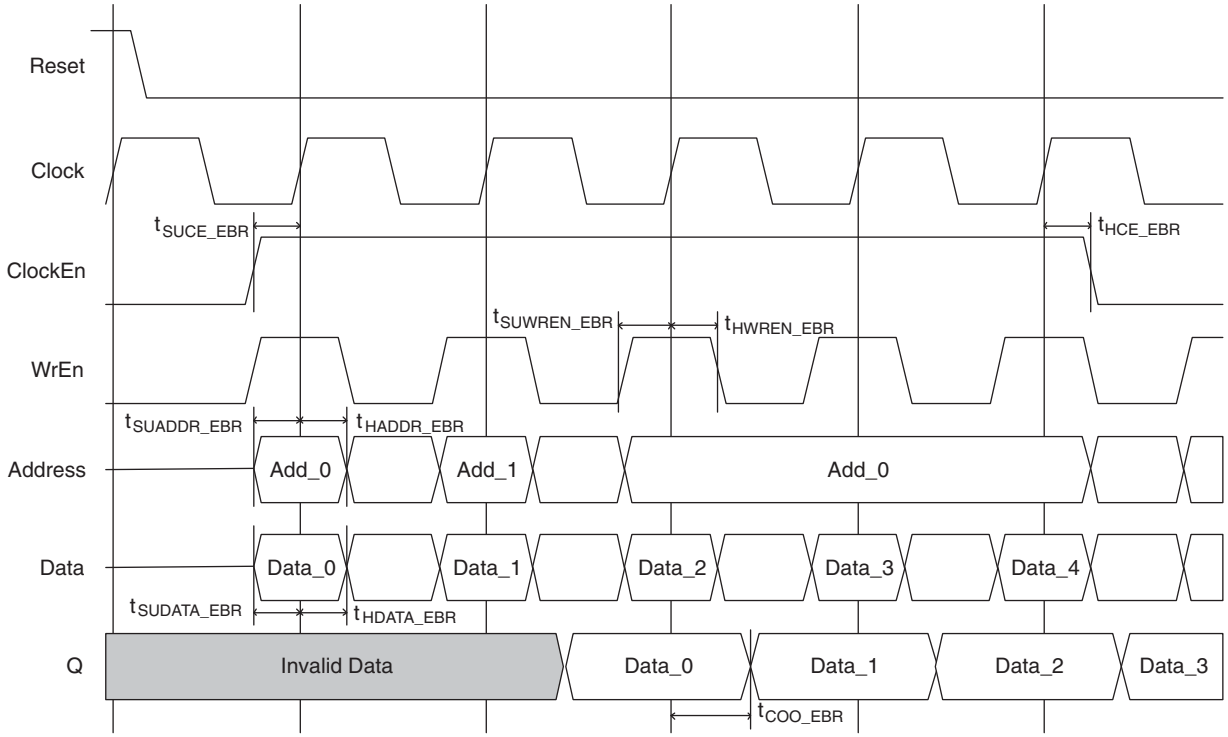


Figure 9-14. Single Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

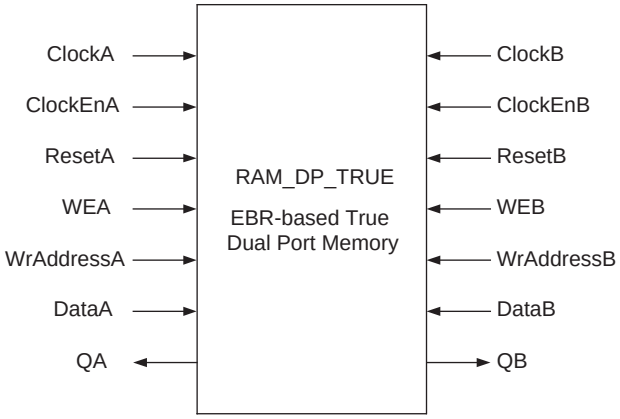


True Dual Port RAM (RAM_DP_TRUE) – EBR Based

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

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

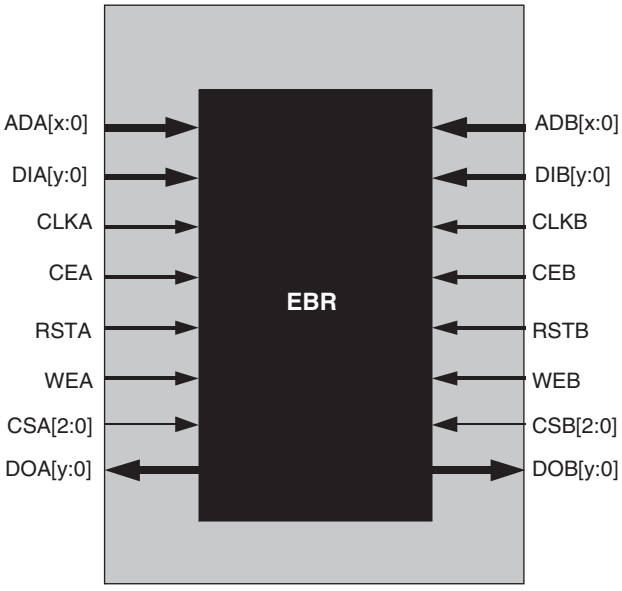
Figure 9-15. True Dual Port Memory Module Generated by IPexpress



The generated module makes use of the RAM_DP_TRUE primitive. For memory sizes smaller than one EBR block, the module will be created in one EBR block. In cases where the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width (as required to create these sizes).

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

Figure 9-16. True Dual Port RAM Primitive or RAM_DP_TRUE for LatticeECP/EC and LatticeXP Devices



In True Dual Port RAM mode, the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

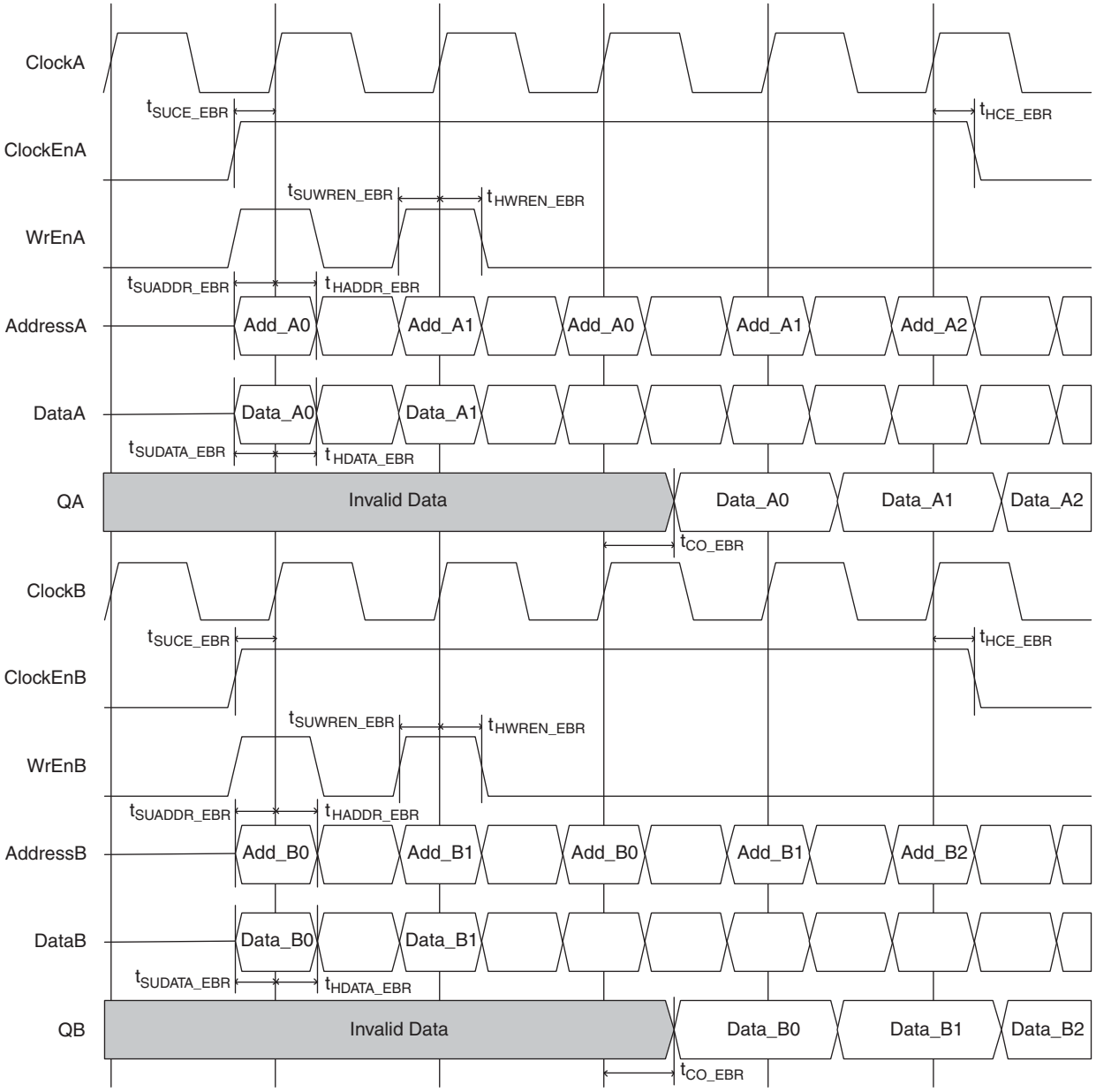


Figure 9-18. True Dual Port RAM Timing Waveform – NORMAL Mode with Output Registers

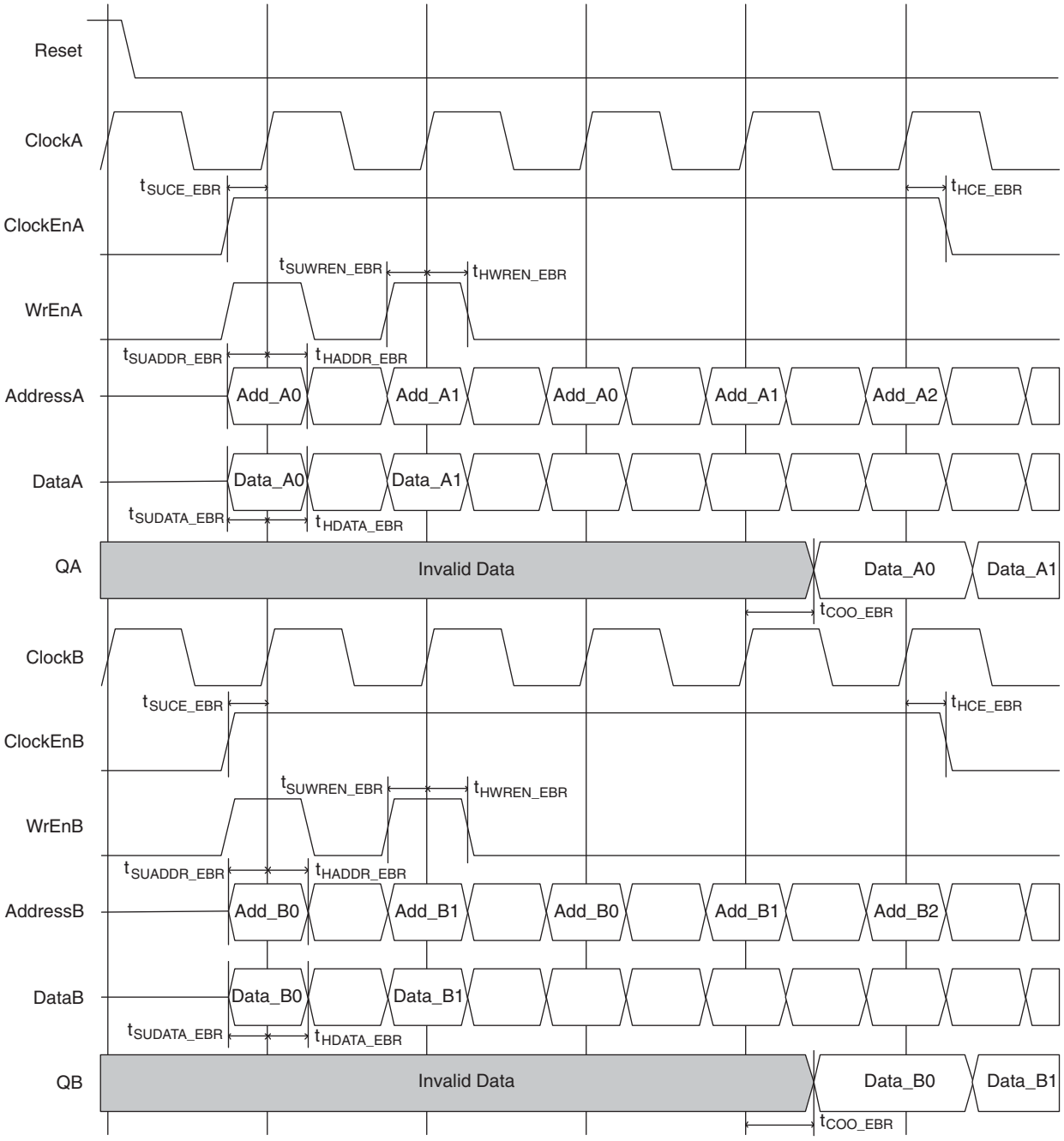


Figure 9-19. True Dual Port RAM Timing Waveform – READ BEFORE WRITE Mode, without Output Registers

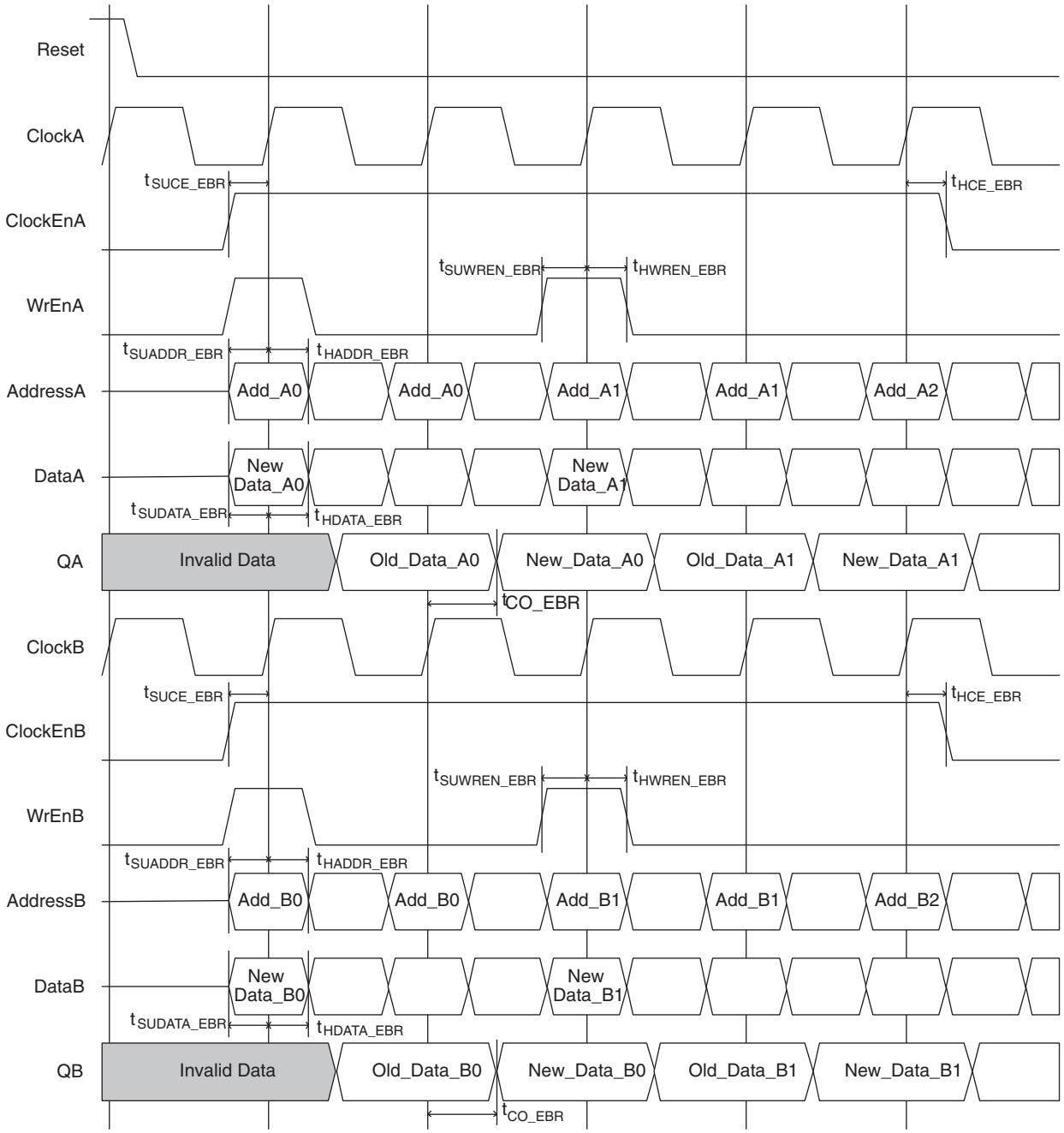


Figure 9-20. True Dual Port RAM Timing Waveform – READ BEFORE WRITE Mode, with Output Registers

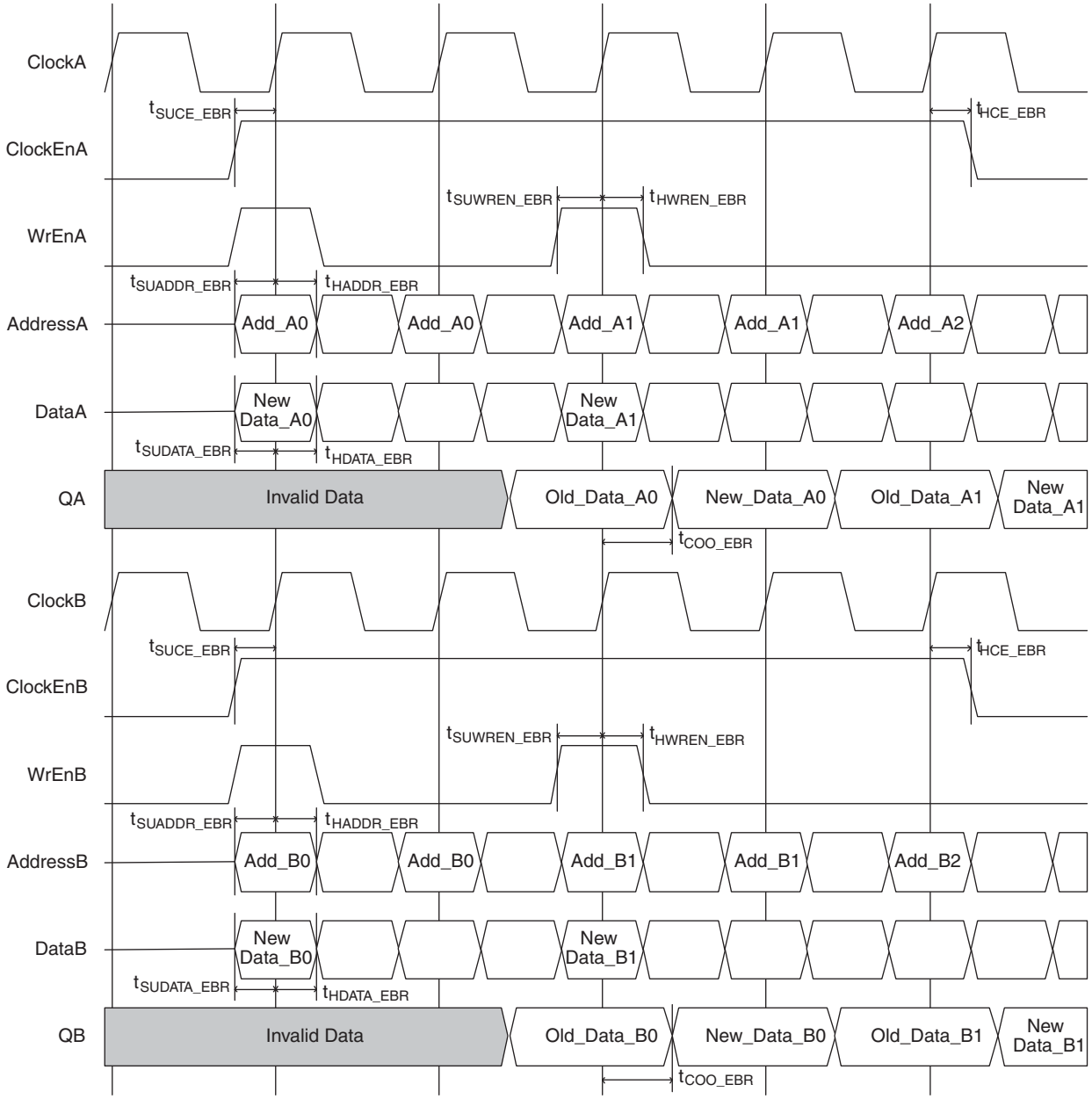


Figure 9-21. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, without Output Registers

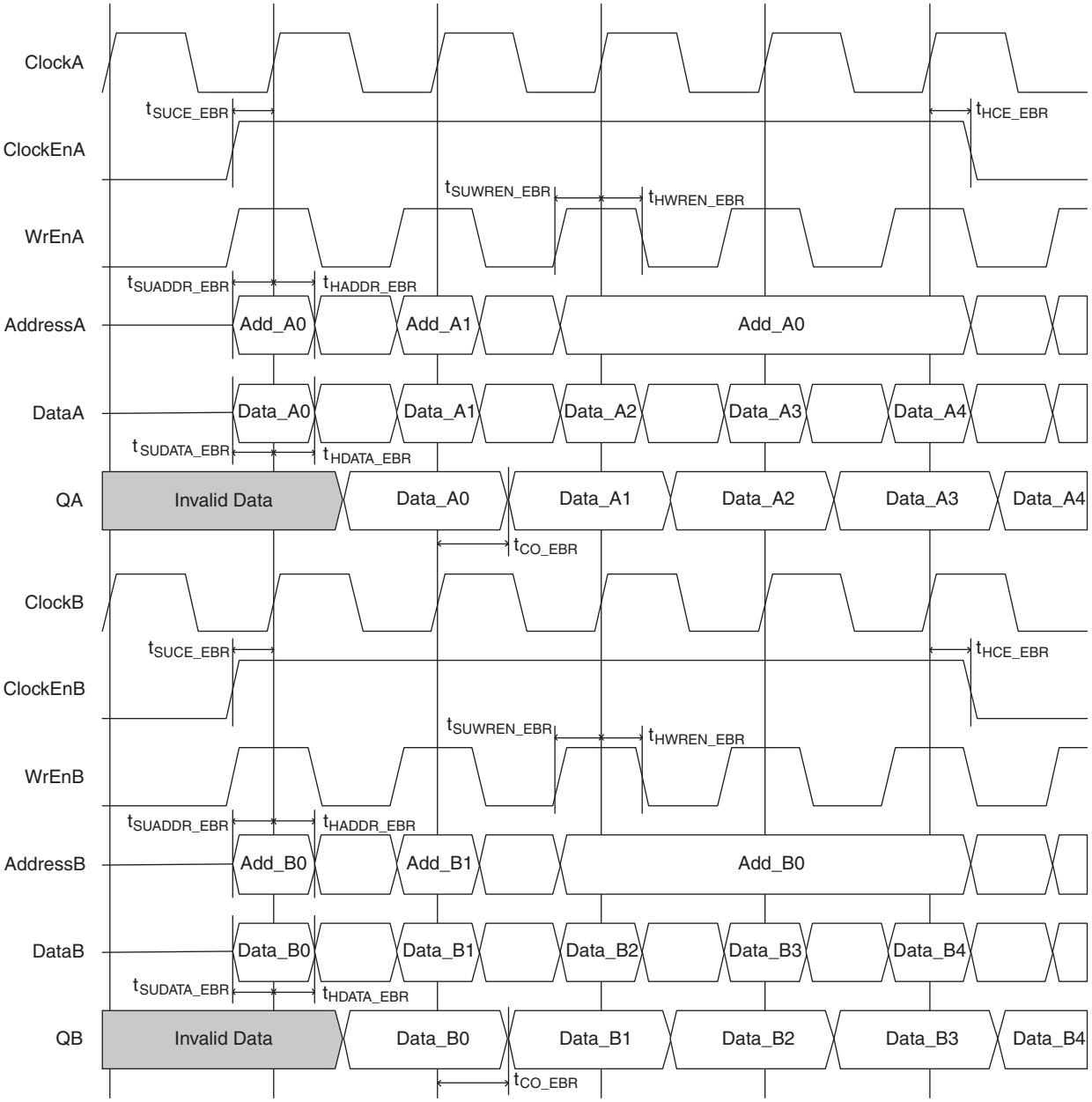
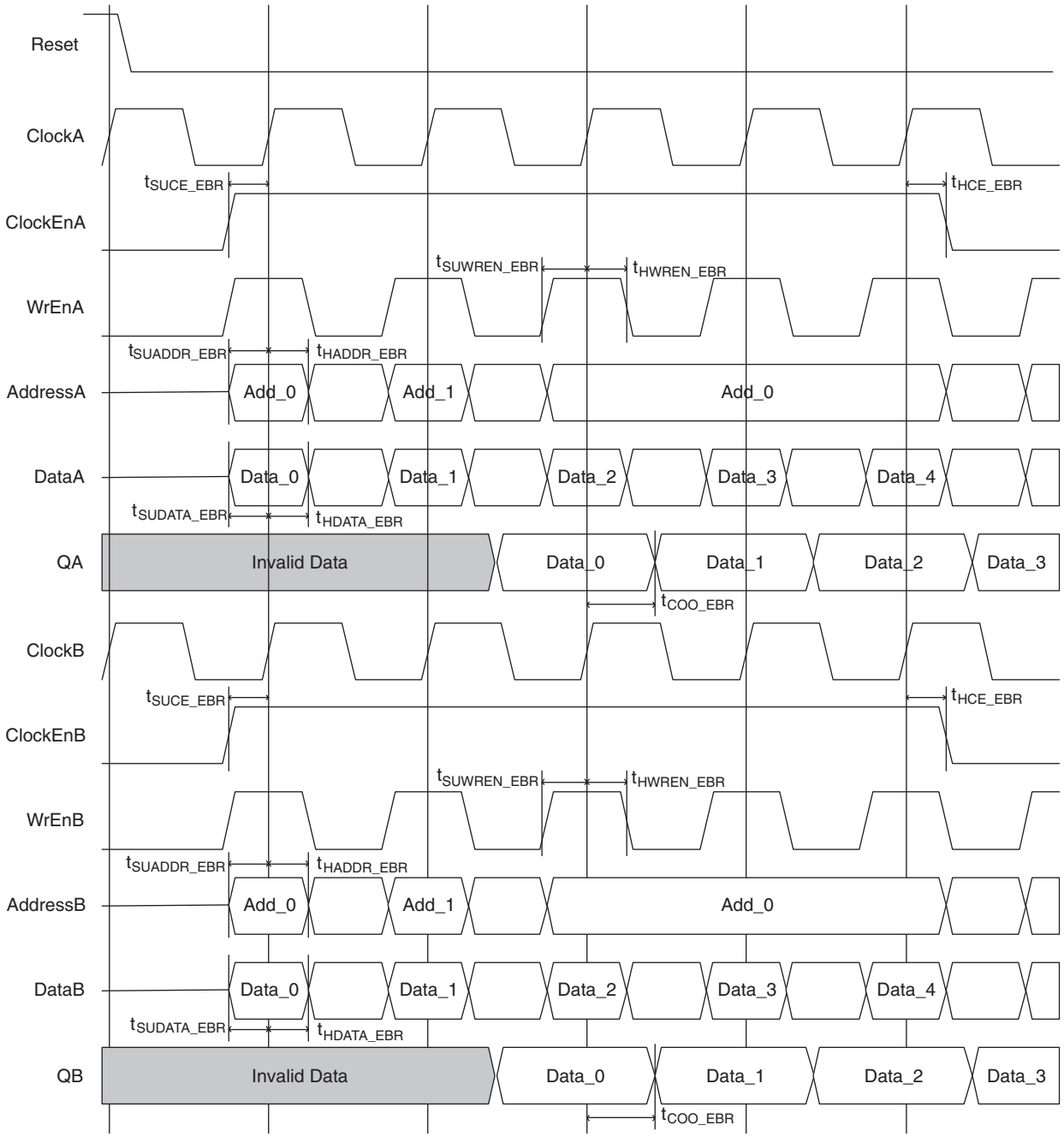


Figure 9-22. True Dual Port RAM Timing Waveform – WRITE THROUGH Mode, with Output Registers

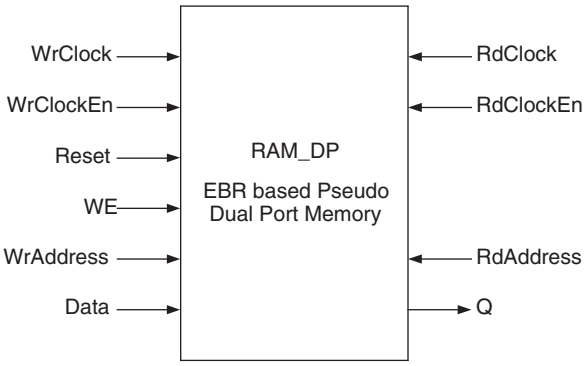


Pseudo Dual Port RAM (RAM_DP) – EBR-Based

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

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

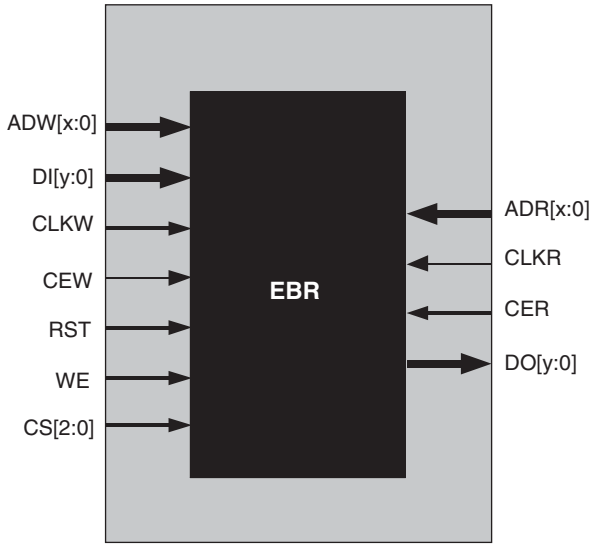
Figure 9-23. Pseudo Dual Port Memory Module Generated by IPexpress



The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR block can be cascaded, in depth or width (as required to create these sizes).

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

Figure 9-24. Pseudo Dual Port RAM primitive or RAM_DP for LatticeECP/EC and LatticeXP Devices



In the Pseudo Dual Port RAM mode, the input data and address for the ports are registered at the input of the memory array. The output data of the memory is optionally registered at the output.

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

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

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

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

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

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

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

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

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

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

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

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

Figure 9-25. PSEUDO DUAL PORT RAM Timing Diagram – without Output Registers

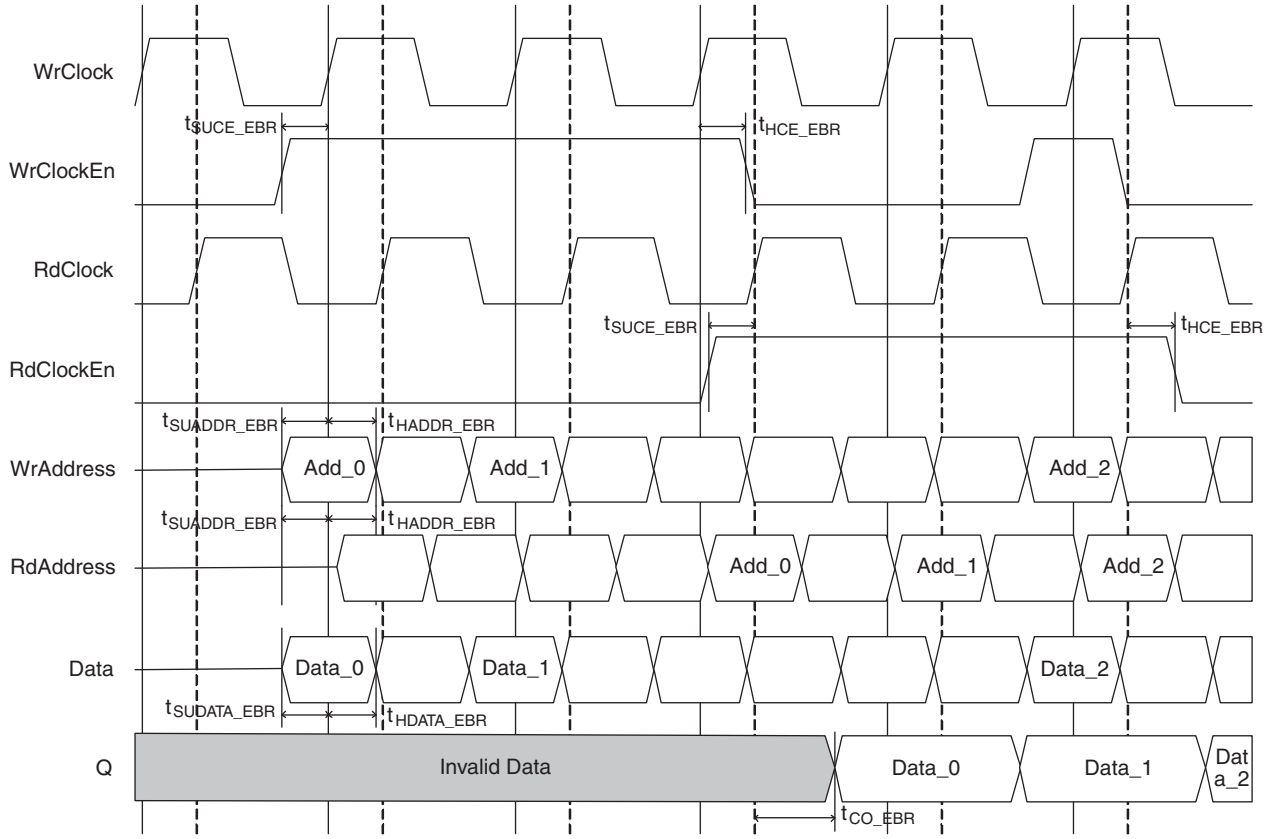
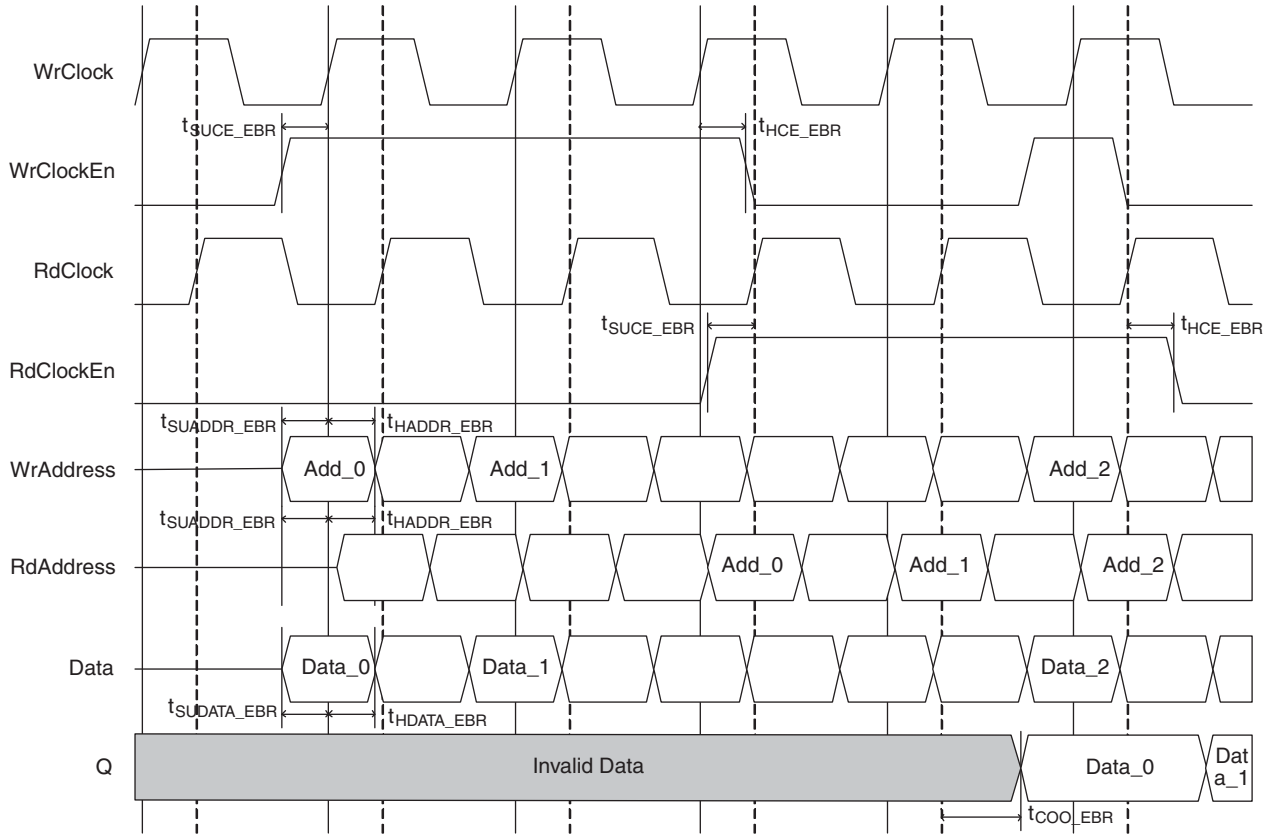


Figure 9-26. PSEUDO DUAL PORT RAM Timing Diagram – with Output Registers

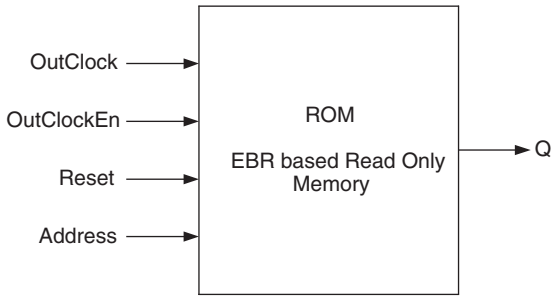


Read Only Memory (ROM) – EBR Based

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

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

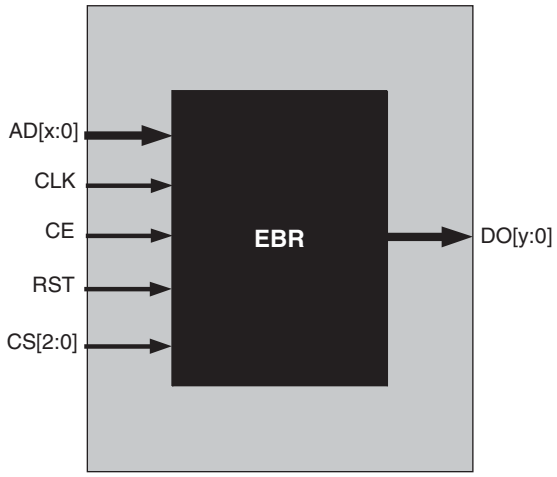
Figure 9-27. ROM - Read Only Memory Module Generated by IPexpress



The generated module makes use of these EBR blocks or primitives. For memory sizes smaller than an EBR block, the module will be created in one EBR block. If the specified memory is larger than one EBR block, multiple EBR blocks can be cascaded, in depth or width (as required to create these sizes).

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

Figure 9-28. ROM Primitive for LatticeECP/EC and LatticeXP Devices



In the ROM mode the address for the port is registered at the input of the memory array. The output data of the memory is optionally registered at the output.

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

Table 9-10. EBR-based ROM Port Definitions

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

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

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

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

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

Figure 9-29. ROM Timing Waveform – without Output Registers

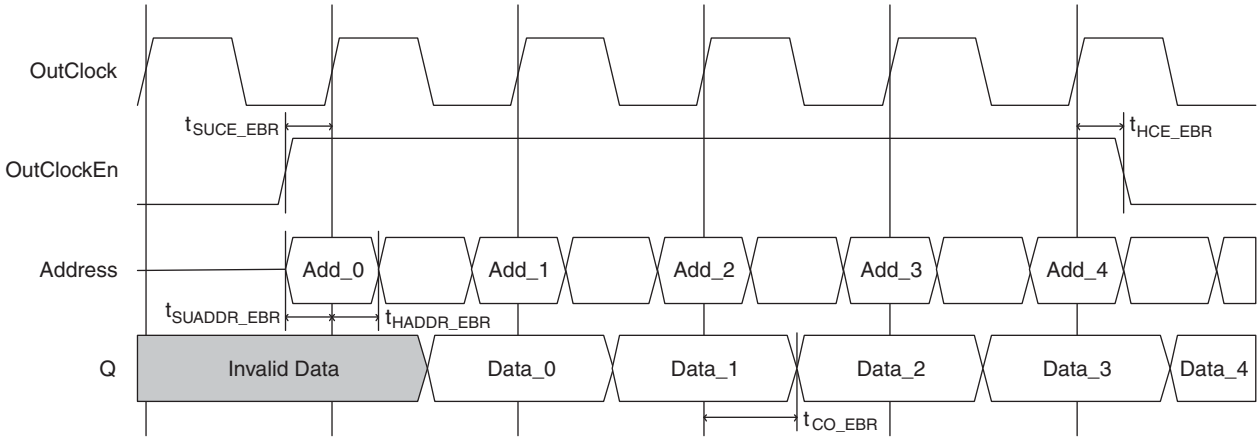
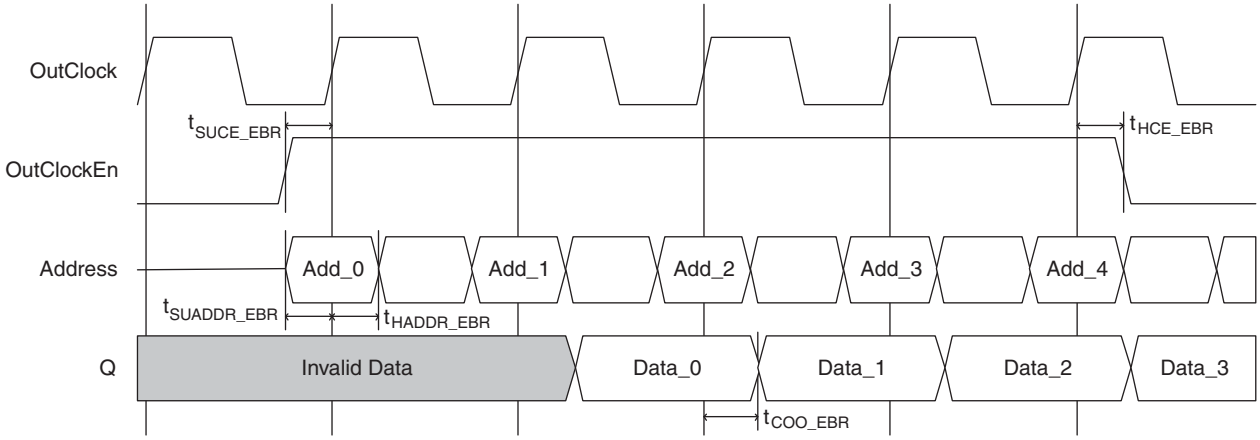


Figure 9-30. ROM Timing Waveform – with Output Registers



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

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

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

Figure 9-31. FIFO Module Generated by IPexpress

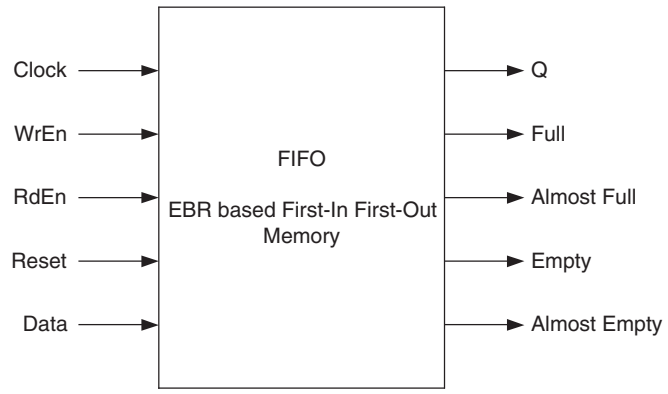
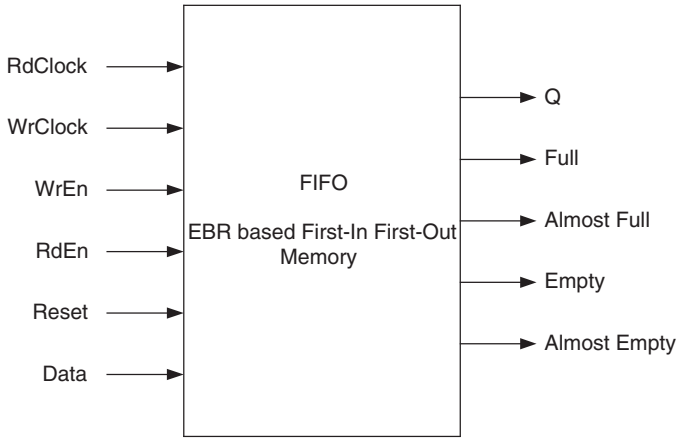


Figure 9-32. FIFO_DC Module Generated by IPexpress



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

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

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

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

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

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

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

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

FIFO Flags

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

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

Table 9-13. FIFO Flag Settings

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

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

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

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

FIFO Operation

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

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

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

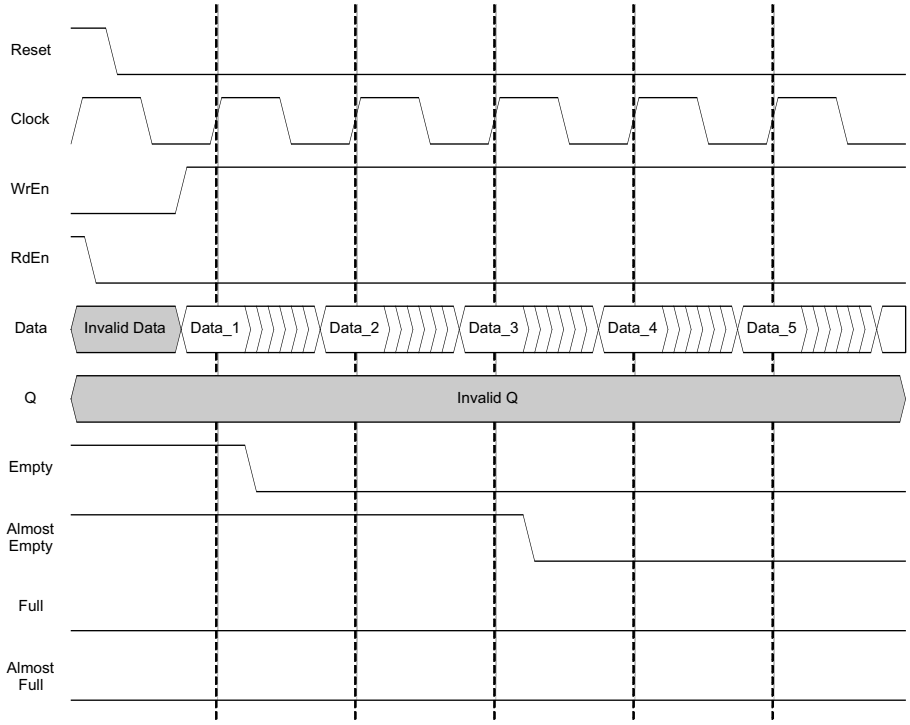
First In First Out (FIFO) Memory: The FIFO or the single clock FIFO is an emulated FIFO. The address logic and the flag logic is implemented in the FPGA fabric around the RAM.

The ports available on the FIFO are:

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

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

Figure 9-33. FIFO Without Output Registers, Start of Data Write Cycle

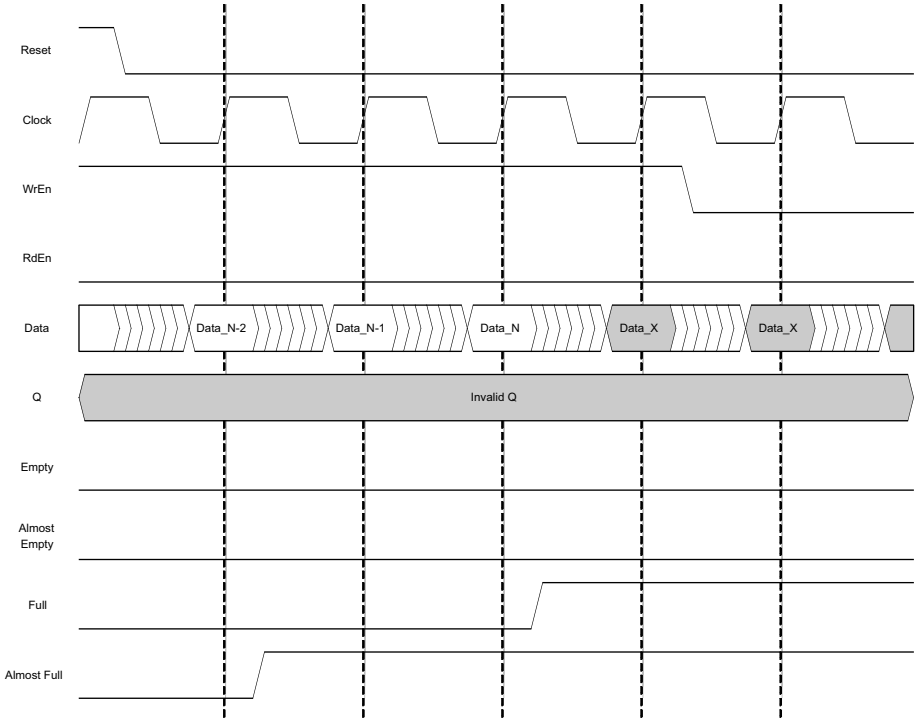


The WrEn signal has to be high to start writing into the FIFO. The Empty and Almost Empty flags are high to begin and Full and Almost full are low.

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

Now let is assume that we continue to write into the FIFO to fill it. When the FIFO is filled, the Almost Full and Full Flags are asserted. Figure 9-34 shows the behavior of these flags. In this figure we assume that FIFO depth is ‘N’.

Figure 9-34. FIFO Without Output Registers, End of Data Write Cycle

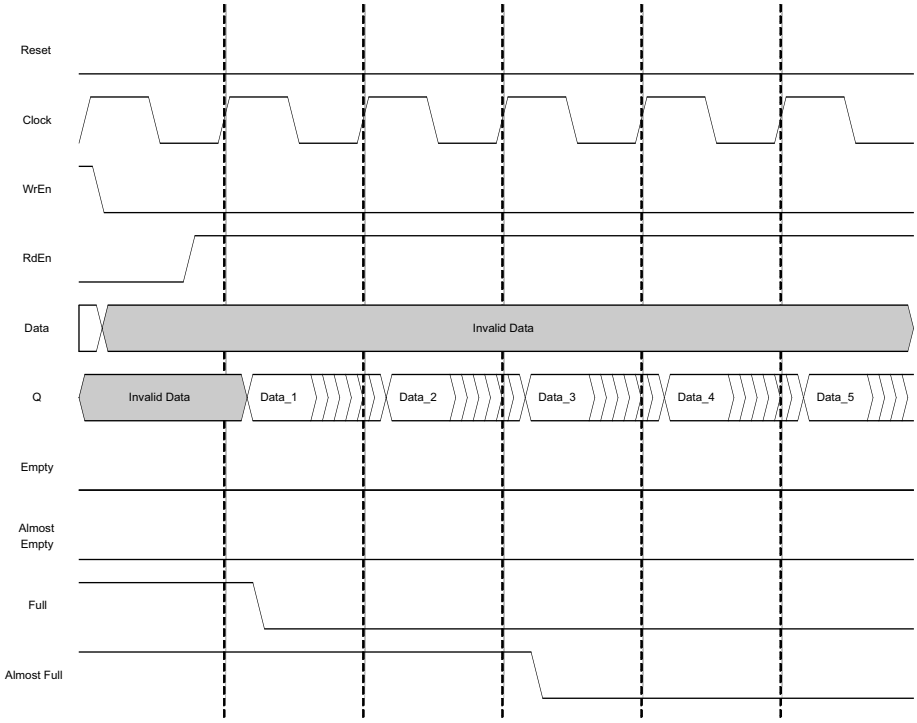


In this case, as seen above, the Almost Full flag is IN location 2 before the FIFO is filled. The Almost Full flag is asserted when N-2 location is written, and Full flag is asserted when the last word is written into the FIFO.

Data_X data inputs do not get written as the FIFO is full (Full flag is high).

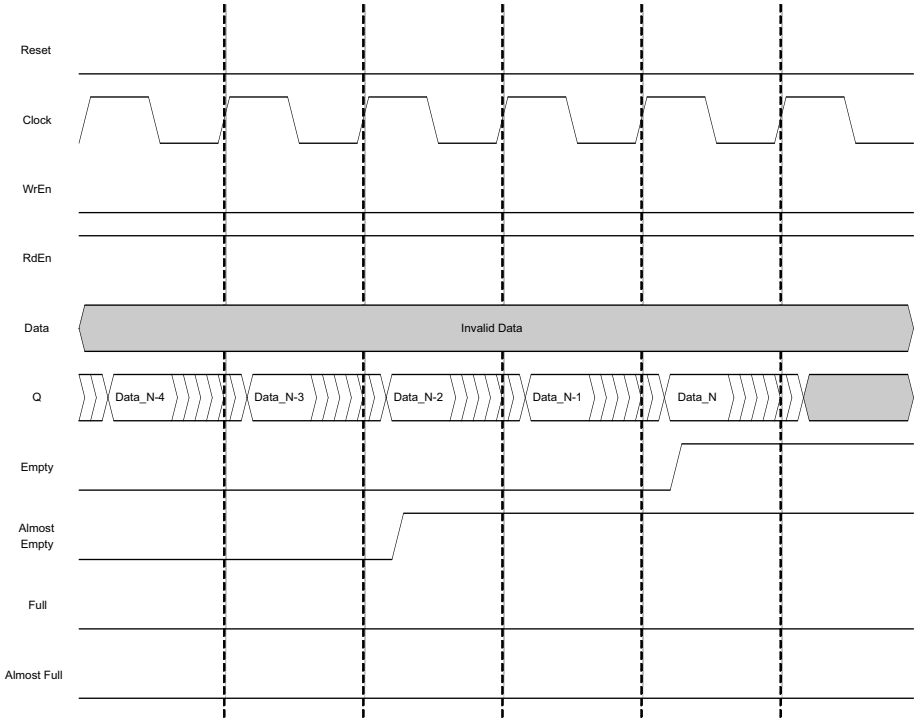
Now let us look at the waveforms when the contents of the FIFO are read out. Figure 9-35 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags gets de-asserted as shown.

Figure 9-35. FIFO Without Output Registers, Start of Data Read Cycle



Similarly as the data is read out, and FIFO is emptied, the Almost Empty and Empty flags are asserted. Below is the

Figure 9-36. FIFO Without Output Registers, End of Data Read Cycle



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

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

Figure 9-37. FIFO with Output Registers, Start of Data Write Cycle

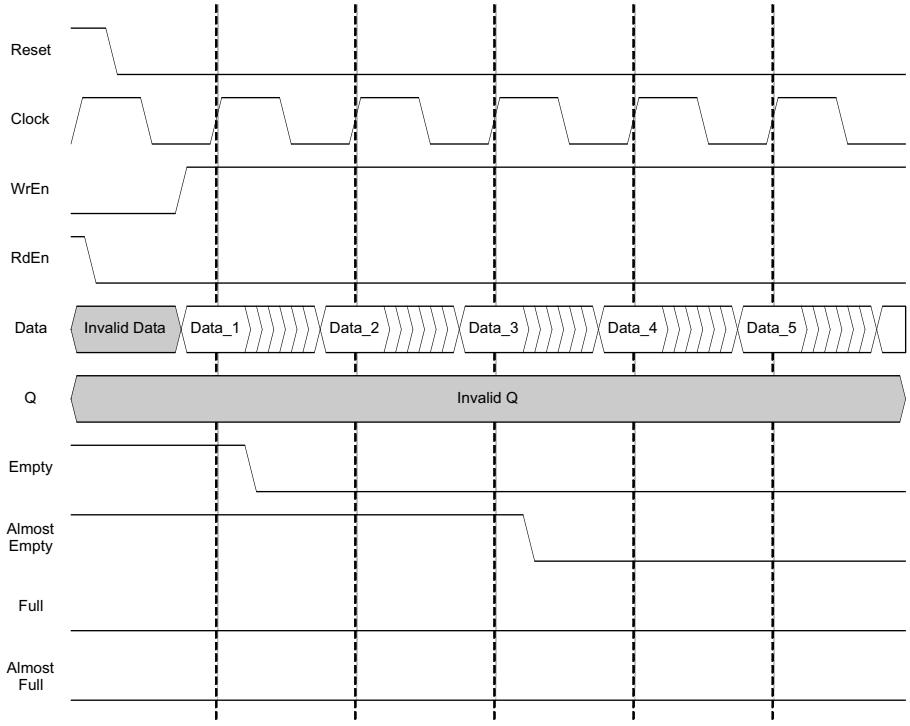


Figure 9-38. FIFO with Output Registers, End of Data Write Cycle

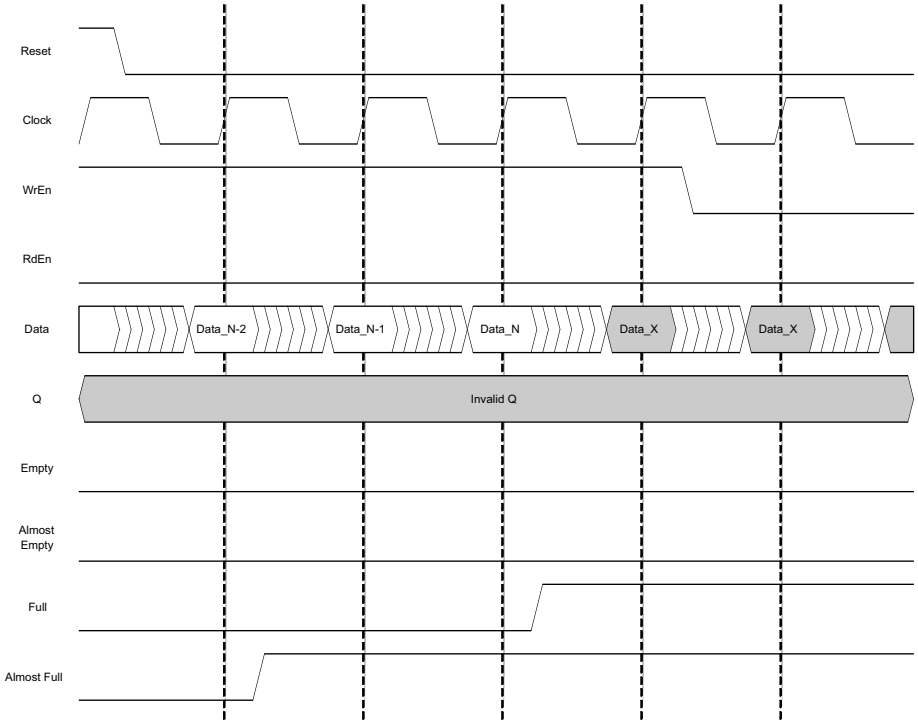


Figure 9-39. FIFO with Output Registers, Start of Data Read Cycle

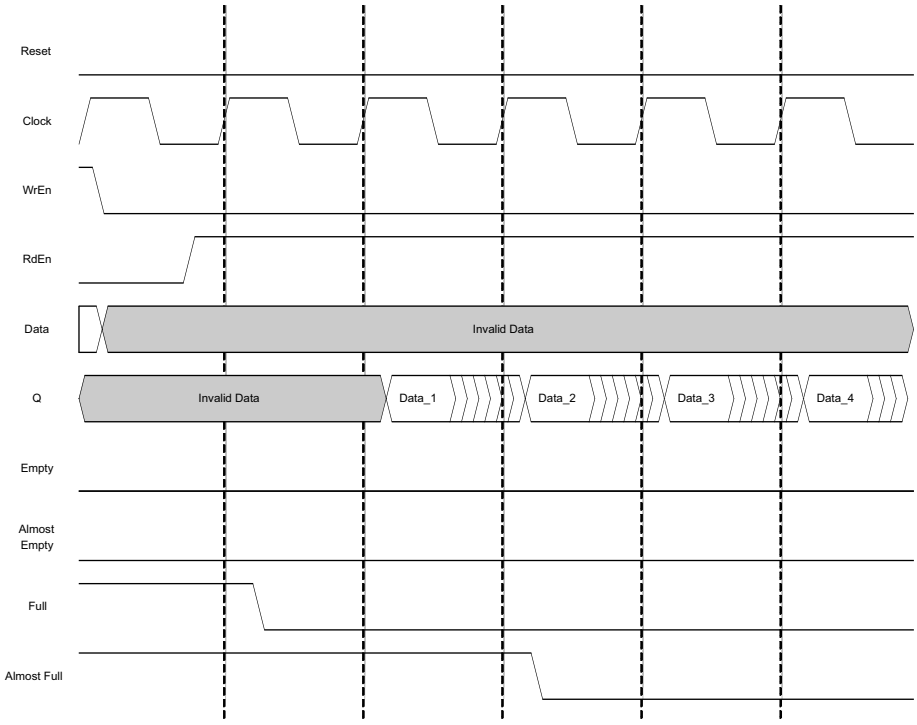
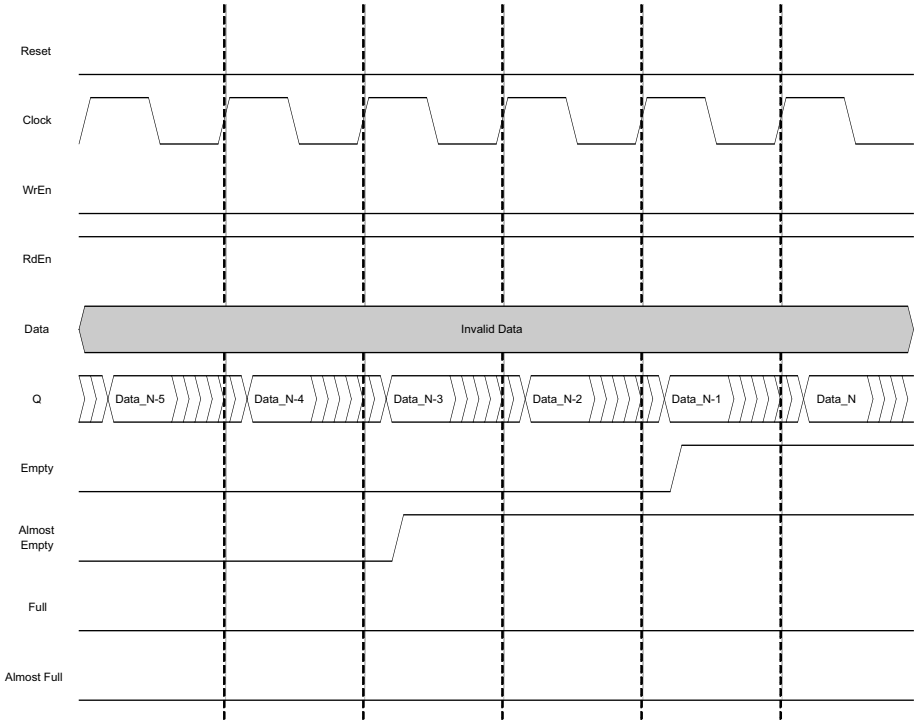
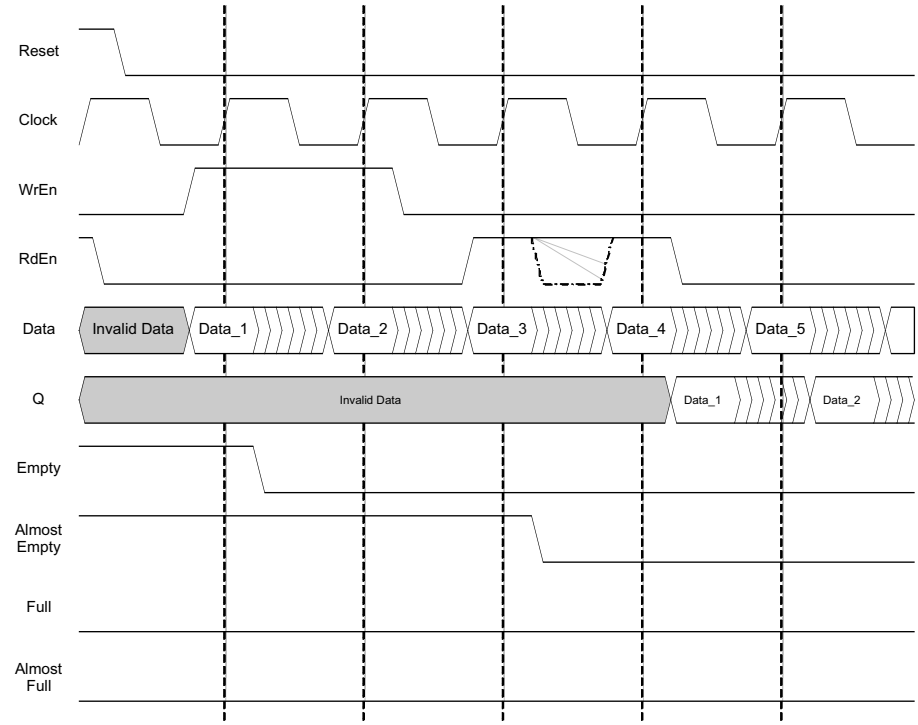


Figure 9-40. FIFO with Output Registers, End of Data Read Cycle



And finally, if you select the option enable output register with RdEn, it still delays the data out by one clock cycle (as compared to the non-pipelined FIFO), and the RdEn should be high also during that clock cycle, otherwise the data takes an extra clock cycle when the RdEn goes true.

Figure 9-41. FIFO with Output Registers and RdEn on Output Registers



Dual Clock First In First Out (FIFO_DC) Memory: The FIFO_DC or the dual clock FIFO is also an emulated FIFO. Again the address logic and the flag logic is implemented in the FPGA fabric around the RAM.

The ports available on the FIFO_DC are:

- Reset
- RPReset
- WrClock
- RdClock
- WrEn
- RdEn
- Data
- Q
- Full Flag
- Almost Full Flag
- Empty Flag
- Almost Empty Flag

FIFO_DC Flags: FIFO_DC, as an emulated FIFO, required the flags to be implemented in the FPGA logic around the block RAM. Because of the two clocks, the flags are required to change clock domains from read clock to write clock and vice versa. This adds latency to the flags either during assertion or during de-assertion. The latency can be avoided only in one of the cases (either assertion or de-assertion).

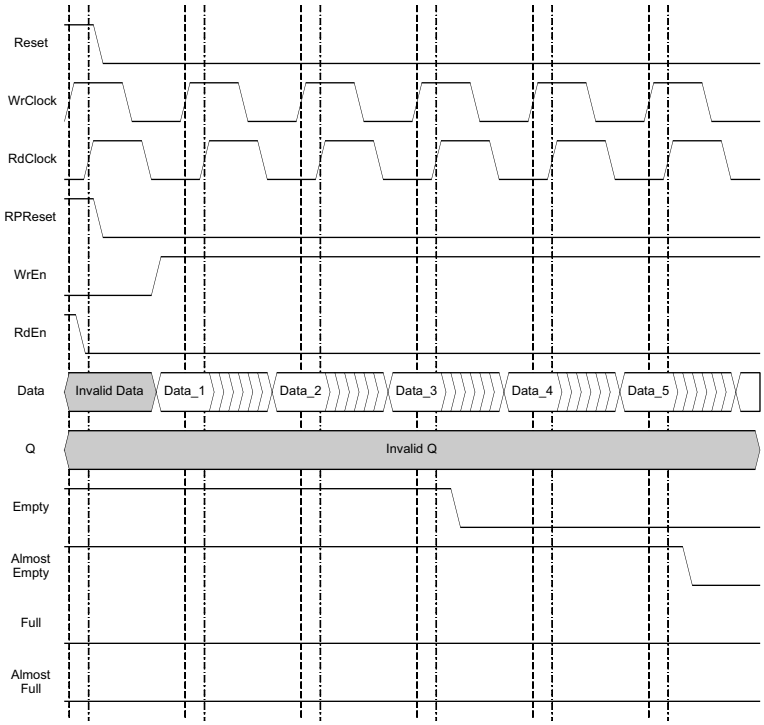
In the current emulated FIFO, there is no latency during assertion of these flags. Thus, when these flag go true, there is no latency. However this causes the latency during the de-assertion.

Let us assume that we start to write into the FIFO_DC to fill it. The write operation is controlled by WrClock and WrEn, however it takes extra RdClock cycles for de-assertion of Empty and Almost Empty flags.

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

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

Figure 9-42. FIFO_DC Without Output Registers, Start of Data Write Cycle

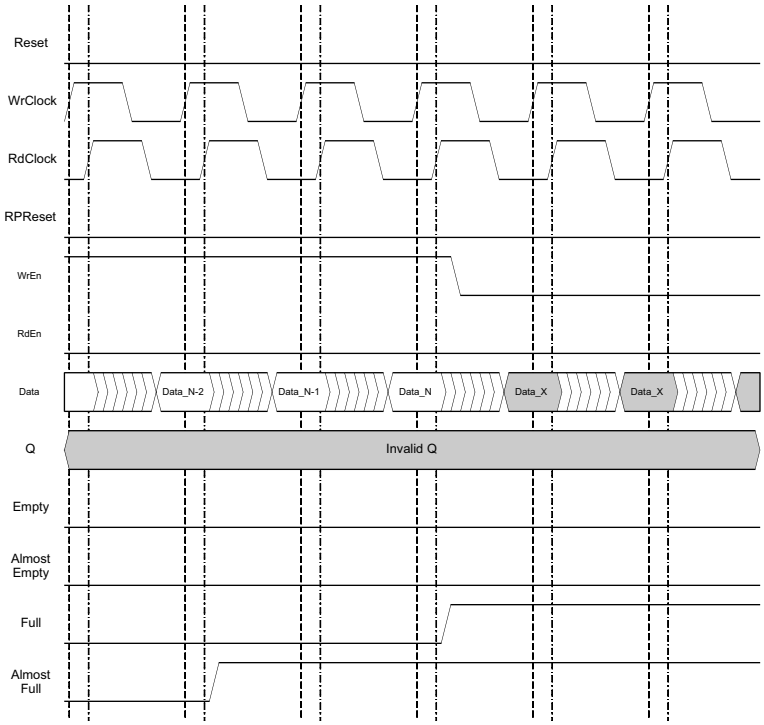


The WrEn signal has to be high to start writing into the FIFO_DC. The Empty and Almost Empty flags are high to begin and Full and Almost full are low.

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

Now let is assume that we continue to write into the FIFO_DC to fill it. When the FIFO_DC is filled, the Almost Full and Full Flags are asserted. Figure 9-43 shows the behavior of these flags. In this figure we assume that FIFO_DC depth is ‘N’.

Figure 9-43. FIFO_DC Without Output Registers, End of Data Write Cycle



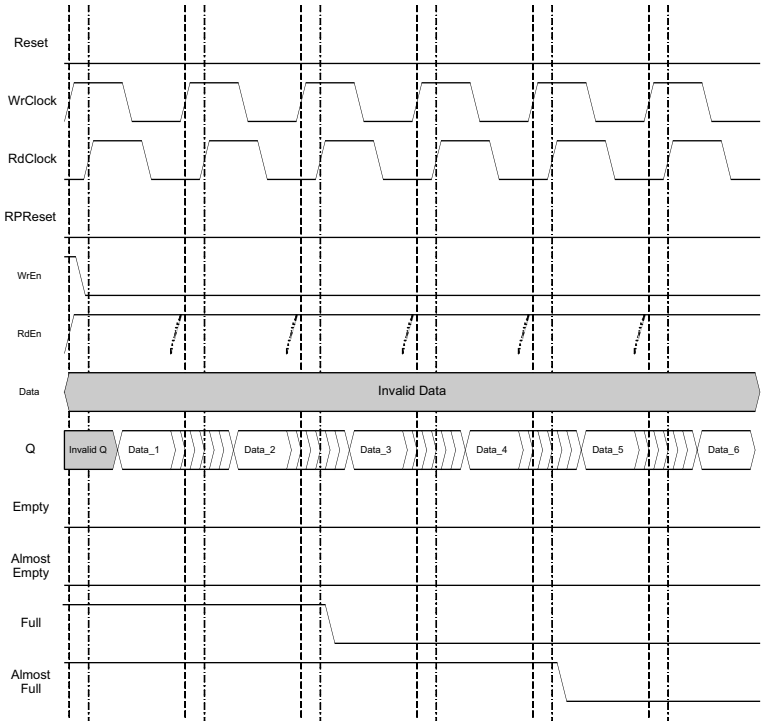
In this case, the Almost Full flag is in location 2 before the FIFO_DC is filled. The Almost Full flag is asserted when N-2 location is written, and Full flag is asserted when the last word is written into the FIFO_DC.

Data_X data inputs do not get written as the FIFO_DC is full (Full flag is high).

Note that the assertion of these flags is immediate and there is no latency when they go true.

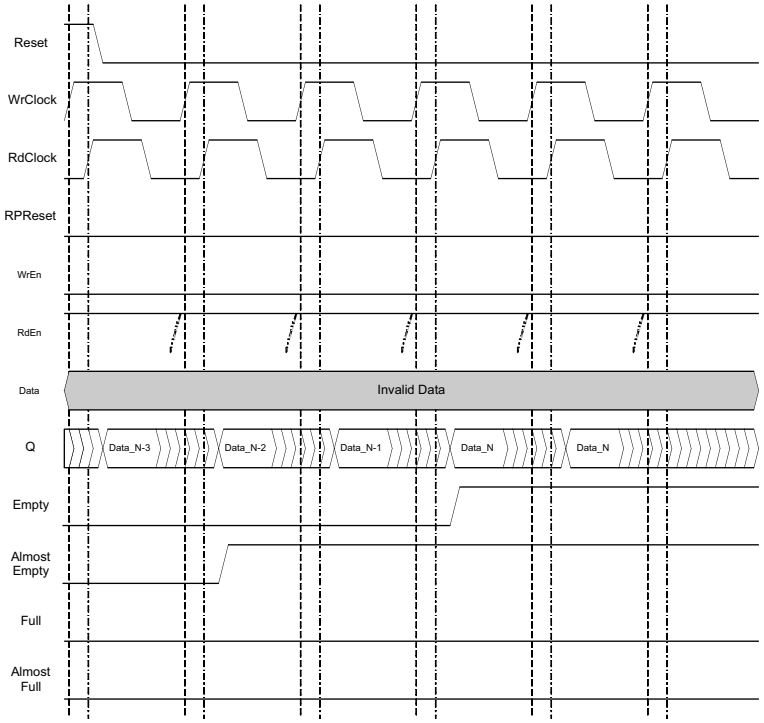
Now let us look at the waveforms when the contents of the FIFO_DC are read out. Figure 9-44 shows the start of the read cycle. RdEn goes high and the data read starts. The Full and Almost Full flags get de-asserted as shown. In this case, note that the de-assertion is delayed by two clock cycles.

Figure 9-44. FIFO_DC Without Output Registers, Start of Data Read Cycle



Similarly, as the data is read out and FIFO_DC is emptied, the Almost Empty and Empty flags are asserted. Below is the

Figure 9-45. FIFO_DC Without Output Registers, End of Data Read Cycle



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

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

Figure 9-46. FIFO_DC With Output Registers, Start of Data Write Cycle

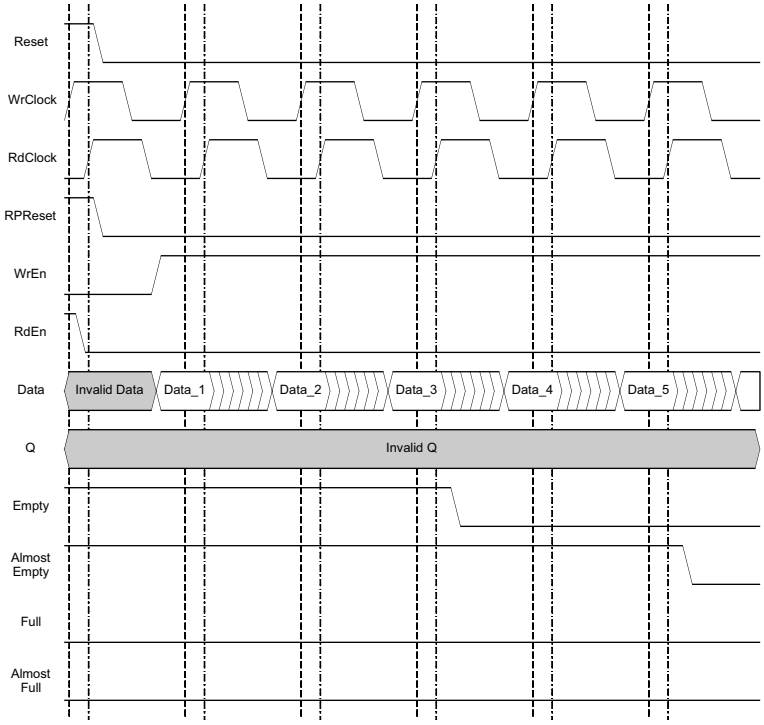


Figure 9-47. FIFO_DC With Output Registers, End of Data Write Cycle

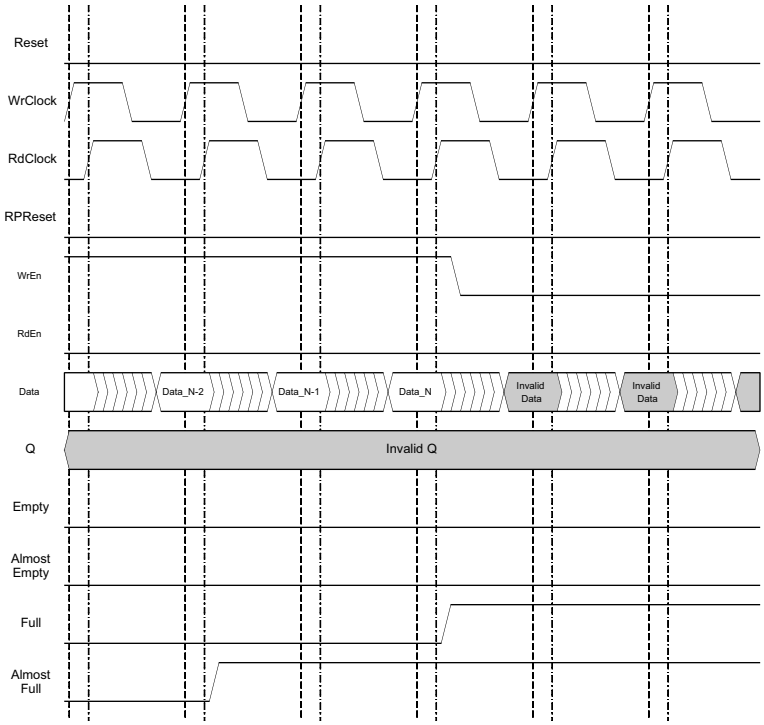


Figure 9-48. FIFO_DC With Output Registers, Start of Data Read Cycle

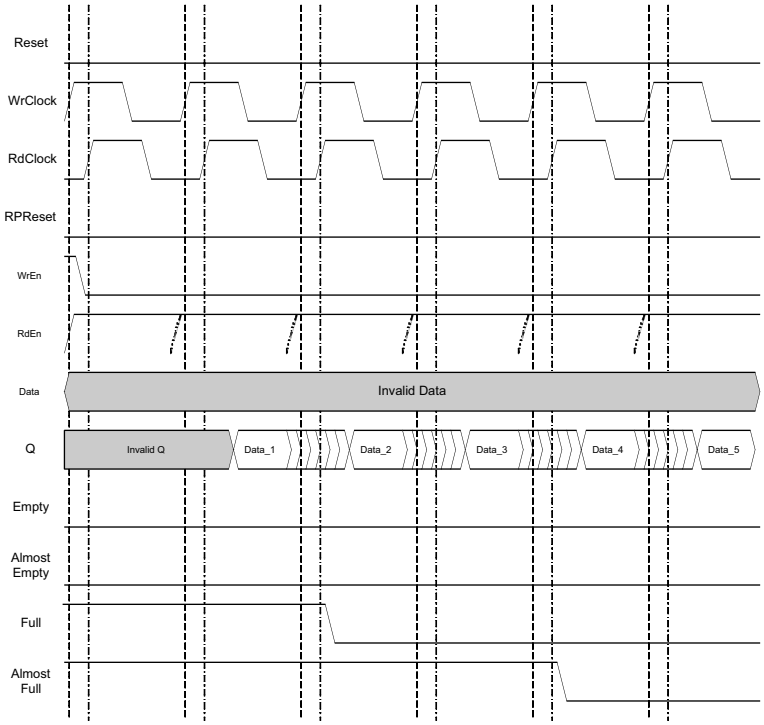
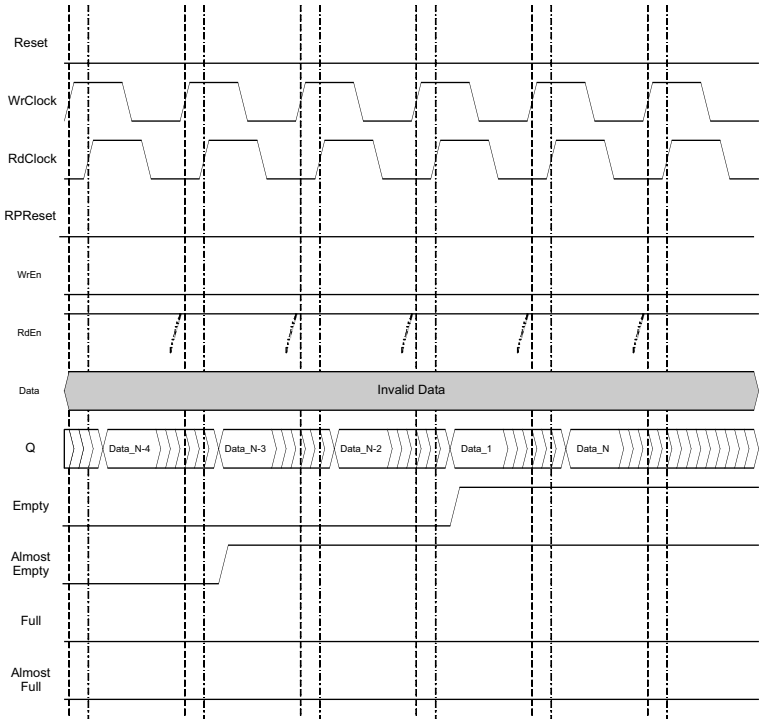
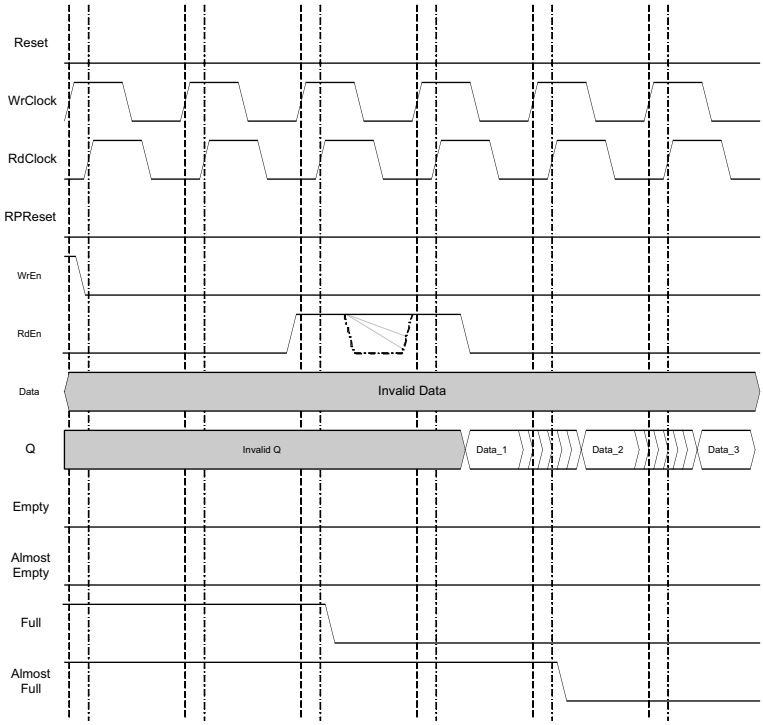


Figure 9-49. FIFO_DC With Output Registers, End of Data Read Cycle



Finally, if you select the option enable output register with RdEn, it still delays the data out by one clock cycle (as compared to the non-pipelined FIFO_DC), and the RdEn should be high also during that clock cycle. Otherwise the data takes an extra clock cycle when the RdEn is goes true.

Figure 9-50. FIFO_DC With Output Registers and RdEn on Output Registers

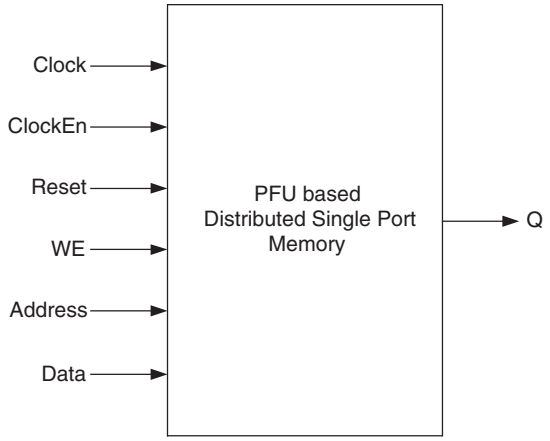


Distributed Single Port RAM (Distributed_SPRAM) – PFU Based

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

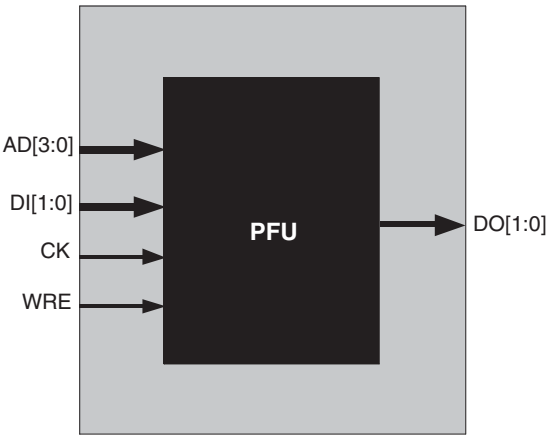
Figure 9-51 shows the Distributed Single Port RAM module as generated by the IPexpress.

Figure 9-51. Distributed Single Port RAM Module Generated by IPexpress



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

Figure 9-52. Distributed Single Port RAM (Distributed_SPRAM) for LatticeECP/EC and LatticeXP Devices



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

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

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

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

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

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

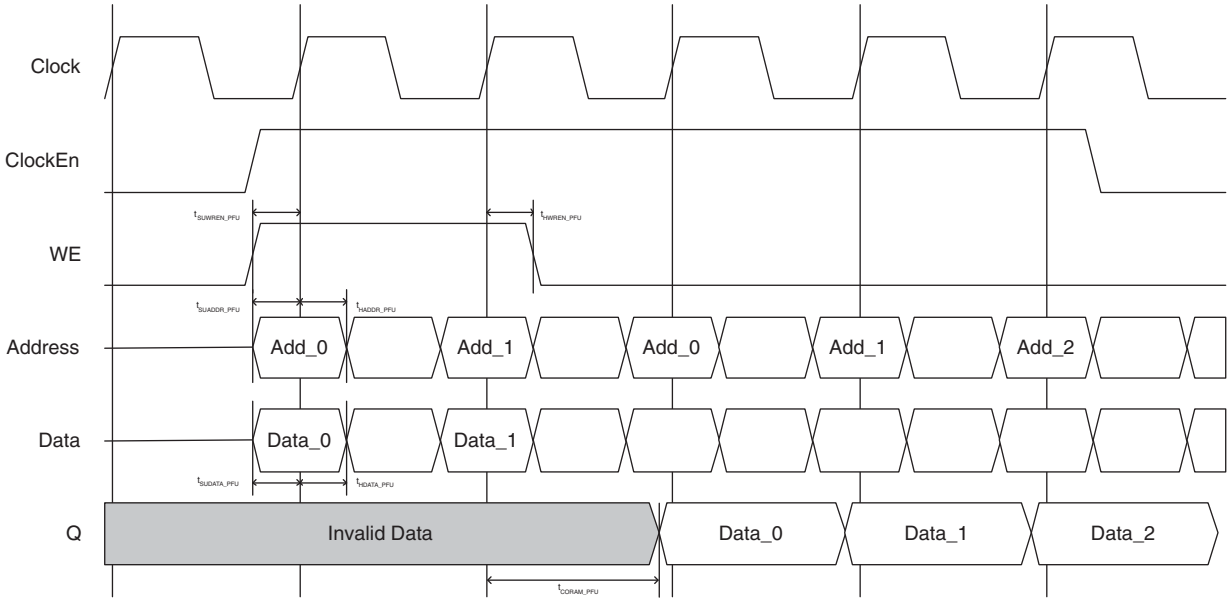
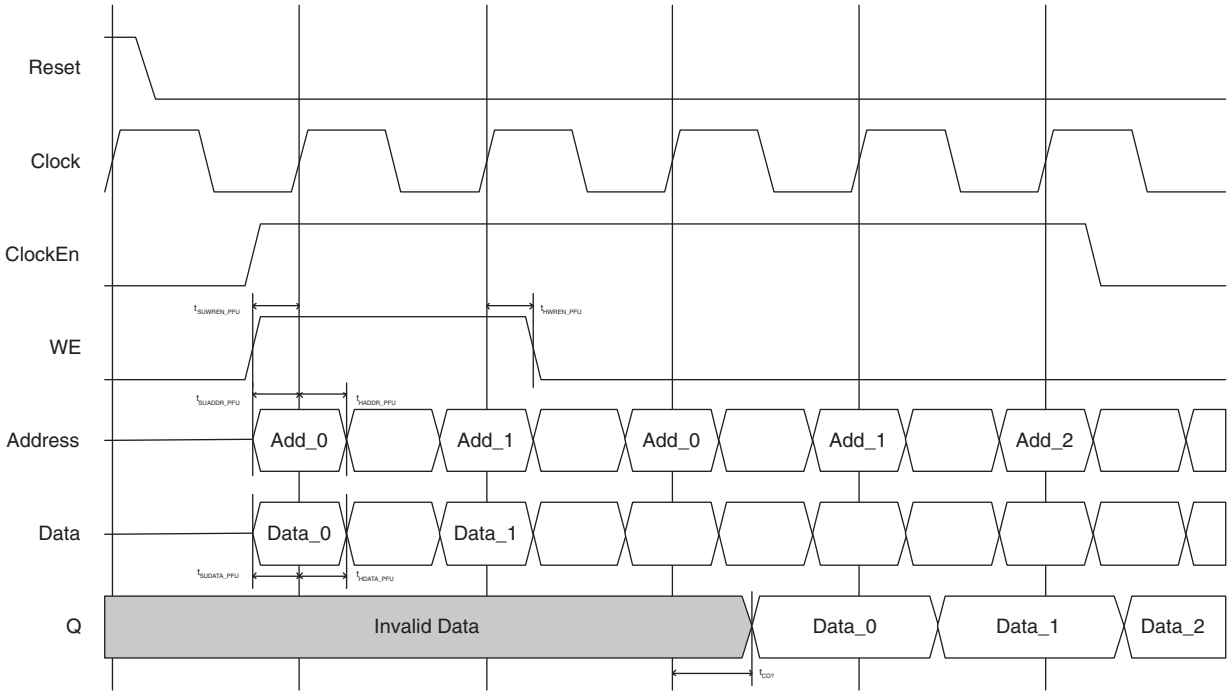


Figure 9-54. PFU Based Distributed Single Port RAM Timing Waveform – with Output Registers

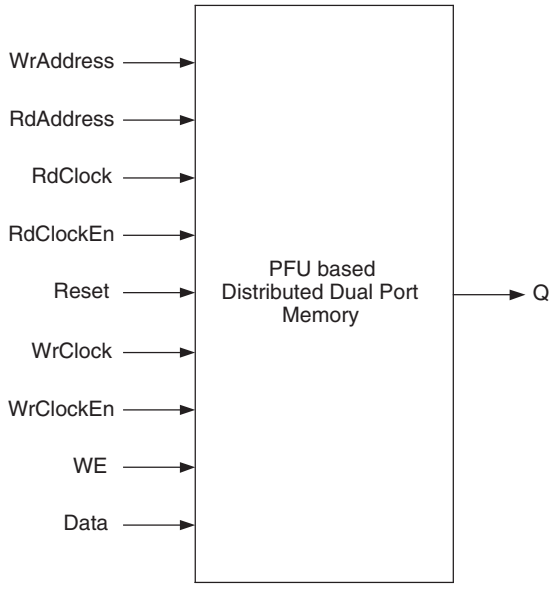


Distributed Dual Port RAM (Distributed_DPRAM) – PFU Based

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

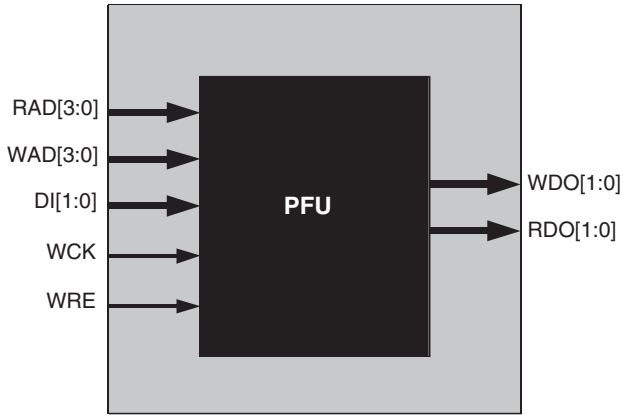
Figure 9-55 shows the Distributed Single Port RAM module as generated by IPexpress.

Figure 9-55. Distributed Dual Port RAM Module Generated by IPexpress



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

Figure 9-56. PFU-based Distributed Dual Port RAM for LatticeECP/EC and LatticeXP Devices



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

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

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

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

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

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

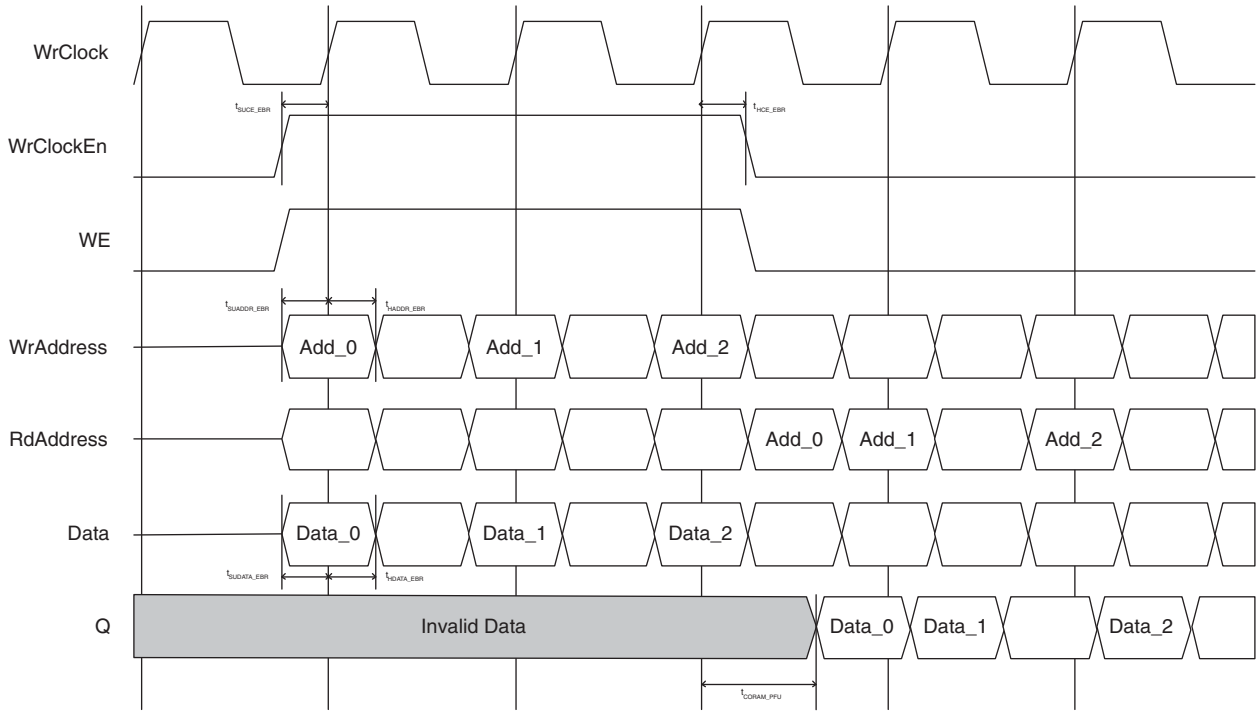
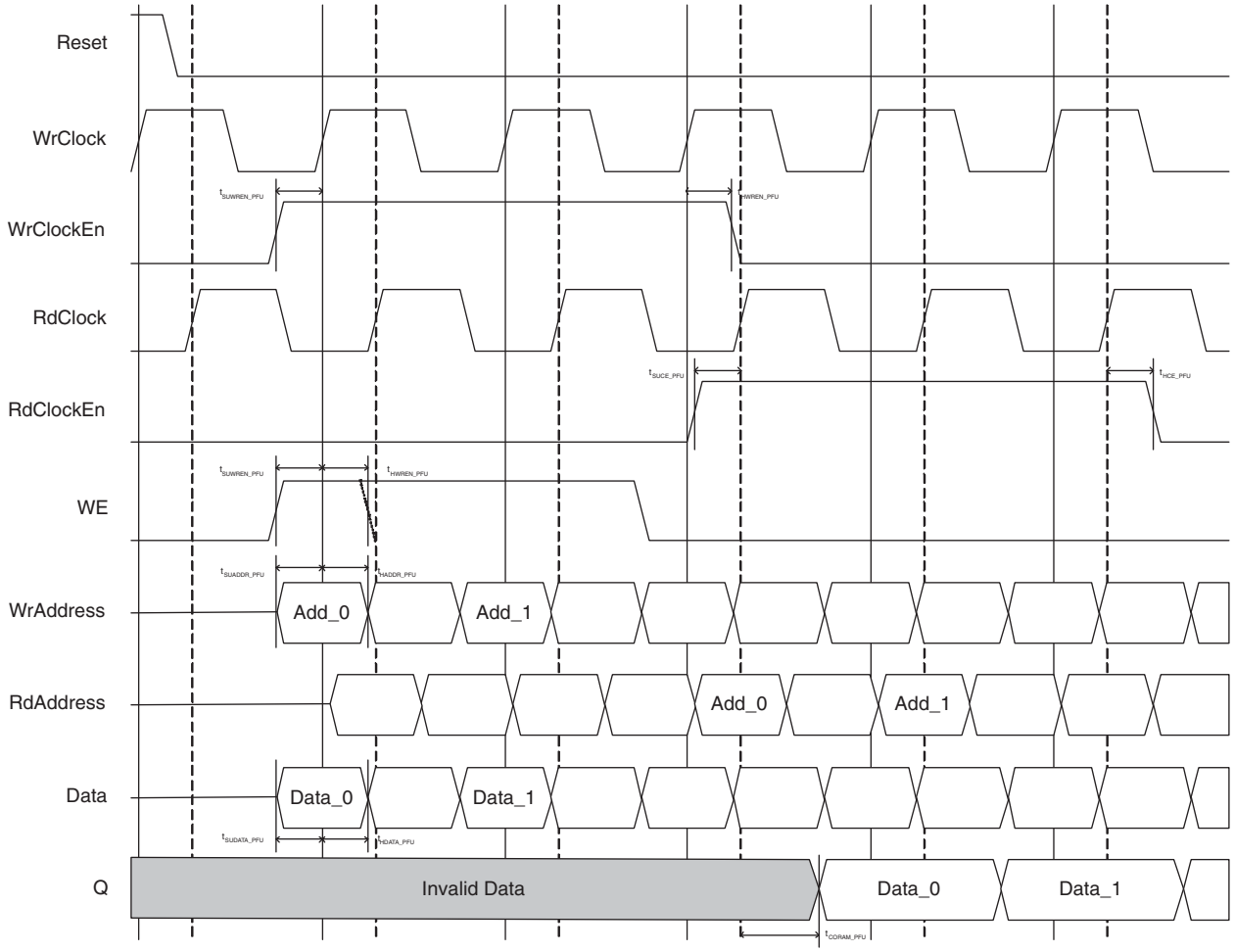


Figure 9-58. PFU Based Distributed Dual Port RAM Timing Waveform – With Output Registers

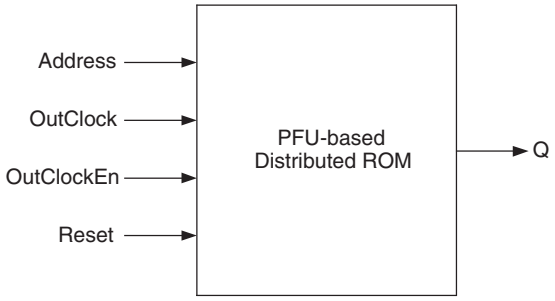


Distributed ROM (Distributed_ROM) – PFU Based

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

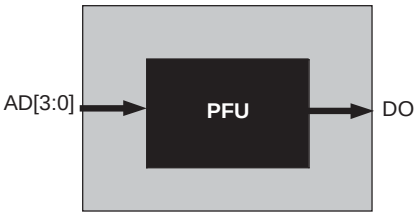
Figure 9-59 shows the Distributed Single Port RAM module as generated by IPexpress.

Figure 9-59. Distributed ROM Generated by IPexpress



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

Figure 9-60. PFU-based Distributed ROM (Sync_ROM) for LatticeECP/EC and LatticeXP Devices



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

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

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

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

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

Figure 9-61. PFU Based ROM Timing Waveform – without Output Registers

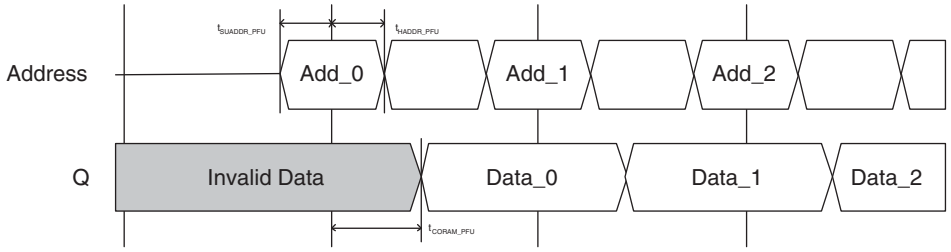
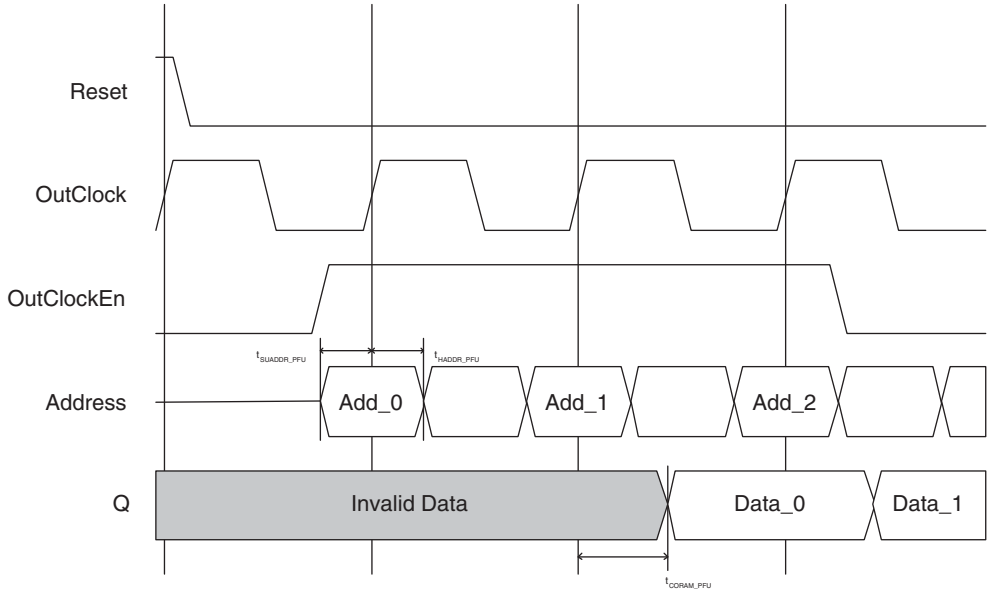


Figure 9-62. PFU Based ROM Timing Waveform – with Output Registers



Initializing Memory

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

Initialization File Format

The initialization file is an ASCII file, which a user can create or edit using any ASCII editor. IPexpress supports three types of memory file formats:

1. Binary file
2. Hex File
3. Addressed Hex

The file name for the memory initialization file is *.mem *(<file_name>.mem).* Each row depicts the value to be stored in a particular memory location. The number of characters (or the number of columns) represents the number of bits for each address (or the width of the memory module).

The Initialization File is primarily used for configuring the ROMs. The EBR in RAM mode can optionally use this Initialization File also to preload the memory contents.

Binary File

The file is essentially a text file of 0's and 1's. The rows indicate the number of words and columns indicate the width of the memory.

```
Memory Size 20x32
00100000010000000010000001000000
00000001000000010000000100000001
00000010000000100000001000000010
00000011000000110000001100000011
00000100000001000000010000000100
00000101000001010000010100000101
00000110000001100000011000000110
00000111000001110000011100000111
00001000010010000000100001001000
00001001010010010000100101001001
00001010010010100000101001001010
00001011010010110000101101001011
00001100000011000000110000001100
00001101001011010000110100101101
00001110001111100000111000111110
00001111001111110000111100111111
00010000000100000001000000010000
00010001000100010001000100010001
00010010000100100001001000010010
00010011000100110001001100010011
```

Hex File

The Hex file is essentially a text file of Hex characters arranged in a similar row-column arrangement. The number of rows in the file is same as the number of address locations, with each row indicating the content of the memory location.

```
Memory Size 8x16
A001
0B03
1004
CE06
0007
040A
0017
02A4
```

Addressed Hex (ORCA)

Addressed Hex consists of lines of address and data. Each line starts with an address, followed by a colon, and any number of data. The format of memfile is address: data data data data ... where address and data are hexadecimal numbers.

```
-A0 : 03 F3 3E 4F
-B2 : 3B 9F
```

The first line puts 03 at address A0, F3 at address A1, 3E at address A2,and 4F at address A3. The second line puts 3B at address B2 and 9F at address B3.

There is no limitation on the values of address and data. The value range is automatically checked based on the values of addr_width and data_width. If there is an error in an address or data value, an error message is printed. Users need not specify data at all address locations. If data is not specified at a certain address, the data at that

location is initialized to 0. IPexpress makes memory initialization possible both through the synthesis and simulation flows.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
June 2004	01.0	Initial release.
July 2004	01.1	Minor updates only.
October 2004	01.2	Minor updates only.
February 2005	01.3	Replace LatticeEC™ and LatticeECP™ LatticeEC™, LatticeECP™ and LatticeXP™
		Replace LatticeECP/ECLatticeEC/ECP and LatticeXP
		Replace LatticeEC or LatticeECP LatticeEC, LatticeECP or LatticeXP
		Added Hardware related information for the LatticeEC/ECP/XP devices
		Update Figure 8-4
		Figure 8-40 Replace “_EBR” with “_PFU” in the figure’s timing parameters.
April 2005	01.4	Updated the Trual Dual Port RAM and Module Manager Flow sections.
October 2005	01.5	Updated the block diagrams of modules generated by the Module Manager. Added section for Module Manager flow example. Added section for PMI flow.
February 2006	01.6	Removed the PMI support section
April 2006	01.7	Updated the Initializing Memory section
October 2006	01.8	Updated the FIFO section. Added dual port memory access notes in Appendix A.

Appendix A. Attribute Definitions

DATA_WIDTH

Data width is associated with the RAM and FIFO elements. The DATA_WIDTH attribute will define the number of bits in each word. It takes the values as defined in the RAM size tables in each memory module.

REGMODE

REGMODE or the Register mode attribute is used to enable pipelining in the memory. This attribute is associated with the RAM and FIFO elements. The REGMODE attribute takes the NOREG or OUTREG mode parameter that disables and enables the output pipeline registers.

RESETMODE

The RESETMODE attribute allows users to select the mode of reset in the memory. This attribute is associated with the block RAM elements. RESETMODE takes two parameters: SYNC and ASYNC. SYNC means that the memory reset is synchronized with the clock. ASYNC means that the memory reset is asynchronous to clock.

CSDECODE

CSDECODE or the Chip select decode attributes are associated to block RAM elements. CS, or Chip Select, is the port available in the EBR primitive that is useful when memory requires multiple EBR blocks cascaded. The CS signal would form the MSB for the address when multiple EBR blocks are cascaded. CS is a 3-bit bus, so it can cascade 8 memories easily. CSDECODE takes the following parameters: “000”, “001”, “010”, “011”, “100”, “101”, “110”, and “111”. CSDECODE values determine the decoding value of CS[2:0]. CSDECODE_W is chip select decode for write and CSDECODE_R is chip select decode for read for Pseudo Dual Port RAM. CSDECODE_A and CSDECODE_B are used for true dual port RAM elements and refer to the A and B ports.

WRITEMODE

The WRITEMODE attribute is associated with the block RAM elements. It takes the NORMAL, WRITETHROUGH, and READBEFOREWRITE mode parameters.

In NORMAL mode, the output data does not change or get updated, during the write operation. This mode is supported for all data widths.

In WRITETHROUGH mode, the output data is updated with the input data during the write cycle. This mode is supported for all data widths.

In READBEFOREWRITE mode, the output data port is updated with the existing data stored in the write address, during a write cycle. This mode is supported for x9, x18 and x36 data widths.

WRITEMODE_A and WRITEMODE_B are used for dual port RAM elements and refer to the A and B ports in case of a True Dual Port RAM.

For all modes (of the True Dual Port module), simultaneous read access from one port and write access from the other port to the same memory address is not recommended. The read data may be unknown in this situation. Also, simultaneous write access to the same address from both ports is not recommended. (When this occurs, the data stored in the address becomes undetermined when one port tries to write a 'H' and the other tries to write a 'L'.)

It is recommended that the designer implements control logic to identify this situation if it occurs and either:

1. Implement status signals to flag the read data as possibly invalid, or
2. Implement control logic to prevent the simultaneous access from both ports.

GSR

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

LatticeECP/EC and LatticeXP DDR Usage Guide

February 2007 Technical Note TN1050

Introduction

LatticeECP™, LatticeEC™ and LatticeXP™ devices support various Double Data Rate (DDR) and Single Data Rate (SDR) interfaces using the logic built into the Programmable I/O (PIO). SDR applications capture data on one edge of a clock while the DDR interfaces capture data on both the rising and falling edges of the clock, thus doubling the performance. This document will address in detail how to utilize the capabilities of the LatticeECP/EC and LatticeXP devices to implement both generic DDR and DDR memory interfaces.

DDR SDRAM Interfaces Overview

DDR SDRAM interfaces rely on the use of a data strobe signal, called DQS, for high-speed operation. When reading data from the external memory device, data coming into the device is edge aligned with respect to the DQS signal. This DQS strobe signal needs to be phase shifted 90 degrees before FPGA logic can sample the read data. When writing to a DDR SDRAM the memory controller (FPGA) must shift the DQS by 90 degrees to center align with the data signals (DQ). DQ and DQS are bi-directional ports. The same two signals are used for both write and read operations. A clock signal is also provided to the memory. This clock is provided as a differential clock (CLKP and CLKN) to minimize duty cycle variations. The memory also uses these clock signals to generate the DQS signal during a read via a DLL inside the memory. The skew between CLKP or CLKN and the SDRAM-generated DQS signal is specified in the DDR SDRAM data sheet. Figures 10-1 and 10-2 show DQ and DQS relationships for read and write cycles.

During read, the DQS signal is LOW for some duration after it comes out of tristate. This state is called Preamble. The state when the DQS is LOW before it goes into Tristate is the Postamble state. This is the state after the last valid data transition.

DDR SDRAM also require a Data Mask (DM) signals to mask data bits during write cycles. SDRAM interfaces typically are implemented with x8, x16 and x32 bits for each DQS signal. Note that the ratio of DQS to data bits is independent of the overall width of the memory. An 8-bit interface will have one strobe signal.

Figure 10-1. Typical DDR Interface

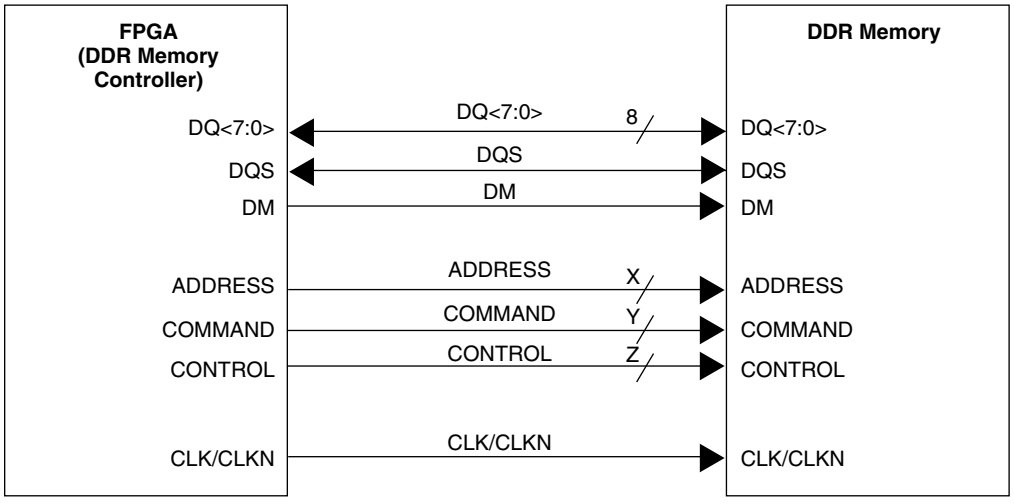


Figure 10-2. DQ-DQS During READ

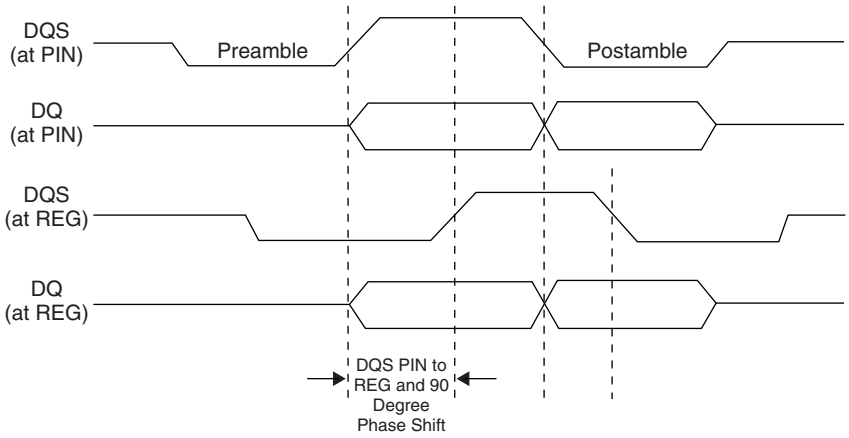
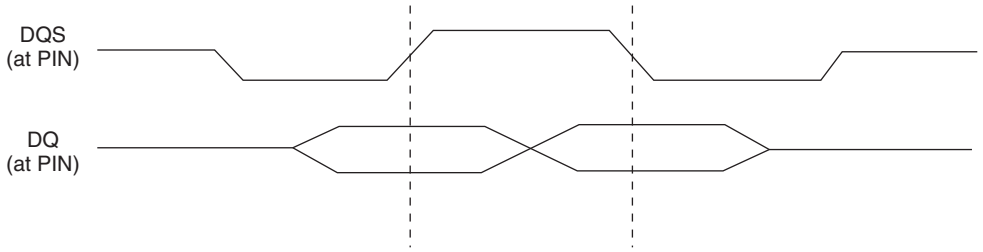


Figure 10-3. DQ-DQS During WRITE



Implementing DDR Memory Interfaces with the LatticeECP/EC Devices

This section describes how to implement the read and write sections of a DDR memory interface. It also provides details of the DQ and DQS grouping rules associated with the LatticeECP/EC and LatticeXP devices.

DQS Grouping

Each DQS group generally consists of at least 10 I/Os (1DQS, 8DQ and 1DM) to implement a complete 8-bit DDR memory interface. In the LatticeECP/EC devices each DQS signal will span across 16 I/Os and in the LatticeXP devices the DQS will span 14 I/Os. Any 10 of these 16 I/Os can be used to implement an 8-bit DDR memory interface. In addition to the DQS grouping, the user must also assign one reference voltage VREF1 for a given I/O bank.

The tables below show the total number of DQS groups available per I/O bank for each device and package.

Table 10-1. Number of DQS Banks in the LatticeECP/EC Device

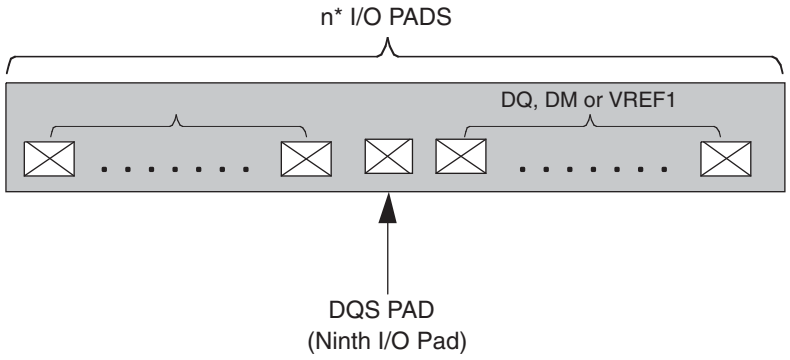
Package	Device	Total x8 DQS Groups	Number of DQS Groups per I/O Bank							
			0	1	2	3	4	5	6	7
100-pin TQFP	LFEC1	3	1	1	0	0	1	0	0	0
	LFEC3	3	1	1	0	0	1	0	0	0
144-pin TQFP	LFEC1	6	1	1	0	1	1	1	1	0
	LFEC3	6	1	1	0	1	1	1	1	0
	LFEC6/LFECP6	6	1	1	0	1	1	1	1	0
208-pin TQFP	LFEC1	6	1	1	0	1	1	1	1	0
	LFEC3	8+2[1]	1+1[1]	1	1	1	1	1+1[1]	1	1
	LFEC6/LFECP6	8+2[1]	1+1[1]	1	1	1	1	1+1[1]	1	1
	LFEC10/LFECP10	8+2[1]	1+1[1]	1	1	1	1	1+1[1]	1	1
256-ball fpBGA	LFEC3	10	2	1	1	1	1	2	1	1
	LFEC6/LFECP6	12	2	1	1	2	1	2	2	1
	LFEC10/LFECP10	12	2	1	1	2	1	2	2	1
	LFEC15/LFECP15	12	2	1	1	2	1	2	2	1
484-ball fpBGA	LFEC6/LFECP6	14	2	2	1	2	2	2	2	1
	LFEC10/LFECP10	18	3	2	2	2	2	3	2	2
	LFEC15/LFECP15	20	3	3	2	2	3	3	2	2
	LFEC20/LFECP20	22	3	3	2	3	3	3	3	2
	LFEC33/LFECP33	22	3	3	2	3	3	3	3	2
672-ball fpBGA	LFEC20/LFECP20	24	4	3	2	3	3	4	3	2
	LFEC33/LFECP33	30	4	4	3	4	4	4	4	3

1. 10 I/Os (1 DQS + 8 DQs + Bank VREF1) can function as a DDR interface in which the FPGA can have a DM output but not a DQS aligned input (in the same DDR bank as the rest of the system).

Table 10-2. Number of DQS Banks in the LatticeXP Device

Package	Device	Total x8 DQS Groups	Number of DQS Groups per I/O Bank							
			0	1	2	3	4	5	6	7
100-pin TQFP	LFXP3C/LFXP3E	2	0	0	0	0	0	1	0	1
	LFXP6C LFXP6E	2	0	0	0	0	0	1	0	1
144-pin TQFP	LFXP3C/LFXP3E	7	1	0	1	1	1	1	1	1
	LFXP6C/LFXP6E	7	1	0	1	1	1	1	1	1
208-pin PQFP	LFXP3C/LFXP3E	8	1	1	1	1	1	1	1	1
	LFXP6C/LFXP6E	8	1	1	1	1	1	1	1	1
256-ball fpBGA	LFXP3C/LFXP3E	12	2	2	1	1	2	2	1	1
	LFXP6C/LFXP6E	12	2	2	1	1	2	2	1	1
	LFXP10C/LFXP10E	16	2	2	2	2	2	2	2	2
	LFXP15C/LFXP15E	16	2	2	2	2	2	2	2	2
	LFXP20C/LFXP20E	20	3	3	2	2	3	3	2	2
388-ball fpBGA	LFXP10C/LFXP10E	16	2	2	2	2	2	2	2	2
	LFXP15C/LFXP15E	20	3	3	2	2	3	3	2	2
	LFXP20C/LFXP20E	20	3	3	2	2	3	3	2	2
484-ball fpBGA	LFXP15C/LFXP15E	20	3	3	2	2	3	3	2	2
	LFXP20C/LFXP20E	20	3	3	2	2	3	3	2	2
672-ball fpBGA	LFXP20C/LFXP20E	24	4	4	2	2	4	4	2	2

Figure 10-4. DQ-DQS Grouping



*For LatticeECP/EC: n = 16, for LatticeXP: n = 14.

Figure 10-4 shows a typical DQ-DQS group for both the LatticeECP/EC device and the LatticeXP device. The ninth I/O of this group of 16 I/Os (for LatticeECP/EC) or 14 I/Os (for LatticeXP) is the dedicated DQS pin. All eight pads before the DQS and seven (for LatticeECP/EC) or four (for LatticeXP) pads after the DQS are covered by this DQS bus span. The user can assign any eight of these I/O pads to be DQ data pins. Hence, to implement a 32-bit wide memory interface you would need to use four such DQ-DQS groups.

When not interfacing with the memory, the dedicated DQS pin can be used as a general purpose I/O. Each of the dedicated DQS pin is internally connected to the DQS phase shift circuitry. The pinout information contained in the LatticeECP/EC and LatticeXP device data sheets shows pin locations for the DQS pads. Table 10-2 shows an extract from the LatticeECP/EC data sheet. In this case, the DQS is marked as LDQS6 (L=left side, 6 =associated PFU row/column). Since DQS is always the fifth true pad in the DQ-DQS group, counting from low to high PFU row/column number, LDQS6 will cover PL2A to PL9B. Following this convention, there are eight pads before and seven pads after DQS for DQ available following counter-clockwise for the left and bottom sides of the device, and following clockwise for the top and right sides of the device. The user can assign any eight of these pads to be DQ data signals. The LatticeXP device follows the same method.

Table 10-3. EC20 Pinout (from LatticeECP/EC Family Data Sheet)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PL2A	7	T	VREF2_7	D4	E3
PL2B	7	C	VREF1_7	E4	E4
PL3A	7	T	—	C3	B1
PL3B	7	C	—	B2	C1
PL4A	7	T	—	E5	F3
PL4B	7	C	—	F5	G3
PL5A	7	T	—	D3	D2
PL5B	7	C	—	C2	E2
PL6A	7	T	LDQS6	F4	D1
PL6B	7	C	—	G4	E1
PL7A	7	T	—	E3	F2
PL7B	7	C	—	D2	G2
PL8A	7	T	LUM0_PLLT_IN_A	B1	F6
PL8B	7	C	LUM0_PLLC_IN_A	C1	G6
PL9A	7	T	LUM0_PLLT_FB_A	F3	H4
PL9B	7	C	LUM0_PLLC_FB_A	E2	G4
PL11A	7	T	—	G5	J4

Table 10-3. EC20 Pinout (from LatticeECP/EC Family Data Sheet)

Ball Function	Bank	LVDS	Dual Function	484 fpBGA	672 fpBGA
PL11B	7	C	—	H6	J5
PL12A	7	T	—	G3	K4

DDR Software Primitives

This section describes the software primitives that can be used to implement DDR interfaces and provides details about how to instantiate them in the software. The primitives described include:

- DQSDLL The DQS delay calibration DLL
- DQSBUF The DQS delay function and the clock polarity selection logic
- INDDRXB The DDR input and DQS to system clock transfer registers
- ODDRXB The DDR output registers

An HDL usage example for each of these primitives is listed in Appendices B and C.

DQSDLL

The DQSDLL will generate a 90-degree phase shift required for the DQS signal. This primitive will implement the on-chip DQSDLL. Only one DQSDLL should be instantiated for all the DDR implementations on one half of the device. The clock input to this DLL should be at the same frequency as the DDR interface. The DLL will generate the delay based on this clock frequency and the update control input to this block. The DLL will update the dynamic delay control to the DQS delay block when this update control (UDDCNTL) input is asserted. Figure 10-5 shows the primitive symbol. The active low signal on UDDCNTL updates the DQS phase alignment and should be initiated at the beginning of READ cycles.

Figure 10-5. DQSDLL Symbol

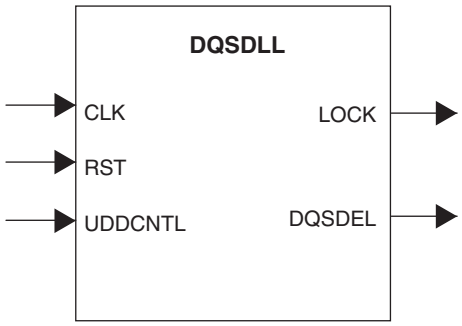


Table 10-4 provides a description of the ports.

Table 10-4. DQSDLL Ports

Port Name	I/O	Definition
CLK	I	System CLK should be at frequency of the DDR interface, from the FPGA core.
RST	I	Resets the DQSDLL
UDDCNTL	I	Provides an update signal to the DLL that will update the dynamic delay. When held low this signal will update the DQSDEL.
LOCK	O	Indicates when the DLL is in phase
DQSDEL	O	The digital delay generated by the DLL should be connected to the DQSBUF primitive.

DQSDLL Configuration Attributes

By default this DLL will generate a 90-degree phase shift for the DQS strobe based on the frequency of the input reference clock to the DLL. The user can control the sensitivity to jitter by using the LOCK_SENSITIVITY attribute. This configuration bit can be programmed to be either “HIGH” or “LOW”.

The DLL Lock Detect circuit has two modes of operation controlled by the LOCK_SENSITIVITY bit, which selects more or less sensitivity to jitter. If this DLL is operated at or above 150 MHz, it is recommended that the LOCK_SENSITIVITY bit be programmed “HIGH” (more sensitive). For operation running at or under 100 MHz it is recommended that the bit be programmed “LOW” (more tolerant). For 133 MHz, the LOCK_SENSITIVITY bit can go either way.

DQSBUF

This primitive implements the DQS Delay and the DQS transition detector logic. Figure 10-6 shows the DQSBUFB function. The preamble detect signal is also generated within this primitive.

Figure 10-6. DQSBUFB Function

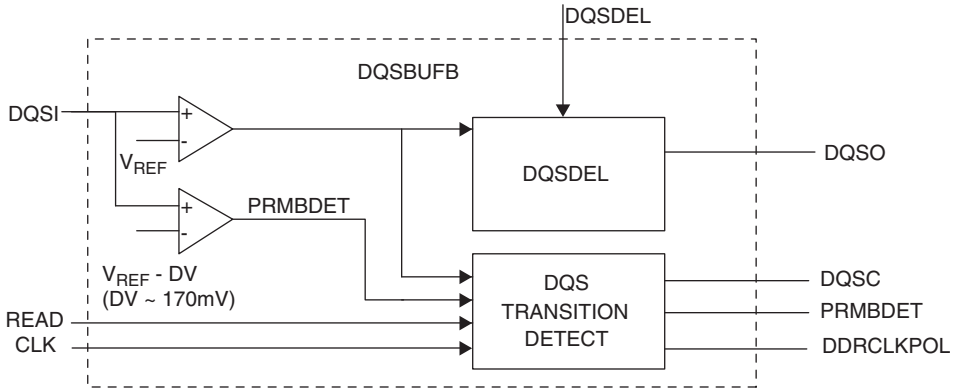


Figure 10-7 shows the primitive symbol and its ports. DQSI is the DQS signal from the memory. PRMBDET is the preamble detect signal that is generated from the DQSI input. READ and CLK are user interface signals coming from the FPGA logic. The DQSDLL block sends digital control line DQSDEL to this block. The DQS is delayed based on this input from the DQSDLL. DQSO is the delayed DQS and is connected to the clock input of the first set of DDR registers.

Figure 10-7. DQSBUFB Symbol

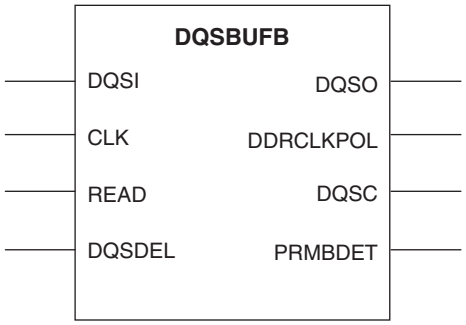


Table 10-5 provides a description of the I/O ports associated with the DQSBUFB primitive.

Table 10-5. DQSBUFB Ports

Port Name	I/O	Definition
DQSI	I	DQS strobe signal from memory
CLK	I	System CLK
READ	I	Read generated from the FPGA core
DQSDEL	I	DQS delay from the DQSDLL primitive
DQSO	O	Delayed DQS Strobe signal, to the input capture register block
DQSC	O	DQS Strobe signal before delay, going to the FPGA core logic
DDRCLKPOL	O	DDR Clock Polarity signal
PRMBDET	O	Preamble detect signal, going to the FPGA core logic

Notes:

1. The DDR Clock Polarity output from this block should be connected to the DDCLKPOL inputs of the input register blocks (IDDRXB).

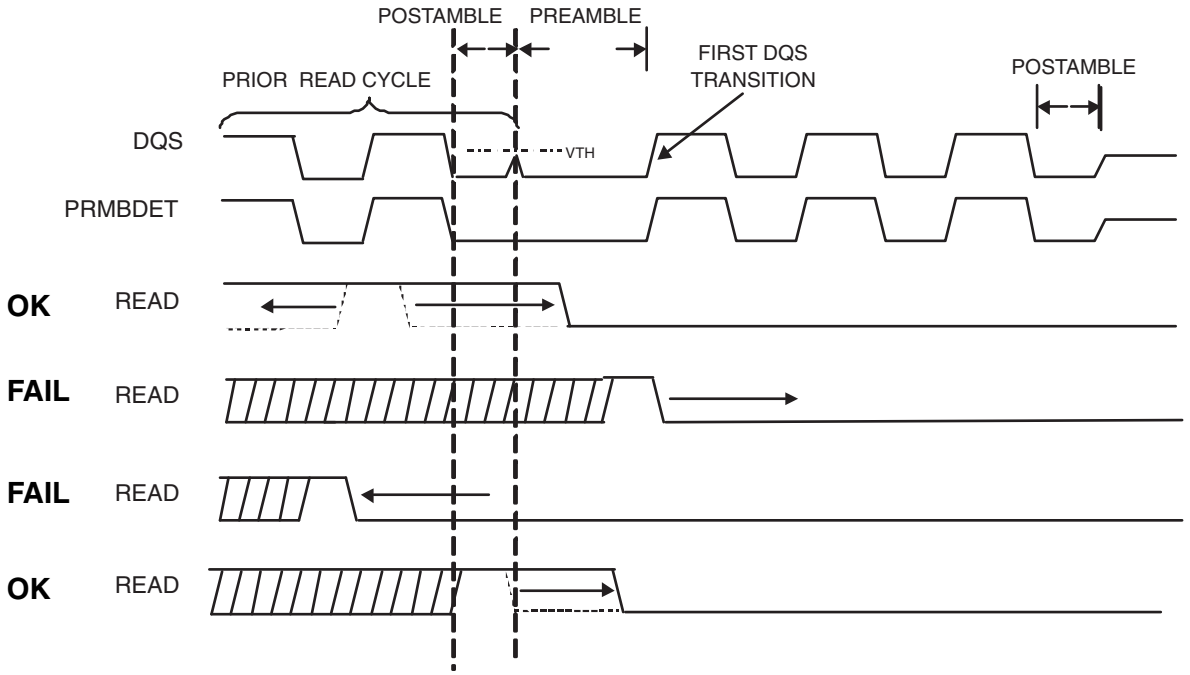
READ Pulse Generation

The READ signal to the DQSBUFB block is internally generated in the FPGA core. The Read signal will go high when the READ command to control the DDR SDRAM is initially asserted. This should normally precede the DQS preamble by one cycle yet may overlap the trailing bits of a prior read cycle. The DQS Detect circuitry of the LatticeECP/EC and LatticeXP devices require the falling edge of the READ signal to be placed within the preamble stage.

The preamble state of the DQS can be detected using the CAS latency and the round trip delay for the signals between the FPGA and the memory device. Note that the internal FPGA core generates the READ pulse. The rise of the READ pulse needs to coincide with the initial READ Command of the Read Burst and needs to go low before the Preamble goes high.

Figure 10-8 shows the READ Pulse Timing Example with respect to the PRMBDET signal.

Figure 10-8. READ Pulse Generation



IDDRXB

This primitive will implement the input register block. The software defaults to CE Enabled unless otherwise specified. The ECLK input is used to connect to the DQS strobe coming from the DQS delay block (DQSBUFB primitive). The SCLK input should be connected to the system (FPGA) clock. The SCLK and CE inputs to this primitive will be used primarily to synchronize the DDR inputs. DDRCLKPOL is an input from the DQS Clock Polarity tree. This signal is generated by the DQS Transition detect circuit in the hardware. Figure 10-9 shows the primitive symbol and the I/O ports.

Figure 10-9. IDDRXB Symbol

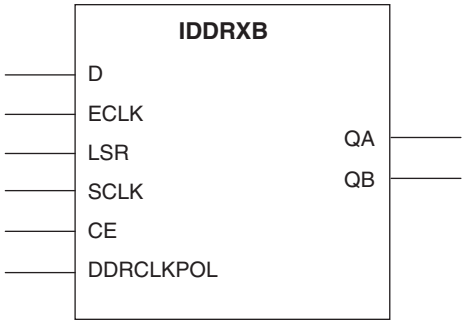


Table 10-6 provides a description of all I/O ports associated with the IDDRXB primitive.

Table 10-6. IDDRXB Ports

Port Name	I/O	Definition
D	I	DDR data
ECLK	I	The phase shifted DQS should be connected to this input
LSR	I	Reset
SCLK	I	System CLK
CE	I	Clock enable
DDRCLKPOL	I	DDR clock polarity signal
QA	O	Data at the positive edge of the CLK
QB	O	Data at the negative edge of the CLK

Note:

1. The DDRCLKPOL input to IDDRXB should be connected to the DDRCLKPOL output of DQSBUFB.

ODDRXB

The ODDRXB primitive implements both the write and the tristate functions. This primitive is used to output DDR data and the DQS strobe to the memory. The CKP and CKN can also be generated using this primitive. All the DDR output tristate implementations are also implemented using the same primitive.

Figure 10-10 shows the ODDRXB primitive symbol and its I/O ports.

Figure 10-10. ODDRXB Symbol

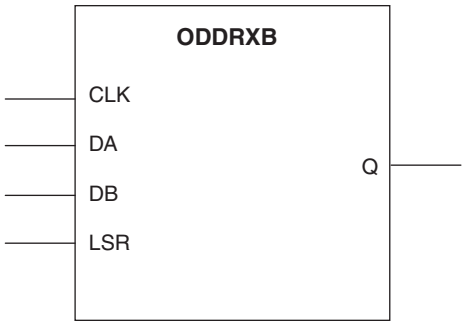


Table 10-7 provides a description of all I/O ports associated with the ODDRXB primitive.

Table 10-7. ODDRXB Ports

Port Name	I/O	Definition
CLK	I	System CLK
DA	I	Data at the positive edge of the clock
DB	I	Data at the negative edge of the clock
LSR	I	Reset
Q	I	DDR data to the memory

Notes:

1. LSR should be held low during DDR Write operation. By default, the software will be implemented CE High and LSR low.
2. DDR output and tristate registers do not have CE support. LSR is available for the tristate DDRX mode (while reading). The LSR will default to set when used in the tristate mode.
3. CE and LSR support is available for the regular (non-DDR) output mode.
4. When asserting reset during DDR writes, it is important to keep in mind that this would only reset the FFs and not the latches.

Memory Read Implementation

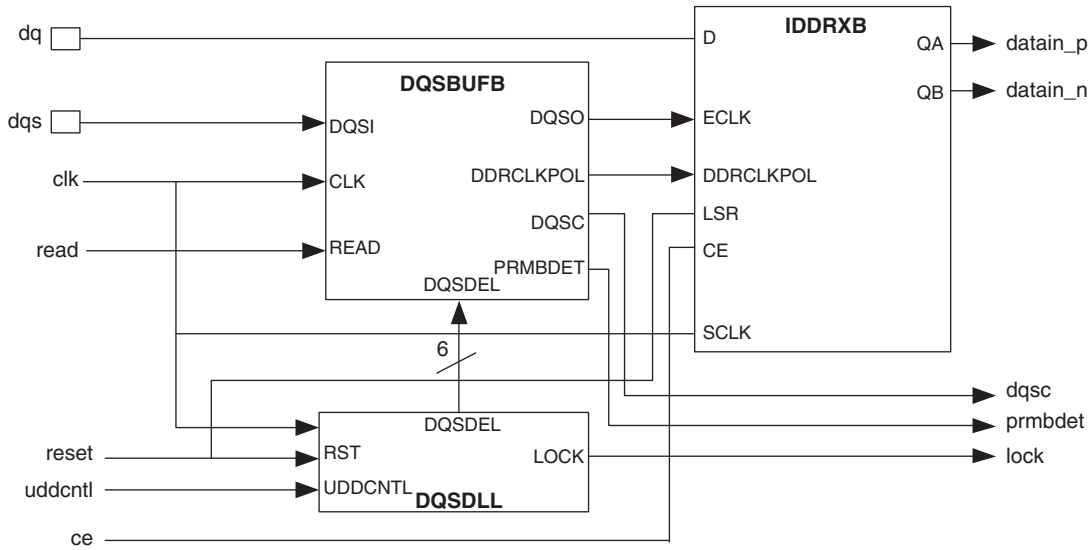
The LatticeECP/EC and LatticeXP devices contain a variety of features to simplify implementation of the read portion of a DDR interface:

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

The LatticeECP/EC and LatticeXP device data sheets detail these circuit elements.

Three primitives in the Lattice ispLEVER® design tools represent the capability of these three elements. The 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 10-11 illustrates how to hook these primitives together to implement the read portion of a DDR memory interface. The DDR Software Primitives section describes each of the primitives and its instantiation in more detail. Appendices A and B provide example code to implement the complete I/O section of a memory interface within a LatticeECP/EC or LatticeXP device.

Figure 10-11. Software Primitive Implementation for Memory READ



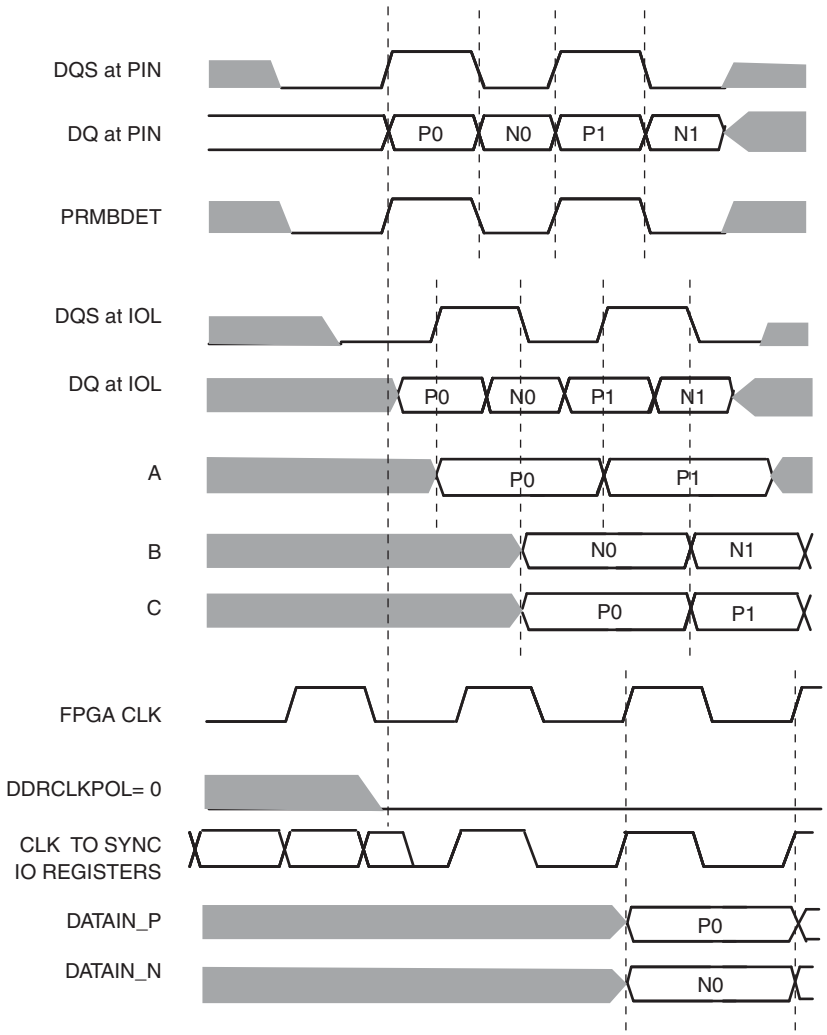
Read Timing Waveforms

Figure 10-12 and Figure 10-13 show READ data transfer for two cases based on the results of the DQS Transition detector logic. This circuitry decides whether or not to invert the phase of FPGA system CLK to the synchronization registers based on the relative phases of PRMBDET and CLK.

- Case 1 – If CLK = 0 on the 1st PRMBDET transition, then DDRCLKPOL = 0, hence no inversion required. (Figure 10-12)
- Case 2 – If CLK=1 on the 1st PRMBDET then DDRCLKPOL = 1, the system clock (CLK) needs to be inverted before it is used for synchronization. (Figure 10-13)

The signals A, B and C illustrate the Read Cycle half clock transfer at different stages of IDDRX registers. The first stage of the register captures data on the positive edge as shown by signal A and negative edge as shown by signal B. The data stream A goes through an additional half clock cycle transfers shown by signal C. Phase aligned data streams B and C are presented to the next stage registers clocked by the FPGA CLK

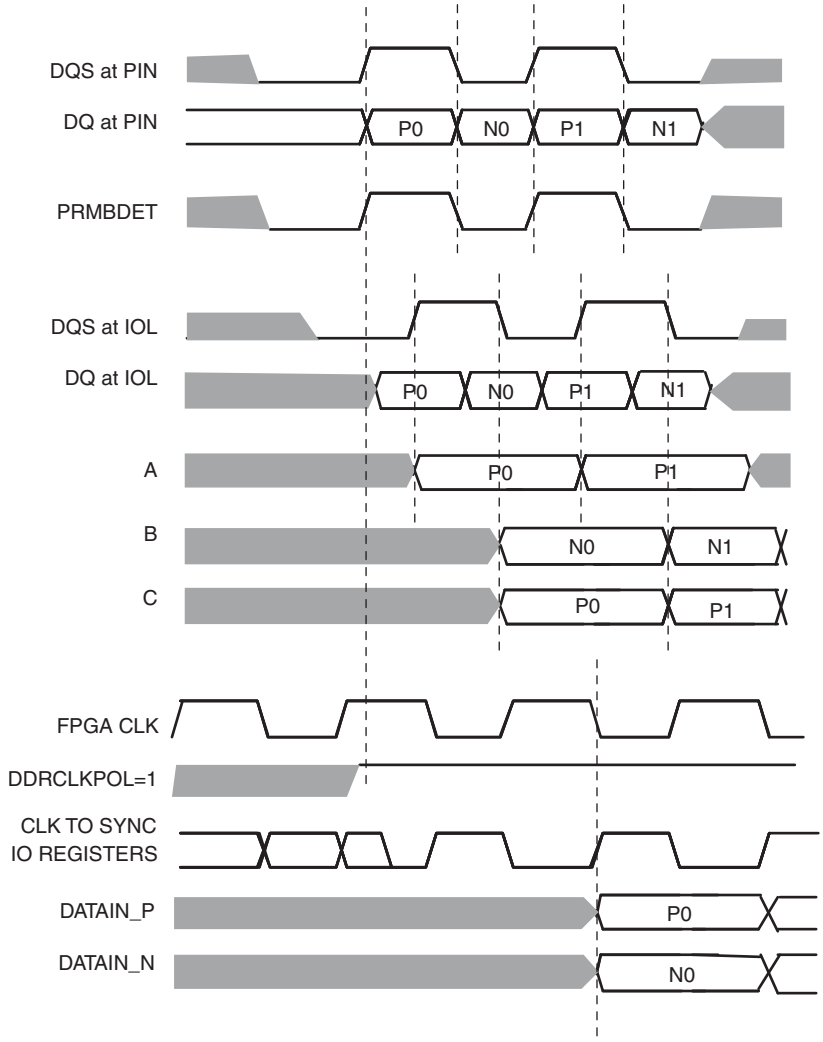
Figure 10-12. READ Data Transfer When DDRCLKPOL=0



Notes -

(1) DDR memory sends DQ aligned to DQS strobe.
(2) The DQS Strobe is delayed by 90 degree using the dedicated DQS logic.
(3) DQ is now center aligned to DQS Strobe.
(4) PRMBDET is the Preamble detect signal generated using the DQSBUFB primitive. This is used to generate the DDRCLKPOL signal.
(5) The first set of IO registers A and B, capture data on the positive edge and negative edge of DQS.
(6) IO register C transfers data so that both data are now aligned to negative edge of DQS.
(7) DDCLKPOL signal generated will determine if the CLK going into the synchronization registers need to be inverted. In this case, the DDRCLKPOL=0 as the CLK is LOW at the 1st rising edge of PRMBDET.
(8) The IO Synchronization registers capture data at on positive edge of the FPGA CLK.

Figure 10-13. Read Data Transfer When DDRCLKPOL=1



Notes -

(1) DDR memory sends DQ aligned to DQS strobe.
(2) The DQS Strobe is delayed by 90 degree using the dedicated DQS logic.
(3) DQ is now center aligned to DQS Strobe.
(4) PRMBDET is the Preamble detect signal generated using the DQSBUFB primitive. This is used to generate the DDRCLKPOL signal.
(5) The first set of IO registers A and B, capture data on the positive edge and negative edge of DQS.
(6) IO register C transfers data so that both data are now aligned to negative edge of DQS.
(7) DDCLKPOL signal generated will determine if the CLK going into the synchronization registers need to be inverted. In this case, the DDRCLKPOL=1 as the CLK is HIGH at the 1st rising edge of PRMBDET.
(8) The IO Synchronization registers capture data at on negative edge of the FPGA CLK.

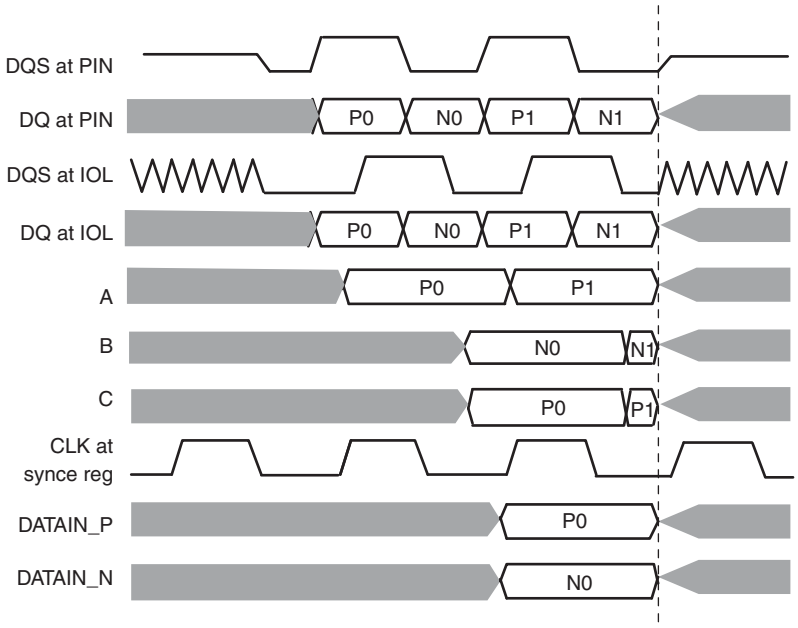
Data Read Critical Path

Data in the second stage DDR registers can be registered either on the positive edge or on the falling edge of FPGA clock depending on the DDRCLKPOL signal. In order to ensure that the data transferred to the FPGA core registers is aligned to the rising edge of system CLK, this path should be constrained with a half clock transfer. This half clock transfer can be forced in the software by assigning a multicycle constraint (multicycle of 0.5 X) on all the data paths to the first PFU register.

DQS Postamble

At the end of a READ cycle, the DDR SDRAM device executes the READ cycle postamble and then immediately tristates both the DQ and DQS output drivers. Since neither the memory controller (FPGA) nor the DDR SDRAM device are driving DQ or DQS at that time, these signals float to a level determined by the off-chip termination resistors. While these signals are floating, noise on the DQS strobe may be interpreted as a valid strobe signal by the FPGA input buffer. This can cause the last READ data captured in the IOL input DDR registers to be overwritten before the data has been transferred to the free running resynchronization registers inside the FPGA.

Figure 10-14. Postamble Effect on READ



LatticeECP/EC and LatticeXP devices have extra dedicated logic in the in the DQS Delay Block that will prevent this postamble problem. The DQS postamble logic is automatically implemented when the user instantiates the DQS Delay logic (DQSBUFB software primitive) in a design.

This postamble solution was implemented in all the devices of the LatticeECP/EC and LatticeXP families except the LFEC20/LFECP20 device. For this device, it is recommended that the user issue an extra READ command to assure correct data has been transferred to the synchronization registers.

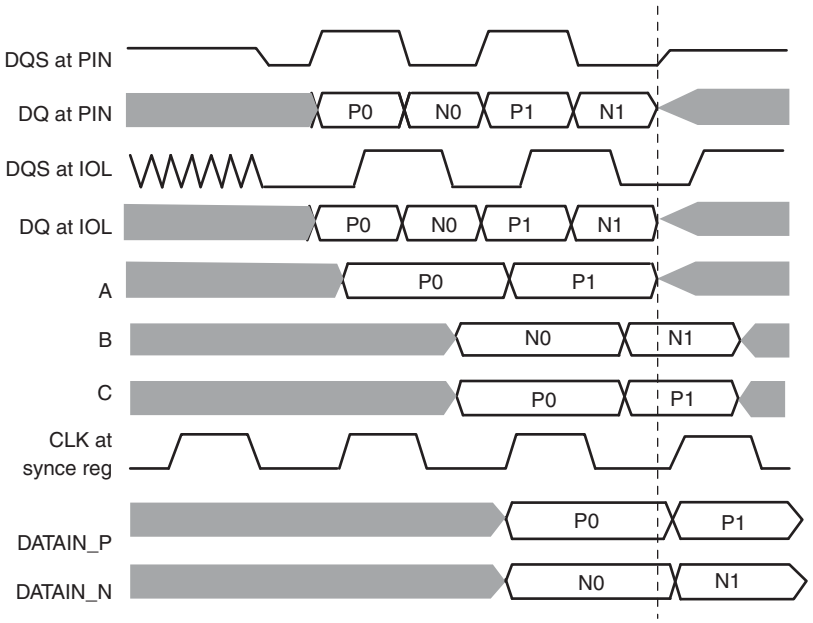
The circumstances under which the extended READ cycle is issued are given in Table 10-8.

Table 10-8. DDR Read Postamble

Current Command	Next Command	Action	Lost Cycles
Read (Row x, Bank y)	Read (Row x, Bank y)	None.	None
Read (any address)	NOP	Extend the current read command.[1]	3
Read (Row x, Bank y)	Read (Row n, Bank y)	Extend the current read to (Row x, Bank y) consecutive to current command	3
Read (Row x, Bank y)	Read (Row x, Bank n)	If the Row x, Bank n was open, do nothing. Else, extend the current read to Row x, Bank y	3
Read (any address)	Write/LMR	Extend the current read command.	3

1. Current read is extended one or more additional clock cycles.

Figure 10-15. Postamble Solution with Extra READ Command



Memory Write Implementation

To implement the write portion of a DDR memory interface, two streams of single data rate data must be multiplexed together with data transitioning on both edges of the clock. In addition, during a write cycle, DQS must arrive at the memory pins center-aligned with data, DQ. Along with the strobe and data this portion of the interface provides the CLKP, CLKN Address/Command and Data Mask (DM) signals to the memory.

LatticeECP/EC and LatticeXP devices contain DDR output and tri-state registers along with PLLs that allow the easy implementation of the write portion of the DDR memory interfaces. The DDR output registers can be accessed in the design tools via the ODDRXB primitive.

All DDR output signals (“ADDR, CMD”, DQS, DQ, DM) are initially aligned to the rising edge of CLK inside the FPGA core. These signals are used for the entire DDR write interface or the controls of DDR read interface. The relative phase of the signals may be adjusted in the IOL logic before departing the FPGA. The adjustments are shown in Figure 16

The adjustments are as follows:

The PLL is used to generate a 90 degree phase shifted clock. This 90 degree phase shifted clock will be used to generate DQS and the differential clocks going to the memory.

The CLKP needs to be centered relative to the ADDR,CMD signal, which is an SDR signal. This is accomplished by inverting the CLKP signal relative to the PLL’s 90 degree phase shifted CLK.

The DDR clock can be generated by assigning “0” to the DA input and “1” to the DB inputs of the ODDRXB primitive as shown in Figure 10-16. This is then fed into a SSTL25 differential output buffer to generate CLKP and CLKN differential clocks. Generating the CLKN in this manner would prevent any skew between the two signals.

The DDR interface specification for t_{DSS} and t_{DSH} parameters, defined as DQS falling to CLKP rising setup and hold times must be met. This is met by making CLKP and DQS identical in phase. DQS is inverted to match CLKP (= CLK + 270). This is accomplished by routing the positive DQS data in core logic to DB, and negative DQS data in core logic to DA inputs of the ODDRXB primitive.

Internally the DQS and ADDR/CMD signals are clocked using the primary FPGA clock. Therefore, the user will need to do a 1/4 (one-quarter) clock transfer from the core logic to the DDR registers. Timing can be hard to meet, so it is recommended that the user first register these signals with the inverted Clock, so that the transfer from the core logic to I/O registers will only require a 1/2 (half) clock transfer.

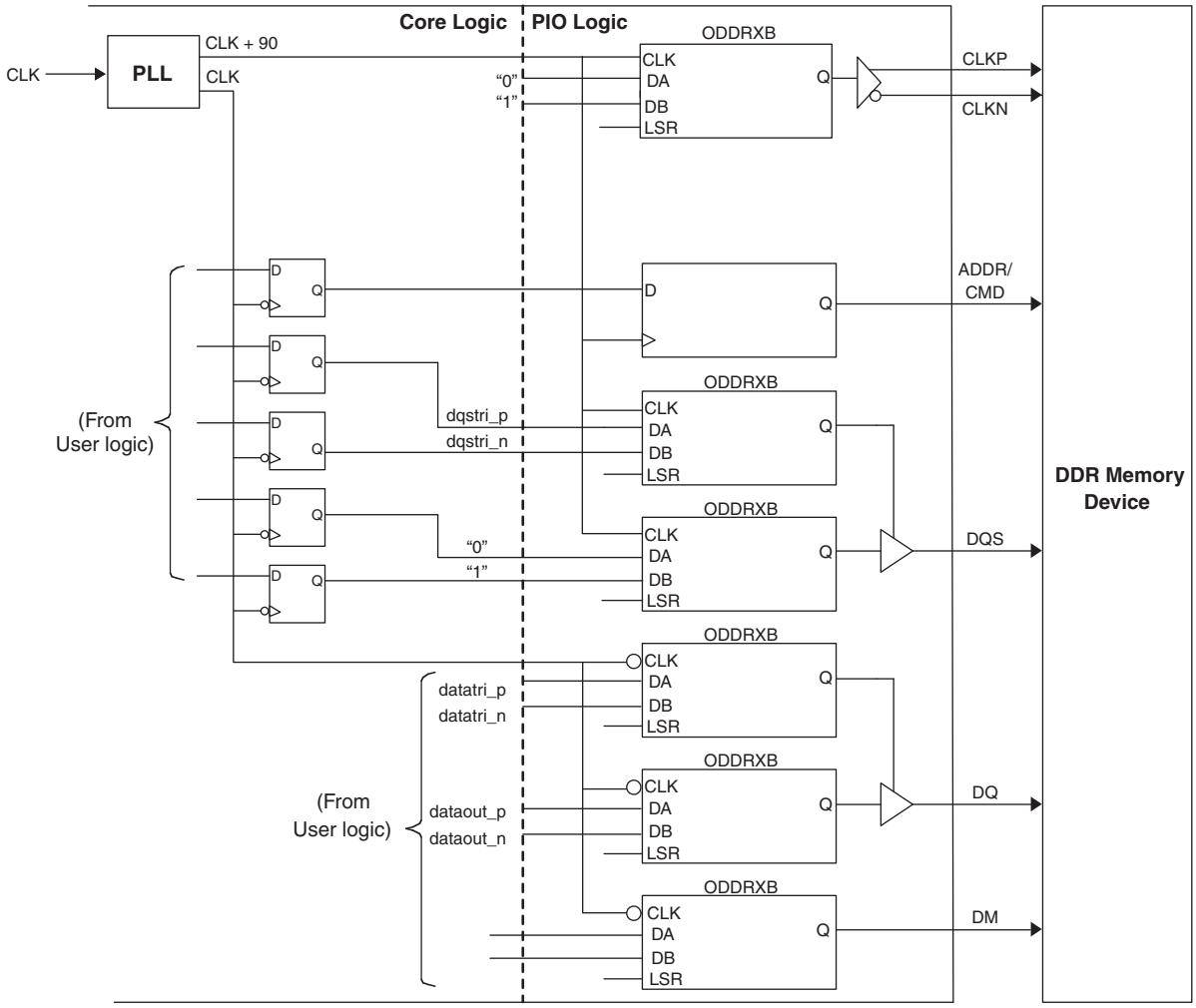
The data DQ and DM needs to be delayed by 90° as it leaves the FPGA. This is to center the data and data mask relative to the DQS when it reaches the DDR memory. This can be accomplished by inverting the CLK to the DQ and DM data.

The DM signal is generated using the same clock as the DQ data pin. The memory masks the DQ signals if the DM pins are driven high.

The tristate control for the data output can also be implemented using the ODDRXB primitive.

Figure 10-16 illustrates how to hook up the ODDRXB primitives and the PLL. The DDR Software Primitives section describes each of the primitives and its instantiation in more detail. Appendix A and Appendix B provide example code for implementing the complete I/O section of a memory interface for a LatticeECP/EC or LatticeXP device.

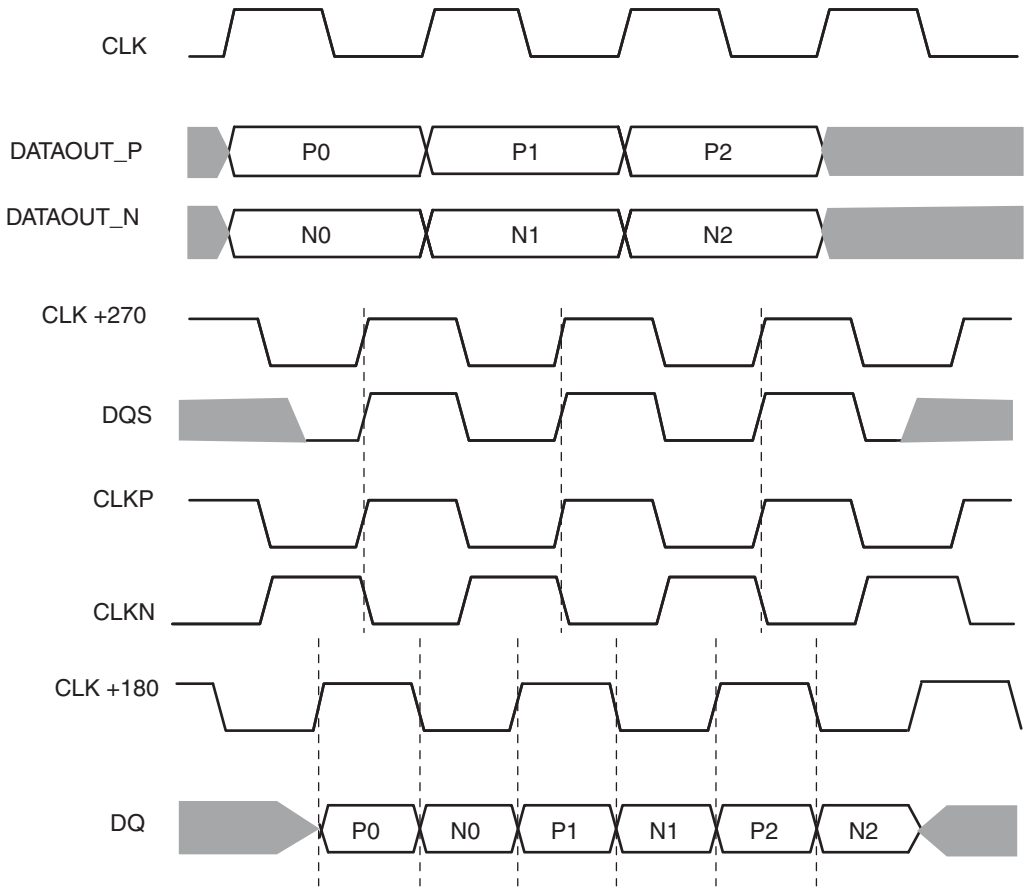
Figure 10-16. Software Primitive Implementation for Memory Write



Write Timing Waveforms

Figure 10-17 shows DDR write side data transfer timing for the DQ Data pad and the DQS Strobe Pad. When writing to the DDR memory device, the DM (Data Mask) and the ADDR/ CMD (Address and Command) signals are also sent to the memory device along with the data and strobe signals.

Figure 10-17. DDR Write Data Transfer for DQ Data



Notes -
(1) DATAOUT_P and DATAOUT_N are inputs to the DDR output registers.
(2) DQS is generated at 270 degree phase of CLK.
(3) CLKP is generated simular to DQS and CLKN is the inverted CLKP.
(4) DQ is generated at 180 degree phase of CLK.
(5) DQ is center aligned with the DQS strobe signal when it reaches the memory.

Design Rules/Guidelines

Listed below are some rules and guidelines to keep in mind when implementing DDR memory interfaces in the LatticeECP/EC and LatticeXP devices.

- The LatticeECP/EC and LatticeXP devices have dedicated DQ-DQS banks. Please refer to the logical signal connections of the groups in the LatticeECP/EC and LatticeXP data sheets before locking these pins.
- There are two DQSDLLs on the device, one for the top half and one for the bottom half. Hence, only one DQSDLL primitive should be instantiated for each half of the device. Since there is only one DQSDLL on each half of the device, all the DDR memory interfaces on that half of the device should run at the same frequency. Each DQSDLL will generate 90 degree digital delay bits for all the DQS delay blocks on that half of the device based on the reference clock input to the DLL.

- The DDR SDRAM interface supports the SSTL25 I/O standard, therefore the interface pins should be assigned as SSTL25 I/O type.
- When implementing a DDR interface, the VREF1 of the bank is used to provide the reference voltage for the interface pins.
- Appendix F shows DDR400 implementation results of the LatticeEC Advanced Evaluation Board.

QDR II Interface

QDR II SRAM is a new memory technology defined by a number of leading memory vendors for high-performance and high-bandwidth communication applications. QDR is a synchronous pipelined burst SRAM with two separate unidirectional data buses dedicated for read and write operations running at double data rate. Both the QDR II read and write interfaces use HSTL 1.8V I/O standard.

A QDR II memory controller can be easily implemented using the LatticeECP/EC and LatticeXP devices by taking advantage of the DDR I/O registers. For LatticeECP/EC and LatticeXP devices, ODDRXB primitives are used on the QDR outputs and PFU registers are used on the QDR inputs to implement the DDR interface. To see the details of this implementation refer to Lattice reference design RD1019, *QDR Memory Controller* on the Lattice web site at www.latticesemi.com.

FCRAM (Fast Cycle Random Access Memory) Interface

FCRAM is a DDR-type DRAM, which performs data output at both the rising and the falling edges of the clock. FCRAM devices operate at a core voltage of 2.5V with SSTL Class II I/O. It has enhanced both the core and peripheral logic of the SDRAM. In FCRAM the address and command signals are synchronized with the clock input, and the data pins are synchronized with the DQS signal. Data output takes place at both the rising and falling edges of the DQS. DQS is in phase with the clock input of the device. The DDR SDRAM and DDR FCRAM controller will have different pin outs.

LatticeECP/EC and LatticeXP devices can implement an FCRAM interface using the dedicated DQS logic, input DDR registers and output DDR registers as described in the Implementing Memory Interfaces section of this document. Generation of address and control signals for FCRAM are different compared to the DDR SDRAM devices. Please refer to the FCRAM data sheets to see detailed specifications. Toshiba, Inc. and Fujitsu, Inc. offer FCRAM devices in 256Mb densities. They are available in x8 or x16 configurations.

Generic High Speed DDR Implementation

In addition to the DDR memory interface, users can use the I/O logic registers to implement a high speed DDR interface. DDR data write operations can be implemented using the DDR output registers similar to the memory interface implementation using the ODDRXB primitives.

On the input side, the read interface can be implemented using the core logic PFU registers. The PFU register next to the I/O cells can be used to de-mux the DDR data to single data rate data. This method of implementing DDR can run at 300 MHz when accompanied by proper preferences in the software. The HDL and the preferences to implement this DDR interface are listed in Appendix D of this document.

Board Design Guidelines

The most common challenge associated with implementing DDR memory interfaces is the board design and layout. Users must strictly follow the guidelines recommended by memory device vendors.

Some common recommendations include matching trace lengths of interface signals to avoid skew, proper DQ-DQS signal grouping, proper termination of the SSTL2 I/O standard, proper VREF and VTT generation decoupling and proper PCB routing.

Some reference documents that discuss board layout guidelines:

- www.idt.com, IDT, PCB Design for Double Data Rate Memory.
- www.motorola.com, AN2582, Hardware and Layout Design Considerations for DDR Interfaces.
- www.micron.com, TN4607, DDR 333 Design Guide for Two DIMM Systems

References

- www.jedec.org – JEDEC Standard 79, Double Data Rate (DDR) SDRAM Specification
- www.micron.com – DDR SDRAM Data Sheets
- www.infinion.com – DDR SDRAM Data Sheets
- www.samsung.com – DDR SDRAM Data Sheets
- www.latticesemi.com – RD1019 *QDR Memory Controller* Reference Design for LatticeECP/EC devices
- www.toshiba.com – DDR FCRAM Data Sheet
- www.fujitsu.com – DDR FCRAM Data Sheet
- www.latticesemi.com – LatticeEC Advanced Evaluation Board User's Guide
- www.latticesemi.com – DDR SDRAM Controller (Pipelined Version for LatticeECP/EC Devices) User's Guide

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
—	—	Previous Lattice releases.
February 2007	03.2	Updated Generic High Speed DDR Implementation section.
		Updated Appendix D.

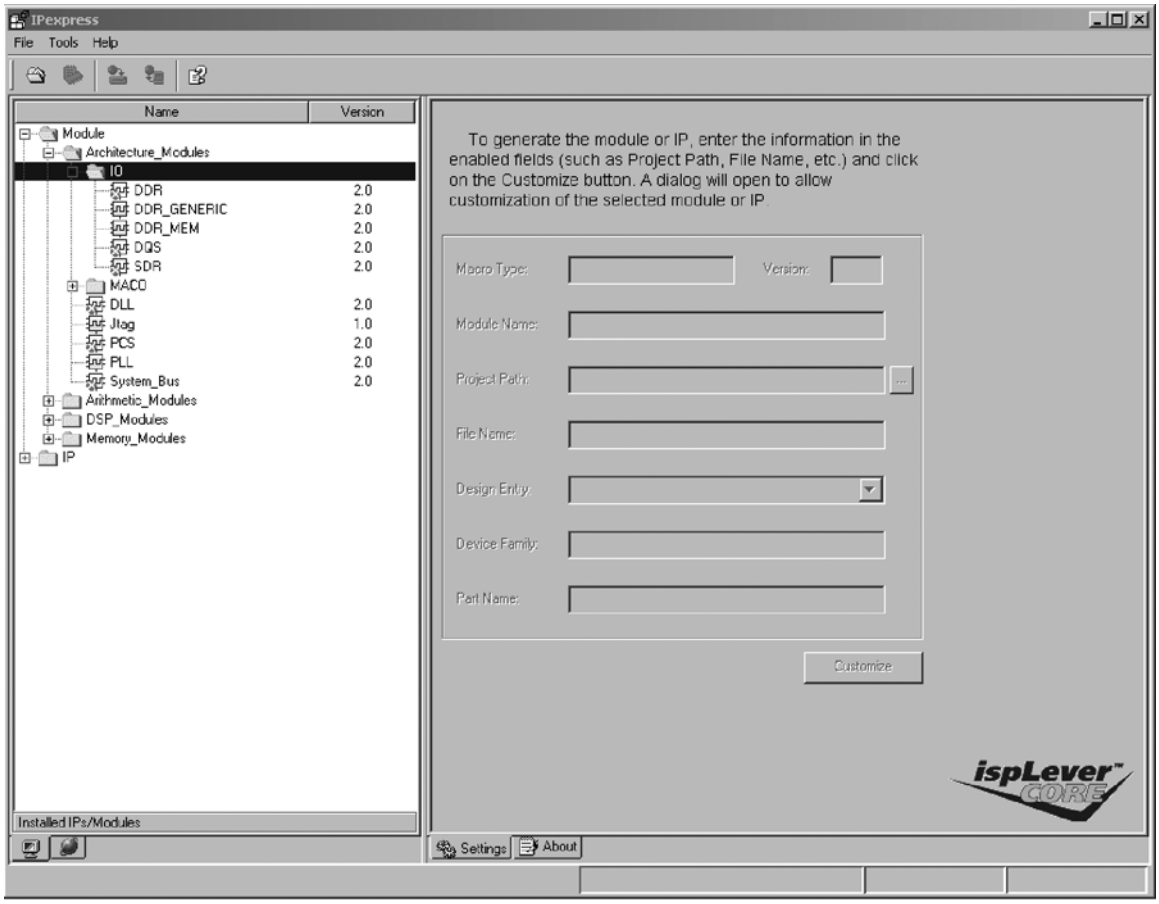
Appendix A. Using IPexpress™ to Generate DDR Modules

The input and output DDR module can be generated using IPexpress. The I/O section under the Architecture modules provides two options to the user:

1. DDR_GENERIC – The option allows generation of a Generic DDR interface, which in the case of LatticeECP/EC and LatticeXP devices, is only the output side DDR. The input side for a Generic DDR interface must be implemented using PFU registers. Appendix D provides the example code for the input side generic DDR.
2. DDR_MEM – This option allows the user to generate a complete DDR memory interface. It will generate both the read and write side interface required to interface with the memory.

IPexpress generates only the modules that are implemented within the IOLOGIC. Any logic required in the FPGA core to complete the memory interface must be implemented by the user.

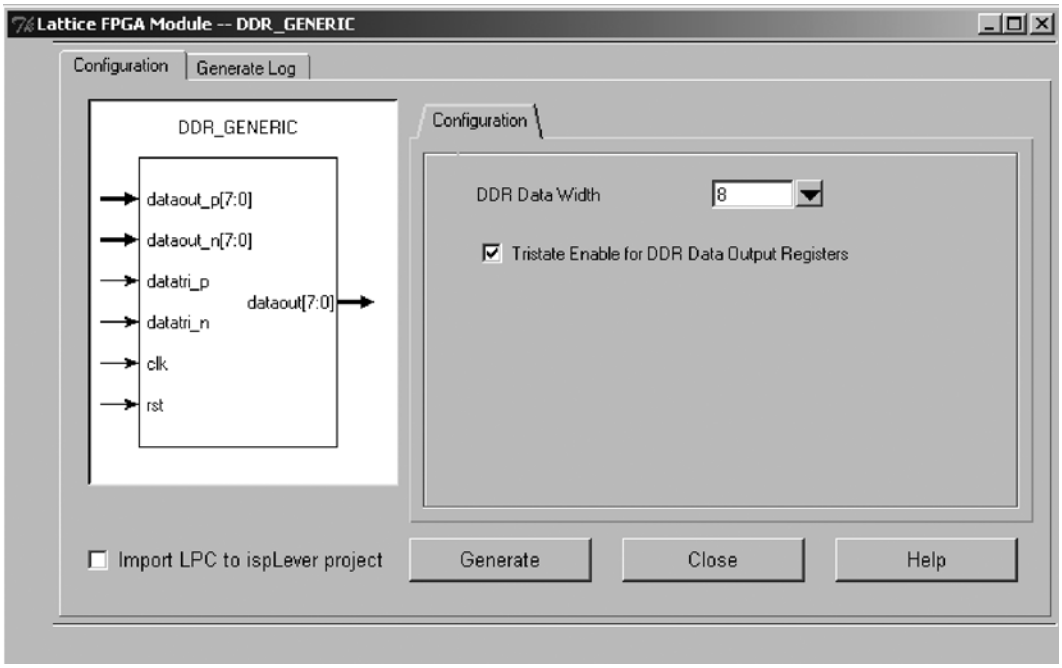
Figure 10-18. IPexpress I/O Section



DDR Generic

DDR Generic will generate the output DDR (ODDRXB) primitives for a given bus width. The user has the option to enable or disable tristate control to the output DDR registers. Figure 10-19 shows the DDR Generic views of IPexpress.

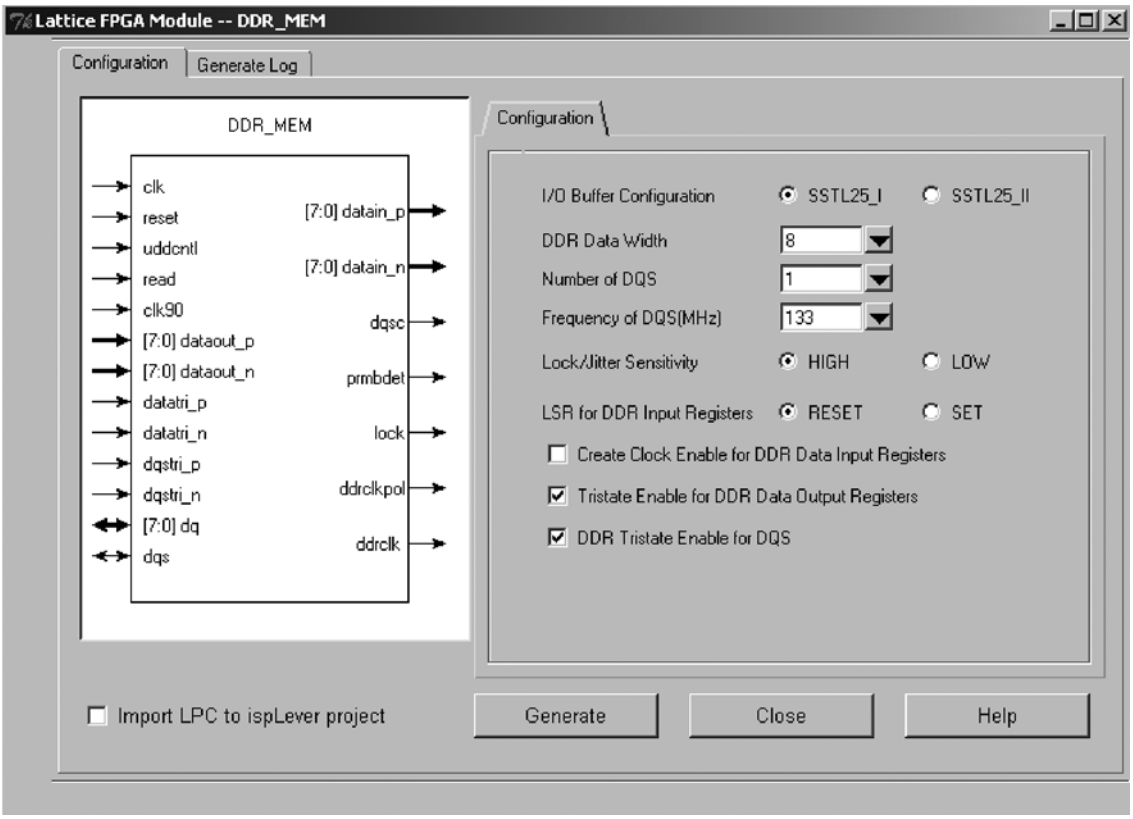
Figure 10-19. DDR Generic Configuration Options



DDR Memory Interface

This IPexpress option generates both read and write interfaces using all DDR primitives for a given bus width. Figure 10-20 shows the options under this section.

Figure 10-20. Configuration Options for DDR Memory Interface



Appendix B. Verilog Example for DDR Input and Output Modules

```
module ddr_mem (dq, dqs, clk, reset, uddcntl, read, datain_p, datain_n, dqsc, prmbdet, lock,
ddrclkpol, clk90, dqstri_p, dqstri_n, datatri_p, datatri_n, dataout_p, dataout_n, ddrclk);

    inout [7:0] dq/* synthesis IO_TYPE="SSTL25_II"*/;
    inout dqs/* synthesis IO_TYPE="SSTL25_II"*/;

    --clk is the core clock and clk90 is the 90 degree phase shifted clock coming from the PLL
    input clk, clk90;
    input reset, uddcntl, read;
    input [7:0] dataout_p, dataout_n;
    input [7:0] datatri_p, datatri_n;
    input dqstri_p, dqstri_n;

    output [7:0] datain_p;
    output[7:0] datain_n;
    output dqsc, prmbdet, lock, ddrclkpol;

    output ddrclk /* synthesis IO_TYPE="SSTL25D_II"*/ ;
    wire vcc_net,gnd_net;
    wire dqsbuf, dqsdel, clk, ddrclkpol_sig;
    wire [7:0] ddrin, ddrout, tridata;
    wire dqsout, tridqs, dqsin, ddrclk;


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


    //-------Bidirectional Buffers -----------------------------------------------------
    BB bidiInst0 (.I(ddrout[0]), .T(tridata[0]), .O(ddrin[0]), .B(dq[0]));
    BB bidiInst1 (.I(ddrout[1]), .T(tridata[1]), .O(ddrin[1]), .B(dq[1]));
    BB bidiInst2 (.I(ddrout[2]), .T(tridata[2]), .O(ddrin[2]), .B(dq[2]));
    BB bidiInst3 (.I(ddrout[3]), .T(tridata[3]), .O(ddrin[3]), .B(dq[3]));
    BB bidiInst4 (.I(ddrout[4]), .T(tridata[4]), .O(ddrin[4]), .B(dq[4]));
    BB bidiInst5 (.I(ddrout[5]), .T(tridata[5]), .O(ddrin[5]), .B(dq[5]));
    BB bidiInst6 (.I(ddrout[6]), .T(tridata[6]), .O(ddrin[6]), .B(dq[6]));
    BB bidiInst7 (.I(ddrout[7]), .T(tridata[7]), .O(ddrin[7]), .B(dq[7]));

    //Bidirectional Strobe, DQS
    BB bidiInst8(.I(dqsout), .T(tridqs), .O(dqsin), .B(dqs));
    //-------------------------------------------------------------------------------------

    //-----------DDR Input ----------------------------------------------------------


    DQSBUFB U8 (.DQSI(dqsin), .CLK(clk), .READ(read), .DQSDEL(dqsdel), .DDRCLKPOL(ddrclkpol_sig),
                    .DQSC(dqsc), .PRMBDET(prmbdet), .DQSO(dqsbuf));

    DQSDLL U9 (.CLK(clk), .UDDCNTL(uddcntl), .RST(reset), .DQSDEL(dqsdel), .LOCK(lock));

    IDDRXB UL0 (.D(ddrin[0]), .ECLK(dqsbuf), .SCLK(clk), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
                   .LSR(reset), .QA(datain_p[0]), .QB(datain_n[0]));

    IDDRXB UL1 (.D(ddrin[1]), .ECLK(dqsbuf), .SCLK(clk), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
                    .LSR(reset), .QA(datain_p[1]), .QB(datain_n[1]));
```

```
    IDDRXB UL2 (.D(ddrin[2]), .ECLK(dqsbuf), .SCLK(clk), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
                    .LSR(reset), .QA(datain_p[2]), .QB(datain_n[2]));

    IDDRXB UL3 (.D(ddrin[3]), .ECLK(dqsbuf), .SCLK(clk), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
                    .LSR(reset), .QA(datain_p[3]), .QB(datain_n[3]));

    IDDRXB UL4 (.D(ddrin[4]), .ECLK(dqsbuf), .SCLK(clk), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
                    .LSR(reset), .QA(datain_p[4]), .QB(datain_n[4]));

    IDDRXB UL5 (.D(ddrin[5]), .ECLK(dqsbuf), .SCLK(clk), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
                    .LSR(reset), .QA(datain_p[5]), .QB(datain_n[5]));

    IDDRXB UL6 (.D(ddrin[6]), .ECLK(dqsbuf), .SCLK(clk), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
                    .LSR(reset), .QA(datain_p[6]), .QB(datain_n[6]));

    IDDRXB UL7 (.D(ddrin[7]), .ECLK(dqsbuf), .SCLK(clk), .CE(vcc_net), .DDRCLKPOL(ddrclkpol_sig),
                    .LSR(reset), .QA(datain_p[7]), .QB(datain_n[7]));


  //-------------------------------------------------------------------------------------
  //----TRISTATE Instantiations-----------------------------------------------------------
  // DDR Trisate for data, DQ

  ODDRXB T0 (.DA(datatri_p[0]), .DB(datatri_n[0]), .LSR(reset), .CLK(~clk), .Q(tridata[0]));
  ODDRXB T1 (.DA(datatri_p[1]), .DB(datatri_n[1]), .LSR(reset), .CLK(~clk), .Q(tridata[1]));
  ODDRXB T2 (.DA(datatri_p[2]), .DB(datatri_n[2]), .LSR(reset), .CLK(~clk), .Q(tridata[2]));
  ODDRXB T3 (.DA(datatri_p[3]), .DB(datatri_n[3]), .LSR(reset), .CLK(~clk), .Q(tridata[3]));
  ODDRXB T4 (.DA(datatri_p[4]), .DB(datatri_n[4]), .LSR(reset), .CLK(~clk), .Q(tridata[4]));
  ODDRXB T5 (.DA(datatri_p[5]), .DB(datatri_n[5]), .LSR(reset), .CLK(~clk), .Q(tridata[5]));
  ODDRXB T6 (.DA(datatri_p[6]), .DB(datatri_n[6]), .LSR(reset), .CLK(~clk), .Q(tridata[6]));
  ODDRXB T7 (.DA(datatri_p[7]), .DB(datatri_n[7]), .LSR(reset), .CLK(~clk), .Q(tridata[7]));

  // DDR Trisate for strobe, DQS
  ODDRXB T8 (.DA(dqstri_p), .DB(dqstri_n), .LSR(reset), .CLK(clk90), .Q(tridqs));

  //-------------------------------------------------------------------------------------

  //-----------DQ output-----------------------------------------------------------------
  ODDRXB O0 (.DA(dataout_p[0]), .DB(dataout_n[0]), .LSR(reset), .CLK(~clk), .Q(ddrout[0]));
  ODDRXB O1 (.DA(dataout_p[1]), .DB(dataout_n[1]), .LSR(reset), .CLK(~clk), .Q(ddrout[1]));
  ODDRXB O2 (.DA(dataout_p[2]), .DB(dataout_n[2]), .LSR(reset), .CLK(~clk), .Q(ddrout[2]));
  ODDRXB O3 (.DA(dataout_p[3]), .DB(dataout_n[3]), .LSR(reset), .CLK(~clk), .Q(ddrout[3]));
  ODDRXB O4 (.DA(dataout_p[4]), .DB(dataout_n[4]), .LSR(reset), .CLK(~clk), .Q(ddrout[4]));
  ODDRXB O5 (.DA(dataout_p[5]), .DB(dataout_n[5]), .LSR(reset), .CLK(~clk), .Q(ddrout[5]));
  ODDRXB O6 (.DA(dataout_p[6]), .DB(dataout_n[6]), .LSR(reset), .CLK(~clk), .Q(ddrout[6]));
  ODDRXB O7 (.DA(dataout_p[7]), .DB(dataout_n[7]), .LSR(reset), .CLK(~clk), .Q(ddrout[7]));
  //-------------------------------------------------------------------------------------
  //--------------- DQS output-----------------------------------------------------------
  ODDRXB O8 (.DA(gnd_net), .DB(vcc_net), .LSR(reset), .CLK(clk90), .Q(dqsout));
  //-------------------------------------------------------------------------------------

  //--------------- CLKOUTP and CLKOUTN Generation--------------------------------------
  ODDRXB O9 (.DA(gnd_net), .DB(vcc_net), .LSR(reset), .CLK(clk90), .Q(ddrclk));
  //-------------------------------------------------------------------------------------
endmodule
```

Appendix C. VHDL Example for DDR Input and Output Modules

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

entity ddr_mem is
        port( dq       : inout std_logic_vector(7 downto 0 );
              dqs      : inout std_logic;
              clk      : in std_logic; -- core clock
              clk90       : in std_logic; -- 90 degree phase shifted clock from the pll
              reset       : in std_logic;
              uddcntl     : in std_logic;
              read        : in std_logic;
              dataout_p   : in std_logic_vector(7 downto 0);
              dataout_n   : in std_logic_vector(7 downto 0);
              datatri_p   : in std_logic_vector(7 downto 0);
              datatri_n   : in std_logic_vector(7 downto 0);
              dqstri_p    : in std_logic;
              dqstri_n    : in std_logic;
              ddrclk      : out std_logic;
              datain_p    : out std_logic_vector(7 downto 0);
              datain_n    : out std_logic_vector(7 downto 0);
              dqsc        : out std_logic;
              prmbdet     : out std_logic;
              lock        : out std_logic;
              ddrclkpol   : out std_logic);

--*****DDR interface signals assigned SSTL25 IO Standard *************
ATTRIBUTE IO_TYPE               :  string;
ATTRIBUTE IO_TYPE OF ddrclk     :  SIGNAL IS "SSTL25D_II";
ATTRIBUTE IO_TYPE OF dq         :  SIGNAL IS "SSTL25_II";
ATTRIBUTE IO_TYPE OF dqs        :  SIGNAL IS "SSTL25_II";

end ddr_mem;

architecture structure of ddr_mem is

--*****DDR Input register*********************************************
 component IDDRXB
    port(
       D    : in STD_LOGIC;
       ECLK : in STD_LOGIC;
       SCLK : in STD_LOGIC;
       CE   : in STD_LOGIC;
       LSR  : in STD_LOGIC;
       DDRCLKPOL  : in STD_LOGIC;
       QA   : out STD_LOGIC;
       QB   : out STD_LOGIC);
 end component;
```

```
--*******DDR Output register ****************************************
 component ODDRXB
    port(
       CLK  : in STD_LOGIC;
       DA   : in STD_LOGIC;
       DB   : in STD_LOGIC;
       LSR  : in STD_LOGIC;
       Q    : out STD_LOGIC);
 end component;

--*******Bidirectional Buffer*********************************************
 component BB
    port(
       I  : in STD_LOGIC;
       T  : in STD_LOGIC;
       O  : out STD_LOGIC;
       B  : inout STD_LOGIC);
 end component;

--******DQS DLL Component**************************************************
 component DQSDLL
    port(
       CLK         : in STD_LOGIC;
       RST         : in STD_LOGIC;
       UDDCNTL     : in STD_LOGIC;
       LOCK        : out STD_LOGIC;
       DQSDEL      : out STD_LOGIC);
 end component;

--****** DQS Delay block***************************************************
 component DQSBUFB
     port(
       DQSI        : in STD_LOGIC;
       CLK         : in STD_LOGIC;
       READ        : in STD_LOGIC;
       DQSDEL      : in STD_LOGIC;
       DQSO        : out STD_LOGIC;
       DDRCLKPOL   : out STD_LOGIC;
       DQSC        : out STD_LOGIC;
       PRMBDET     : out STD_LOGIC);
 end component;


 signal dqsbuf : std_logic;
 signal dqsdel : std_logic;
 signal ddrclkpol_sig : std_logic;
 signal ddrin : std_logic_vector(7 downto 0 );
 signal ddrout : std_logic_vector(7 downto 0 );
 signal tridata : std_logic_vector(7 downto 0 );
 signal dqsout : std_logic;
 signal tridqs : std_logic;
 signal dqsin : std_logic;
 signal vcc_net : std_logic;
 signal gnd_net : std_logic;
```

```
 signal clkinv : std_logic;
 signal ddrclk : std_logic;


 begin
   vcc_net <= '1';
   gnd_net <= '0';
   clkinv<= not clk;
   ddrclkpol<=ddrclkpol_sig;

--************BIDIRECTIONAL BUFFERS*****************************************************
    bidiInst0 : BB PORT MAP( I => ddrout(0),T => tridata(0),O => ddrin(0),B => dq(0));
    bidiInst1 : BB PORT MAP( I => ddrout(1),T => tridata(1),O => ddrin(1),B => dq(1));
    bidiInst2 : BB PORT MAP( I => ddrout(2),T => tridata(2),O => ddrin(2),B => dq(2));
    bidiInst3 : BB PORT MAP( I => ddrout(3),T => tridata(3),O => ddrin(3),B => dq(3));
    bidiInst4 : BB PORT MAP( I => ddrout(4),T => tridata(4),O => ddrin(4),B => dq(4));
    bidiInst5 : BB PORT MAP( I => ddrout(5),T => tridata(5),O => ddrin(5),B => dq(5));
    bidiInst6 : BB PORT MAP( I => ddrout(6),T => tridata(6),O => ddrin(6),B => dq(6));
    bidiInst7 : BB PORT MAP( I => ddrout(7),T => tridata(7),O => ddrin(7),B => dq(7));

    bidiInst8 : BB PORT MAP( I=> dqsout, T=> tridqs, O=> dqsin, B=> dqs);

--*******************************************************************************

--************DDRInput*****************************************************************
--DQSDLL, generates the DQS delay
    I0: DQSDLL PORT MAP(CLK=>clk, UDDCNTL=> uddcntl, RST=> reset, DQSDEL=> dqsdel,
                        LOCK => lock);

    I1: DQSBUFB PORT MAP( DQSI=> dqsin, CLK=>clk, READ=> read, DQSDEL=> dqsdel,
                        DDRCLKPOL=> ddrclkpol_sig, DQSC=> dqsc, PRMBDET=> prmbdet,
                        DQSO=> dqsbuf);

--DDR INPUT primitives
    I2 : IDDRXB PORT MAP(D=> ddrin(0), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(0),
                        QB => datain_n(0));
    I3 : IDDRXB PORT MAP(D=> ddrin(1), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(1),
                        QB => datain_n(1));
    I4 : IDDRXB PORT MAP(D=> ddrin(2), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(2),
                        QB => datain_n(2));
    I5 : IDDRXB PORT MAP(D=> ddrin(3), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(3),
                        QB => datain_n(3));
    I6 : IDDRXB PORT MAP(D=> ddrin(4), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(4),
                        QB => datain_n(4));
    I7 : IDDRXB PORT MAP(D=> ddrin(5), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                        DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(5),
                        QB => datain_n(5));
```

```
    I8 : IDDRXB PORT MAP(D=> ddrin(6), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                         DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(6),
                         QB => datain_n(6));
    I9 : IDDRXB PORT MAP(D=> ddrin(7), ECLK=> dqsbuf, SCLK => clk, CE => vcc_net,
                         DDRCLKPOL=> ddrclkpol_sig, LSR => reset, QA =>datain_p(7),
                         QB => datain_n(7));
--*************************************************************************************

--*****TRISTATE Instantiations*****************************************************
-- DDR Trisate for data, DQ
    T0 : ODDRXB PORT MAP( DA => datatri_p(0), DB => datatri_n(0), LSR => reset,
                         CLK => clkinv, Q => tridata(0));
    T1 : ODDRXB PORT MAP( DA => datatri_p(1), DB => datatri_n(1), LSR => reset,
                         CLK => clkinv, Q => tridata(1));
    T2 : ODDRXB PORT MAP( DA=>  datatri_p(2), DB => datatri_n(2), LSR => reset,
                         CLK => clkinv, Q => tridata(2));
    T3 : ODDRXB PORT MAP( DA => datatri_p(3), DB => datatri_n(3), LSR => reset,
                         CLK => clkinv, Q => tridata(3));
    T4 : ODDRXB PORT MAP( DA => datatri_p(4), DB => datatri_n(4), LSR => reset,
                         CLK => clkinv, Q => tridata(4));
    T5 : ODDRXB PORT MAP( DA => datatri_p(5), DB => datatri_n(5), LSR => reset,
                         CLK => clkinv, Q => tridata(5));
    T6 : ODDRXB PORT MAP( DA => datatri_p(6), DB => datatri_n(6), LSR => reset,
                         CLK => clkinv, Q => tridata(6));
    T7 : ODDRXB PORT MAP( DA => datatri_p(7), DB => datatri_n(7), LSR => reset,
                         CLK => clkinv, Q => tridata(7));

--DDR Trisate for strobe, DQS
    T8: ODDRXB PORT MAP( DA =>dqstri_p, DB=> dqstri_n, LSR=> reset, CLK=> clk90,
                         Q => tridqs);
--*************************************************************************************

--***************DDR Output*********************************************************
--DQ OUTPUT
    O0 : ODDRXB PORT MAP( DA => dataout_p(0), DB  => dataout_n(0), LSR => reset,
                         CLK  => clkinv, Q => ddrout(0));
    O1 : ODDRXB PORT MAP( DA => dataout_p(1), DB  => dataout_n(1), LSR => reset,
                         CLK  => clkinv, Q => ddrout(1));
    O2 : ODDRXB PORT MAP( DA => dataout_p(2), DB  => dataout_n(2), LSR => reset,
                         CLK  => clkinv, Q => ddrout(2));
    O3 : ODDRXB PORT MAP( DA => dataout_p(3), DB  => dataout_n(3), LSR => reset,
                         CLK  => clkinv, Q => ddrout(3));
    O4 : ODDRXB PORT MAP( DA => dataout_p(4), DB  => dataout_n(4), LSR => reset,
                         CLK  => clkinv, Q => ddrout(4));
    O5 : ODDRXB PORT MAP( DA => dataout_p(5), DB  => dataout_n(5), LSR => reset,
                         CLK  => clkinv, Q => ddrout(5));
    O6 : ODDRXB PORT MAP( DA => dataout_p(6), DB  => dataout_n(6), LSR => reset,
                         CLK  => clkinv, Q => ddrout(6));
    O7 : ODDRXB PORT MAP( DA => dataout_p(7), DB  => dataout_n(7), LSR => reset,
                         CLK  => clkinv, Q => ddrout(7));
```

```
--DQS output
    O8: ODDRXB PORT MAP( DA => gnd_net, DB => vcc_net, LSR => reset, CLK => clk90,
                         Q => dqsout);

--clkp and clkn Generation

    O9 : ODDRXB PORT MAP( DA => vcc_net, DB => gnd_net, LSR => reset, CLK => clk90,
                        Q => ddrclk);
--********************************************************************************

end structure;
```

Appendix D. Generic (Non-Memory) High-Speed DDR Interface

The following HDL implements the DDR input interface using PFU registers for non-memory DDR applications.

VHDL Implementation

```
library IEEE;
use IEEE.std_logic_1164.all;

library ec;
use ec.components.all;

entity ddrin is
 port (rst : in std_logic;
       ddrclk: in std_logic;
       ddrdata: in std_logic_vector(7 downto 0);
       datap: out std_logic_vector(7 downto 0);
       datan: out std_logic_vector(7 downto 0));


end ddrin;
architecture structure of ddrin is
-- parameterized module component declaration
component pll90
    port (CLK: in std_logic; RESET: in std_logic; CLKOP: out std_logic;
        CLKOS: out std_logic; LOCK: out std_logic);
end component;


        signal pos0 : std_logic_vector( 7 downto 0 );
        signal pos1 : std_logic_vector( 7 downto 0 );
        signal neg0 : std_logic_vector( 7 downto 0 );

        signal clklock : std_logic;
        signal ddrclk0: std_logic;
        signal ddrclk90: std_logic;
        signal vcc_net : std_logic;
        signal gnd_net: std_logic;

attribute syn_useioff : boolean;
attribute syn_useioff of structure : architecture is false;

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


       -- parameterized module component instance
        I0 : pll90
        port map (CLK=>ddrclk, RESET=>rst, CLKOP=>ddrclk0, CLKOS=>ddrclk90, LOCK=>clklock);

demux: process (rst, ddrclk90)
begin
        if  rst = '1' then
              pos0    <= (others => '0');
              neg0    <= (others => '0');
              pos1    <= (others => '0');
        elsif rising_edge(ddrclk90) then
              pos0    <= ddrdata;
        elsif falling_edge(ddrclk90) then
```

```
          neg0  <=ddrdata;
          pos1  <=pos0;
        end if;
    end  process demux;

    synch: process (rst, ddrclk90)
    begin
        if  rst = '1' then
           datap  <= (others => '0');
           datan  <= (others => '0');
        elsif rising_edge(ddrclk90) then
           datap<= pos1;
        elsif falling_edge(ddrclk90) then
               datan<= neg0;
        end if;
    end  process synch;

end structure;
```

Verilog Example

```
module ddrin (rst, ddrclk, ddrdata, datap, datan)/*synthesis syn_useioff = 0*/;
// Inputs
input             rst;
input             ddrclk;
input     [7:0]   ddrdata;
// Outputs
output    [7:0]   datap, datan;


reg [7:0] pos0/*synthesis syn_keep=1*/;
reg [7:0] pos1/*synthesis syn_keep=1*/;
reg [7:0] neg0/*synthesis syn_keep=1*/;
reg [7:0] datap, datan/*synthesis syn_keep=1*/;

//PLL signals
wire ddrclk0;
wire ddrclk90;

pll I0 (.CLK(ddrclk), .RESET(rst), .CLKOP(ddrclk0), .CLKOS(ddrclk90), .LOCK(clklock));

always @ ( posedge ddrclk90)
begin
    if (rst)
    begin
    pos0 <= 0;
    end
    else
    begin
        pos0 <= ddrdata;
    end
end

always@ (negedge ddrclk90)
begin
    if (rst)
    begin
        neg0<=0;
        pos1<=0;
    end
    else
    begin
        neg0<=ddrdata;
        pos1<=pos0;
    end
end
always @ (posedge ddrclk90)
begin
    if (rst)
    begin
        datap<= 0;
        datan<= 0;
    end
    else
    begin
        datap<= pos1;
        datan<= neg0;
    end
```

```
end
endmodule
```

Preference File

In order to run the above DDR PFU Implementation at 300MHZ, the following preferences were added to the software preference file.

```
COMMERCIAL;
FREQUENCY NET "ddrclk90" 300.000000 MHz ;
INPUT_SETUP PORT "ddrdata_0" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_1" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_2" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_3" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_4" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_5" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_6" 0.800000 ns CLKNET "ddrclk90" ;
INPUT_SETUP PORT "ddrdata_7" 0.800000 ns CLKNET "ddrclk90" ;
BLOCK ASYNCPATHS ;
```

Appendix E. List of Compatible DDR SDRAM

Below are the criteria used to list the DDR SDRAM part numbers.

1. The memory device should support one DQS strobe for every eight DQ data bits.
2. 4-bit, 8-bit and 16-bit configurations. For 16-bit configurations, each data byte must have an independent DQS strobe.
3. The memory device uses SSTL2 I/O interface standard.
4. Data transfer rate DDR400, DDR333 or DDR266 for LatticeECP/EC devices and DDR333 or DDR266 for LatticeXP devices.
5. Clock transfer rate of 200MHz, 167MHz or 133MHz for LatticeECP/EC devices and 167MHz or 133MHz for LatticeXP devices.

Table 10-9 lists the DDR SDRAM part numbers that can be used with the LatticeECP/EC and LatticeXP devices.

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

Table 10-9. List of Compatible DDR SDRAM

DDR SDRAM Vendor	Part Number	Configuration	Max Data Rate	Clock Speed
Micron 128MB	MT46V32M4TG	32Mx4	DDR266	133MHz
Micron 128MB	MT46V16M8TG	16Mx8	DDR333 DDR266	167MHz 133MHz
Micron 128MB	MT46V8M16TG	8Mx16	DDR266	133MHz
Micron 256MB	MT46V64M4FG	64Mx4	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 256MB	MT46V64M4TG	64Mx4	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 256MB	MT46V32M8FG	32Mx8	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 256MB	MT46V32M8TG	32Mx8	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 256MB	MT46V16M16FG	16Mx16	DDR266	133MHz
Micron 256MB	MT46V16M16TG	16Mx16	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 512MB	MT46V128M4FN	128Mx4	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 512MB	MT46V128M4TG	128Mx4	DDR266	133MHz
Micron 512MB	MT46V64M8FN	64Mx8	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 512MB	MT46V64M8TG	64Mx8	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 512MB	MT46V32M16FN	32Mx16	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz

Table 10-9. List of Compatible DDR SDRAM (Continued)

DDR SDRAM Vendor	Part Number	Configuration	Max Data Rate	Clock Speed
Micron 512MB	MT46V32M16TG	32Mx16	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
Micron 1GB	MT46V256M4TG	256Mx4	DDR266	133MHz
Micron 1GB	MT46V128M8TG	128Mx8	DDR266	133MHz
Micron 1GB	MT46V64M16TG	64Mx16	DDR333 DDR266	167MHz 133MHz
Samsung 128MB E die	K4H280438E-TC/LB3	32Mx4	DDR333	167MHz
	K4H280838E-TC/LB3	16Mx8	DDR333	167MHz
	K4H281638E-TC/LB3	8Mx16	DDR333	167MHz
Samsung 256 Mb E-die	K4H560438E-TC/LB3	64Mx4	DDR333	167MHz
	K4H560438E-NC/LB3	64Mx4	DDR333	167MHz
	K4H560438E-GC/LB3, CC	64Mx4	DDR333 DDR400	167MHz 200MHz
	K4H560838E-TC/LB3, CC	32Mx8	DDR333 DDR400	167MHz 200MHz
	K4H560838E-NC/LB3, CC	32Mx8	DDR333 DDR400	167MHz 200MHz
	K4H560838E-GC/LB3, CC	32Mx8	DDR333 DDR400	167MHz 200MHz
Samsung 512Mb B die	K4H510838B-TC/LB3, CC	64Mx8	DDR333 DDR400	167MHz 200MHz
	K4H510838B-NC/LB3, CC	64Mx8	DDR333 DDR400	167MHz 200MHz
	K4H511638B-TC/LB3, CC	32Mx16	DDR333 DDR400	167MHz 200MHz
	K4H510438B-TC/LB3	128Mx4	DDR333	167MHz
	K4H510438B-NC/LB3	128Mx4	DDR333	167MHz
Infineon 128Mb	HYB25D128400AT	32Mx4	DDR266	133MHz
	HYB25D128400CT	32Mx4	DDR266	133MHz
	HYB25D128400CE	32Mx4	DDR266	133MHz
	HYB25D128800AT	16Mx8	DDR266	133MHz
	HYB25D128800CT	16Mx8	DDR333	167MHz
	HYB25D128800CE	16Mx8	DDR333	167MHz
	HYB25D128160AT	8Mx16	DDR333 DDR266	167MHz 133MHz
	HYB25D128160CT	8Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D128160CE	8Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D128160CC	8Mx16	DDR333	167MHz

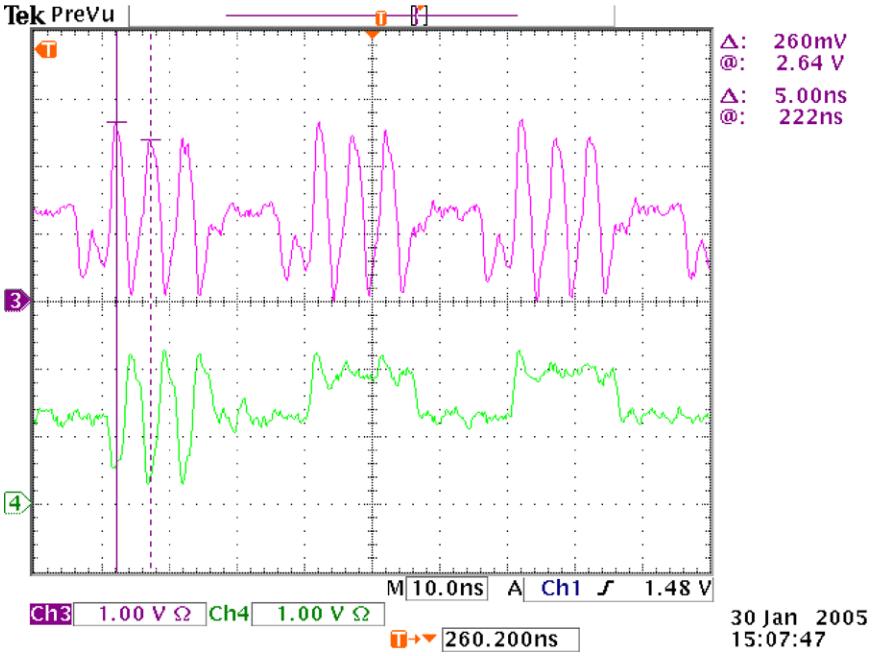
Table 10-9. List of Compatible DDR SDRAM (Continued)

DDR SDRAM Vendor	Part Number	Configuration	Max Data Rate	Clock Speed
Infineon 256Mb	HYB25D256400BT	64Mx4	DDR266	133MHz
	HYB25D256400CT	64Mx4	DDR266	133MHz
	HYB25D256400CE	64Mx4	DDR266	133MHz
	HYB25D256800BT	32Mx8	DDR333	167MHz
	HYB25D256800CT	32Mx8	DDR333	167MHz
	HYB25D256800CE	32Mx8	DDR333 DDR400	167MHz 200MHz
	HYB25D256160BT	16Mx16	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
	HYB25D256160CT	16Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D256160CE	16Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D256400BC	64Mx16	DDR400 DDR333 DDR266	200MHz 167MHz 133MHz
	HYB25D256400CF	64Mx16	DDR333	167MHz
	HYB25D256400CC	64Mx16	DDR333	167MHz
	HYB25D256160BC	16Mx16	DDR333 DDR266	167MHz 133MHz
	HYB25D256160CC	16Mx16	DDR333 DDR400	167MHz 200MHz
Infineon 512Mb	HYB25D512400AT	128Mx4	DDR266	133MHz
	HYB25D512400BT	128Mx4	DDR333	167MHz
	HYB25D512400BE	128Mx4	DDR333	167MHz
	HYB25D1G400BG	256Mx4	DDR266	133MHz
	HYB25D512800AT	64Mx8	DDR333 DDR266	167MHz 133MHz
	HYB25D512800BT	64Mx8	DDR333 DDR400	167MHz 200MHz
	HYB25D512800BE	64Mx8	DDR333 DDR400	167MHz 200MHz
	HYB25D512160AT	32Mx16	DDR333 DDR266	167MHz 133MHz
	HYB25D512160BT	32Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D512160BE	32Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D512400BC	128Mx4	DDR333 DDR400	167MHz 200MHz
	HYB25D512400BF	128Mx4	DDR333 DDR400	167MHz 200MHz
	HYB25D512800BC	64Mx8	DDR333	167MHz
	HYB25D512800BF	64Mx8	DDR333 DDR400	167MHz 200MHz
	HYB25D512160BC	32Mx16	DDR333 DDR400	167MHz 200MHz
	HYB25D512160BF	32Mx16	DDR333 DDR400	167MHz 200MHz

Appendix F. DDR400 Interface using the LatticeEC Evaluation Board

The DDR400 interface was implemented using the LatticeEC20 device on the LatticeEC Advanced Evaluation Board. Figures 10-21, 10-22 and 10-23 show the READ, WRITE and WRITE to READ transition operations running at 200MHz. For more information on the evaluation board, refer to *LatticeEC Advanced Evaluation Board User's Guide* available on the Lattice web site at www.latticesemi.com.

Figure 10-21. READ Function Running at 200MHz



Note: An extra READ command is implemented in the LatticeEC20 device to protect the data during postamble. This extra READ is not required for other LatticeEC devices. Refer to the DQS Postamble section of this document for more information.

Figure 10-22. WRITE Function Running at 200MHz

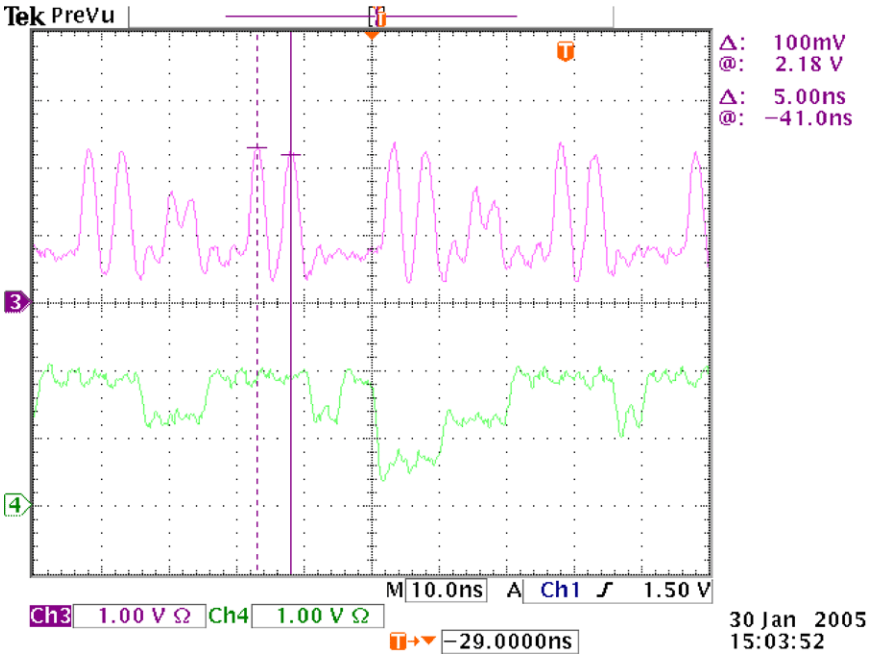
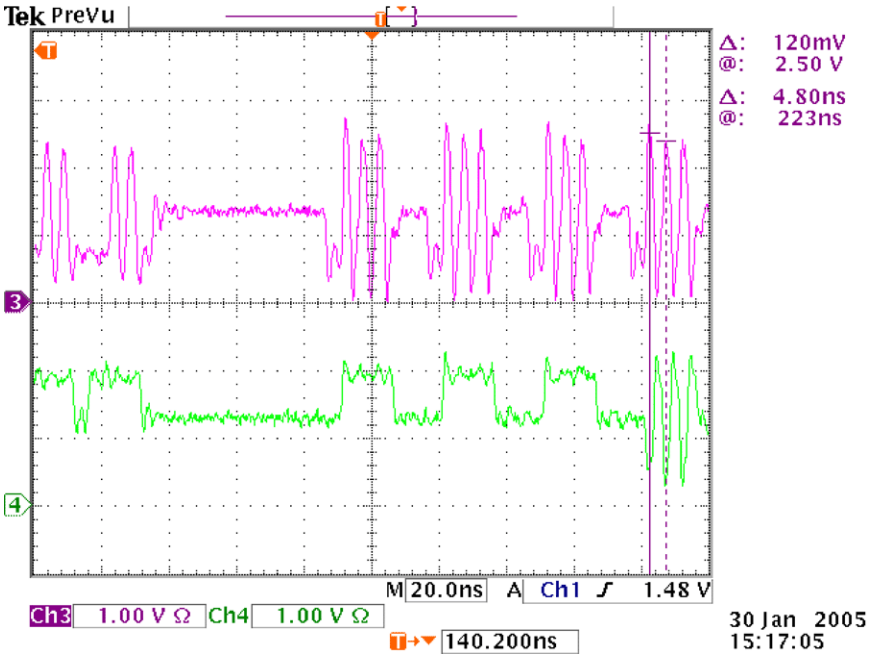


Figure 10-23. WRITE to READ Transition Running at 200MHz



Note: An extra READ command is implemented in the LatticeEC20 device to protect the data during postamble. This extra READ is not required for other LatticeEC devices. Refer to the DQS Postamble section of this document for more information.

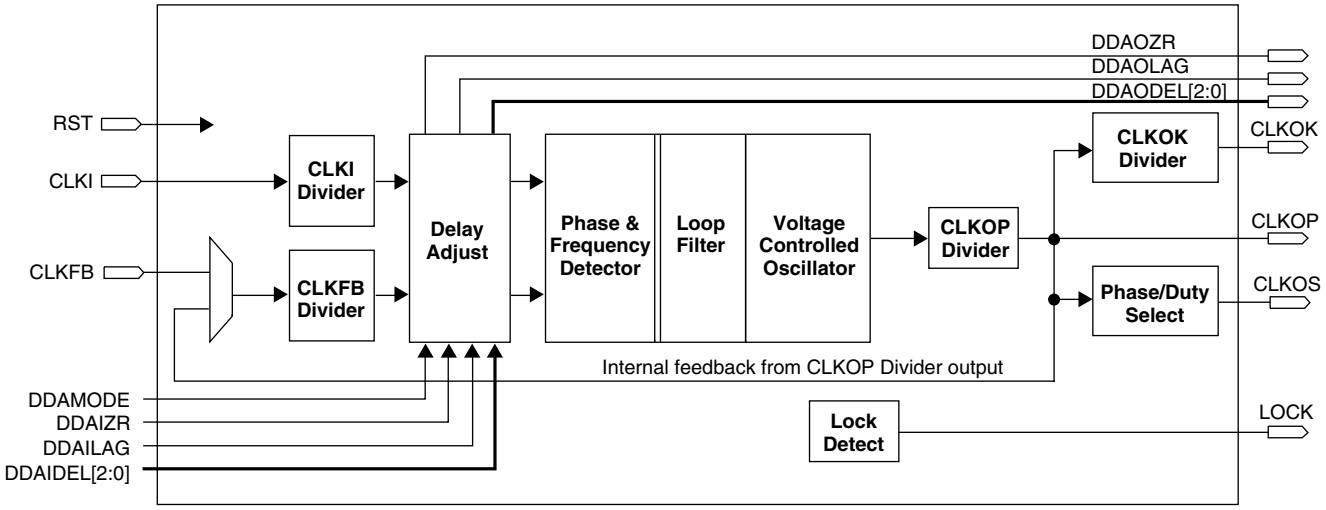
LatticeECP/EC and LatticeXP sysCLOCK PLL Design and Usage Guide

September 2006 **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™, LatticeEC™ and LatticeXP™ device families to help designers manage clocks within their designs. The PLL components in the LatticeECP/EC and LatticeXP device families share the same architecture. This technical note describes the features and functionalities of the PLLs and their configuration in the ispLEVER® design tool. Figure 11-1 shows the block diagram of the PLL.

Figure 11-1. LatticeECP/EC and LatticeXP sysCLOCK PLL Block Diagram



Features

- Clock synthesis
- Phase shift/duty cycle selection
- Internal and external feedback
- Dynamic delay adjustment
- No external components required
- Lock detect output

Functional Description

PLL Divider and Delay Blocks

Input Clock (CLKI) Divider

The CLKI divider is used to control the input clock frequency into the PLL block. It can be set to an integer value of 1 to 16. The divider setting directly corresponds to the divisor of the output clock. The input and output of the input divider must be within the input and output frequency ranges specified in the device data sheet.

Feedback Loop (CLKFB) Divider

The CLKFB divider is used to divide the feedback signal. Effectively, this multiplies the output clock, because the divided feedback must speed up to match the input frequency into the PLL block. The PLL block increases the output frequency until the divided feedback frequency equals the input frequency. Like the input divider, the feedback

loop divider can be set to an integer value of 1 to 16. The input and output of the feedback divider must be within the input and output frequency ranges specified in the device data sheet.

Delay Adjustment
The delay adjust circuit provides programmable clock delay. The programmable clock delay allows for step delays in increments of 250ps (nominal) for a total of 2.00ns lagging or leading. The time delay setting has a tolerance. See device data sheet for details. Under this mode, CLKOP, CLKOS and CLKOK are identically affected. The delay adjustment has two modes of operation:

- **Static Delay Adjustment** – In this mode, the user-selected delay is configured at power-up.
- **Dynamic Delay Adjustment (DDA)** – In this mode, a simple bus is used to configure the delay. The bus signals are available to the general purpose FPGA.

Output Clock (CLKOP) Divider
The CLKOP divider serves the dual purposes of squaring the duty cycle of the VCO output and scaling up the VCO frequency into the 420MHz to 840MHz range to minimize jitter. Refer to Table 11-3 for CLKOP Divider value.

CLKOK Divider
The CLKOK divider feeds the global clock net. It divides the CLKOP signal of the PLL by the value of the divider. It can be set to values of 2, 4, 6,....126,128.

PLL Inputs and Outputs

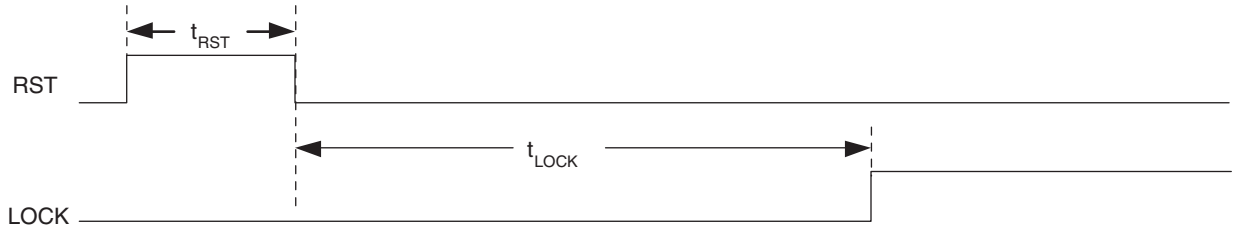
CLKI Input
The CLKI signal is the reference clock for the PLL. It must conform to the specifications in the data sheet in order for the PLL to operate correctly. The CLKI can be derived from a dedicated dual-purpose pin or from routing.

RST Input
The PLL reset occurs under two conditions. At power-up an internal power-up reset signal from the configuration block resets the PLL. The user controlled PLL reset signal RST is provided as part of the PLL module that can be driven by an internally generated reset function or a pin. This RST signal resets all internal PLL counters. When RST goes inactive, the PLL will start the lock-in process, and will take the t_{LOCK} time to complete the PLL lock.

Note: For LatticeECP/EC, RST must be asserted to re-start the locking process after losing lock. Refer to the LatticeECP/EC Family Data Sheet for the RST pulse width requirement. For LatticeXP, RST may be tied to GND.

Figure 11-2 shows the timing diagram of the RST Input.

Figure 11-2. RST Input Timing Diagram



CLKFBK Input
The feedback signal to the PLL, which is fed through the feedback divider can be derived from the global clock net, a dedicated dual-purpose pin, or directly from the CLKOP divider. Feedback must be supplied in order for the PLL to synchronize the input and output clocks. External feedback allows the designer to compensate for board-level clock alignment.

CLKOP Output
The sysCLOCK PLL main clock output, CLKOP, is a signal available for selection as a primary clock.

CLKOS Output with Phase and Duty Cycle Select
The sysCLOCK PLL auxiliary clock output, CLKOS, is a signal available for selection as a primary clock. The CLKOS is used when phase shift and/or duty cycle adjustment is desired. The programmable phase shift allows for different phase in increments of 45° to 315°. The duty select feature provides duty select in 1/8th of the clock period.

CLKOK Output with Lower Frequency
The CLKOK is used when a lower frequency is desired. It is a signal available for selection as a primary clock.

Dynamic Delay Control I/O Ports
Refer to Table 11-1 and Table 11-6 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. PLL RST must be applied to re-synchronize the PLL to the reference clock. The LOCK signal is available to the FPGA routing to implement generation of RST.

PLL Attributes

The PLL utilizes several attributes that allow the configuration of the PLL through source constraints. The following section details these attributes and their usage.

FIN
The input frequency can be any value within the specified frequency range based on the divider settings.

CLKI_DIV, CLKFB_DIV, CLKOP_DIV, CLKOK_DIV
These dividers determine the output frequencies of each output clock. The user is not allowed to input an invalid combination; determined by the input frequency, the dividers, and the PLL specifications.

Frequency_Pin_CLKI, Frequency_Pin_CLKOP and Frequency_Pin_CLKOK
These output clock frequencies determine the divider values.

FDEL
The FDEL attribute is used to pass the Delay Adjustment step associated with the Output Clock of the PLL. This allows the user to advance or retard the Output Clock by the step value passed multiplied by 250ps(nominal). The step ranges from -8 to +8 resulting the total delay range to +/- 2ns.

PHASEADJ
The PHASEADJ attribute is used to select Coarse Phase Shift for CLKOS output. The phase adjustment is programmable in 45° increments.

DUTY
The DUTY attribute is used to select the Duty Cycle for CLKOS output. The Duty Cycle is programmable at 1/8 of the period increment.

FB_MODE
There are three sources of feedback signals that can drive the CLKFB Divider: internal, clocktree and external feedback. Clocktree feedback is used by default. Internal feedback takes the CLKOP output at CLKOP Divider output before the Clocktree to minimize the feedback path delay. The external feedback is driven from the pin.

DELAY_CNTL
This attribute is designed to select the Delay Adjustment mode. If the attribute is set to “DYNAMIC” the delay control switches between Dynamic and Static depending upon the input logic of DDAMODE pin. If the attribute is set to “STATIC”, Dynamic Delay inputs are ignored in this mode.

LatticeECP/EC and LatticeXP PLL Primitive Definitions

The PLL primitive name is EHXPLLB. Figure 11-3 shows the LatticeECP/EC and LatticeXP PLL primitive library symbol. Some features and I/Os are optional as described in Table 11-1 and Table 11-2.

Figure 11-3. LatticeECP/EC and LatticeXP PLL Primitive Symbol

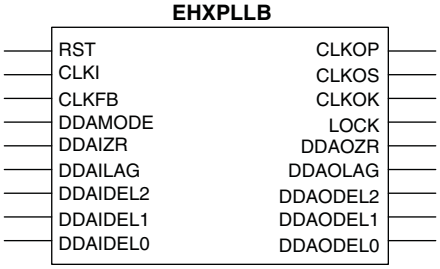


Table 11-1. LatticeECP/EC and LatticeXP PLL I/O Definitions

Signal	I/O	Description	Optional
CLKI	I	PLL reference clock input. From internal logic or dedicated clock pin.	No
CLKFB[1]	I	Feedback clock input. From internal node, CLKOP or dedicated pin.	No
RST	I	"1" to reset PLL	No
CLKOP	O	PLL output clock to clock tree	No
CLKOS	O	PLL output clock to clock tree with optional phase shift/duty cycle	Yes
CLKOK	O	PLL output clock to clock tree through K-divider for lower frequency	Yes
LOCK[2]	O	"1" indicates PLL locked to CLKI	Yes
DDAMODE	I	DDA Mode. "1": Pin Control (dynamic), "0": fuse control (static)	Yes
DDAIZR	I	DDA Delay Zero. "1" delay=0, "0": delay=[DDILAG+DDAIDEL].	Yes
DDAILAG	I	DDA Lag/Lead. "1": Lead, "0": Lag.	Yes
DDAIDEL	I	DDA Delay	Yes
DDAOZR	O	DDA Delay Zero Output	Yes
DDAOLAG	O	DDA Lag/Lead Output	Yes
DDAODEL[2:0]	O	DDA Delay Output	Yes

1. When internal feedback or clocktree feedback is selected in the IPexpress™ GUI, software uses CLKOP as the source of CLKFB. CLKOS is not recommended as the source of CLKFB even in external feedback mode.
2. ModelSim® simulation models take two to four clock cycles from RST release to LOCK high.

PLL Attributes Definitions

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

Table 11-2. LatticeECP/EC and LatticeXP PLL Attributes

User Accessible	IPexpress GUI Access	Attribute Name	Preference Language Support	Preference Editor Support	Value	Default Value	Units
CLKI Frequency	Y	FREQUENCY_PIN_CLKI	N	N	Note 5	100	MHz
CLKOP Frequency	Y	FREQUENCY_PIN_CLKOP	N	N	Note 5	100	MHz
CLKOK Frequency	Y	FREQUENCY_PIN_CLKOK	N	N	Note 5	50	MHz
CLKOP Frequency Tolerance	Y		N	N	0.0, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0	0.0	%
CLKOP Actual Frequency	Y		N	N			MHz
CLKOK Frequency Tolerance	Y		N	N	0.0, 0.1, 0.2, 0.5,1.0, 2.0, 5.0, 10.0	0.0	%

Table 11-2. LatticeECP/EC and LatticeXP PLL Attributes (Continued)

User Accessible	IPexpress GUI Access	Attribute Name	Preference Language Support	Preference Editor Support	Value	Default Value	Units
CLKOK Actual Frequency	Y		N	N			MHz
CLKI Divider Setting	Y	CLKI_DIV[4, 6]	Y	N	1 to 16 (1 to 15)	1	
CLKFB Divider Setting	Y	CLKFB_DIV[6]	Y	N	1 to 16 (1 to 15)	1	
CLKOP Divider Setting	Y	CLKOP_DIV[6]	Y	N	Note 3	8[2] (4 or 6)	
CLKOK Divider Setting	Y	CLKOK_DIV	Y	N	2, 4, 6,..,126, 128	2	
Fine Delay Adjust	N	FDEL	Y	Y	-8 to 8	0	ps
Coarse Phase Shift Selection (O)	Y	PHASEADJ	Y	N	0, 45, 90...315	0	Degrees
Duty Cycle Selection (1/8 increment)	Y	DUTY	Y	N	1 to 7	4	
Delay Control	Y	DELAY_CNTL1	Y	N	DYNAMIC/STATIC	STATIC	
Feedback Mode	Y	FB_MODE	N	N	INTERNAL/CLOCKTREE/EXTERNAL	CLOCKTREE	
CLKOS Select	Y		N	N			
CLKOK Select	Y		N	N			

1. DYNAMIC: This mode switches delay control between Dynamic and Static depending upon the input logic of the DDAMODE pin. STATIC: This is Static Control Only mode.
2. The CLKOP_DIV value is calculated to maximize the f_{VCO} within the specified range. For LatticeXP devices, if CLKOS is not used, the default value is 6. If CLKOS is used, the value is 4.
3. The CLKOP Divider values are 2, 4, 6, 8,..32 (2, 4, 6, 8..16 for LatticeXP devices) if CLKOS is not used. The CLKOP Divider values are 2, 4, 8, 16, 32 (2, 4, 8,16 for LatticeXP devices) if CLKOS is used.
4. All divider settings are user transparent in Frequency Mode. These are user attributes in Divider Mode.
5. Refer to data sheet for frequency limits.
6. Values in parentheses are for LatticeXP devices.
7. This attribute is not available in the IPexpress GUI. After reviewing the trace report file, users can determine the amount of delay that will best fit the clocking in their design. Further information on FDEL settings is described in the following section.

FDEL Settings

There are four ways the user can enter the desired FDEL value.

1. Although the FDEL entry is not available in the IPexpress GUI, the module generated by IPexpress includes the attribute with default value, “0”. Users can replace it with a desired value.

   Example of source code with default FDEL value:

   ```
   attribute FDEL of ehxpll_mod_0_0 : label is "0";
   generic map (…
      FDEL=>"0",
      …
      ")
   ```

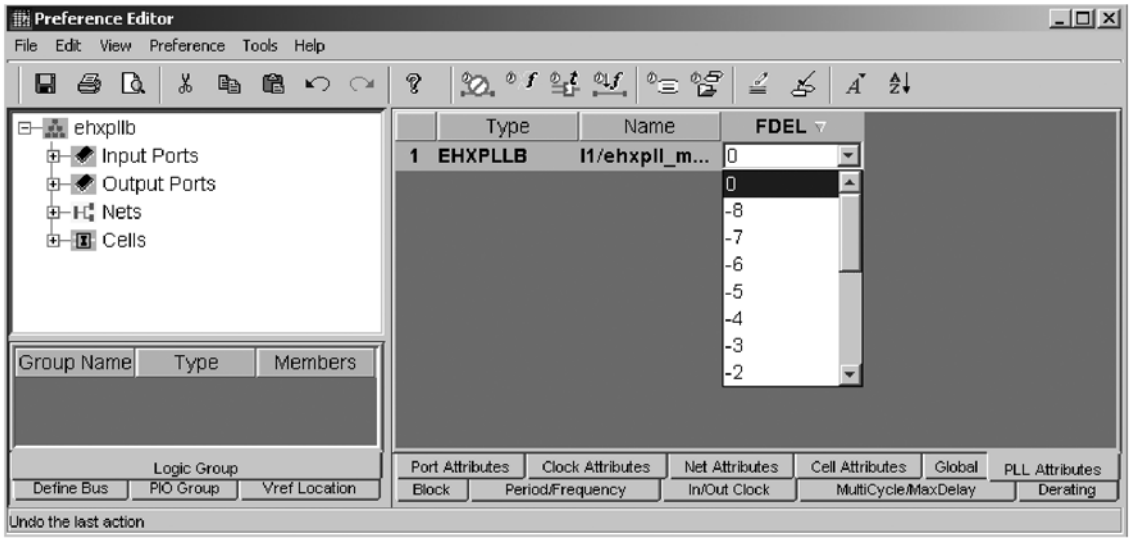
2. Preference File: User may specify the preference in the Preference file.

   Example:

   ```
   ASIC "FDEL_CODE_0_0" TYPE "EHXPLLB" FDEL="-2"  ;
   ```

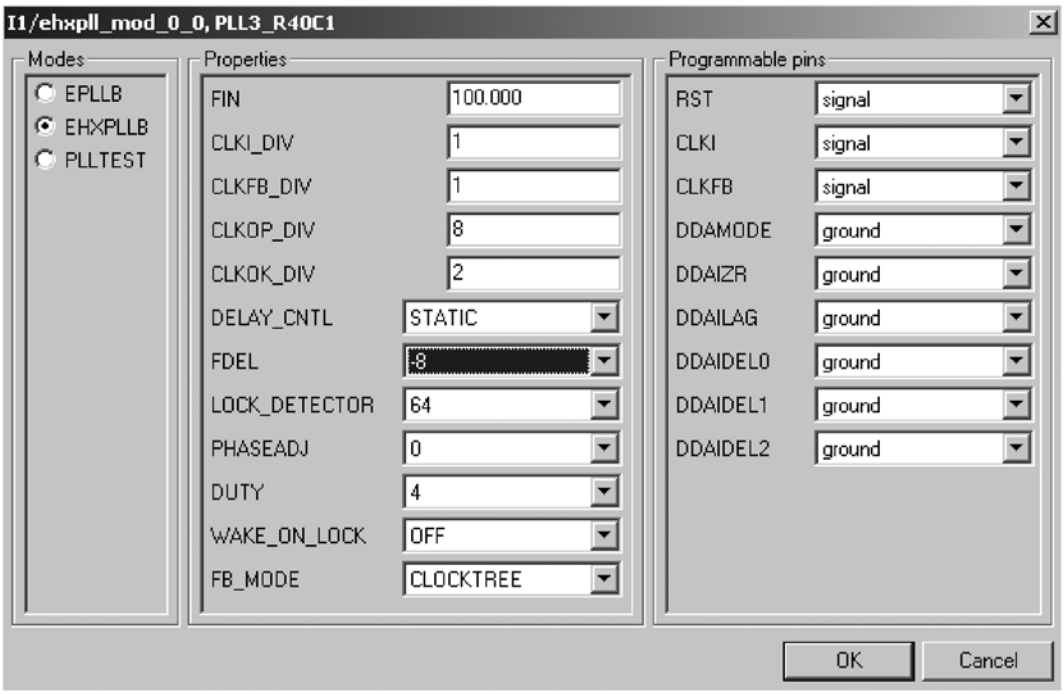
3. Pre-Map Preference Editor: Users can enter the FDEL value in the Pre-Map Preference Editor as shown in Figure 11-4.

Figure 11-4. Pre-Map Preference Editor



4. EPIC Device Editor: Users can edit their preferences in the EPIC Device Editor as shown in Figure 11-5.

Figure 11-5. EPIC Preferences Edit Window



Dynamic Delay Adjustment

The Dynamic Delay Adjustment is controlled by the DDAMODE input. When the DDAMODE input is set to “1”, the delay control is handled through the inputs, DDAIZR, DDAILAG and DDAIDEL(2:0). For this mode, the attribute “DELAY_CNTL” must be set to “DYNAMIC”. Table 11-3 shows the delay adjustment values based on the attribute/input settings.

In this mode, the PLL may come out of lock due to the abrupt change of phase. RST must be asserted to re-lock the PLL. Upon de-assertion of RST, the PLL will start the lock-in process and will take the t_{LOCK} time to complete the PLL lock.

Table 11-3. Delay Adjustment

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

Note: t_{DLY} = Unit Delay Time = 250 ps (nominal). See the data sheet for the tolerance of this delay

PLL Usage in IPexpress

Including sysCLOCK PLLs in a Design

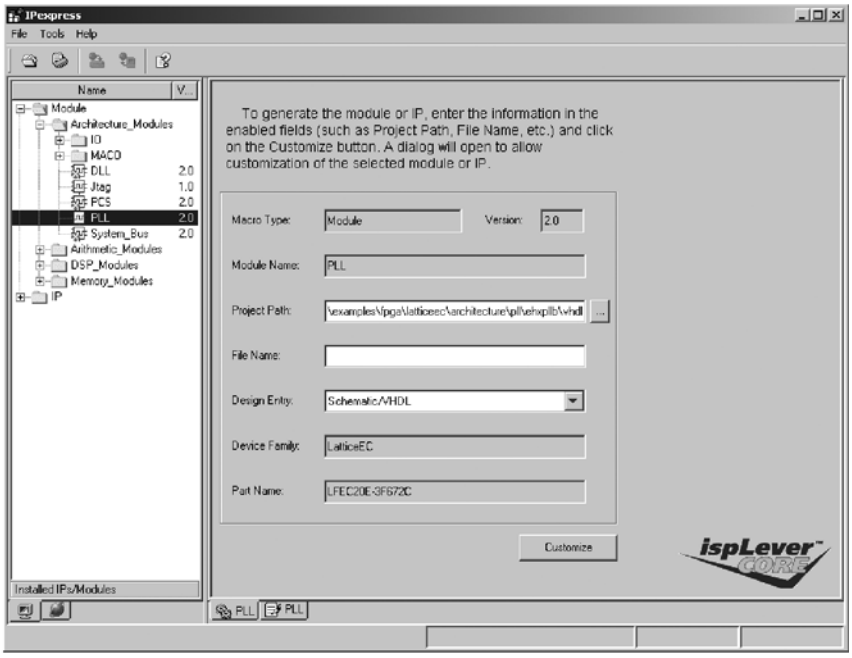
The sysCLOCK PLL capability can be accessed through the IPexpress GUI. The following section describes the usage of IPexpress.

IPexpress Usage

The LatticeECP/EC and LatticeXP PLL is fully supported in IPexpress in the ispLEVER software. IPexpress allows the user to define the desired PLL using a simple, easy-to-use GUI. Following definition, a VHDL or Verilog module that instantiates the desired PLL is created. This module can be included directly in the user's design.

Figure 11-6 shows the main window when PLL is selected. The only entry required in this window is the module name. After entering the module name, clicking on "Customize" will open the "Configuration" window as shown in Figure 11-7.

Figure 11-6. IPexpress Main Window

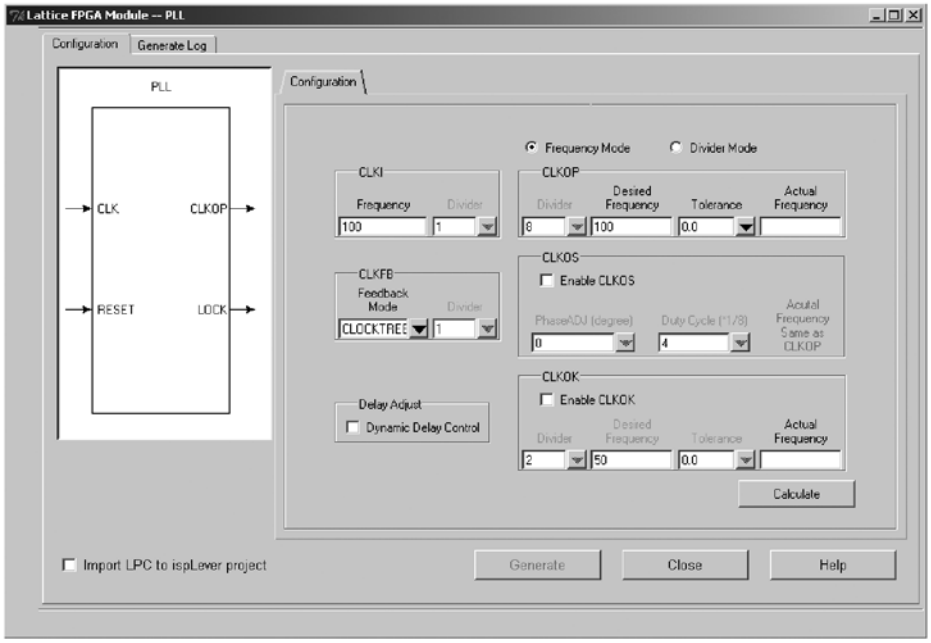


Configuration Tab
The Configuration Tab lists all user accessible attributes. Default values are set initially.

There are two modes in the Configuration Tab which can be used to configure the PLL, Frequency Mode and Divider Mode.

Frequency Mode: In this mode, the user enters input and output clock frequencies and the software calculates the divider settings for the user. If the output frequency the user entered is not achievable, the nearest frequency will be displayed in the 'Actual' text box. After input and output frequencies are entered, clicking the 'Calculate' button will display the divider values. If the desired output frequency is not achievable with the given frequency tolerance, the software generates an error. Users may increase the frequency tolerance or change the output frequencies. Figure 11-7 shows the Configuration Tab window.

Figure 11-7. Configuration Tab



Divider Mode: In this mode, the user sets the input frequency and divider settings. It is assumed the user is familiar with the PLL operation. The user must choose the CLKOP Divider value to maximize the f_{VCO} to achieve optimum PLL performance. After input frequency and divider settings are set, clicking the 'Calculate' button will display the output frequencies. If the divider settings are out of the PLL specification, the software will generate an error.

EHXPLLB Example Projects

ispLEVER provides example PLL projects for first time PLL users.

In the ispLEVER Project Navigator, go to the **File** menu and select **Open Examples...**.

Select the **FPGA** folder. The LatticeEC and LatticeXP folders include PLL example projects in both Verilog and VHDL.

Equations for Generating Input and Output Frequency Ranges

The values of f_{IN}, f_{OUT} and f_{VCO} are the absolute frequency ranges for the PLL. The values of f_{INMIN}, f_{INMAX}, f_{OUTMIN} and f_{OUTMAX} are the calculated frequency ranges based on the divider settings. These calculated frequency ranges become the limits for the specific divider settings used in the design.

Table 11-4. Frequency Limits

Parameter	LatticeECP/EC	LatticeXP
f_{IN}	Note 1	
f_{OUT}	Note 1	
f_{OUTK}	Note 1	
f_{VCO} (Hz)	Note 1	
CLKI Divider	1 to 16	1 to 15
CLKFB Divider	1 to 16	1 to 15
CLKOP Divider	See Table 11-2	
CLKOK Divider	2, 4, 6, 8,.. ,126, 128	
Maximum (N*V)	32	30
f_{PFD} (f_{IN}/M) (Hz)	Note 1	

Note: Refer to data sheet for the latest data.

The divider names are abbreviated with legacy names as:

- CLKI DIVIDER:M
- CLKFB DIVIDER:N
- CLKOP DIVIDER:V
- CLKOK DIVIDER:K

for use in the equations below.

f_{VCO} Constraint

From the loop:

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

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:

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

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

$$f_{INMIN} = f_{PFDMIN} * M = 25 * M \text{ (assume } f_{PFDMIN} = 25) \quad (8)$$

Equation (5) becomes:

$$f_{INMIN} = ((f_{VCOMIN} / (V*N))*M, \text{ if below } 25 * M \text{ round up to } 25 * M \quad (9)$$

From the loop:

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

Assume $f_{INMAX} = 420$

Equation (6) becomes:

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

From equation (1):

$$f_{OUTMIN} = f_{INMIN} * (N/M), \text{ if below } 25 * N \text{ round up to } 25 * N \quad (12)$$
$$f_{OUTMAX} = f_{INMAX} * (N/M), \text{ if above 420 round down to 420} \quad (13)$$

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

Clock Distribution in LatticeECP/EC and LatticeXP

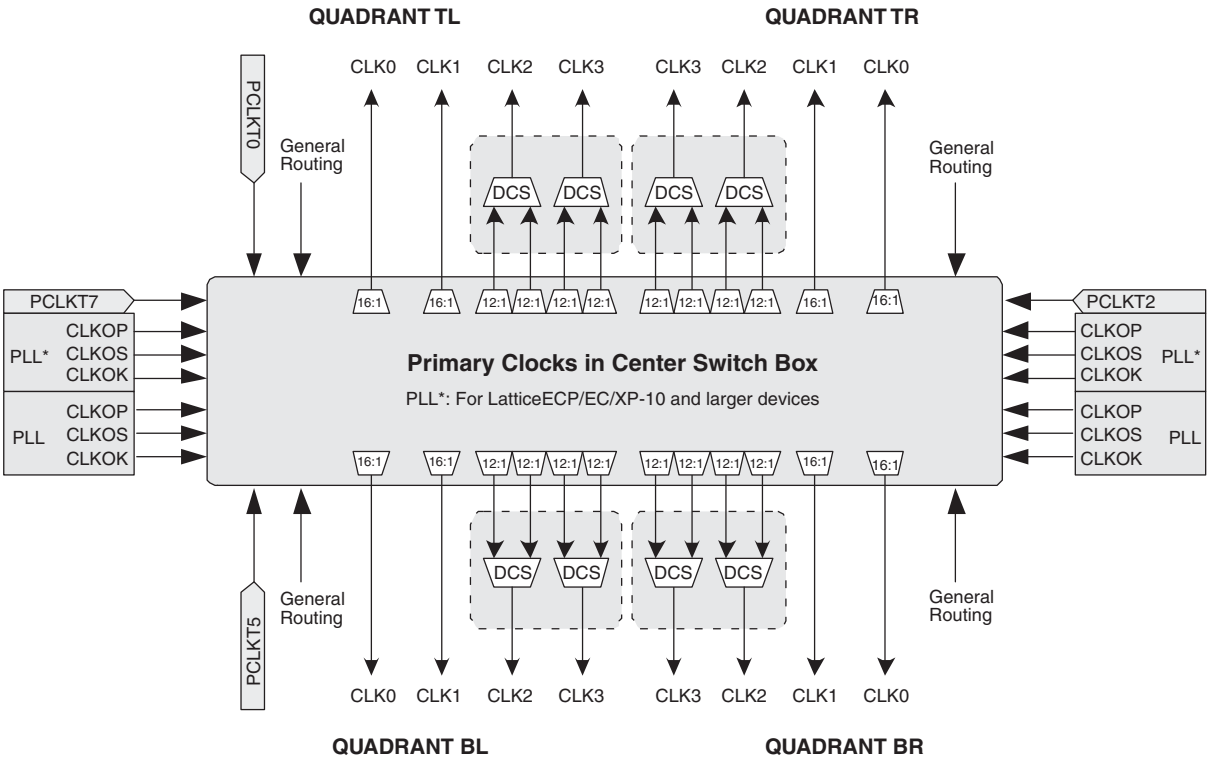
The clock inputs are selected from external I/Os, the sysCLOCK PLLs or general routing. These clock inputs are fed through the chip via a clock distribution system.

LatticeECP/EC and LatticeXP devices provide a quadrant-based primary and secondary clock structure.

Primary Clock Sources and Distribution

Each quadrant has four primary clock nets: CLK0, CLK1, CLK2 and CLK3. CLK2 and CLK3 provide dynamic clock selection (DCS) capability. Figure 11-8 illustrates the block diagram of the primary clock distribution.

Figure 11-8. Primary Clocks and Center Switch Boxes



Note: Two PLLs are available in LatticeECP/EC/XP-6 or smaller devices.

Note on the Primary Clock
The CLKOP must be used as the feedback source to optimize the PLL performance.

Most designers use the PLL for the clock tree injection removal mode and the CLKOP should be assigned as the primary clock. This is done automatically by the software unless the user specifies otherwise.

CLKOP can route to CLK0 and CLK1 only. CLKOS/CLKOK can route to all primary clocks (CLK0 to CLK3).

When CLK2 or CLK3 is used as a primary clock and there is only one clock input to the DCS, the DCS is assigned as a buffer mode by the software. Please see the DCS section of this document for further information.

Clock Net Preferences

There are two clock nets, primary clock and secondary clock.

As illustrated in Figure 11-9, users can set each clock to the desired clock net in the Pre-map Preference Editor or write in the Preference File as shown in the examples below.

Primary-Pure and Primary-DCS

Primary Clock Net can be assigned to either Primary-Pure (CLK0 and CLK1) or Primary-DCS (CLK2 and CLK3).

Syntax Example

```
USE PRIMARY DCS NET "bf_clk";
```

Global Primary Clock and Quadrant Primary Clock

Global Primary Clock
If a primary clock is not assigned as a quadrant clock, the software assumes it is a Global Clock.

There are two Global Primary/Pure Clocks and two Global Primary/DCS Clocks available.

Quadrant Primary Clock
Any Primary Clock may be assigned to a Quadrant Clock. The Clock may be assigned to a single quadrant or to two adjacent quadrants (not diagonally adjacent).

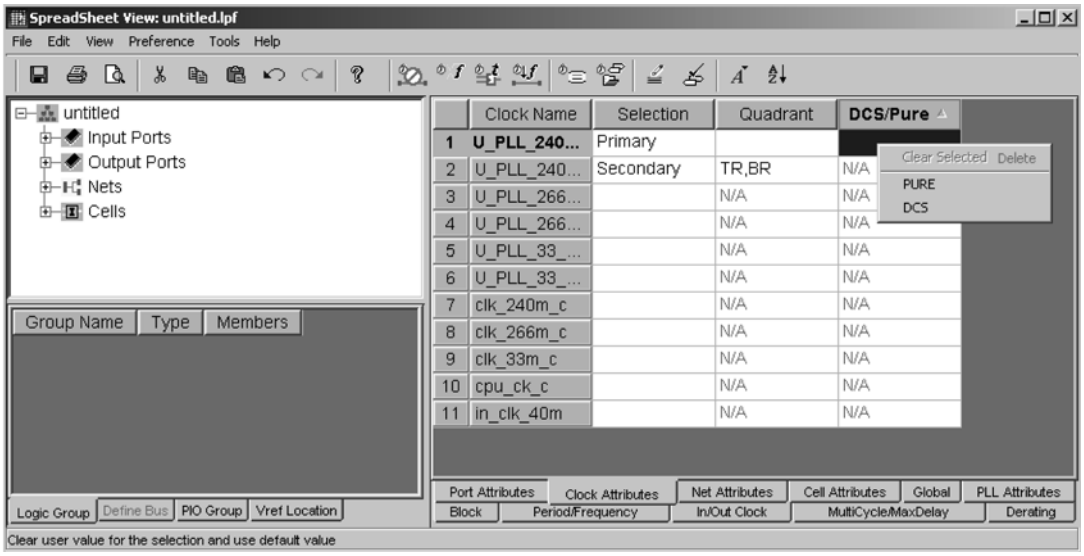
When the quadrant clock net is used, users must ensure that the registers each clock drives can be assigned in that quadrant without any routing issues.

With the Quadrant Primary Clocking scheme, the maximum number of Primary Clocks is 16 as long as all the Primary Clock Sources are avaialble.

Syntax Example

```
USE PRIMARY PURE NET "bf_clk" QUADRANT_TL;
```

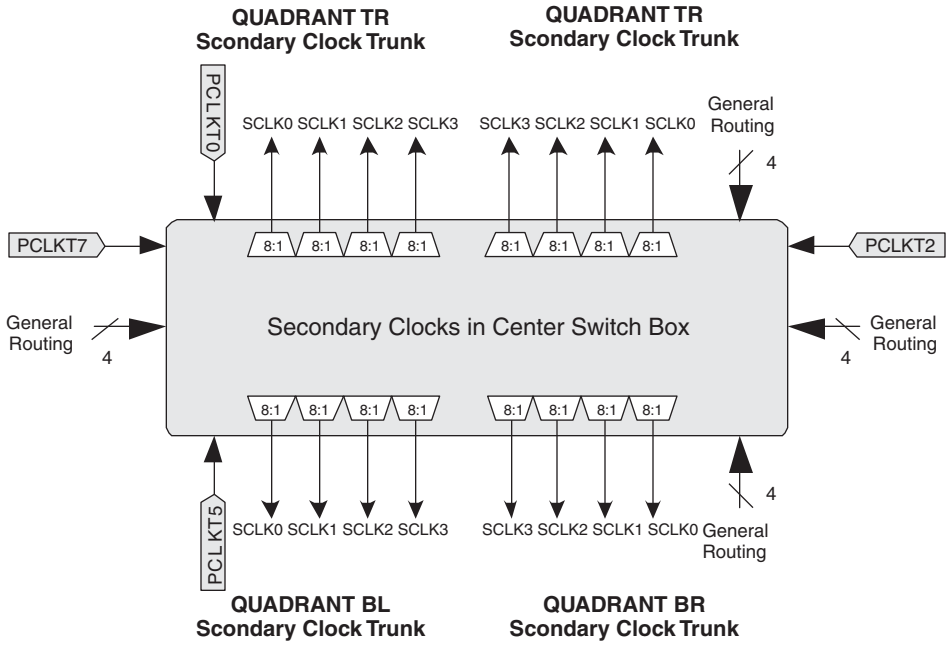
Figure 11-9. Clock Preferences in the Pre-map Preference Editor



Secondary Clock Sources and Distribution

LatticeECP/EC and LatticeXP devices support quadrant base Secondary Clocks. Figure 11-10 describes the Secondary Clock arrangement.

Figure 11-10. Secondary Clock Center Switch Box



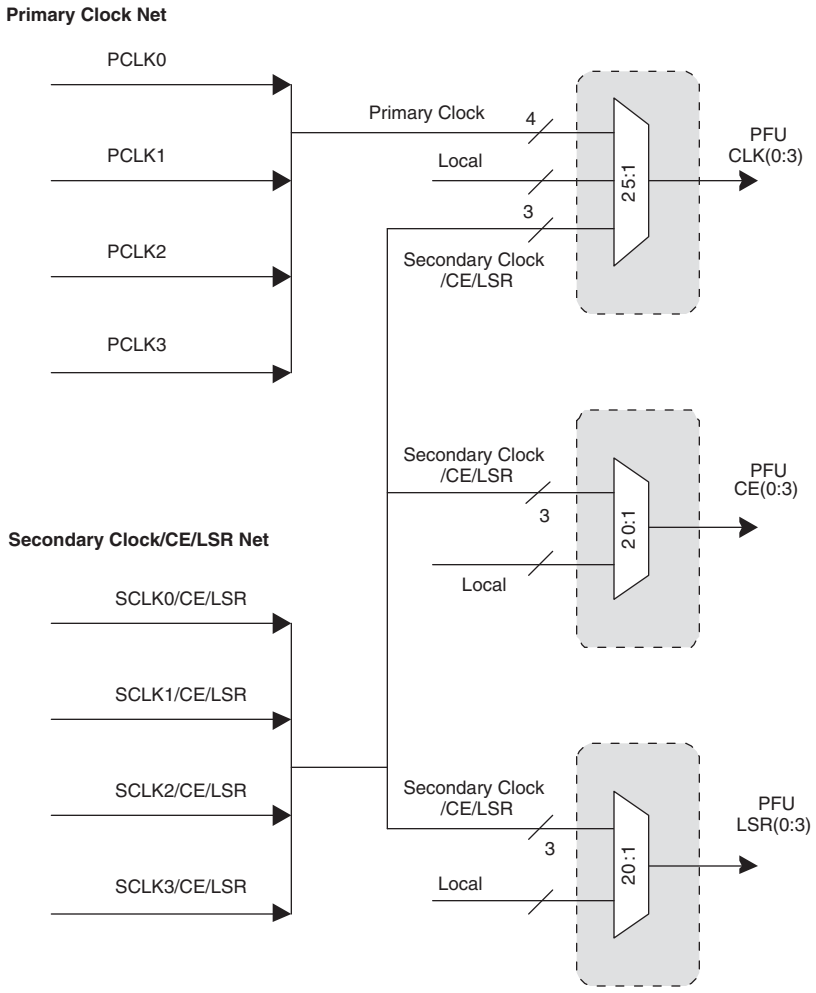
Limitations on Secondary Clock Availability

As illustrated in Figure 11-11, three secondary clocks are shared with CLK, CE and LSR.

This routing scheme limits the secondary clocks available per quadrant base to three, which results in a maximum of 12 available secondary clocks per device. Figure 11-11 illustrates the primary and secondary clock distribution structure of the PFUs.

LFEC6/LFXP6 and smaller devices have limited routing resources and can implement a maximum of nine secondary clocks per device.

Figure 11-11. Primary Clock and Secondary Clock/CE/LSR Distribution



Dynamic Clock Selection (DCS)

DCS is a global clock buffer incorporating a smart multiplexer function that takes two independent input clock sources and avoids glitches or runt pulses on the output clock, regardless of when the enable signal is toggled. The DCS blocks are located in pairs at the center of each side of the device. Thus, there are eight of them in every device.

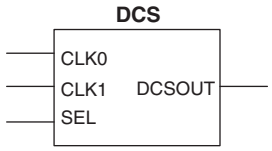
Table 11-5. DCS I/O

I/O	Name	Description
Input	SEL	Input Clock Select
	CLK0	Primary Clock Input 0
	CLK1	Primary Clock Input 1
Output	DCSOUT	To Primary Clock

Table 11-6. DCS Attributes

Attribute Name	Description	Output		Value
		SEL=0	SEL=1	
DCS MODE	Rising edge triggered, latched state is high	CLK0	CLK1	POS (Default)
	Falling edge triggered, latched state is low	CLK0	CLK1	NEG
	Sel is active high, Disabled output is low	0	CLK1	HIGH_LOW
	Sel is active high, Disabled output is high	1	CLK1	HIGH_HIGH
	Sel is active low, Disabled output is low	CLK0	0	LOW_LOW
	Sel is active low, Disabled output is high	CLK0	1	LOW_HIGH
	Buffer for CLK0	CLK0	CLK0	CLK0
	Buffer for CLK1	CLK1	CLK1	CLK1

Figure 11-12. DCS Primitive Symbol



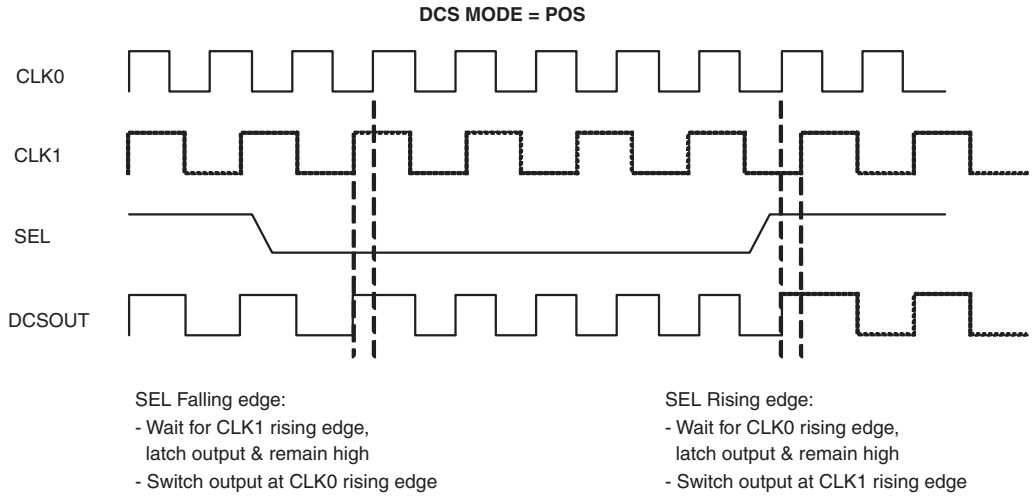
DCS Waveforms

The DCSOUT waveform timing is described in Figure 11-13 for each mode. The 'POS' and 'NEG' modes describe DCSOUT timing at both the falling and rising edges of SEL.

Figure 11-13. DCS Waveforms

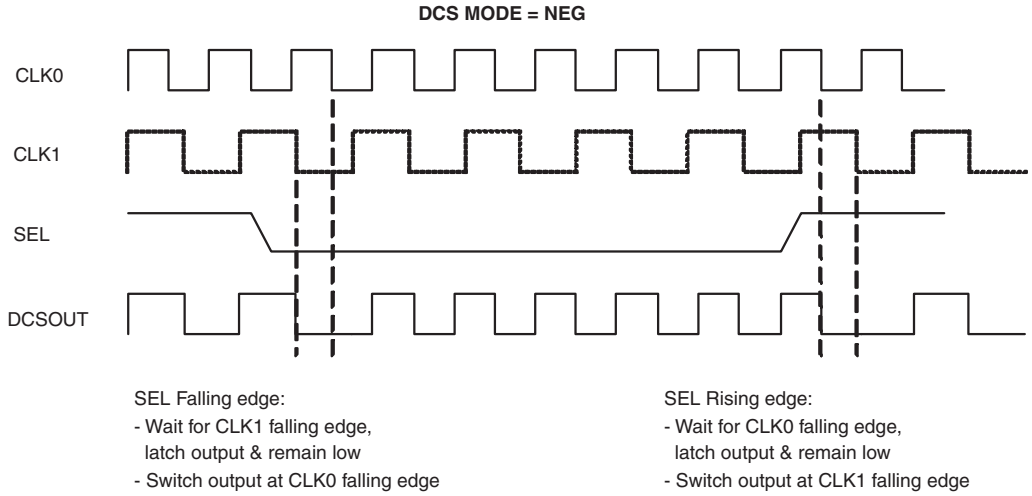
DCS MODE = POS

At the rising edge (POS) of SEL, the DCSOUT changes from CLK0 to CLK1. This mode is the default mode.



DCS MODE = NEG

At the falling edge (NEG) of SEL, the DCSOUT changes from CLK0 to CLK1.

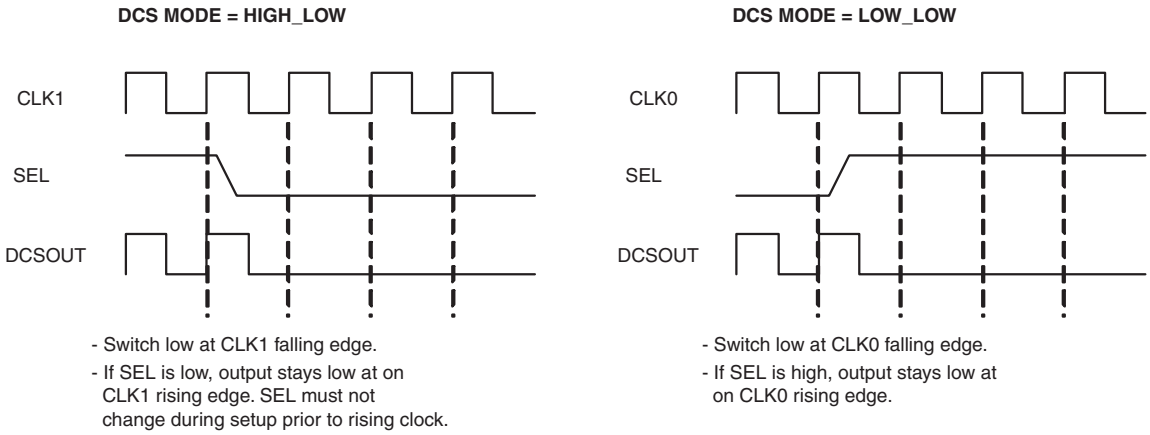


DCS MODE = HIGH_LOW

SEL is active high (HIGH) to select CLK1, and the disabled output is LOW.

DCS MODE = LOW_LOW

SEL is active low (LOW) to select CLK0, and the disabled output is LOW.

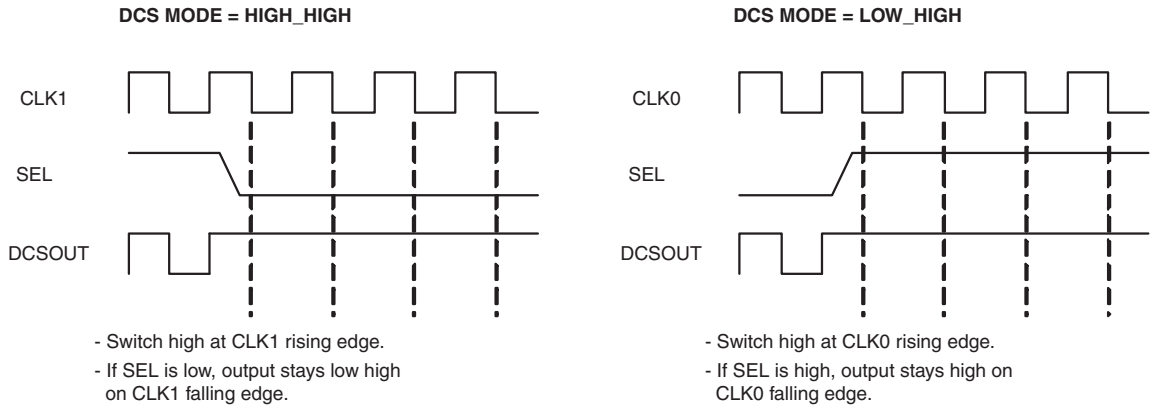


DCS MODE = HIGH_HIGH

SEL is active high (HIGH) to select CLK1, and the disabled output is HIGH.

DCS MODE = LOW_HIGH

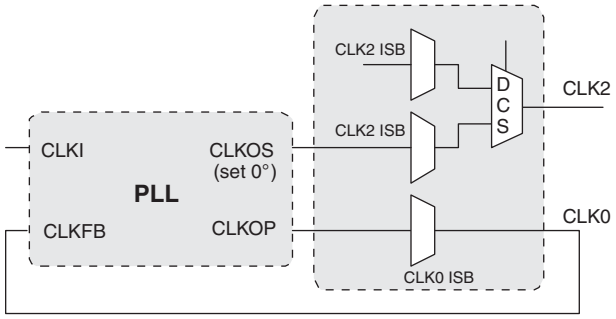
SEL is active low (LOW) to select CLK0, and the disabled output is HIGH.



Use of DCS with PLL

The four PLL CLKOP sources reach CLK0 and CLK1 of the quadrant clock. When using the DCS, the PLL needs a free-running feedback path to keep the PLL in lock. The user should use CLKOP as this feedback path, and CLKOS as the input into the DCS. CLKOP does not reach CLK2 or CLK3 to prevent the user from using the PLL improperly with DCS. See Figure 11-14.

Figure 11-14. Implementation of Dynamic Clock Select for a PLL Clock (Must Use Both CLKOP and CLKOS)



Other Design Considerations

Jitter Considerations

The Clock Output jitter specifications assume that the reference clock is free of jitter. Even if the clock source is clean, there are a number of sources that place noise in the PLL clock input. While intrinsic jitter is not avoidable, there are ways to minimize the input jitter and output jitter.

Signal inputs that share the same I/O bank with PLL clock inputs are preferably less noisy inputs and slower switching signals. Try to avoid placing any high speed and noisy signals in the same I/O bank with clock signals if possible. Use differential signaling if possible.

When external feedback is used, the PCB path must be well designed to avoid reflection as well as noise coupling from adjacent signal sources. A shorter PCB feedback path length does not necessarily reduce feedback input jitter.

Simulation Limitations

- Simulation does not compensate for external delays and dividers in the feedback loop.
- The LOCK signal is not simulated according to the t_{LOCK} specification. The LOCK signal will appear active shortly after the simulation begins, but will remain active throughout the simulation.
- The jitter specifications are not included.

PCB Layout Recommendations for VCCPLL and GNDPLL if Separate Pins are Available

It is best to connect VCCPLL to VCC at a single point using a filter and to create a separate GNDPLL plane directly under it (tied via a single point to GND).

Separate islands for both VCCPLL and GNDPLL are recommended if applicable.

DCS Usage with Verilog

```
module dcs(clk0,clk1,sel,dcsout);

input clk0, clk1, sel;
output dcsout;

DCS DCSInst0 (.SEL(sel),.CLK0(clk0),.CLK1(clk1),.DCSOUT(dcsout));
defparam DCSInst0.DCSMODE = "CLK0";

endmodule
```

DCS Usage with VHDL

```
COMPONENT DCS
-- synthesis translate_off
        GENERIC       (
                      DCSMODE : string :=  "POS"
                      );
-- synthesis translate_on

        PORT          (
                      CLK0           :IN    std_logic;
                      CLK1           :IN    std_logic;
                      SEL            :IN    std_logic;
                      DCSOUT         :OUT   std_logic
                      );
END COMPONENT;

    attribute DCSMODE : string;
    attribute DCSMODE of DCSinst0 : label is "POS";

begin

DCSInst0: DCS
-- synthesis translate_off

        GENERIC MAP(
                      DCSMODE  => "POS"
                      )
-- synthesis translate_on

        PORT MAP      (
                      SEL            =>  clksel,
                      CLK0           =>  dcsclk0,
                      CLK1           =>  sysclk1,
                      DCSOUT         =>  dcsclk
);
```

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
June 2004	01.0	Initial release.
October 2004	01.1	RST description with timing diagram added.
		Primitive 'Wake_On_Lock' removed.
		Added RST input in EPLLB.
		Fin min 33 replaced with 25.
		FB_MODE default = CLOCKTREE
		CLKOP_DIV values for EHXPLLB 2, 4, 8,16, 32.
December 2004	02.0	Appendices C and D integrated to body of the document.
		DCS source code example moved to Appendix A.
January 2005	03.0	LatticeXP information added.
		Figures 6 and 7 updated.
		CLKOP_FREQ, CLKOK_FREQ user attributes added.
		FB_MODE added.
October 2005	04.0	Clock Distribution section added.
		Example code section removed and referred to help file.
		GUI screen shots updated.
		MM/IP Manager renamed as IPexpress.
		Epllb definition section removed.
		Clkos/clkok select attributes added.
September 2006	04.1	Detailed clock distribution information added.

Appendix A. Clock Preferences

A few key clock preferences are introduced below. Refer to the 'Help' file for other preferences and detailed information.

ASIC

The following preference command assigns a phase of 90° to the CIMDLLA CLKOP.

```
ASIC "my_pll" TYPE "EXHXPLLB" CLKOS_PHASE=90;
```

FREQUENCY

The following physical preference command assigns a frequency of 100 MHz to a net named clk1.

```
FREQUENCY NET "clk1" 100 MHz;
```

The following preference specifies a hold margin value for each clock domain.

```
FREQUENCY NET "RX_CLKA_CMOS_c" 100.000 MHz HOLD_MARGIN 1 ns;
```

MAXSKEW

The following command assigns a maximum skew of 5 ns to a net named NetB.

```
MAXSKEW NET "NetB" 5 NS;
```

MULTICYCLE

The following command will relax the period to 50 ns for the path starting at COMPA to COMPB (NET1).

```
MULTICYCLE "PATH1" START COMP "COMPA" END COMP "COMPB" NET "NET1" 50 NS ;
```

PERIOD

The following command assigns a clock period of 30 ns to the port named Clk1.

```
PERIOD PORT "Clk1" 30 NS;
```

PROHIBIT

This command prohibits the use of a primary clock to route a clock net named bf_clk.

```
PROHIBIT PRIMARY NET "bf_clk";
```

CLOCK_TO_OUT

Specifies a maximum allowable output delay relative to a clock.

Below are two preferences using both the CLKPORT and CLKNET keywords showing the corresponding scope of TRACE reporting.

The CLKNET will stop tracing the path before the PLL, so you will not get PLL compensation timing numbers.

```
CLOCK_TO_OUT PORT "RxAddr_0" 6.000000 ns CLKNET "pll_rxclk" ;
```

The above preference will yield the following clock path:

Physical Path Details:

Clock path pll_inst/pll_utp_0_0 to PFU_33:

Name	Fanout	Delay (ns)	Site	Resource
ROUTE	49	2.892	ULPPLL.MCLK to	R3C14.CLK0 pll_rxclk

2.892 (0.0% logic, 100.0% route), 0 logic levels.

If CLKPORT is used, the trace is complete back to the clock port resource and provides PLL compensation timing numbers.

```
CLOCK_TO_OUT PORT "RxAddr_0" 6.000000 ns CLKPORT "RxClk" ;
```

The above preference will yield the following clock path:

Clock path RxClk to PFU_33:

Name	Fanout	Delay (ns)	Site	Resource
IN_DEL	---	1.431	D5.PAD to	D5.INCK RxClk
ROUTE	1	0.843	D5.INCK to	ULPPLL.CLKIN RxClk_c
MCLK_DEL	---	3.605	ULPPLL.CLKIN to	ULPPLL.MCLK pll_inst/pll_utp_0_0
ROUTE	49	2.892	ULPPLL.MCLK to	R3C14.CLK0 pll_rxclk

8.771 (57.4% logic, 42.6% route), 2 logic levels.

INPUT_SETUP

Specifies an setup time requirement for input ports relative to a clock net.

```
INPUT_SETUP PORT "datain" 2.000000 ns HOLD 1.000000 ns CLKPORT "clk" PLL_PHASE_BACK ;
```

PLL_PHASE_BACK

This preference is used with INPUT_SETUP when the user wants a trace calculation based on the previous clock edge.

This preference is useful when setting the PLL output phase adjustment. Since there is no negative phase adjustment provided, the PLL_PHASE_BACK preference works as if negative phase adjustment is available.

For example:

If phase adjustment of -90° of CLKOS is desired, the user can set the phase to 270° and set the INPUT_SETUP preference with PLL_PHASE_BACK.

Estimating Power Using the Power Calculator for LatticeECP/EC and LatticeXP Devices

November 2006 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 Supply Sequencing and Hot Socketing

LatticeECP™, LatticeEC™ and LatticeXP devices have eight sysIO™ buffer banks in addition to the $V_{CC,}$ V_{CCAUX} and V_{CCJ} power supplies; each is capable of supporting multiple I/O standards. Each sysIO bank has its own I/O supply voltage (V_{CCIO}), and two voltage references V_{REF1} and V_{REF2} resources allowing each bank to be completely independent from each other.

The LatticeECP/EC and LatticeXP devices are designed to ensure predictable behavior during power-up and power-down. Power supplies can be sequenced in any order. The I/Os remain in tristate until the power supply voltage is high enough to ensure reliable operation during power up and power-down sequences and the leakage into I/O pins is controlled to within specified limits. Refer to the Typical I/O Behavior During Power-up and Hot Socketing sections of the device data sheet for more details.

Power Calculator Hardware Assumptions

The power consumption for a device can be coarsely broken down into the DC portion and the AC portion.

The power calculator reports the power dissipation in terms of:

1. DC portion of the power consumption.
2. AC portion of the power consumption.

The DC power (or the static power consumption) is the total power consumption of the used and unused resources. These components are fixed for each resource used and depend upon the number of resource units utilized. The DC component also includes the static power dissipation for the unused resources of the device.

The AC portion of power consumption is associated with the used resources and it is the dynamic part of the power consumption. Its power dissipation is directly proportional to the frequency at which the resource is running and the number of resource units used.

Power Calculator

Power Calculator is a powerful tool which allows users to make an estimate of the power consumption at two different levels:

1. Estimate of the utilized resources before completing place and route
2. Post place and route design

For first level estimation, the user provides estimates of device usage in the Power Calculator Wizard and the tool provides a rough estimate of the power consumption.

The second level is a more accurate approach where the user imports the actual device utilization by importing the post Place and Route netlist (NCD) file.

Power Calculator Equations

The power equations used in the Power Calculator have the following general form:

Total DC Power (Resource)
= 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 sum of Quiescent Power and the individual DC power of the resources in the Power Calculator.

Total DC Power (I_{CCAUX})
= $K_{RESOURCE}$ * 500 µA + Typical Standby I_{CCAUX}

Where:

KRESOURCE	is the number of reference input I/O such as HSTL/SSTL. For LVDS KRESOURCE is number of inputs divided by two.
I_{CCAUX}	is a DC current that does not change with I/O toggle rate or temperature.

Typical Standby I_{CCAUX} is found in the data sheet.

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

Total AC Power (Resource)
= $K_{RESOURCE}$ * 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, measured in mW/MHz.
f_{MAX}	is the maximum frequency at which the resource is running, measured in MHz.
$AF_{RESOURCE}$	is the activity factor for the resource group, as a percentage (%) of switching frequency.

Based on the above equations, if we wish to calculate the power consumption of the Slice portion, it will be as follows:

Total DC Power (Slice)
= 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}

The DC and AC power, for a dedicated block, like DSP in LatticeECP devices, is governed by the following equations.

Total DC Power (Resource)
= DC Leakage per Resource * $N_{RESOURCE}$

Total AC Power (Resource)

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

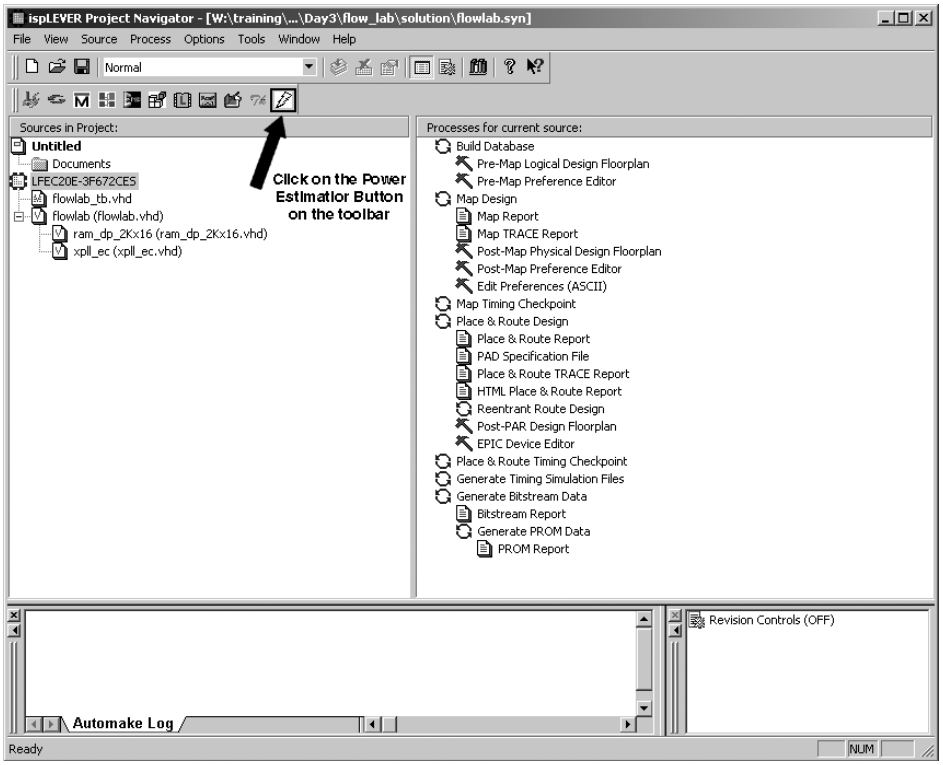
Where:

$N_{RESOURCE}$	is the total number of resources in a device.
$K_{RESOURCE}$	is the power constant for the resource, measured in mW/MHz.
f_{MAX}	is the maximum frequency at which the resource is running, measured in MHz.

Starting the Power Calculator

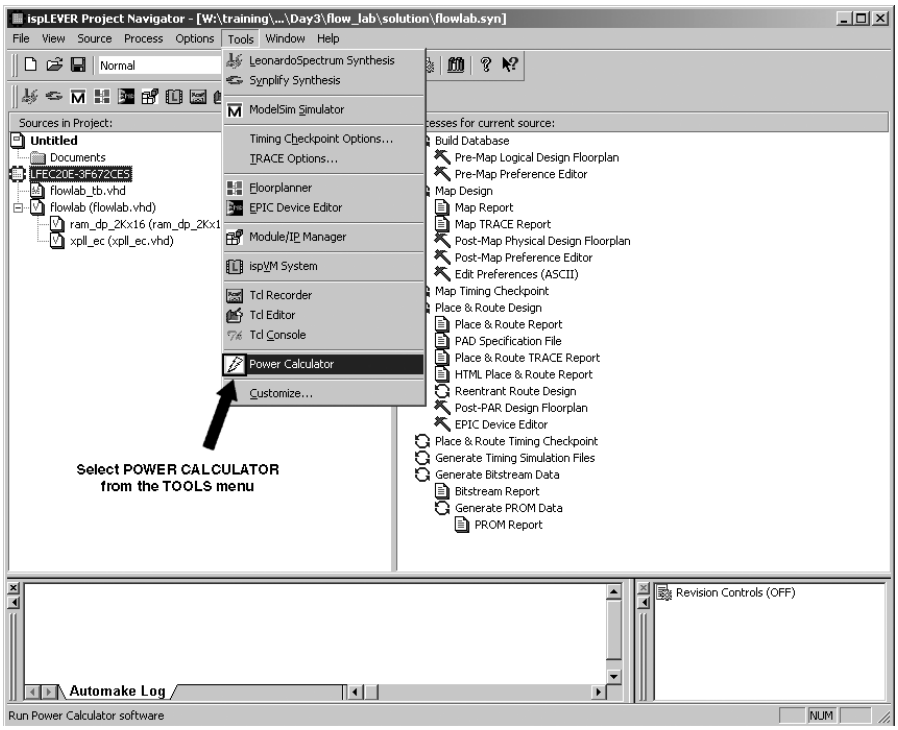
The user can launch the Power Calculator by one of the two methods. The first method is by clicking the Power Calculator button in the toolbar as shown in Figure 12-1.

Figure 12-1. Starting Power Calculator from Toolbar



Alternatively, users can launch the Power Calculator by going to the Tools menu and selecting the option Power Calculator as shown in Figure 12-2.

Figure 12-2. Starting Power Calculator from Tools Menu

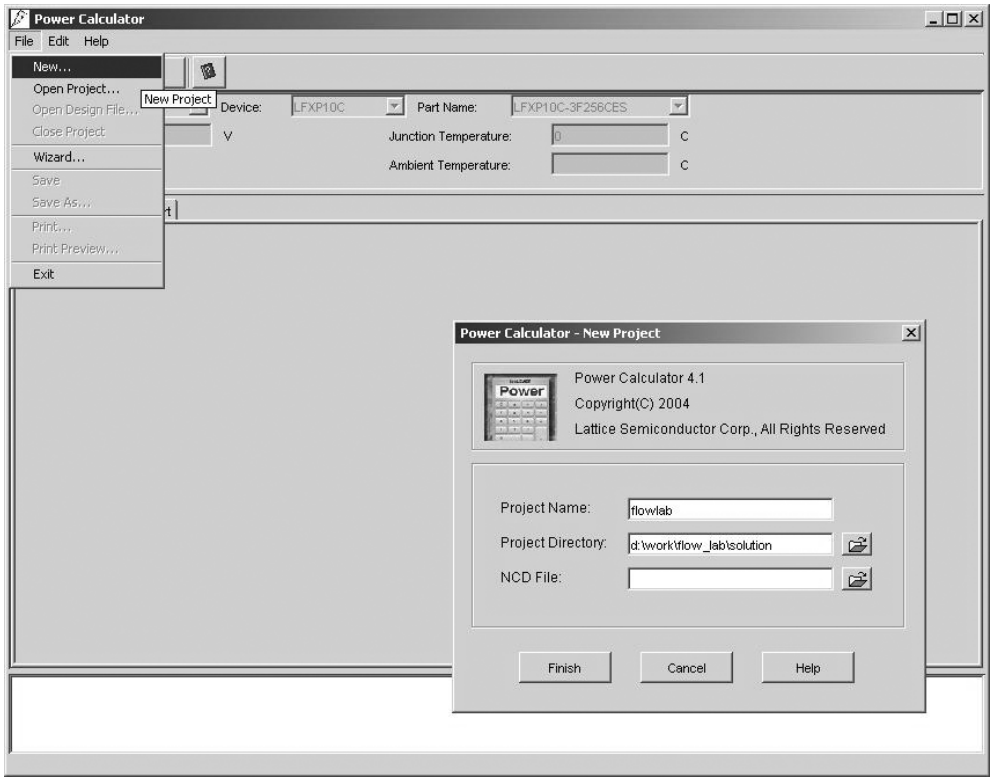


The Power Calculator does not support some of Lattice's older devices. The toolbar button and menu item is only present when supported devices are selected.

Starting a Power Calculator Project

Once the Power Calculator has been started, the Power Calculator window appears. Click on **File ->Menu**, and select **New** to get to the Start Project window, as shown in Figure 12-3.

Figure 12-3. Power Calculator Start Project Window (Create New Project)



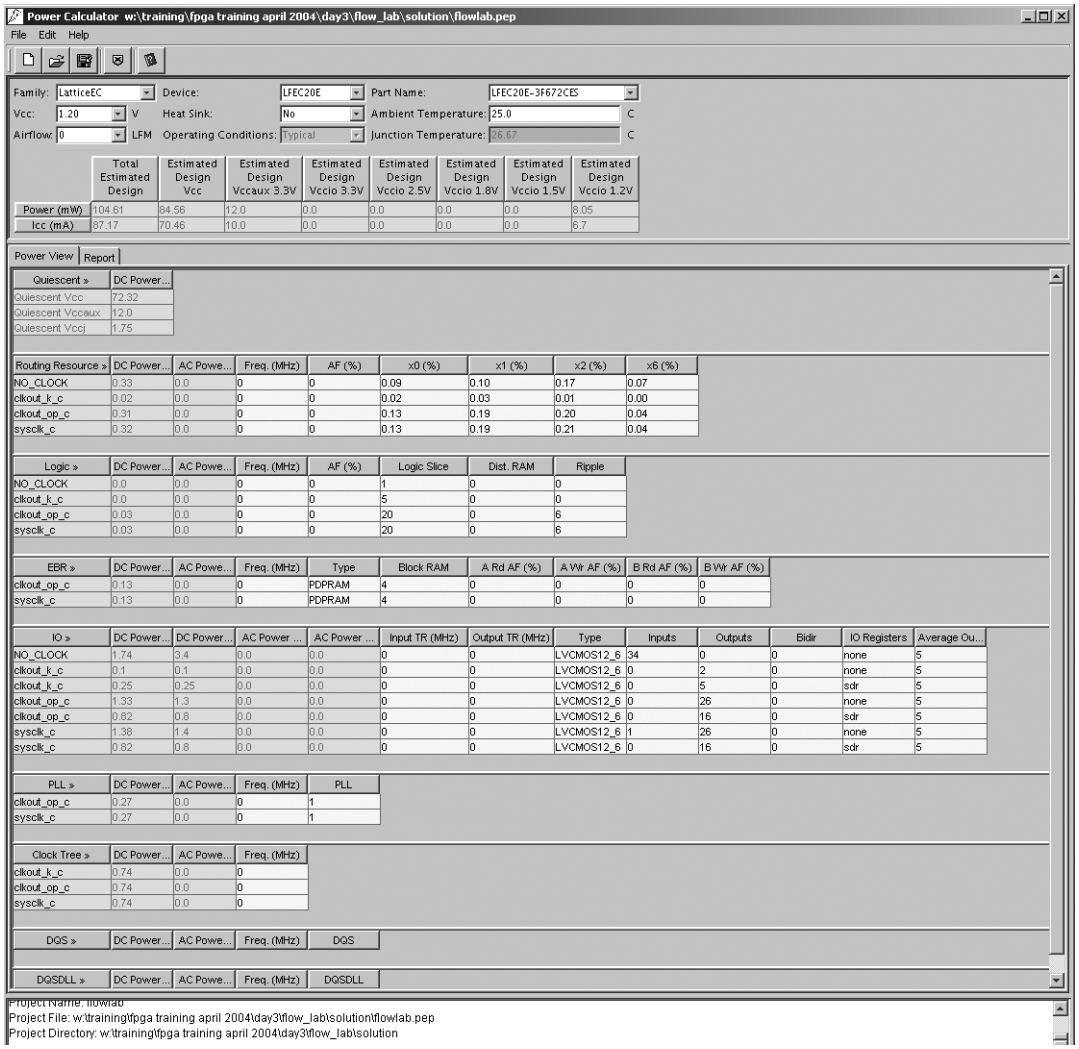
The Start Project window is used to create a new Power Calculator Project (*.pep project). Three pieces of data are input in the Start Project window.

1. The Power Calculator project name by default is same as the Project Navigator project name. The name can be changed, if desired.
2. Project Directory is where the Power Calculator project (*.pep) file will be stored. By default, the file is stored in the main project folder.
3. Input an NCD file (if available) or browse to the NCD file in a different location.

Power Calculator Main Window

The main power calculator window is shown in Figure 12-4.

Figure 12-4. Power Calculator Main Window (Type View)



The top pane of the window shows information about the device family, device and the part number as it appears in the Project Navigator. The V_{CC} used for the Power Calculation is also listed. Users have an option to provide the ambient temperature, and the junction temperature is calculated based on that.

Users can also enter values of airflow in Linear Feet per Minute (LFM) along with heat sink to get the junction temperature. A table in the top part of the Power Calculator summarizes the currents and power consumption associated with each type of power supply for the device. This also takes into consideration the I/O power supplies.

In the middle pane of the window, there are two tabs:

1. Power View
2. Report

The first tab is the Power view. Under this tab, the Power Calculator tool has an interactive spreadsheet type interface.

The second and third columns, which are shaded blue in the tool, provide the DC (or static) and AC (or dynamic) power consumption, respectively.

In case of the I/O, there are four columns that are shaded blue. These provide the DC and AC power for I/Os for the core voltage, V_{CC} and the I/O voltage supply, $V_{CCIO.}$

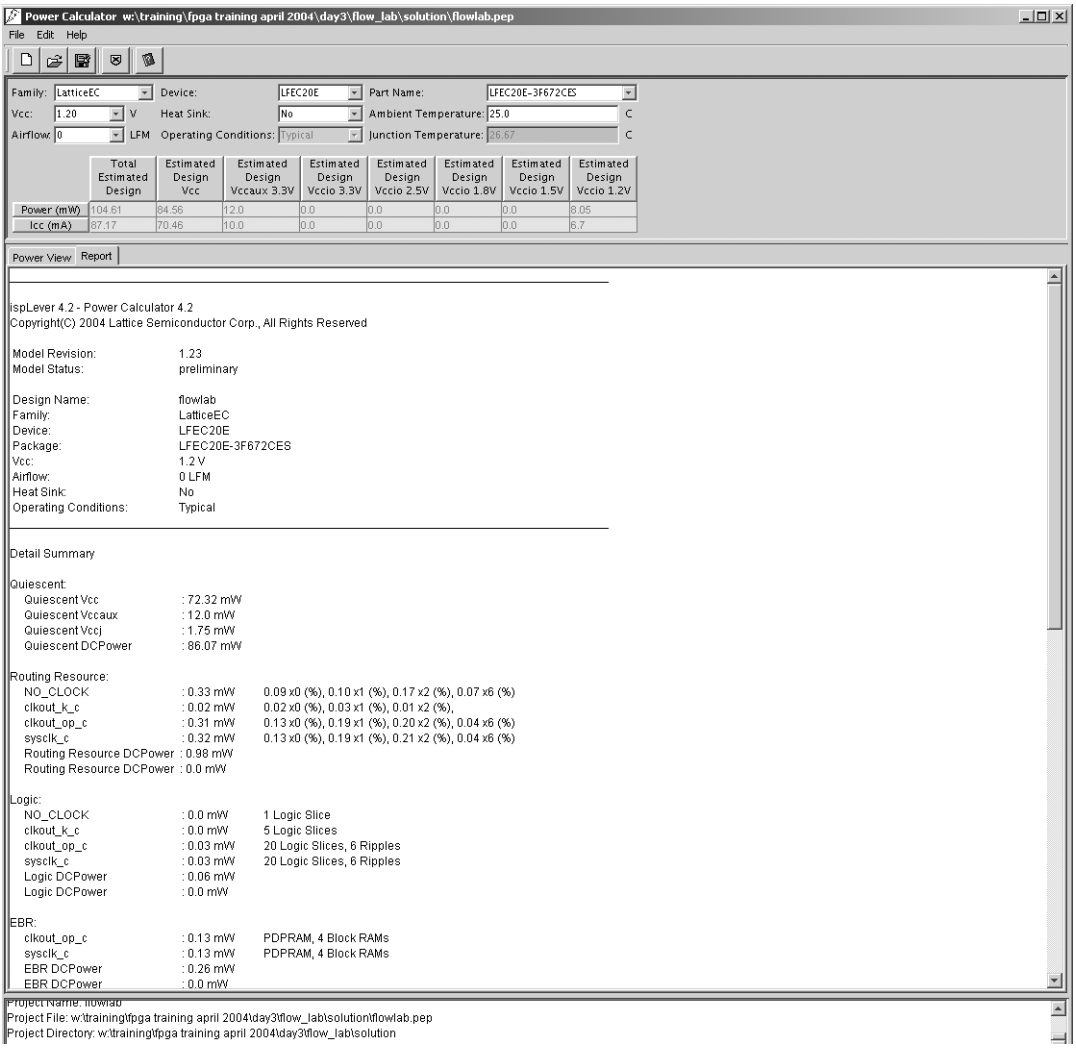
The first three rows show the Quiescent Power for $V_{CC,}$ V_{CCAUX} and $V_{CCJ.}$ These are DC power numbers for a blank device or device with no resource utilization.

Some of the cells are shaded yellow in the tool. These cells are editable cells and users can type in values such as frequency, activity factors and resource utilization.

The second tab or the Report tab is the summary of the Power View. This report is in text format that provides the details of the power consumption.

The final pane or the lower pane of the window is the log pane where users can see the log of the various operations in the Power Calculator.

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

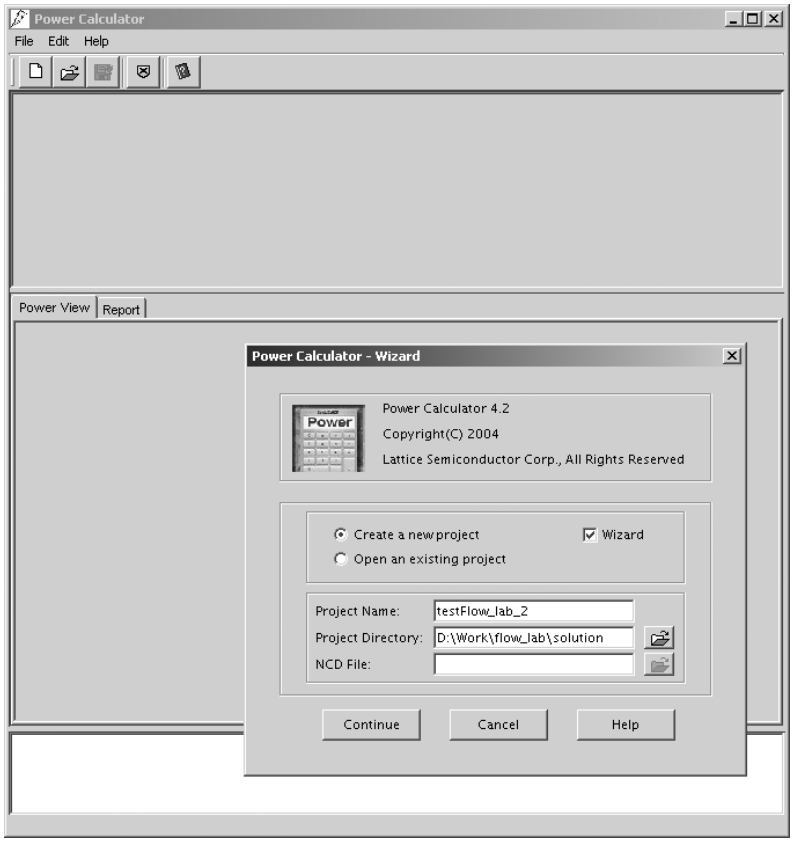


Power Calculator Wizard

The Power Calculator Wizard allows users to estimate the power consumption of a design. This estimation is done before actually creating the design. The user must understand the logic requirements of the design. The wizard asks the user to provide these parameters and then estimates the power consumption of the device.

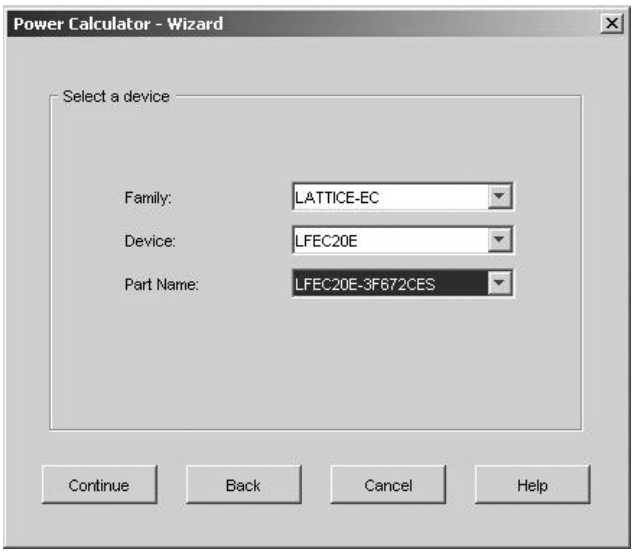
To start the Power Calculator in the wizard mode, go to the **File** menu and select **Wizard**. Alternatively, click on the **Wizard** button and get the **Power Calculator - Wizard** window, as shown in Figure 12-6. Select the option **Create a New Project** and check the **Wizard** check box in the Power Calculator Start Project window. Users provide the project name and the project folder and click **Continue**. Since power is being estimated before the actual design, no NCD file is required.

Figure 12-6. Power Calculator Start Project Window (Using the New Project Window Wizard)



The next screen, as shown in Figure 12-7, allows users to select the device family, device and appropriate part number. After making proper the selections, click **Continue**. This is shown in Figure 12-7.

Figure 12-7. Power Calculator Wizard Mode Window - Device Selection



In the following screens, as shown in Figures 12-8-12-12, users can select further resources such as I/O types and provide a clock name, frequency at which the clock in running and other parameters, by selecting the appropriate resource using the pull-down Type menu:

1. Routing Resources
2. Logic
3. EBR
4. I/O
5. PLL
6. Clock Tree

The numbers in these windows refers to the number of clocks and the index corresponds to each of the clocks. By default, the clock names are clk_1, clk_2, and so on. The name of each clock can be changed by typing in the Clock Name text box. For each clock domain and resource users can specify parameters such as frequency, activity factor, etc. Users can click the Create button for each clock-driven resource using the pull-down Type menu.

These parameters are then used in the Power Type View window (see Figure 12-13) which can be seen by clicking Finish.

Figure 12-8. Power Calculator Wizard Mode Window - Resource Specification - Logic

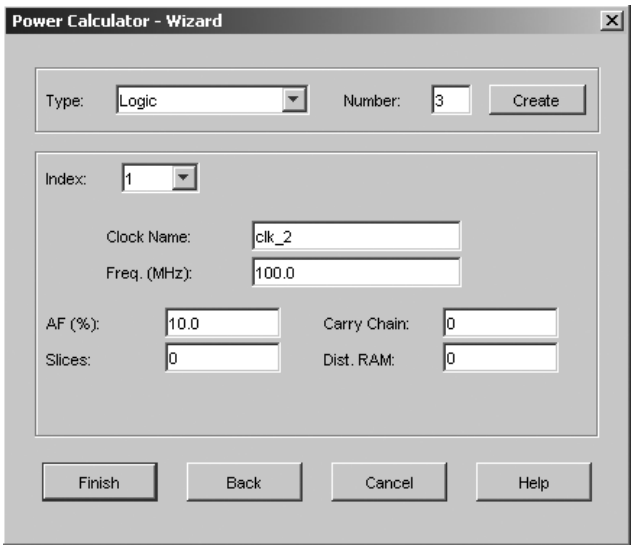


Figure 12-9. Power Calculator Wizard Mode Window - Resource Specification - EBR

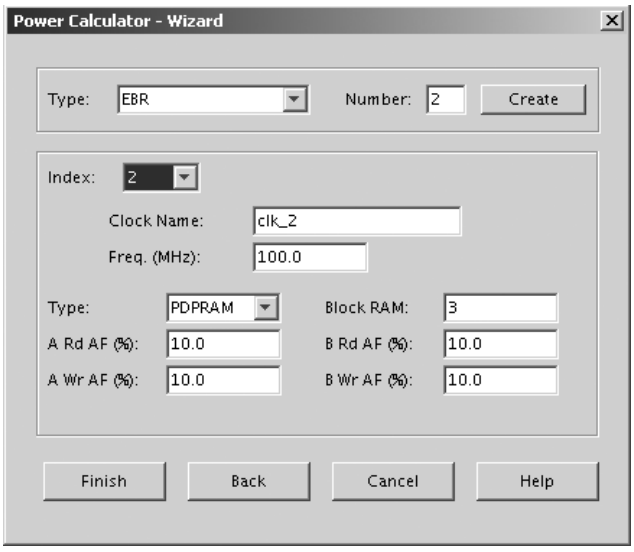


Figure 12-10. Power Calculator Wizard Mode Window - Resource Specification - PLL

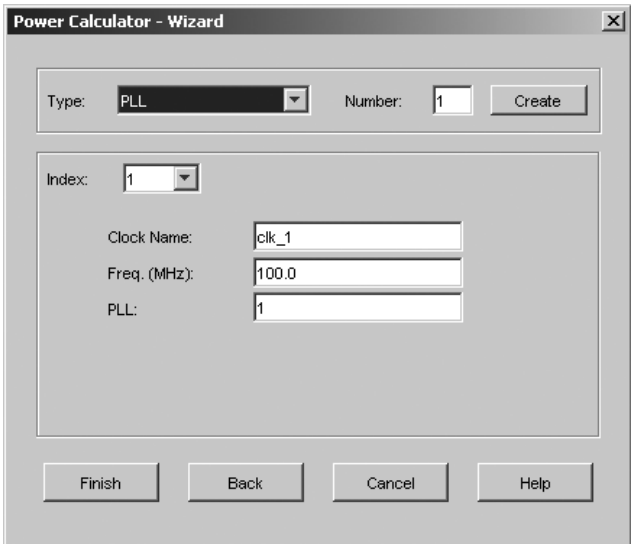


Figure 12-11. Power Calculator Wizard Mode Window - Resource Specification - Routing Resources

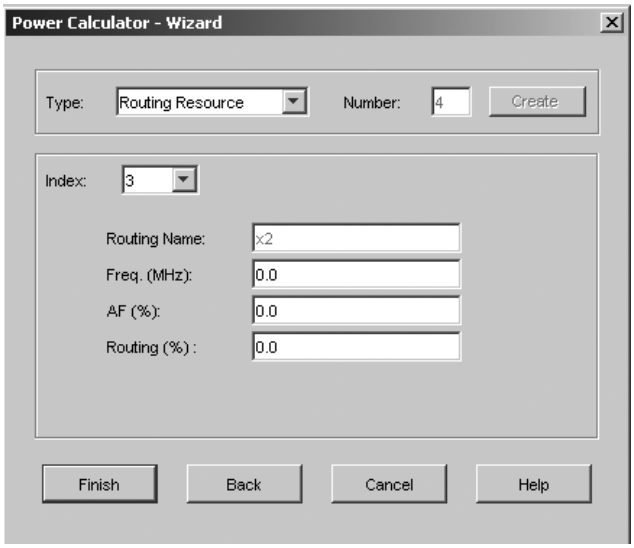


Figure 12-12. Power Calculator Wizard Mode Window - Resource Specification - I/Os

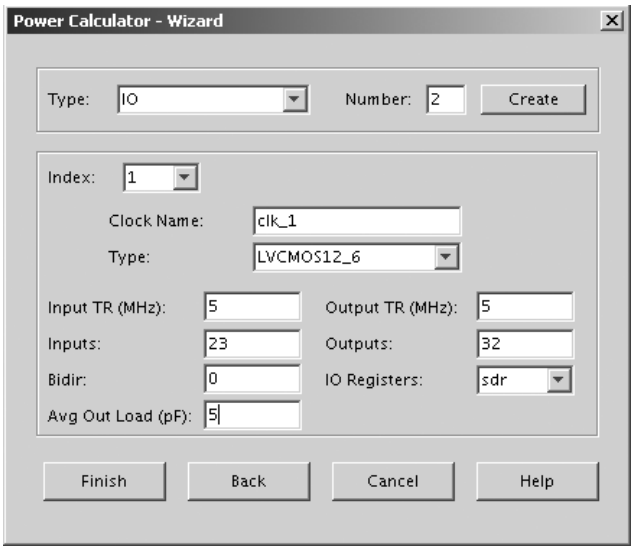
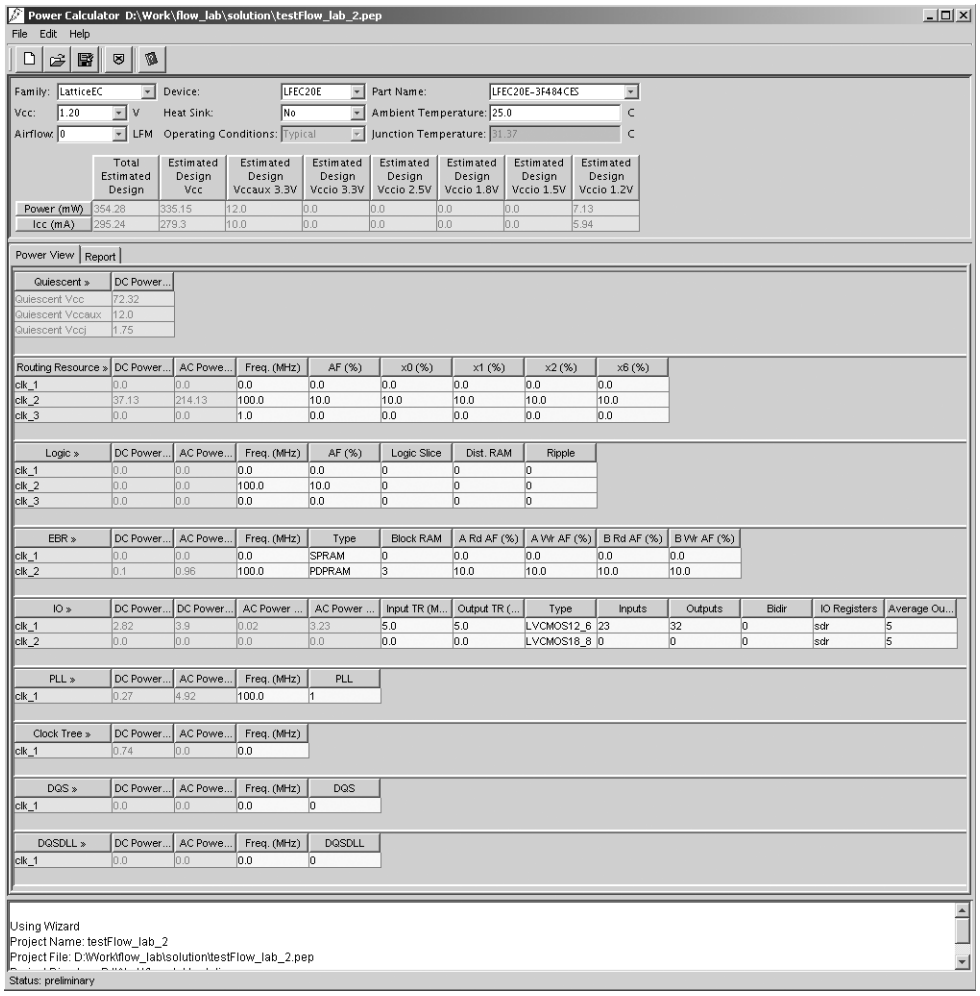


Figure 12-13. Power Calculator Wizard Mode - Main Window



Power Calculator – Creating a New Project Without the NCD File

A new project can be started without the NCD file by either using the Wizard (as discussed above) or by selecting the **Create a New Project** option in the **Power Calculator – Start Project**. A project name and project directory must be provided. After clicking **Continue**, the Power Calculator main window will be displayed.

However, in this case there are no resources added. The power estimation row for the Routing resources is always available in the Power Calculator. Users are then asked to add more information like the slice, EBR, I/O, PLL and clock tree utilization to calculate the power consumption.

For example, to add logic resources (as shown in Figure 12-14), right-click on **Logic >>** and then select **Add** in the menu that pops up.

Figure 12-14. Power Calculator Main Window – Adding Resources

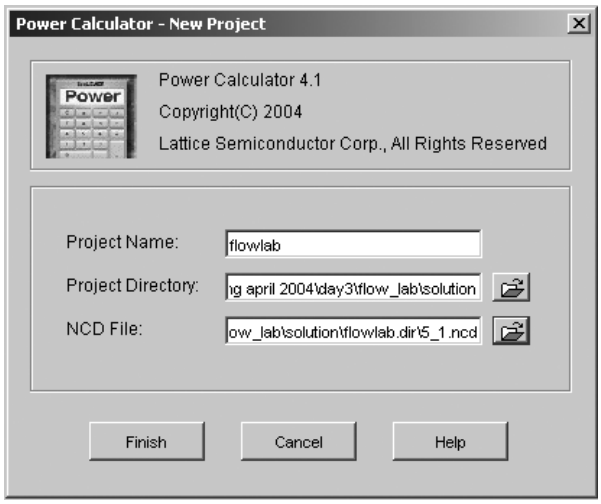
This adds a new row for the logic resource utilization with clock domain as clk_1.

Similarly, other resources like EBR, I/Os, PLLs and routing can be added. Each of these resources is for AC power estimation and categorized by clock domains.

Power Calculator – Creating a New Project With the NCD File

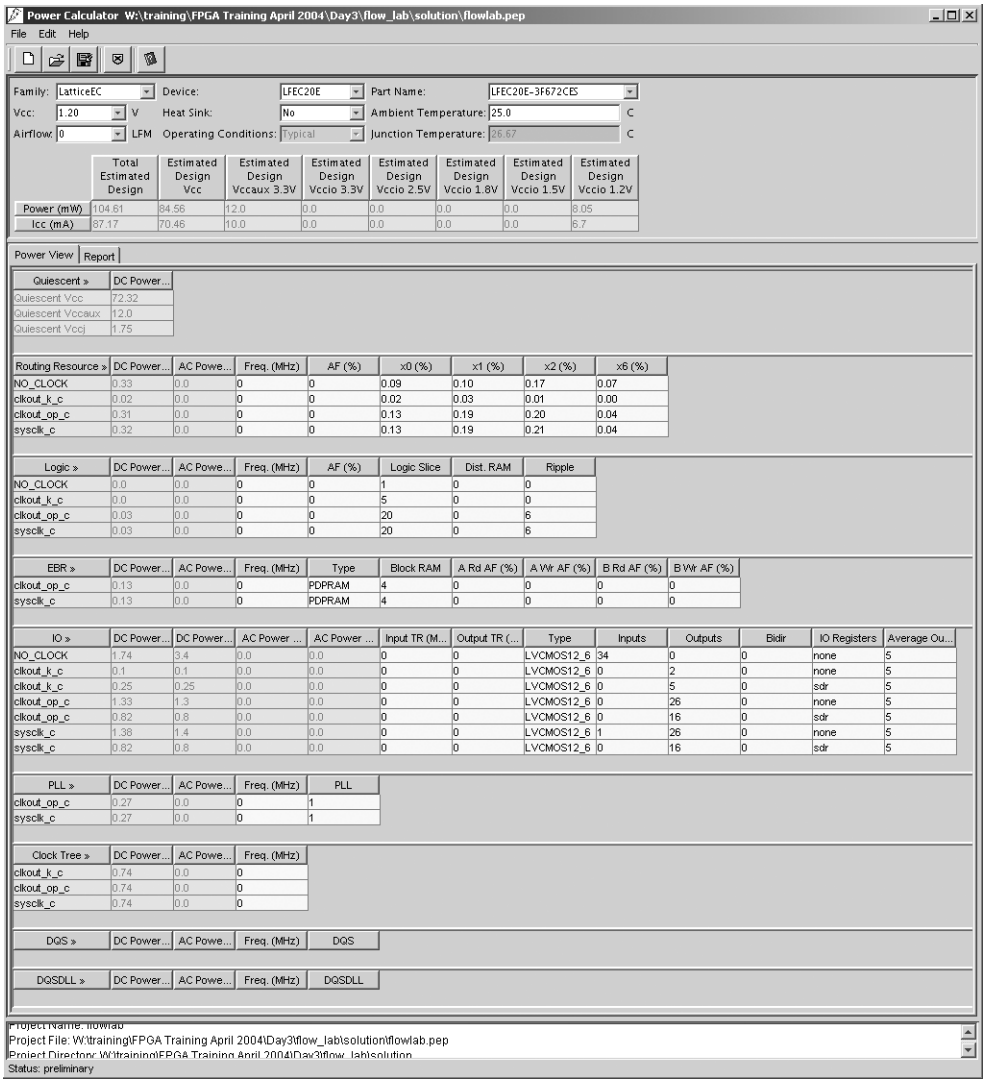
If the post place and routed NCD file is available, the Power Calculator can use it to import the accurate information about the design data and resource utilization and calculate the power. When the Power Calculator is started, the NCD file is automatically placed in the NCD File option, if available in the project directory. Otherwise, the user can browse to the NCD file in the Power Calculator.

Figure 12-15. Power Calculator Start Project Window – With Post Place and Route NCD File



The information from the NCD file is automatically inserted into the correct rows and the Power Calculator uses the Clock names from the design, as shown in Figure 12-16.

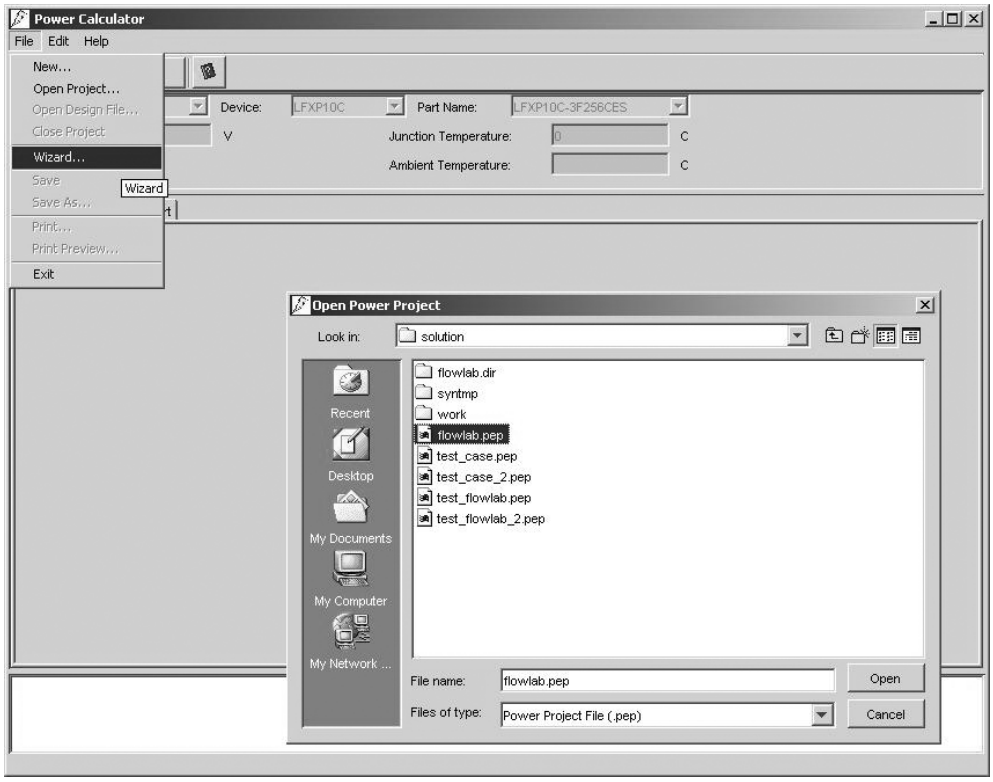
Figure 12-16. Power Calculator Main Window – Resource Utilization Picked Up From the NCD File



Power Calculator – Open Existing Project

The Power Calculator – Start Project window also allows users to open an existing project. Select the option **Open Existing Project** and browse to the *.pep project file and click **Continue**. This opens the existing project in similar windows as discussed above. This is shown in Figure 12-17.

Figure 12-17. Opening Existing Project in Power Calculator

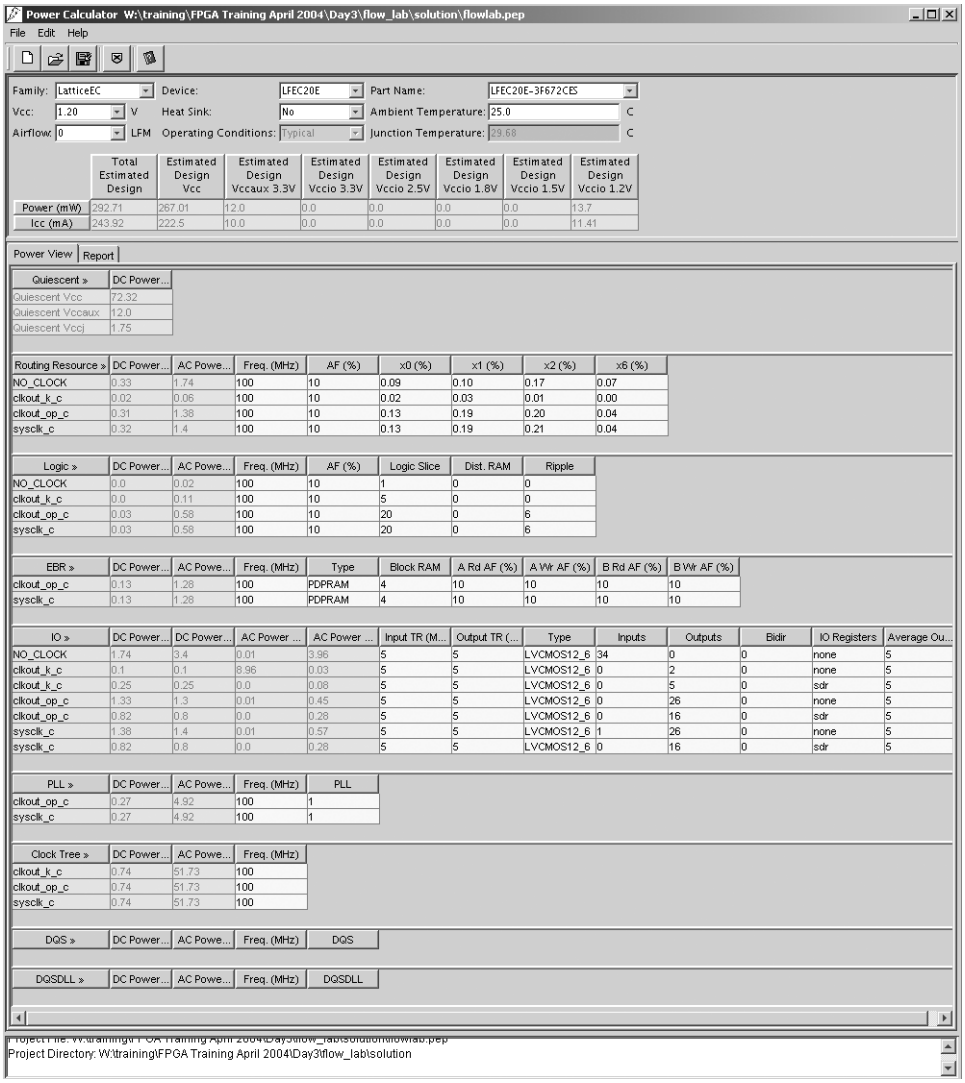


Power Calculator – Total Power

The Power Calculator project created or opened using any of the methods discussed here would allow a user to calculate the power consumption for the device running with their design.

The estimated power is indicated in the Total section at the bottom of the table as shown in Figure 12-18.

Figure 12-18. Calculated Power in the Power Calculator Main Window



The second and third columns from the left indicate the DC (or static) and AC (or dynamic) power consumption. The total power consumption for the design can be seen in the same table. Scroll down to the row labeled **Total**.

Activity Factor

Activity Factor % (or AF%) is defined as the percentage of frequency (or time) that a signal is active or toggling of the output.

Most of the resources associated with a clock domain are running or toggling at some percentage of the frequency at which the clock is running. Users are required to provide this value as a percentage under the AF% column in the Power Calculator tool.

Another term used for I/Os is the I/O Toggle Rate or the I/O Toggle Frequency. The AF% is applicable to the PFU, Routing and Memory Read Write Ports, etc. The activity of I/Os is determined by the signals provided by the user (in the case of inputs) or as an output of the design (in the case of outputs). So, the rates at which I/Os toggle define their activity. The Toggle Rate (or TR) in MHz of the output is defined as:

Toggle Rate (MHz) = 1/2 * f_{MAX} * AF%

Users are required to provide the TR (MHz) value for the I/O instead of providing the Frequency and AF% in case of other resources.

The AF can be calculated for each routing resource, output or PFU, however it involves long calculations. The general recommendation of a design occupying roughly 30% to 70% of the device is that the AF% used can be between 15% to 25%. This is an average value that can be seen most of the design. The accurate value of an AF depends upon clock frequency, stimulus to the design and the final output.

Ambient and Junction Temperature and Airflow

A common method of characterizing a packaged device's thermal performance is with thermal resistance, Θ. For a semiconductor device, thermal resistance indicates the steady state temperature rise of the die junction above a given reference for each watt of power (heat) dissipated at the die surface. Its units are °C/W.

The most common examples are Θ_{JA}, thermal resistance junction-to-ambient (in °C/W) and Θ_{JC}, thermal resistance junction-to-case (also in °C/W). Another factor is Θ_{JB}, thermal resistance junction-to-board (in °C/W).

Knowing the reference (i.e. ambient, case or board) temperature, the power, and the relevant Θ value, the junction temperature can be calculated as per following equations.

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

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.

The Power Calculator utilizes the 25°C junction temperature as its basis to calculate power, per Equation 1 above. Users can also provide the airflow values (in LFM) and ambient temperature to get a calculated value of the junction temperature based on the power estimate.

Managing Power Consumption

One of the most critical design factors today is reducing system power consumption, especially for modern handheld devices and electronics. There are several design techniques that designers can use to significantly reduce overall system power consumption. Some of these include:

1. Reducing operating voltage.
2. Operating within the specified package temperature limitations.
3. Using optimum clock frequency reduces power consumption, as the dynamic power is directly proportional to the frequency of operation. Designers must determine if a portion of their design can be clocked at a lower rate that will reduce power.
4. Reducing the span of the design across the device. A more closely placed design utilizes fewer routing resources for less power consumption.

5. Reducing the voltage swing of the I/Os where possible.

6. Using optimum encoding where possible. For example, a 16-bit binary counter has, on average, only 12% Activity Factor and a 7-bit binary counter has an average of 28% Activity Factor. On the other hand, a 7-bit Linear Feedback Shift Register could toggle as much as 50% Activity Factor, which causes higher power consumption. A gray code counter, where only one bit changes at each clock edge, will use the least amount of power, as the Activity Factor would be less than 10%.

7. Minimize the operating temperature, by the following methods:
   a. Use packages that can better dissipate heat. For example, packages with lower thermal impedance.
   b. Place heat sinks and thermal planes around the device on the PCB.
   c. Better airflow techniques using mechanical airflow guides and fans (both system fans and device mounted fans).

Power Calculator Assumptions

Following are the assumptions made in the Power Calculator:

1. The Power Calculator tool is based on equations with constants based on room temperature of 25°C.

2. The user can define the Ambient Temperature (Ta) for device Junction Temperature (Tj) calculation based on the power estimation. Tj is calculated from user-entered Ta and power calculation of typical room temperature.

3. The I/O power consumption is based on output loading of 5pF. Users have ability to change this capacitive loading.

4. The current version of the Power Calculator allows users to get an estimate of the power dissipation and the current for each type of power supplies, that are $V_{CC,}$ $V_{CCIO,}$ V_{CCJ} and $V_{CCAUX.}$ For $V_{CCAUX,}$ only static I_{CCAUX} values are provided in the Calculator.

   Additional V_{CCAUX} contributions due to differential output buffers, differential input buffers and reference input buffers must be added per pair for differential buffers or per pin for reference input buffers according to the user's design. See the equation given in this technical note for Total DC Power (I_{CCAUX}).

5. The nominal V_{CC} is used by default to calculate the power consumption. Users can choose a lower or higher V_{CC} from a list of available values. For example, the nominal V_{CC} of 1.2V is used by default for the LatticeECP/EC and LatticeXP families of devices.

6. The current versions also allows users to enter an airflow in Linear Feet per Minute (LFM) along with the Heat Sink option to calculate the Junction Temperature.

7. The default value of the I/O types for the LatticeEC and LatticeXP devices is LVCMOS12, 6mA.

8. The Activity Factor (AF) is defined as the toggle rate of the registered output. For example, assuming that the input of a flip-flop is changing at every clock cycle, 100% AF of a flip-flop running at 100MHz is 50MHz.

Revision History

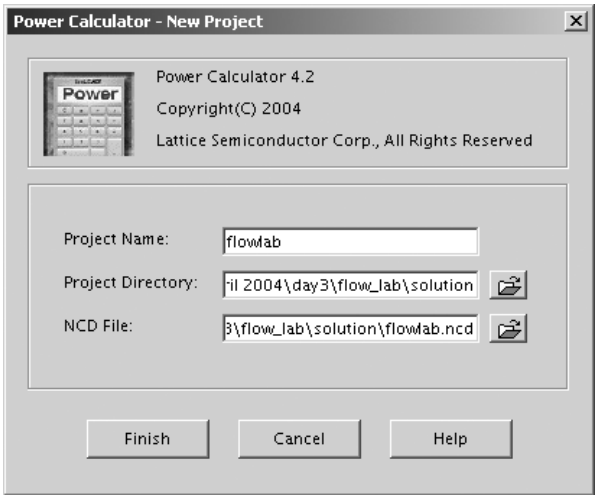
Date	Version	Change Summary
June 2004	01.0	Initial release.
July 2004	01.1	Provided additional description of the assumptions used in the power model.
October 2004	01.2	Updated screen shots for ispLEVER 4.1.
February 2005	02.0	Added support for LatticeXP family throughout.
		Added DC and AC power for a dedicated block like DSP for LatticeECP devices.
May 2005	02.1	Updated the Power Supply Sequencing and Hot Socketing section.
		Updated the total DC Power consumption to be sum of Quiescent and the DC power of resources.
November 2006	02.2	Added calculation of I_{CCAUX} in Power Calculator Equations section.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Appendix A. Power Calculator Project Example

This example assumes that you have post Place and Route NCD netlist in the design folder. Click on File > New or click on the New Project button. The New Project Window will open as shown below.



The various fields are filled in automatically with the project name the same as the ispLEVER project name and the directory also the same as the design folder. If the Post Place and Route NCD file is available, the NCD File field is also automatically filled. Users can also browse to the particular location to change the folder where they wish to create the Power Calculator project. Users can also browse to the NCD file in case it is not available at the root design folder.

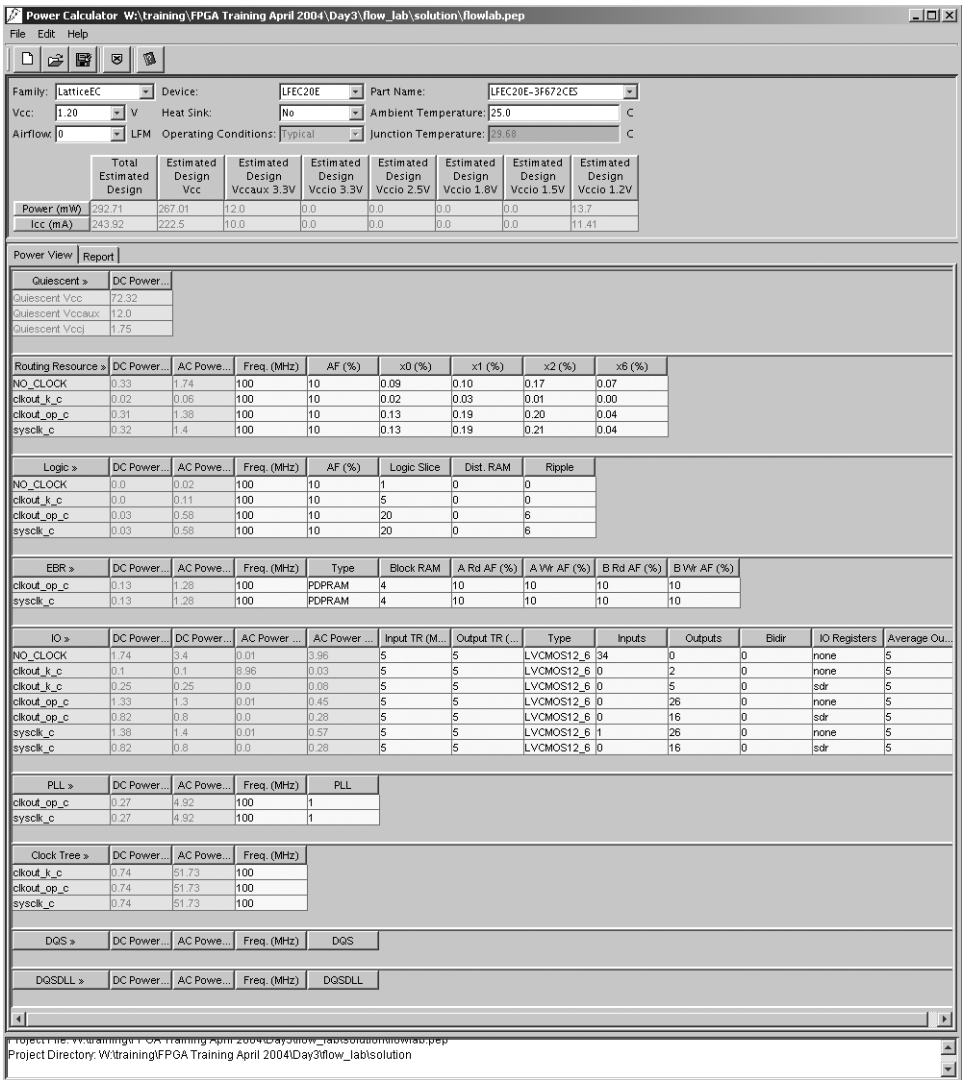
Click Finish. This opens the Main Power Calculator project window, as shown below.

Note that the project window above imports all the resource utilization information from the NCD file. It does not, however, include information such as the frequency at which the design is operating or the activity factors at which the various components are toggling. This information is to be filled in by the user.

The top portion of the Power Calculator window shows information such as the device family and device being considered for power calculation, the $V_{CC,}$ which is by default the nominal V_{CC} for the device, and operating conditions. Operating conditions users can enter include the ambient temperature, and heat sink available. Users can also select the air flow values.

There is a grayed box for junction temperature that shows Tj based on the given conditions and the calculated power.

If we assume the design is running at 100 MHz with a 10% Activity Factor, the final Power Calculator will be as shown below.


Power Calculator W:\training\FPGA Training April 2004\Day3\flow_lab\solution\flowlab.pep
File Edit Help
Family: LatticeEC
Device: LFEC20E
Part Name: LFEC20E-3F672CES
Vcc: 1.20 V
Heat Sink: No
Ambient Temperature: 25.0 C
Airflow: 0 LFM
Operating Conditions: Typical
Junction Temperature: 29.68 C
Total Estimated Design
Estimated Design Vcc
Estimated Design Vccaux 3.3V
Estimated Design Vccio 3.3V
Estimated Design Vccio 2.5V
Estimated Design Vccio 1.8V
Estimated Design Vccio 1.5V
Estimated Design Vccio 1.2V
Power (mW) 292.71 267.01 12.0 0.0 0.0 0.0 0.0 13.7
Icc (mA) 243.92 222.5 10.0 0.0 0.0 0.0 0.0 11.41
Power View
Report
Quiescent » DC Power...
Quiescent Vcc 72.32
Quiescent Vccaux 12.0
Quiescent Vccj 1.75
Routing Resource » DC Power... AC Powe... Freq. (MHz) AF (%) x0 (%) x1 (%) x2 (%) x6 (%)
NO_CLOCK 0.33 1.74 100 10 0.09 0.10 0.17 0.07
clkout_k_c 0.02 0.06 100 10 0.02 0.03 0.01 0.00
clkout_op_c 0.31 1.38 100 10 0.13 0.19 0.20 0.04
sysclk_c 0.32 1.4 100 10 0.13 0.19 0.21 0.04
Logic » DC Power... AC Powe... Freq. (MHz) AF (%) Logic Slice Dist. RAM Ripple
NO_CLOCK 0.0 0.02 100 10 1 0 0
clkout_k_c 0.0 0.11 100 10 5 0 0
clkout_op_c 0.03 0.58 100 10 20 0 6
sysclk_c 0.03 0.58 100 10 20 0 6
EBR » DC Power... AC Powe... Freq. (MHz) Type Block RAM A Rd AF (%) A Wr AF (%) B Rd AF (%) B Wr AF (%)
clkout_op_c 0.13 1.28 100 PDPRAM 4 10 10 10 10
sysclk_c 0.13 1.28 100 PDPRAM 4 10 10 10 10
IO » DC Power... DC Power... AC Power ... AC Power ... Input TR (M... Output TR (... Type Inputs Outputs Bidir IO Registers Average Ou...
NO_CLOCK 1.74 3.4 0.01 3.96 5 5 LVCMOS12_6 34 0 0 none 5
clkout_k_c 0.1 0.1 8.96 0.03 5 5 LVCMOS12_6 0 2 0 none 5
clkout_k_c 0.25 0.25 0.0 0.08 5 5 LVCMOS12_6 0 5 0 sdr 5
clkout_op_c 1.33 1.3 0.01 0.45 5 5 LVCMOS12_6 0 26 0 none 5
clkout_op_c 0.82 0.8 0.0 0.28 5 5 LVCMOS12_6 0 16 0 sdr 5
sysclk_c 1.38 1.4 0.01 0.57 5 5 LVCMOS12_6 1 26 0 none 5
sysclk_c 0.82 0.8 0.0 0.28 5 5 LVCMOS12_6 0 16 0 sdr 5
PLL » DC Power... AC Powe... Freq. (MHz) PLL
clkout_op_c 0.27 4.92 100 1
sysclk_c 0.27 4.92 100 1
Clock Tree » DC Power... AC Powe... Freq. (MHz)
clkout_k_c 0.74 51.73 100
clkout_op_c 0.74 51.73 100
sysclk_c 0.74 51.73 100
DQS » DC Power... AC Powe... Freq. (MHz) DQS
DQSDLL » DC Power... AC Powe... Freq. (MHz) DQSDLL
Project Directory: W:\training\FPGA Training April 2004\Day3\flow_lab\solution

LatticeXP sysCONFIG Usage Guide

February 2006 Technical Note TN1082

Introduction

The memory in the LatticeXP™ FPGAs is built using Flash cells, along with SRAM cells, so that configuration memory can be loaded automatically at power-up, or at any time the user wishes to update the device. In addition to "instant-on" capability, on-chip Flash memory greatly increases design security by getting rid of the external configuration bitstream; while maintaining the ease of use and reprogrammability of an SRAM-based FPGA.

While an external device is not required, the LatticeXP does support several external configuration modes. The available external configuration modes are:

- Slave Serial
- Master Serial
- Slave Parallel
- ispJTAG™ (1149.1 interface)

This guide will cover all the configuration options available for the LatticeXP.

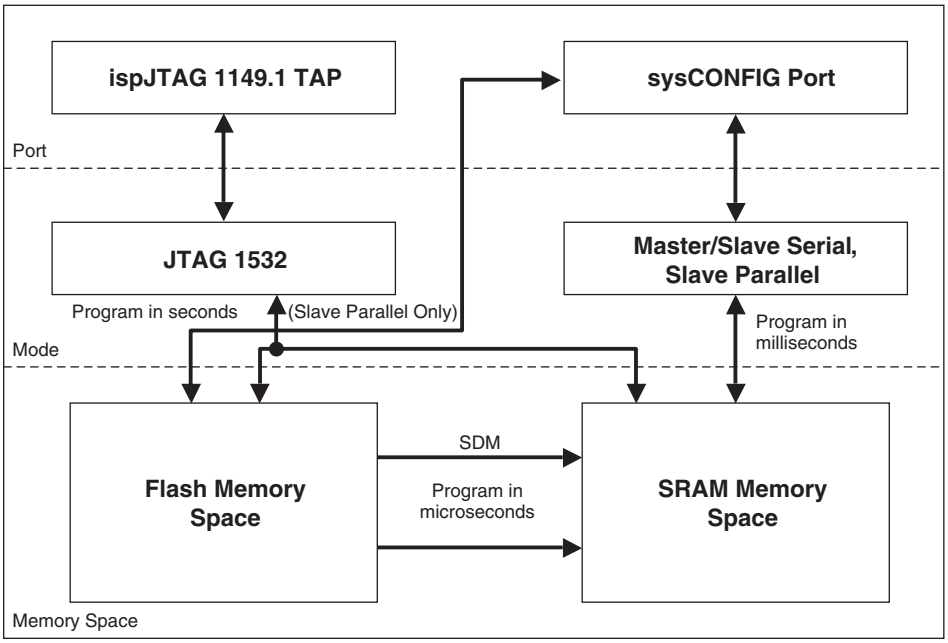
Programming Overview

The LatticeXP contains two types of memory, SRAM and Flash (refer to Figure 13-1). SRAM contains the FPGA configuration, essentially the "fuses" that define the circuit connections; Flash provides an internal storage space for the configuration data.

The SRAM can be configured using JTAG, one of the external configuration modes, or by using the data stored in on-chip Flash. The configuration process consists of SRAM initialization (clear the RAM and the address pointers), loading the SRAM with the configuration data, and setting the FPGA into user mode (waking up the FPGA).

On-chip Flash can be programmed by using JTAG or by using the external Slave Parallel port. JTAG Flash programming can be performed any time the device is powered up. The Slave Parallel port uses the sysCONFIG™ pins and can program the Flash directly or in the background. Direct programming takes place during config mode, background programming during user mode. The FPGA enters config mode at power up, when the PROGRAMN pin is pulled low, or when a refresh command is issued via JTAG; it enters user mode when it wakes up, i.e. when the device begins running user code. These two programming modes, direct and background, will be referred to in this document as Flash Direct and Flash Background.

Figure 13-1. Programming Block Diagram



Configuration Pins

The LatticeXP supports 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.

Two configuration mode pins, along with a programmable option, control the dual-purpose configuration pins. The configuration mode pins (CFG) are generally hard wired on the PCB and determine which configuration mode will be used; the programmable option is accessed via preferences in Lattice ispLEVER® design software, or as HDL source file attributes, and allows the user to protect the configuration pins from accidental use by the user or the place-and-route software. The LatticeXP devices also support ispJTAG for configuration, including transparent readback, and for JTAG testing. The following sections describe the functionality of the sysCONFIG and JTAG pins. Note that JTAG and ispJTAG will be used interchangeably in this document. Table 13-1 is provided for reference.

Table 13-1. Configuration Pins for the LatticeXP Device

Pin Name	I/O Type	Pin Type	Mode Used
CFG[1:0]	Input, weak pull-up	Dedicated	All
PROGRAMN	Input, weak pull-up	Dedicated	All
INITN	Bi-Directional Open Drain, weak pull-up	Dedicated	All
DONE	Bi-Directional Open Drain with weak pull-up or Active Drive	Dedicated	All
CCLK	Input or Output	Dedicated	All
DIN	Input, weak pull-up	Dual-Purpose	Serial
DOUT/CSON	Output	Dual-Purpose	Serial or Parallel
CSN	Input, weak pull-up	Dual-Purpose	Parallel
CS1N	Input, weak pull-up	Dual-Purpose	Parallel
WRITEN	Input, weak pull-up	Dual-Purpose	Parallel
BUSY	Output, tri-state, weak pull-up	Dual-Purpose	Parallel
D[0:7]	Input or Output	Dual-Purpose	Parallel
TDI	Input, weak pull-up	JTAG	

Table 13-1. Configuration Pins for the LatticeXP Device (Continued)

Pin Name	I/O Type	Pin Type	Mode Used
TDO	Output	JTAG	
TCK	Input with Hysteresis	JTAG	
TMS	Input, weak pull-up	JTAG	

Note: Weak pull-ups consist of a current source of 30uA to 150uA. The pull-ups for CFG and PROGRAMN track V_{CC} (core); the pull-ups for TDI and TMS track V_{CCJ}; all other pull-ups track the V_{CCIO} for that pin.

Dedicated Pins

Following is a description of the dedicated sysCONFIG pins for the LatticeXP device. These pins are used to control or monitor the configuration process. These pins are used for non-JTAG programming sequences only. The JTAG pins will be explained later in the ispJTAG Pins section of this document.

CFG[1:0]

The Configuration Mode pins, CFG[1:0], are dedicated inputs with weak pull-ups. The CFG pins are used to select the configuration mode for the LatticeXP, i.e. what type of device the LatticeXP will configure from. At Power-On-Reset (POR), or when the PROGRAMN pin is driven low, and depending on the configuration mode selected, different groups of dual-purpose pins will be used for device configuration.

Table 13-2. LatticeXP Configuration Modes

Configuration Mode	CFG[1]	CFG[0]
Slave Serial	0	0
Master Serial	0	1
Slave Parallel	1	0
Self Download Mode (SDM)	1	1

When both CFG pins are high the device will configure itself by reading the data stored in on-chip Flash; this is referred to as SDM, or Self Download Mode. See the Self-Download section of this document for more information regarding SDM.

PROGRAMN

The PROGRAMN pin is a dedicated input with a weak pull-up. This pin is used to initiate a non-JTAG SRAM configuration sequence. A high to low signal applied to PROGRAMN sets the device into configuration mode. The PROGRAMN pin can be used to trigger configuration at any time. If the device is using JTAG then PROGRAMN will be ignored until the device is released from JTAG mode.

If the CFG pins are not both high (not in SDM) then the configuration sequence will proceed using the selected configuration port. If both CFG pins are high (SDM), and the Flash has been programmed, then the configuration sequence will proceed using the data in on-chip Flash.

If both CFG pins are high (SDM), and the Flash has not been programmed, the configuration sequence will pause and wait for the Flash done bit to be programmed. Once the Flash has been programmed, and PROGRAMN is brought high, the configuration sequence will continue.

INITN

The INITN pin is a dedicated bi-directional open drain pin with a weak pull-up. INITN is capable of driving a low pulse out as well as detecting a low pulse driven in.

During SRAM configuration from an external device INITN going low indicates that the SRAM is being initialized; INITN going high indicates that the FPGA is ready to accept configuration data. 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. After configuration has started INITN is used to indicate a bitstream error. The INITN pin will be driven low if the cal-

culated CRC and the configuration data CRC do not match; DONE will then remain low and the LatticeXP will not wake up.

During SRAM configuration from on-chip Flash INITN is not used or monitored and is driven low.

When programming on-chip Flash the INITN pin is only used to indicate an error during erase or program. If an error occurs INITN will be driven low. During Flash Direct programming an error will prevent the FPGA from configuring from the Flash, during Flash Background programming an error will not affect the configuration already running in SRAM.

DONE
The DONE pin is a dedicated bi-directional open drain with a weak pull-up (default), or an actively driven pin. DONE 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 go high (or in the case of SDM just PROGRAMN goes high), and the internal DONE bit is programmed, the DONE pin will be released (or driven high, if it is an actively driven pin). The DONE pin can be held low externally and, depending on the wake-up sequence selected, the device will not become functional until the DONE pin is externally brought high.

Reading the DONE bit is a good way for an external device to tell if the FPGA is configured.

When using JTAG to configure SRAM the DONE pin is driven by the boundary scan cell, so the state of the DONE pin has no meaning during JTAG configuration.

CCLK
CCLK is a dedicated bi-directional pin; direction depends on whether a Master or Slave mode is selected. If a Master mode is selected via the CFG pins, the CCLK pin will become an output pin; otherwise CCLK is an input pin.

If the CCLK pin becomes an output, the internal programmable oscillator is connected to the CCLK and is driven out to slave devices. CCLK will stop 100 to 500 clock cycles after the DONE pin is brought high and the device wake-up sequence completed. The extra clock cycles ensure that enough clocks are provided to wake-up other devices in the chain. When stopped, CCLK becomes an input (tri-stated output). CCLK will restart (become an output) on the next configuration initialization sequence.

The MCCLK_FREQ parameter (see ispLEVER software documentation) controls the CCLK master frequency (see Table 13-3). Until changed during configuration CCLK will be 2.5 MHz. One of the first things loaded during configuration is the MCCLK_FREQ parameter; once this parameter is loaded the frequency changes to the selected value using a glitchless switch. Care should be exercised not to exceed the frequency specification of the slave devices or the signal integrity capabilities of the PCB layout.

Table 13-3. Master Clock Frequency Selections

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	-

Note: Default is the lowest frequency, 2.5 MHz.

Table 13-4. Maximum Configuration Bits

Density	Bitstream Size (Mb)
LFXP3	1
LFXP6	1.6
LFXP10	2.8
LFXP15	4
LFXP20	4.9

Table 13-5. SDM Pin Usage

Configuration Mode	SDM (Self Download Mode)			
CFG[1:0]	[1, 1]			
Flash Programming Mode	Direct	Background	Direct	Background
Port	sysCONFIG		ispJTAG[1]	
Pins	CCLK, CSN, CS1N, WRITEN, D[0:7]		TAP	
User I/O States	Tristate	User	BSCAN	User
PROGRAMN	↓	Keep at High	Keep At High[2]	
BUSY	Status	Status	Not Used	
INITN	Pass/Fail	Pass/Fail	Not Used[3]	
DONE	Done	Not Used	Keep at High[4]	
Persistent Bit	Don't Care	ON	Don't Care	

1. ispJTAG can be used to program the Flash regardless of the state of the CFG pins, however only if the device is in SDM can Flash be used to configure SRAM
2. The state of the PROGRAMN pin is ignored by the device during JTAG Flash programming but the pin should be held high as a low will inhibit Flash to SRAM data transfer.
3. The state of the INITN pin is ignored by the device during JTAG Flash programming but the pin should be allowed to float high using the internal pull-up.
4. The state of the DONE pin is ignored by the device during JTAG Flash programming but the pin should be allowed to float high using the internal pull-up as a low can keep the device from waking up.

Table 13-6. Pins Used for Memory Access

CFG Pins			On-Chip Flash	SRAM	
1	0	CFG Mode	Write or Read[2]	Write From	Readback[2, 3]
X[1]	X[1]	JTAG	TAP	TAP	TAP
1	1	SDM	sysCONFIG	On-Chip Flash	sysCONFIG
1	0	Slave Parallel	N/A[4]	sysCONFIG	sysCONFIG
0	1	Master Serial	N/A[4]	sysCONFIG	N/A[5]
0	0	Slave Serial	N/A[4]	sysCONFIG	N/A[5]

1. The ispJTAG port is always available independent of the CFG setting.
2. Readback can only be disabled by programming the security bit.
3. Set the PERSISTENT bit to ON to retain the sysCONFIG port for background readback.
4. Flash access is not allowed in this mode.
5. SRAM readback is not allowed in this mode.

Programming Sequence

There are three types of programming, SRAM, Flash Direct, and Flash Background. This section goes through the process for each showing how the dedicated pins are used.

SRAM: When not using SDM (Self Download Mode, on-chip Flash) to program SRAM the sequence begins when the internal power-on reset (POR) is released or the PROGRAMN pin is driven low (see Figure 13-2). The LatticeXP then drives INITN low, tri-states the I/Os, and initializes the internal SRAM and control logic. When this is

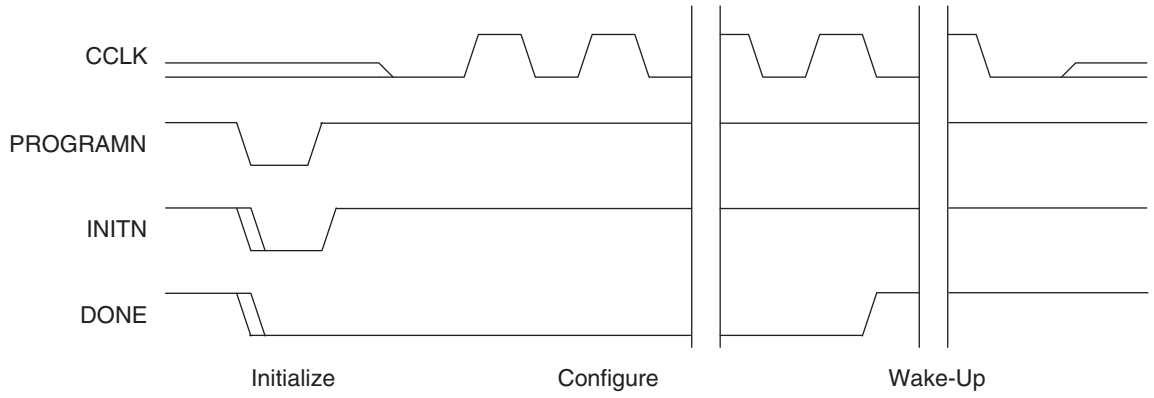
complete, if PROGRAMN is high, INITN will be released. If INITN is held low externally the LatticeXP will wait until it goes high. When INITN goes high the LatticeXP begins looking for the configuration data preamble on the selected configuration port, as determined by the CFG pins.

Once configuration is complete the internal DONE bit is set, the DONE pin goes high, and the FPGA wakes up (enters user mode). If a CRC error is detected when reading the bitstream INITN will go low, the internal DONE bit will not be set, the DONE pin will stay low, and the LatticeXP will not wake up.

When using SDM to program SRAM the sequence is similar but INITN is not used or monitored (INITN is driven low). The sequence begins when the internal power-on reset (POR) is released or the PROGRAMN pin is driven low (see Figure 13-2). The LatticeXP then tri-states the I/Os and initializes the internal SRAM and control logic. When initialization is complete the LatticeXP begins loading configuration data from on-chip Flash.

As with non-SDM, once configuration is complete the internal DONE bit is set, the DONE pin goes high, and the FPGA wakes up (enters user mode).

Figure 13-2. SRAM Configuration Timing Diagram



Flash Direct: Flash Direct programming is possible using the Slave Parallel port if both CFG pins are high (SDM). Serial ports may not be used to program the Flash. Flash Direct is only valid if the DONE pin is low (the SRAM is blank).

The sequence begins when the PROGRAMN pin is driven low. The LatticeXP tri-states the I/Os, and initializes the internal SRAM and control logic. The LatticeXP waits for WRITEN and both CSN and CS1N pins to go low and then looks for the programming preamble followed by the erase, program, and verify commands. Data is written and read on the D[0:7] pins.

Once the Flash is programmed the PROGRAMN pin can be brought high to start the transfer from Flash to SRAM.

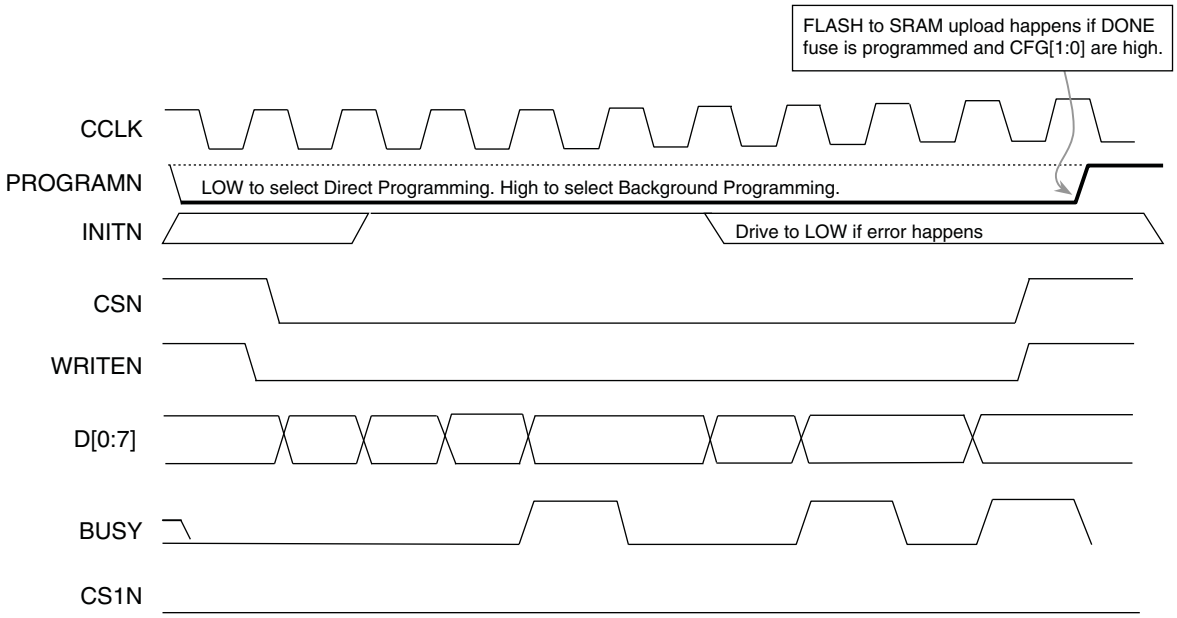
Flash Background: Flash Background programming is possible using the Slave Parallel port if both CFG pins are high (SDM). Serial ports may not be used to program the Flash. Flash Background will not disturb the FPGA's present configuration.

Flash Background programming may be used in both config mode and user mode (Done bit = 0 or 1). To support Flash Background programming in user mode the PERSISTENT bit must be set to ON.

When WRITEN goes low, and CSN and CS1N are low, the FPGA will wait for the preamble and then look for the proper commands. A low on INITN indicates an error during a Flash erase or program. Data is written and read on the D[0:7] pins.

After programming the Flash the user may toggle the PROGRAMN pin to transfer the Flash data to SRAM.

Figure 13-3. Flash Programming Timing Diagram



Dual-Purpose sysCONFIG Pins

The following is a list of the dual-purpose sysCONFIG pins. These pins are available as general purpose I/O after configuration. If a dual-purpose pin is to be used both for configuration and as a general purpose I/O the user must adhere to the following:

- The general purpose I/O (GPIO) must maintain the same direction as it has during configuration. In other words, if the pin is an input during configuration it must remain an input as a GPIO. If it's an output during configuration it must remain an output as a GPIO. If it's a bi-directional it must remain a bi-directional as a GPIO.
- The I/O type must remain the same. In other words, if the pin is a 3.3V CMOS pin (LVCMOS33) during configuration it must remain a 3.3V CMOS pin as a GPIO.
- The Persistent option must be set to OFF. The Persistent option can be accessed by using the Preference Editor in ispLEVER.
- The user is responsible for insuring that no internal or external logic will interfere with device configuration.

After configuration these pins, if not used as GPIO, are tri-stated and weakly pulled up.

DIN
DIN (data input) is a dual-purpose input with a weak pull-up. DIN is used for the serial bitstream configurations.

DOUT/CSON
The DOUT/CSON is a dual-purpose output that is used in Chain Mode (daisy chaining). This pin can be used in serial or parallel modes and has two uses.

For serial and parallel configuration modes, when BYPASS Chain Mode is selected, this pin will become DOUT. In a serial configuration mode, when the device becomes fully configured, a BYPASS instruction will be executed and the data on DIN will be presented on the DOUT pin through a bypass register. In this way the serial data is passed to the next device. In parallel configuration mode the data will be serialized and then presented on DOUT; D[0] (MSb) will be shifted out first followed by D[1], D[2] and so on to D[7] (LSb).

For parallel configuration mode, when FLOW_THROUGH Chain Mode is selected, this pin will become Chip Select Out (CSON). In FLOW_THROUGH Chain Mode, when the device is fully configured (the internal DONE bit goes

high), the Flow-Through instruction will be executed and the CSON pin will be driven low to enable the next device's chip select pin.

The DOUT/CSON bypass register will drive out a high upon power up and continue to do so until the execution of the Bypass or Flow-Through instruction within the bitstream.

Chain Mode is not supported when configuring from internal Flash (SDM).

CSN and CS1N
Both CSN and CS1N are active low input pins with weak pull-ups and are used in parallel mode only. These inputs are OR'ed and used to enable the D[0:7] data pins to receive or output a byte of data.

In non-SDM, when CSN or CS1N are high, the D[0:7], INITN, and BUSY pins are tri-stated. When both CSN and CS1N are driven to a high value, they will reset the Bypass/Flow-Through register. CSN and CS1N are interchangeable when controlling the D[0:7], INITN, and BUSY pins.

When SDM is selected and CSN or CS1N are high, the D[0:7], INITN, and BUSY pins are tri-stated. If the Flash has not been programmed a high on both CSN and CS1N will cause the LatticeXP to drive the INITN pin low to reset the internal FPGA configuration circuitry. The LatticeXP will then monitor D[0:7] waiting for the configuration preamble. CSN and CS1N are interchangeable when controlling the D[0:7], INITN, and BUSY pins.

If SRAM or Flash will need to be accessed while the device is in user mode (the DONE pin is high) then the PERSISTENT preference must be set to ON in order to preserve these pins as CSN and CS1N.

WRITEN
The WRITEN pin is an active low input with a weak pull-up and used for parallel mode only. The WRITEN pin is used to determine the direction of the data pins D[0:7]. The WRITEN pin must be driven low when a byte of data is to be clocked into the device and driven high when data is to be read from the device.

If SRAM or Flash will need to be accessed while the part is in user mode (the DONE pin is high) then the PERSISTENT preference must be set to ON in order to preserve this pin as WRITEN.

BUSY
In parallel mode the BUSY pin is a tri-stated output with a weak pull-up. The BUSY pin will be driven low by the LatticeXP device only when it is ready to receive a byte of data from the D[0:7] pins or a byte of data is ready for reading. The BUSY pin can be used to support asynchronous peripheral mode (handshaking). This pin is used to indicate that the LatticeXP needs extra time to execute a command.

If SRAM or Flash will need to be accessed while the part is in user mode (the DONE pin is high) then the PERSISTENT preference must be set to ON in order to preserve this pin as BUSY.

D[0:7]
The D[0:7] pins support slave parallel mode only. The D[0:7] pins are tri-statable bi-directional I/O pins used for data write and read. When the WRITEN signal is low, and the CSN and CS1N pins are low, the D[0:7] pins become data inputs. When the WRITEN signal is driven high, and the CSN and CS1N pins are low, the D[0:7] pins become data outputs. If either CSN or CS1N is high D[0:7] will be tri-state. D[0] is the most significant bit and D[7] is the least significant bit.

If SRAM or Flash will need to be accessed while the part is in user mode (the DONE pin is high) then the PERSISTENT preference must be set to ON in order to preserve these pins as D[0:7].

Care must be exercised during read back of EBR or PFU memory. It is up to the user to ensure that reading these RAMs will not cause data corruption, i.e. these RAMs may not be read while being accessed by user code.

ispJTAG Pins

The ispJTAG pins are standard IEEE 1149.1 TAP (Test Access Port) pins. The ispJTAG pins are dedicated pins and are always accessible when the LatticeXP device is powered up. When programming the SRAM via ispJTAG the

dedicated programming pins, such as DONE, cannot be used to determine programming progress. This is because the state of the boundary scan cell will drive the pin, per JTAG 1149.1, rather than normal internal logic.

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 instructions 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 the state of TMS, a transition will be made in the TAP controller state machine. An internal pull-up resistor on the TMS pin is provided. The internal resistor is pulled up to $V_{CCJ.}$

TCK
The test clock pin, TCK, provides the clock to run the TAP controller, which loads and unloads 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 the value is shown in the DC parameter table of the data sheet. The TCK pin does not have a pull-up. A pull-down on the PCB of 4.7 K is recommended to avoid inadvertent clocking of the TAP controller as V_{CC} ramps up.

Optional TRST
Test Reset, TRST, in not supported on the LatticeXP device.

VCCJ
JTAG V_{CC} (V_{CCJ}) supplies independent power to the JTAG port to allow chaining with other JTAG devices at a common voltage. V_{CCJ} must be connected even if JTAG is not used. This voltage may also power the JTAG download cable. Valid voltage levels are 3.3V, 2.5V, 1.8V, 1.5V, and 1.2V.

Please see *In-System Programming Design Guidelines for ispJTAG Devices,* available on the Lattice web site at www.latticesemi.com, for further JTAG chain information.

Configuration and JTAG Voltage Levels

All of the control pins and programming pins default to LVCMOS. CFG and PROGRAMN are linked to V_{CC} (core); TCK, TDI, TDO, and TMS track V_{CCJ}; all other pins track the V_{CCIO} for that pin.

Configuration Modes and Options

The LatticeXP device supports several configuration modes, utilizing serial or parallel data inputs, as well as self-configuration. On power up, or upon driving the PROGRAMN pin low, the CFG[1:0] pins are sampled to determine the mode that will be used to configure the LatticeXP device. The CFG pins are generally hard wired on the PCB and determine which port the device will use to retrieve its configuration data. CONFIG_MODE is a programmable option accessed via preferences in Lattice ispLEVER design software, or as HDL source file attributes, and allow the user to protect the configuration pins from accidental use by the user or the place-and-route software.

Table 13-7 shows the mode, CFG[1:0], and the software CONFIG_MODE parameter. The following sections breakdown each configuration mode.

Table 13-7. Configuration Modes for the LatticeXP

Configuration Mode	CFG[1]	CFG[0]	CONFIG_MODE[1]	Chain Mode[2]
Slave Serial (no overload option)	0	0	Slave_Serial	Disable
Slave Serial (Bypass ON)	0	0	Slave_Serial	Bypass
Master Serial (no overload option)	0	1	Master_Serial	Disable
Master Serial (Bypass ON)	0	1	Master_Serial	Bypass
Slave Parallel (no overload option)	1	0	Slave_Parallel	Disable
Slave Parallel (Bypass ON)	1	0	Slave_Parallel	Bypass
Slave Parallel (Flow Through ON)	1	0	Slave_Parallel	Flowthrough
Self Download Mode (SDM)	1	1	None/Slave_Parallel[4]	Disable
ispJTAG (1149.1 interface)	X[3]	X[3]	None[5]	

1. CONFIG_MODE can be found in the ispLEVER Preference Editor.
2. CHAIN_MODE can be found in the ispLEVER bitgen options (right-click on Generate Bitstream Data and click on Properties).
3. The ispJTAG interface is always on.
4. If ispJTAG is used exclusively to access the on-chip Flash and SRAM select None, if Slave Parallel is used to access the Flash and/or the SRAM select Slave_Parallel.
5. The None selection indicates that no dual-purpose pins are reserved for configuration. This is the default.

Configuration Options

Several configuration options are available for each CONFIG_MODE.

- When daisy chaining multiple FPGA devices a configuration overflow option is provided. Configuration data overflow occurs once the first FPGA has completed its download, the remaining data in the configuration storage device is then output through the first FPGA to subsequent FPGAs. Configuration data overflow is not supported when using SDM.
- When using a master clock, the master clock frequency can be set.
- A security bit is provided to prevent SRAM or Flash readback.

By setting the proper parameters in the Lattice ispLEVER design software the selected configuration options are set in the generated bitstream. As the bitstream is loaded into the device the selected configuration options take effect. These options are described in the following sections.

Bypass Overflow Option

The Bypass overflow option can be used in serial and parallel device daisy chains. When the first device has completed configuration data download, and the Bypass option preference is selected, data coming into the device configuration port on the sysCONFIG pins will overflow serially out of DOUT and into the DIN pin of the next slave serial device. The Bypass option is selected in ispLEVER by right-clicking on Generate Bitstream Data and clicking on Properties.

In serial configuration mode, once all of the configuration data has been loaded into the first device, the Bypass option connects the DIN pin to DOUT pin via a bypass register. The bypass register is initialized with a ‘1’ at the beginning of configuration.

In parallel configuration mode, once all of the configuration data has been loaded into the first device, the Bypass option causes the data coming from D[0:7] to be serially shifted to DOUT. The serialized data is shifted to DOUT through the bypass register. D[0] of the byte wide data will be shifted out first followed by D[1], D[2] and so on.

Once the Bypass option starts, the device will remain in Bypass until the wake up sequence completes. The Bypass option can be aborted by setting both CSN and CS1N high.

Flow-Though Overflow Option

The Flow-Through overflow option is used in parallel mode only. The Flow-Through option causes the CSON pin to go low when the FPGA has received all of its configuration data, driving the chip select on the next device in the

daisy chain so that it will start reading configuration data from D[0:7]. The Flow Through Option will also tri-state the D[0:7], INITN, and BUSY pins, once all of the configuration data has been received, in order to prevent interference with other devices in the daisy chain.

Once the Flow-Through option starts, the device will remain in Flow-Through until the wake up sequence completes. The Flow-Through option can be aborted by setting both CSN and CS1N high.

Master Clock
If the CFG pins indicate that this is a Master device the CCLK pin will become an output with the frequency set by the user. The default Master Clock Frequency is 2.5 MHz.

The user can determine the Master Clock frequency by setting the MCCLK_FREQ preference in the Lattice ispLEVER design software. One of the first things loaded during configuration is the MCCLK_FREQ parameter; once this parameter is loaded the frequency changes to the selected value using a glitchless switch. Care should be exercised not to exceed the frequency specification of the slave devices or the signal integrity capabilities of the PCB layout. See Table 13-3 for available options.

Security Bit
Setting the security bit prevents readback of the SRAM and Flash from JTAG or the sysCONFIG pins. When the security bit is set the only operations available are erase and write. The security bit is updated as the last operation of SRAM configuration or Flash programming. By using on-chip Flash, and setting the security bit, the user can create a very secure device.

The security bit is accessed via the Preference Editor in ispLEVER design software.

More information on device security can be found in the document *FPGA Design Security Issues: Using the ispXPGA Family of FPGAs to Achieve High Design Security,* available on the Lattice Semiconductor web site at www.latticesemi.com.

Slave Serial Mode

Configuration Mode	CFG[1]	CFG[0]	CONFIG_MODE	Chain Mode
Slave Serial (no overload option)	0	0	Slave_Serial	Disable
Slave Serial (Bypass ON)	0	0	Slave_Serial	Bypass

The CCLK pin becomes an input and data at DI is clocked 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 via the DOUT pin as shown in Figure 13-4.

Master Serial Mode

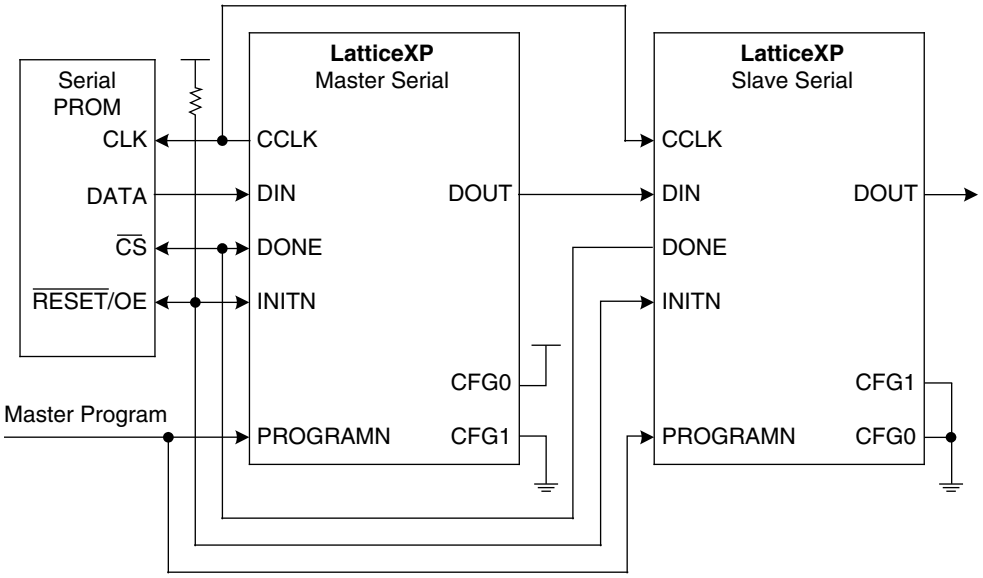
Configuration Mode	CFG[1]	CFG[0]	CONFIG_MODE	Chain Mode
Master Serial (no overload option)	0	1	Master_Serial	Disable
Master Serial (Bypass ON)	0	1	Master_Serial	Bypass

In Master Serial mode the device will drive CCLK out to the Slave Serial devices in the chain and the serial PROM that will provide the bitstream. The Master device accepts the data at DIN on the rising edge of CCLK. The Master Serial device starts driving CCLK at the beginning of the configuration and continues to drive CCLK until the external DONE pin is driven high and an additional 100 to 500 clock cycles have been generated. The CCLK frequency on power up defaults to 2.5 MHz. The master clock frequency default remains unless a new clock frequency is loaded from the bitstream.

If a Master Serial device is daisy chained with Slave Serial devices the Bypass option should be used so that overflow configuration data is directed to the DOUT pin.

Figure 13-4 shows a serial daisy chain. The daisy chain allows multiple Lattice FPGA devices to be configured using one configuration storage device. The center device operates in Master Serial with the Bypass option set while the other Lattice FPGA devices in the daisy chain operate in Slave Serial mode. The RESET/OE pin of the PROM is driven by INITN while the Chip Select pin is driven by the DONE pin of the devices.

Figure 13-4. Master and Slave Serial Daisy Chain



Slave Parallel Mode

Configuration Mode	CFG[1]	CFG[0]	CONFIG_MODE	Chain Mode
Slave Parallel (no overload option)	1	0	Slave_Parallel	Disable
Slave Parallel (Bypass ON)	1	0	Slave_Parallel	Bypass
Slave Parallel (Flow Through ON)	1	0	Slave_Parallel	Flowthrough

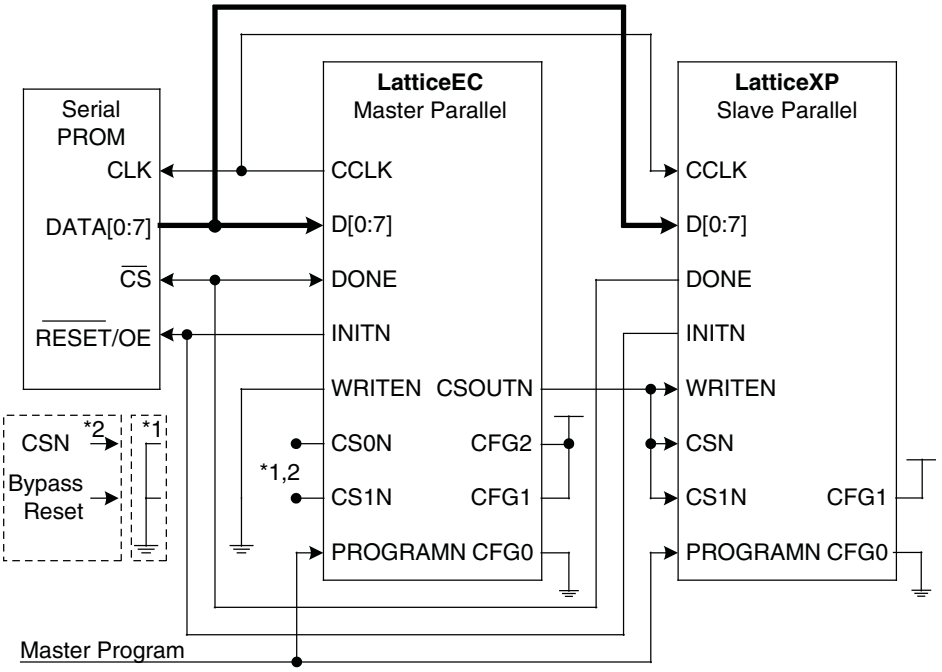
In Slave Parallel mode a host system sends the configuration data in a byte-wide stream to the device. The CCLK, CSN, CS1N, and WRITEN pins are driven by the host system. The Slave Parallel configuration mode allows multiple devices to be chained in parallel, as shown in Figure 13-5.

WRITEN, CSN, and CS1N must be held low to write to the device; data is input from D[0:7]. Slave Parallel mode can also be used for readback of the internal configuration. By driving the WRITEN pin low, and CSN and CS1N low, the device will input the readback instructions on the D[0:7] pins; WRITEN is then driven high and data read on D[0:7]. In order to support readback the PERSISTENT bit in ispLEVER's Preference Editor must be set to ON.

The Slave Parallel mode can support two types of overflow, Bypass and Flow-Through. If the Bypass option is set, after the first device has received all of its configuration data, the data presented to the D[0:7] pins will be serialized and bypassed to the DOUT pin. If the Flow-Through option is set, after the first device has received all of its configuration data, the CSON signal will drive the following parallel mode device's chip select low as shown in Figure 13-5.

To support asynchronous configuration, where the host may provide data faster than the FPGA can accept it, Slave Parallel mode can use the BUSY signal. By driving the BUSY signal high the Slave Parallel device tells the host to pause sending data. See Figure 13-6.

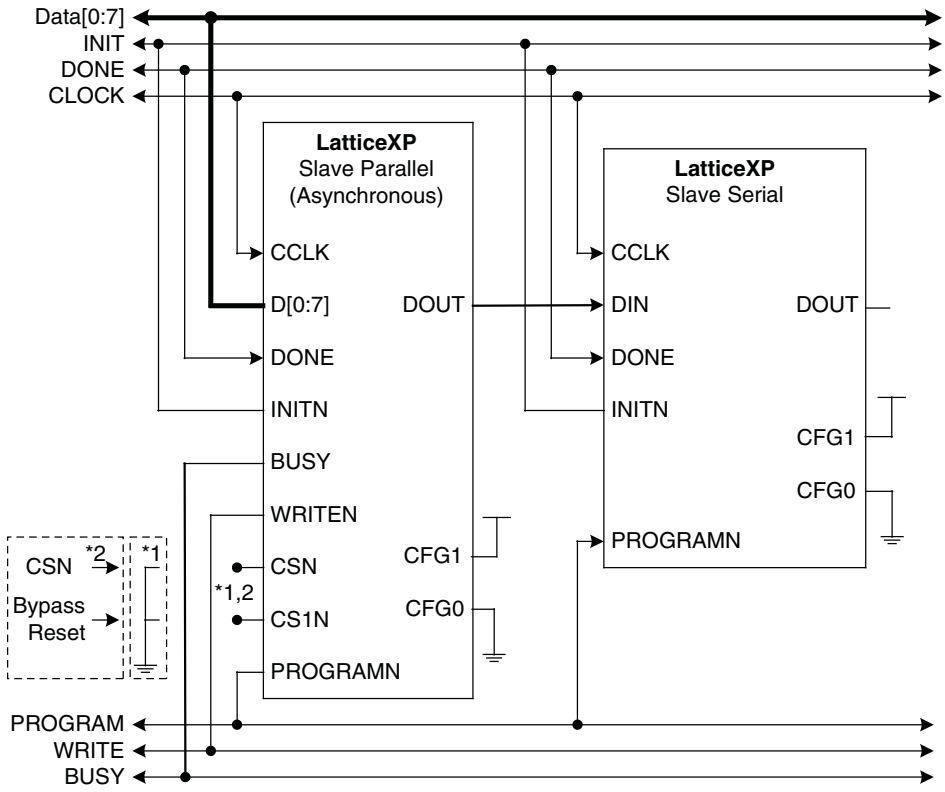
Figure 13-5. Master and Slave Parallel Daisy Chain



*1 Both CS pins can be held or driven low

*2 An option that allows the Bypass and Flow-Through option to be reset

Figure 13-6. Asynchronous Usage of Slave Parallel Configuration Mode



*1 Both CS pins can be held or driven low

*2 An option that allows the Bypass and Flow-Through option to be reset

Figure 13-6 shows an 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 LatticeXP device. When BUSY 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 signal can be used to temporarily stop the write process by setting either to a high state if the host system is busy. The LatticeXP device will resume configuration when the both CSN and CS1N signals are set low again.

If either overflow option is selected, both the CSN and CS1N pins can be set high to reset the Slave Parallel device out of the overflow option.

Self Download Mode

Configuration Mode	CFG[1]	CFG[0]	CONFIG_MODE	Chain Mode
Self Download Mode (SDM)	1	1	None/Slave_Parallel	Disable

Self Download Mode (SDM) allows the FPGA to configure itself without using any external devices, and because the bitstream is not exposed this is also a very secure configuration mode. The user may access on-chip Flash using ispJTAG or the slave parallel port on the sysCONFIG pins.

JTAG may access the on-chip Flash any time the device is powered up, without disturbing device operation. JTAG may also read and write the configuration SRAM. If access to the on-chip Flash and SRAM is limited to JTAG then CONFIG_MODE should be set to None, freeing the dual-purpose pins for use as general purpose I/O.

The slave parallel port can also be used to access on-chip Flash. If the slave parallel port is used then CONFIG_MODE should be set to Slave_Parallel. WRITEN, CSN, and CS1N must be held low to write to on-chip Flash; data is input from D[0:7]. The slave parallel port can also be used for readback of both Flash and SRAM. By driving the WRITEN pin low, and CSN and CS1N low, the device will input the readback instructions on the D[0:7] pins; a bit in the read command will determine if the read is directed to Flash or SRAM. In order to support readback while the device is in user mode (the DONE pin is high) the PERSISTENT bit in ispLEVER's Preference Editor must be set to ON.

SDM does not support overflow.

ispJTAG Mode

Configuration Mode	CFG[1]	CFG[0]	CONFIG_MODE	Chain Mode
ispJTAG (1149.1 interface)	X	X	None	

The LatticeXP device can be configured through the ispJTAG port. The ispJTAG port is always on and available, regardless of the configuration mode selected. A CONFIG_MODE of None can be selected in the Lattice ispLEVER design software to tell the place and route tools that the JTAG port will be used exclusively, i.e. the serial and parallel ports will not be used. Setting the CONFIG_MODE to None allows software to use all of the dual-purpose pins as general purpose I/Os.

ISC 1532

Configuration through the ispJTAG 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 value whenever using the JTAG 1532 mode. Because of this the dedicated pins, such as DONE, cannot be relied upon for valid configuration status.

Transparent Readback
The ispJTAG Transparent Readback mode allows the user to read the content of the device SRAM or Flash while the device remains in a functional state. Care must be exercised when reading EBR and distributed RAM, as it is possible to cause conflicts with accesses from the user design (causing possible data corruption).

The I/O and non-JTAG configuration pins remain active during a Transparent Readback. The device enters the Transparent Readback mode through a JTAG instruction.

Boundary Scan and BSDL Files
BSDL files for this device can be found on the Lattice web site at www.latticesemi.com. The boundary scan ring covers all of the I/O pins, as well as the dedicated and dual-purpose sysCONFIG pins.

Power Save Mode
An I/O Power Save mode option is available for the LatticeXP device and will deactivate portions of the I/O cell drivers. This is only valid when using comparator type inputs pins (pins that use VREF), like HSTL, SSTL, etc.

Power Save mode limits some of the functionality of Boundary Scan. For Boundary Scan testing it is recommended that the I/O Power Save mode be set to OFF so that all of the I/Os will be fully functional.

Wake Up Options

When configuration is complete (the SRAM has been loaded), 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 regardless of whether the DONE pin is held low externally or not, the other waits for the DONE pin to be driven high before starting the wake up process. The DONE_EX preference determines whether the external DONE pin will control the synchronous wake up.

Wake Up Sequence

Table 13-8 provides a list of the wake up sequences supported by the LatticeXP.

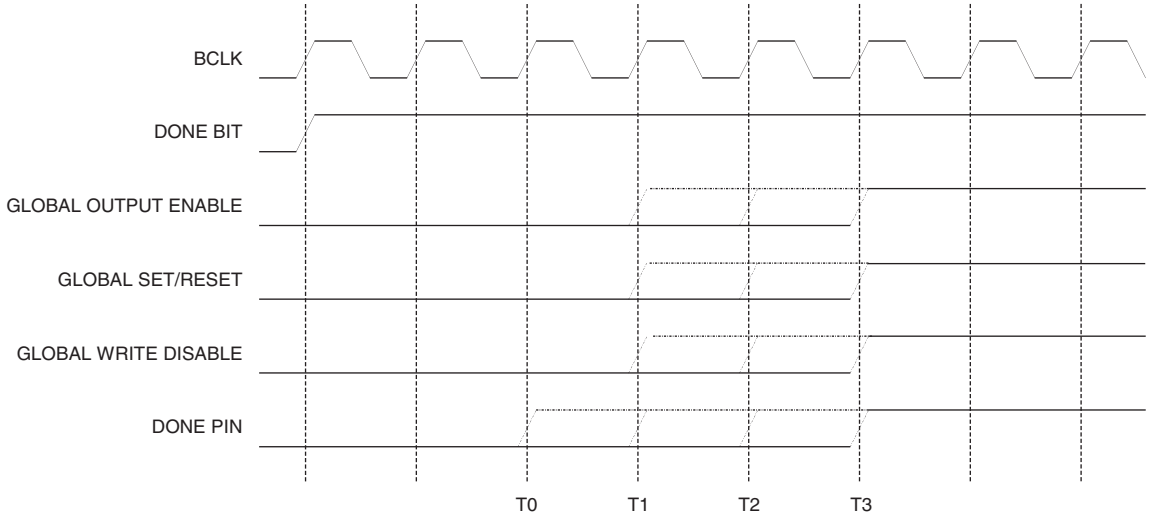
Table 13-8. Wake Up Sequences Supported by LatticeXP

Sequence	Phase T0	Phase T1	Phase T2	Phase T3
1	DONE	GOE, GWDIS, GSR		
2	DONE		GOE, GWDIS, GSR	
3	DONE			GOE, GWDIS, GSR
4	DONE	GOE	GWDIS, GSR	
5	DONE	GOE		GWDIS, GSR
6	DONE	GOE	GWDIS	GSR
7	DONE	GOE	GSR	GWDIS
8		DONE	GOE, GWDIS, GSR	
9		DONE		GOE, GWDIS, GSR
10		DONE	GWDIS, GSR	GOE
11		DONE	GOE	GWDIS, GSR
12			DONE	GOE, GWDIS, GSR
13		GOE, GWDIS, GSR	DONE	
14		GOE	DONE	GWDIS, GSR
15		GOE, GWDIS	DONE	GSR
16		GWDIS	DONE	GOE, GSR
17		GWDIS, GSR	DONE	GOE
18		GOE, GSR	DONE	GWDIS
19			GOE, GWDIS, GSR	DONE
20		GOE, GWDIS, GSR		DONE

Table 13-8. Wake Up Sequences Supported by LatticeXP (Continued)

Sequence	Phase T0	Phase T1	Phase T2	Phase T3
21 (Default)		GOE	GWDIS, GSR	DONE
22		GOE, GWDIS	GSR	DONE
23		GWDIS	GOE, GSR	DONE
24		GWDIS, GSR	GOE	DONE
25		GOE, GSR	GWDIS	DONE

Figure 13-7. Wake Up Sequence to Internal Clock



Synchronous to Internal Done Bit

If the LatticeXP 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 high. The device will then follow the WAKE_UP sequence selected.

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 CCLK, TCK, and User Clock. The Default can be either TCK or CCLK, depending on the programming/configuration method selected. The default clock will be TCK if using ispJTAG and CCLK if using sysCONFIG. The User Clock is chosen in the user's design. The user can select any of the CLK pins of the device, or a net (routing node), as the User Clock source. Figure 13-7 uses BCLK to represent the User Clock. The WAKEUP_CLK defaults to TCK or CCLK.

Software Selectable Options

In order to control the configuration of the LatticeXP device beyond the default settings, software preferences are used. Table 13-9 is a list of the preferences with their default settings.

Table 13-9. Software Preference List for the LatticeXP

Preference Name	Default Setting [List of All Settings]
PERSISTENT	OFF [off, on]
CONFIG_MODE	NONE[SLAVE_PARALLEL, SLAVE_SERIAL, MASTER_SERIAL, NONE]
DONE_OD	ON [off, on]
DONE_EX	OFF [off, on]
MCCLK_FREQ	Lowest Frequency (see Table 13-3)
CONFIG_SECURE	OFF [off, on]
WAKE_UP	21 (DONE_EX = off) 4 (DONE_EX = on)
WAKEUP_CLK	EXTERNAL [external, user]
PWRSAVE	OFF [off, on]

Persistent

In order to use the sysCONFIG port while in user mode to read SRAM or Flash memory, the PERSISTENT preference must be set to ON. PERSISTENT = ON preserves all of the sysCONFIG pins so the FPGA can be accessed by an external device at any time. PERSISTENT = ON lets the software know that all of the dual-purpose configuration pins are reserved and NOT available for use by the fitter or the user. PERSISTENT = ON reserves all of the dual-purpose sysCONFIG pins, without regard to CONFIG_MODE.

Configuration Mode

The device knows which physical sysCONFIG port will be used by reading the state of the CFG[1:0] pins, but the fitter software also needs to know which port will be used. The fitter cannot sample the configuration pins so the user must tell the fitter by selecting the proper CONFIG_MODE. CONFIG_MODE tells the fitter which sysCONFIG pins are not available for use as user I/O.

There are several additional configuration options, such as overflow, that are set by software. These options are selected by clicking Properties under Generate Bitstream Data in ispLEVER. If either overflow option is selected, then the DONE_EX and WAKE_UP selections will be set to correspond (see Table 13-10). Refer to the Configuration Modes and Options section of this document for more details.

Table 13-10. Overflow Option Defaults

Overflow Option	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 (either)	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 configuration is not complete and the device is not ready for the wake up sequence. 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 able 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 is automatically set to ON if the DONE_EX preference is set to ON. See Table 13-11 for more information on the relationship between DONE_OD and DONE_EX.

DONE External

The LatticeXP device can wake up on its own after the Done bit is set or wait for the DONE pin to be driven high externally. Set DONE_EX = ON to delay wake up until the DONE pin is driven high by an external signal synchronous to the clock; select OFF to synchronously wake up when the internal Done bit is set and ignore any external driving of the DONE pin. The default is DONE_EX = OFF. If DONE_EX is set to ON, DONE_OD will be set to ON. If an external signal is driving the DONE pin it should be open drain as well (an external pull-up resistor may need to be added). See Table 13-11 for more information on the relationship between DONE_OD and DONE_EX.

Table 13-11. 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 LatticeXP will be a master configuration device (by properly setting the CFG[1:0] pins), and therefore provide the source clocking for configuration, the CCLK pin becomes an output with the frequency set by the value in MCCLK_FREQ. At the start of configuration the device operates at the default Master Clock Frequency of 2.5 MHz. Some of the first bits in the configuration bitstream are MCCLK_FREQ, once these are read the clock immediately starts operating at the user-defined frequency. The clock frequency is changed using a glitchless switch.

Security

When CONFIG_SECURE is set to ON, NO read back operation will be supported through the sysCONFIG or ispJTAG port of the general contents. The ispJTAG DeviceID area is readable and not considered securable. Default is OFF.

Wake Up Sequence

The WAKE_UP sequence controls three internal signals and the DONE pin. The DONE pin will be driven after configuration and prior to user mode. See the Wake Up Sequence section of this document for an example of the phase controls and information on the wake up selections. The default setting for the WAKE_UP preference is determined by the DONE_EX setting.

Wake Up with DONE_EX = Off (Default Setting)
The WAKE_UP preference for DONE_EX = OFF (default) supports the user selectable options 1 through 25, as shown in Table 13-8. If the user does not select a wake-up sequence, the default, for DONE_EX = OFF, will be wake-up sequence 21.

Wake Up with DONE_EX = On
The WAKE_UP preference for DONE_EX = ON supports the user selectable options 1 through 7, as shown in Table 13-8. If the user does not select a wake-up sequence, the default will be wake-up sequence 4.

Wake Up Clock Selection

The WAKE_UP sequence is synchronized to a clock source. The user can select the clock source to wake up to. The clock sources are either External (CCLK or TCK) or User Clock. The default is External and can be either TCK or CCLK, depending on the programming/configuration method selected. The default clock will be TCK if using ispJTAG and CCLK if using sysCONFIG. The User Clock is chosen in the user's design. The user can select any of the CLK pins of the device, or a net (routing node), as the User Clock source. WAKEUP_CLK defaults to TCK or CCLK.

Power Save

The I/O Power Save option will deactivate portions of the I/O cell drivers. This is only valid when using comparator type inputs pins (pins that use VREF), like HSTL, SSTL, etc.

Power Save mode limits some of the functionality of Boundary Scan. For Boundary Scan testing it is recommended that the I/O Power Save mode be set to OFF so that all of the I/Os will be fully functional.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Lattice ispTRACY Usage Guide

February 2006 Technical Note TN1054

Introduction

This document describes the functionality and usage of ispTRACY™, Lattice's integrated logic analyzer for the ispXPGA®, LatticeSC™, LatticeECP2™, LatticeECP™, LatticeEC™ and LatticeXP™ FPGA families. The ispTRACY tool consists of an Intellectual Property (IP) hardware block and three software tools – Core Generator, Core Linker and ispLA. ispTRACY allows for fast debugging and functional verification inside Lattice FPGA devices without the need for expensive test and measurement equipment. Debugging is accomplished through the hardware IP compiled in the design, on device block RAM and the device JTAG port.

ispTRACY IP Core Features

The ispTRACY IP core is highly configurable. These configurable features include width and depth of data capture lines, multiple edge and level sensitive trigger signals, complex comparison for trigger events, delayed trigger events and more. ispTRACY allows multiple ispTRACY IP cores to be included in a single design. The following table summarizes the features of the ispTRACY IP core.

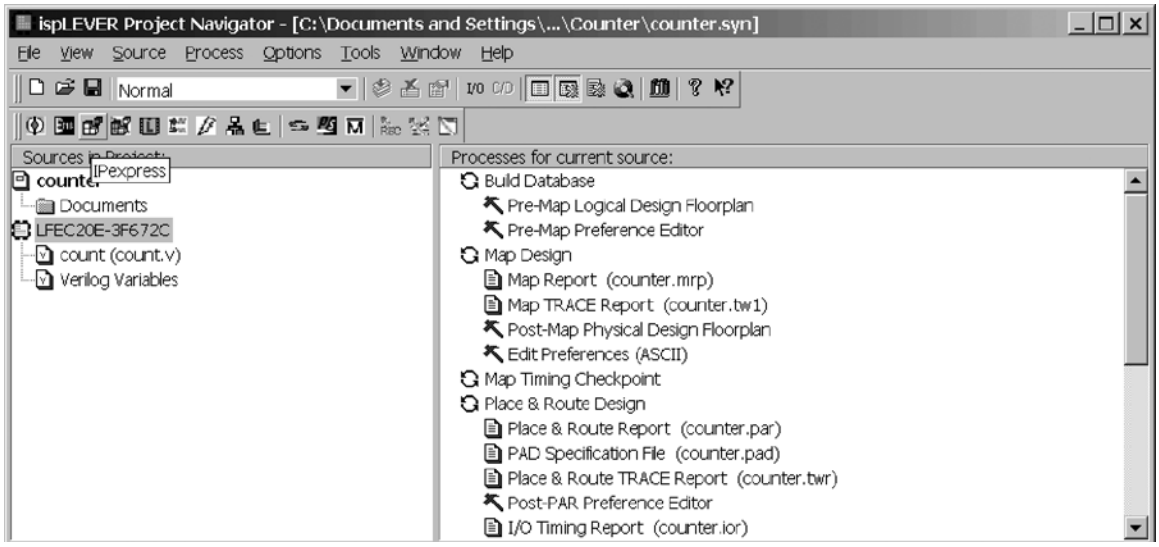
Table 14-1. ispTRACY IP Core Features

Feature	Description
Depth of Memory Capture	256 to 4096 samples
Data Capture Width	8 to 256 bits
Triggering Schemes	Rising/falling edges, level logic, comparison, trigger after combination of events
Number of Triggers	4 to 128 bits, can be a combination of edge and level sensitive signals
Number of Core	Up to 16 ispTRACY cores

ispTRACY IP Module Generator

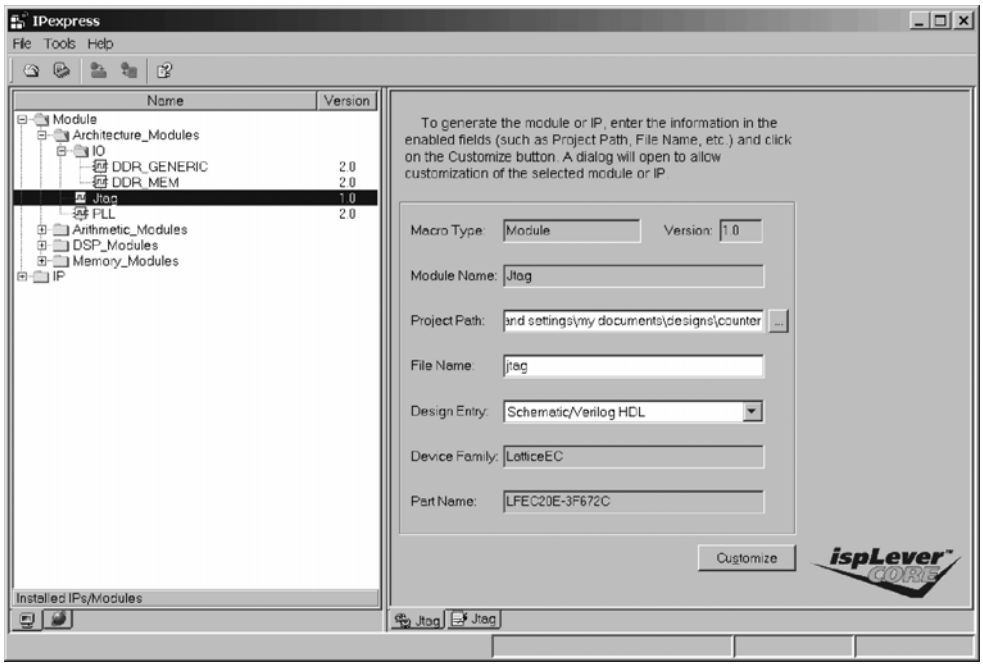
To include ispTRACY cores in a design, the first step is to run IPexpress™ from the ispLEVER® Project Navigator. Figure 14-1 shows the launch button for the IP Manager program.

Figure 14-1. IPexpress Launch Button in ispLEVER Project Navigator



Once IPexpress is launched, you will be presented with the option of generating the ispTRACY IP module by selecting the JTAG module uner Architecture and clicking the Customize button. Figure 14-2 shows the IPexpress window. Selections for the project path and module name are made through this window

Figure 14-2. ispTRACY IP Manager Program Window



ispTRACY Core Generator

The ispTRACY Core Generator under JTAG Module provides all the controls for customizing the ispTRACY core(s). Selections on this page influence the final size of the core(s) inside the FPGA and features available in terms of triggers, size of data bus and depth of memory capture. Figure 14-3 shows the Core Generator window and Table 14-3 contains descriptions of each of the figures available in the IP core. Once the core features are selected, clicking on the Generate button will create the necessary files for the Core Linker Program.

Figure 14-3. ispTRACY Core Generator Window

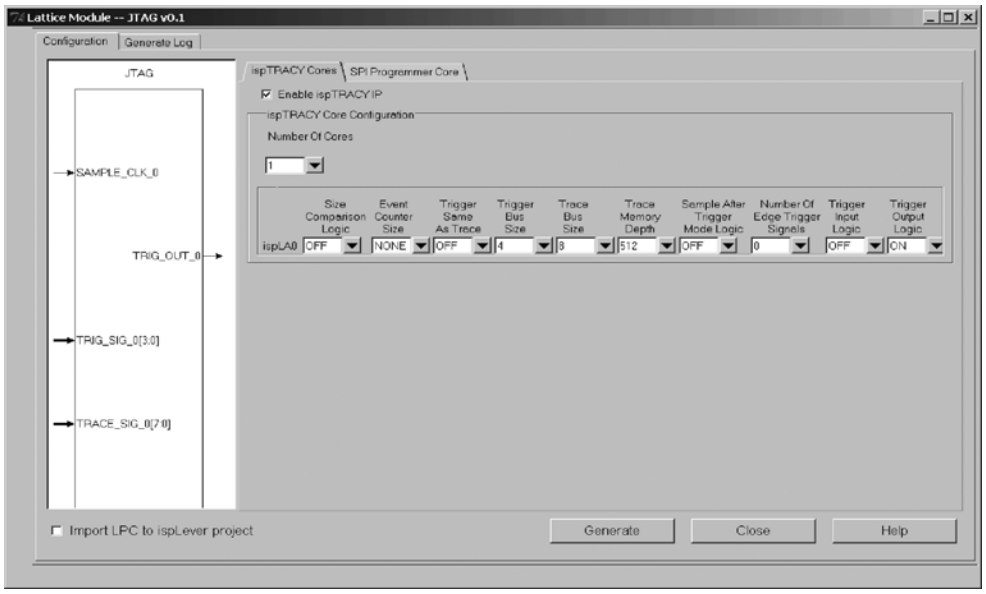


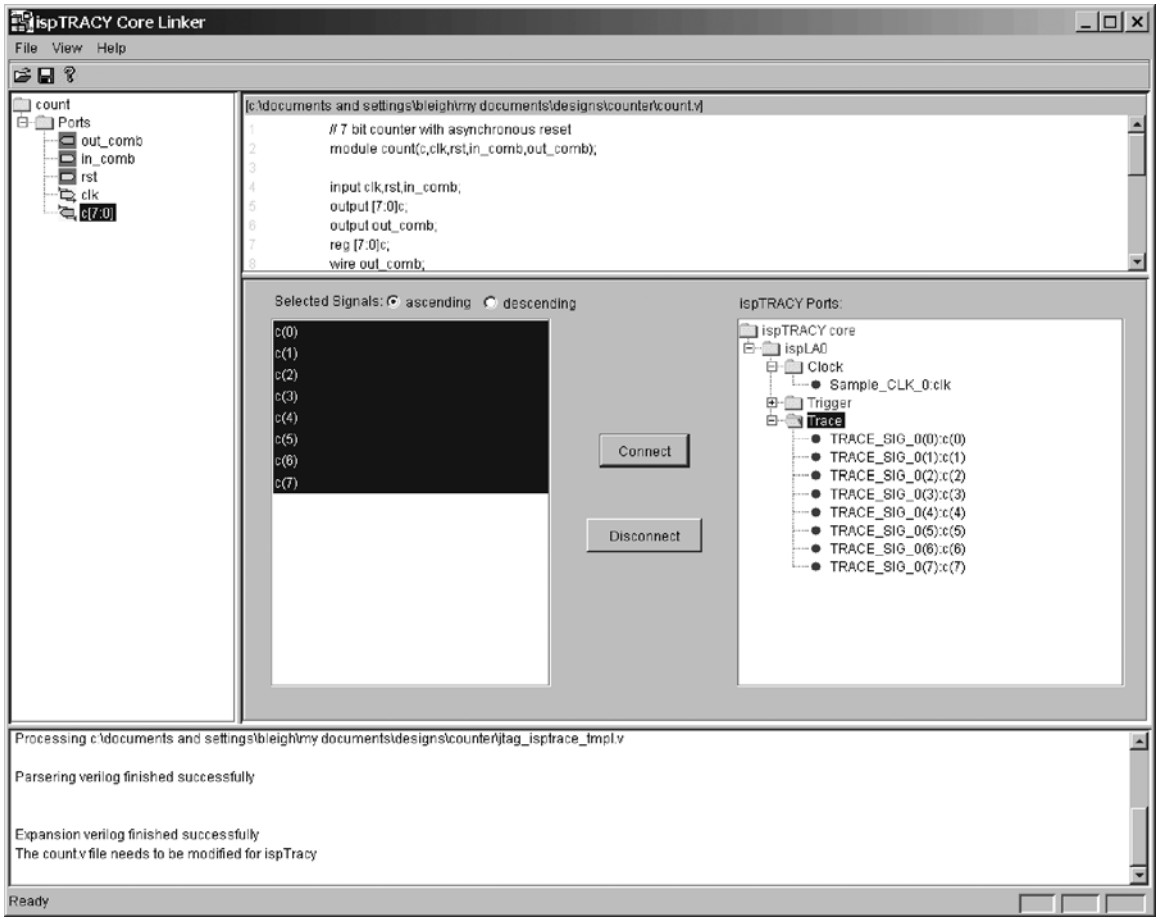
Table 14-2. ispTRACY Core Generator Features and Descriptions

Feature	Description
Number of Core	The number of ispLA(s) in an XPGA can be either 1, 2, 3, ..., or 16. The ispLA needs to be configured before use. After [Generate] button is clicked, the ispTRACY software will generate the required logic for each ispLA based on its own configuration.
ispTRACY Core	Lists the ispLA.
Size Comparison Logic	The comparison logic can compare the trigger bus with the patterns setting by the user. This field needs to be ON for ">", ">=", "<", "<=" comparison. If it's OFF, only = (equal) and <> (not equal) comparison can be preformed.
Event Counter Size	This field configures the size of the event counter. If this field is 8, the counter can be set to the value from 1 to 255. If this field is 16, the counter can be set to the value from 1 to 65535. If this field is "None", the counter logic is removed (i.e. counter value N always equal 1). If the counter value is set to N, then when the pattern occurs N times, the corresponding event will be TRUE.
Trigger Same as Trace	If this is ON, the trace bus and trigger bus are the same bus. If this is OFF, the trace bus and trigger bus are different and they can have different bus sizes.
Trigger Bus Size	This specifies the trigger bus size. It can be 4, 5, 6, ..., up to 128.
Trace Bus Size	This specifies the trace bus size. It can be a multiple of 8 up to 256 (i.e. 8, 16, 24, 32, 40,..., up to 256).
Trace Memory Depth	This is the depth of the trace memory. It defines the number of trace bus samples that ispLA can capture. This field can be set be set to 512, 1024, 2048, or 4096. It can also be set to 256 if the trace bus size is a multiple of 16 (i.e. 16, 32, 48, etc.).
Sample_After_Trigger Mode Logic	The field causes the Sample_After_Trigger mode logic to be removed or not. If this is ON, the trace mode can be set to "One Shot" mode or "Sample After Trigger" mode. If this is OFF, the trace can only be running at "One Shot" mode. When this logic is turned off, the ispLA will use less logic.
Number of Edge Trigger Signals	The trigger bus signals can be either edge sensitive signals or level sensitive signals. The level sensitive trigger signals can only be set to 0, 1 or X (don't care). The edge sensitive trigger signals can be set to 0, 1, X, R (rising edge), F (falling edge) or B (both edges). This field specifies the number of edge sensitive trigger signals.
Number of Level Trigger Signals	This field specifies the number of level sensitive trigger signals. Note that (Number of Edge Trigger Signals) + (Number of Level Trigger Signals) = (Trigger Bus Size).
Trigger Input Logic	This specifies if the "Trigger Input" logic exists. If this field is "None", the logic will be removed and the trigger condition can only be set using EV0 and EV1. If this field is set to "Pin" the trigger input logic exists and the trigger input should come from an ispXPGA device I/O pin. The trigger input can be set to either active low or active high.
Trigger Output Logic	This specifies if the "Trigger Output" logic exists. If this field is "None", the logic will be removed. If this field is set to "Pin" the trigger output logic exists and the trigger output should go out through an XPGA device I/O pin. If this is set to "ispLA", the trigger output will be connected to the trigger input of other ispLAs. You must choose "Trigger Input Logic" of the other ispLAs to be this ispLA. Same as the trigger input, the trigger output can be set to either active low or active high. At least one ispLA should have this option set to "Pin."
Generate	Generates the core.
Cancel	Cancels the action and closes the dialog box without saving any changes.
Help	Displays online Help topics for this dialog box.

ispTRACY Core Linker

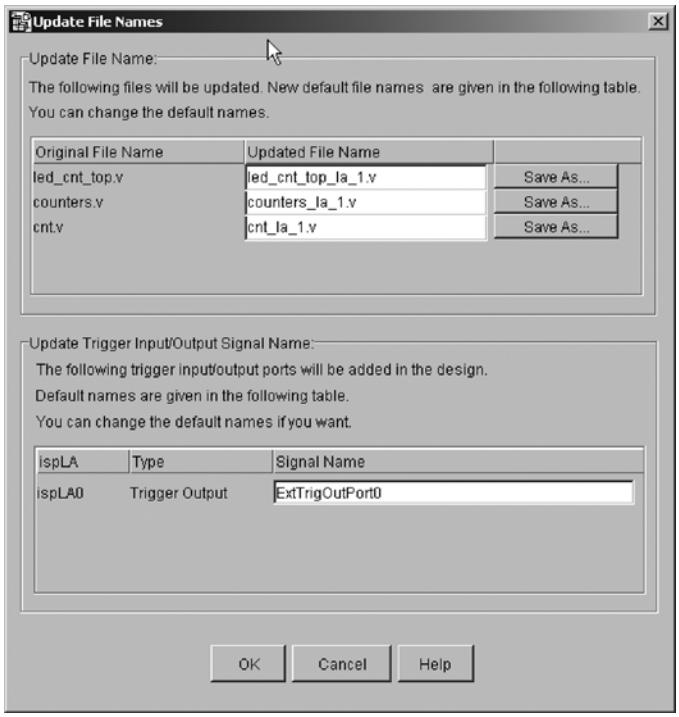
Once the ispTRACY core is created, it must be linked into the target design. This is accomplished through the Core Linker program. The Core Linker program allows the user access to internal and external signals of the target design. The internal signals can be named signals or component ports. This window also displays the available ispTRACY ports. To connect ispTRACY signals, the desired signal(s) are selected in the left-hand signal window. Signals chosen from this window are reflected in the Selected signals window. Signals must be highlighted in this window, the ispTRACY port window and then click on the connect button to connect the signals in the RTL code. Multiple instances (for example, a data bus) can be connected at once by highlighting the first signal, holding down the SHIFT key and clicking on the last desired signal. Figure 14-4 shows the ispTRACY Core Linker window.

Figure 14-4. ispTRACY Core Linker Program Window



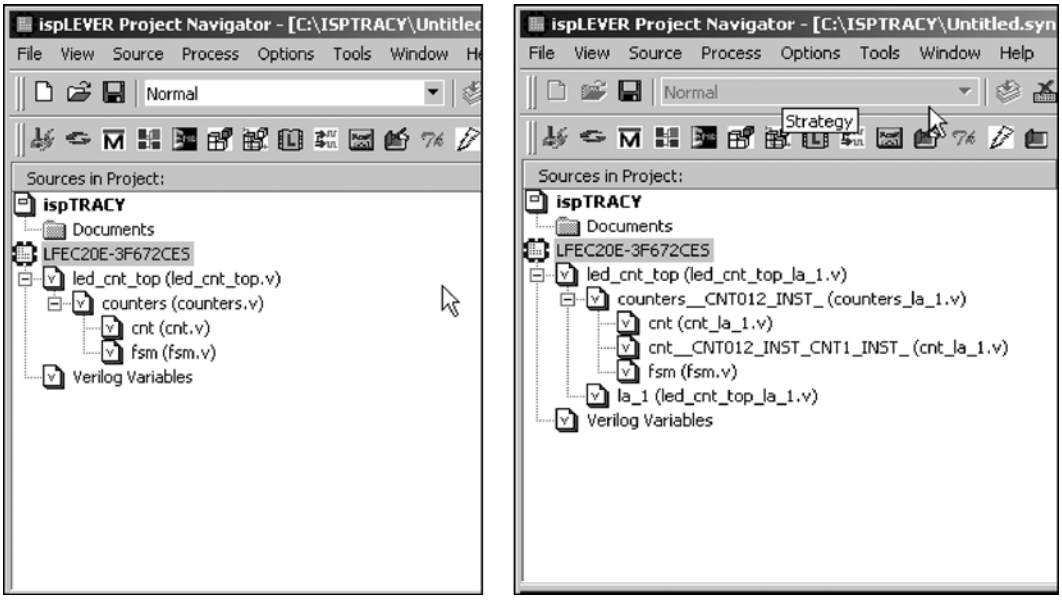
When you click the Save button (or File -> Save menu selection), the Core Linker will create modified versions of your source file, with the ispTRACY core linked into these modified files. Only design files that are directly connected to the core will be modified. A dialogue box will indicate which files have been changed and will need to be replaced in the design project for ispTRACY to function. The design files names will be the original files names with the module name for the ispTRACY core (from the IPexpress ispTRACY core generation) appended. Figure 14-5 shows the changed files dialogue box.

Figure 14-5. ispTRACY Core Linker Output Window



After clicking OK, you will be back in the ispTRACY Core Linker. You may now close this and return to the ispLEVER Project Navigator. At this point, it is necessary to replace the original design files with the ispTRACY Core Linker modified files. Figure 6 shows this file replacement process on a design.

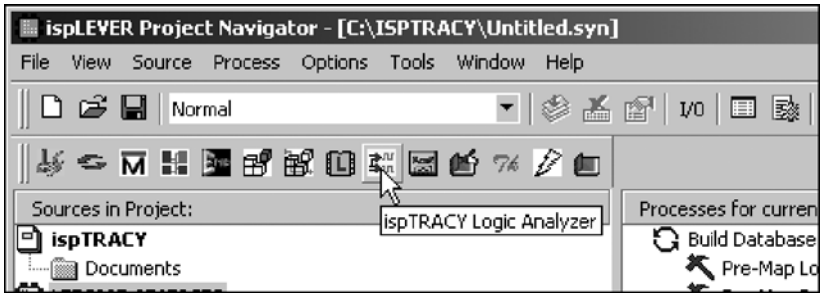
Figure 14-6. Original Project Navigator (left) and ispTRACY Project Navigator (right)



ispTRACY ispLA Program

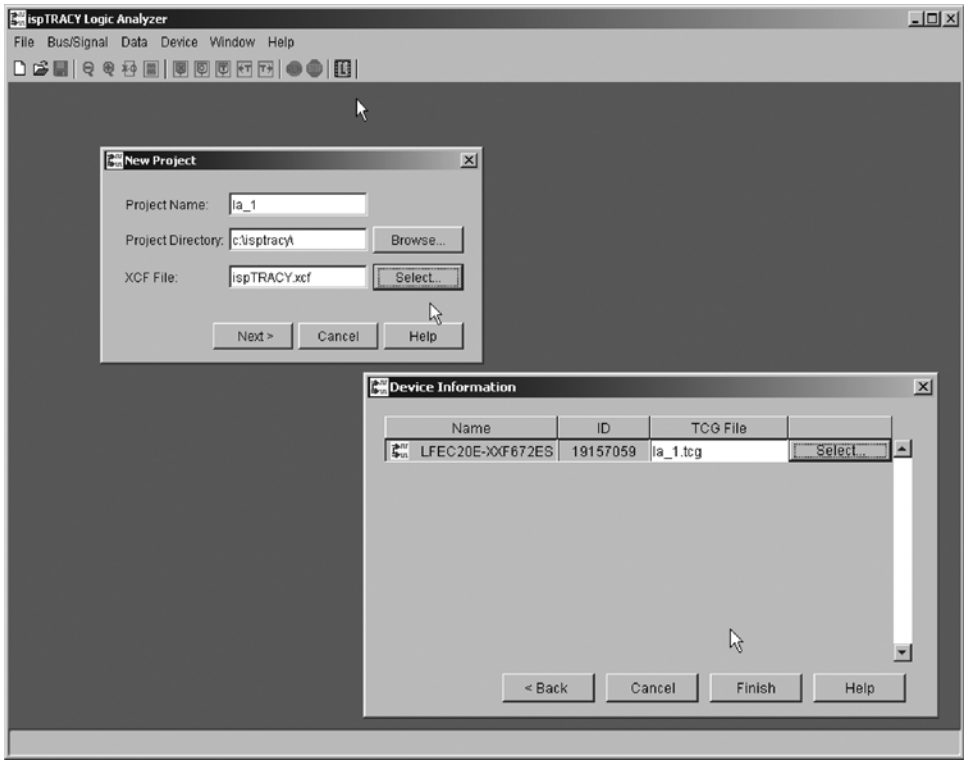
After replacing the original design files with the ispTRACY Core Linker modified files, it is necessary to re-compile the design and then program the device using the ispVM® software (see online documentation for running the ispVM program). Once the device is successfully programmed, the ispLA program needs to be started. Figure 14-7 shows the ispLA launch button from the Project Navigator.

Figure 14-7. ispLA Launch Button in ispLEVER Project Navigator



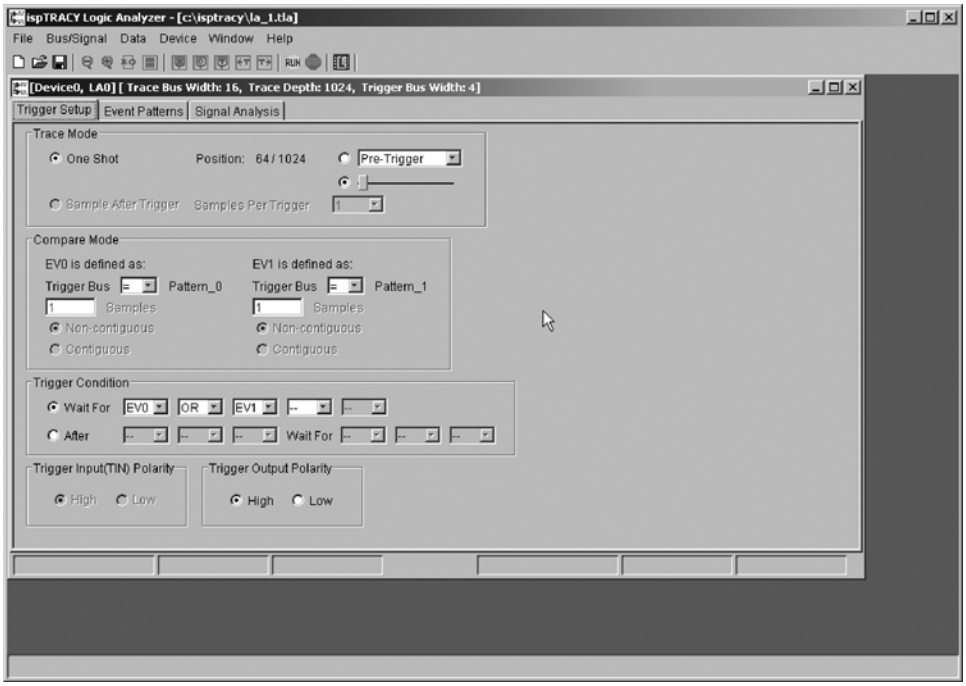
ispLA provides for capture and display of data from the ispTRACY core. The program is used to setup the trigger events counters, trigger location in data buffer, event patterns, event comparisons and data display. Figure 14-8 shows the initial startup windows required to run ispLA.

Figure 14-8. ispLA Program Window – New Project Setup



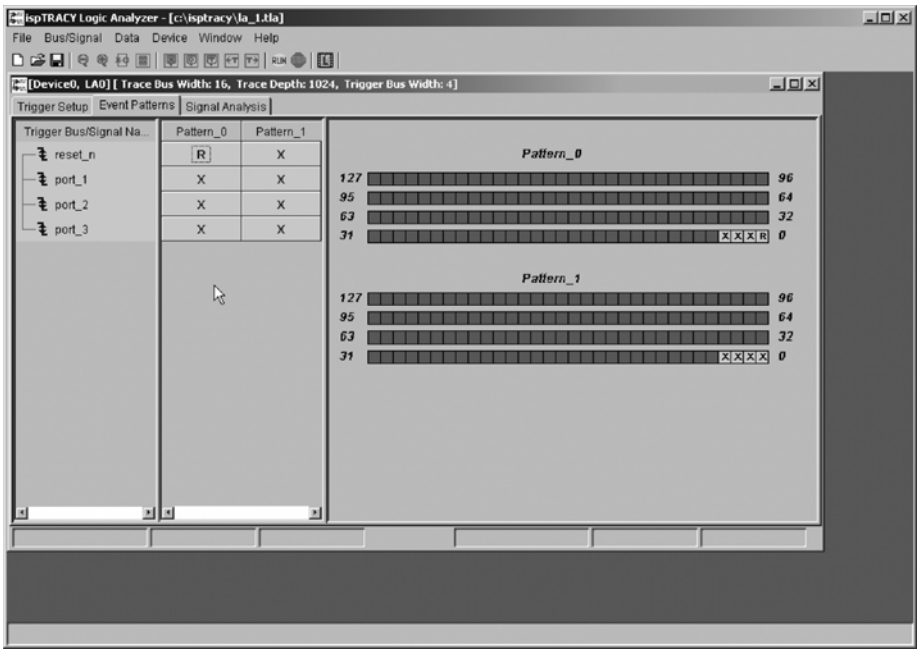
Once the project is set up, you can open up a trigger setup and viewing window for the project by clicking on Window -> Show ispLA Window -> Device 0 -> Device 0 LA0. Figure 9 shows the ispLA window. There are three tabs available under this window, trigger setup, event pattern and signal analysis (data view).

Figure 14-9. ispLA Project Setup Window - Trigger Setup options



The options available in the Trigger setup window are based on selected ispTRACY core options. In the Trace Mode box, One Shot mode will always be available, but Sample After Trigger availability is dependent on selecting Sample_After_Trigger Mode Logic => ON. The position slider can be used to select the trigger point anywhere within the data memory depth. There are three preconfigured trigger positions in the drop-down menu box. They are Pre-Trigger (5% before and 95% after trigger), Center (50% before and 50% after trigger) and Post Trigger (95% before and 5% after trigger). In the Compare Mode box the options are also dependant on core configuration. EV0 and EV1 are always available. The comparisons available and number of samples will be determined by the Size Comparison Logic and Event Counter Size. The equal to comparison is always available. Additions comparisons include >, <. !=, <=, >=. The Trigger Condition box configures which event or combination of events will cause the ispLA program to trigger and upload captured data from the device. In a simple case, this would be set to Wait for EV0. More complex cases could possible be wait for EV0 and EV1, or after EV1 wait until EV0. The final two boxed on this screen control the signal polarity of Trigger In (if available) and Trigger Out.

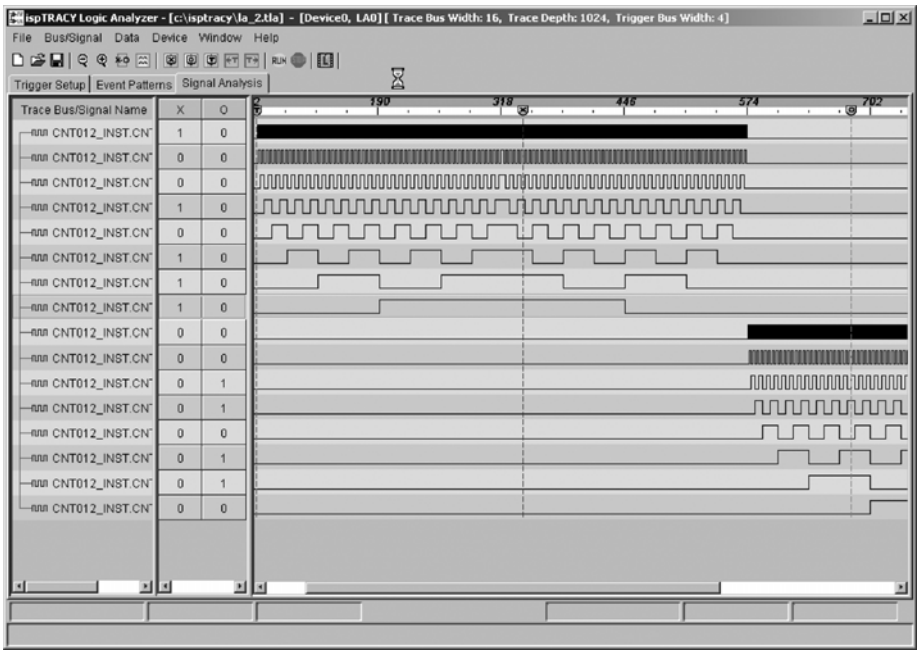
Figure 14-10. ispLA Project Setup Window - Event Pattern Setup



The Event Pattern window configures the patterns for EV0 and EV1. Pattern 0 corresponds to EV0 and Pattern 1 corresponds to EV1. The sample types for trigger signals depends on whether the signal is edge or level sensitive. If the signal is level sensitive, in the middle pattern window, only logic 0 (0) ,logic (1) or don't care (X) are available. For edge sensitive signals, the options are logic 0 (0), logic 1 (1), rising (R), falling (F), both/either edge (B) or don't care (X). In this window, changes to the pattern are made in the center section (Pattern 1 or Pattern 0) and the changes are reflected in the right section.

To begin sampling , click on the green run button. Sampling will complete when the trigger conditions are met. The data will be uploaded through the JTAG cable and results displayed I the Signal Analysis window. Figure 14-11 shows the Signal Analysis window after a data capture cycle.

Figure 14-11. ispLA Signal Analysis Window



Conclusion

ispTRACY is a full-featured logic analysis tool for use in Lattice FPGA products including ispXPGA, LatticeECP/EC and LatticeXP families. Using internal device resources, including PFF/PFU, Embedded Block Ram and the device JTAG port, the user can quickly verify functionality and assist in device debugging. ispTRACY reduces the need for external test and measurement equipment to debug FPGA projects, while providing full access to a wide range of internal signals, components and design elements.

References

- *ispTRACY Usage Guide,* 2/07/04 Rev. 0.1, page 14 of 14.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

HDL Synthesis Coding Guidelines for Lattice Semiconductor FPGAs

October 2005 Technical Note TN1008

Introduction

Coding style plays an important role in utilizing FPGA resources. Although many popular synthesis tools have significantly improved optimization algorithms for FPGAs, it still is the responsibility of the user to generate meaningful and efficient HDL code to guide their synthesis tools to achieve the best result for a specific architecture. This application note is intended to help designers establish useful HDL coding styles for Lattice Semiconductor FPGA devices. It includes VHDL and Verilog design guidelines for both novice and experienced users.

The application note is divided into two sections. The general coding styles for FPGAs section provides an overview for effective FPGA designs. The following topics are discussed in detail:

- Hierarchical Coding
- Design Partitioning
- Encoding Methodologies for State Machines
- Coding Styles for Finite State Machines (FSM)
- Using Pipelines
- Comparing IF Statements and CASE Statements
- Avoiding Non-intentional Latches

The HDL Design with Lattice Semiconductor FPGA Devices section covers specific coding techniques and examples:

- Using the Lattice Semiconductor FPGA Synthesis Library
- Implementation of Multiplexers
- Creating Clock Dividers
- Register Control Signals (CE, LSR, GSR)
- Using PIC Features
- Implementation of Memories
- Preventing Logic Replication and Fanout
- Comparing Synthesis Results and Place and Route Results

General Coding Styles for FPGA

The following recommendations for common HDL coding styles will help users generate robust and reliable FPGA designs.

Hierarchical Coding

HDL designs can either be synthesized as a flat module or as many small hierarchical modules. Each methodology has its advantages and disadvantages. Since designs in smaller blocks are easier to keep track of, it is preferred to apply hierarchical structure to large and complex FPGA designs. Hierarchical coding methodology allows a group of engineers to work on one design at the same time. It speeds up design compilation, makes changing the implementation of key blocks easier, and reduces the design period by allowing the re-use of design modules for current and future designs. In addition, it produces designs that are easier to understand. However, if the design mapping into the FPGA is not optimal across hierarchical boundaries, it will lead to lower device utilization and design performance. This disadvantage can be overcome with careful design considerations when choosing the design hierarchy. Here are some tips for building hierarchical structures:

- The top level should only contain instantiation statements to call all major blocks
- Any I/O instantiations should be at the top level
- Any signals going into or out of the devices should be declared as input, output or bi-directional pins at the top level

- Memory blocks should be kept separate from other code

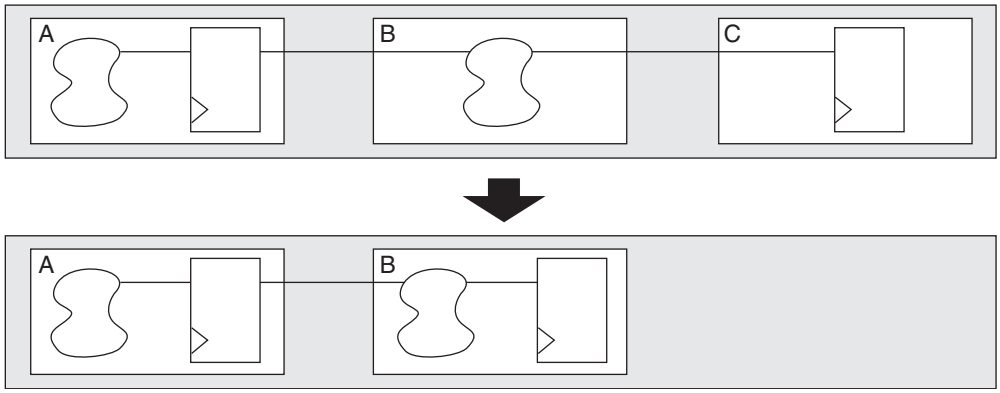
Design Partitioning

By effectively partitioning the design, a designer can reduce overall run time and improve synthesis results. Here are some recommendations for design partitioning.

Maintain Synchronous Sub-blocks by Registering All Outputs

It is suggested to arrange the design boundary such that the outputs in each block are registered. Registering outputs helps the synthesis tool to consider the implementation of the combinatorial logic and registers into the same logic block. Registering outputs also makes the application of timing constraints easier since it eliminates possible problems with logic optimization across design boundaries. Single clock is recommended for each synchronous block because it significantly reduces the timing consideration in the block. It leaves the adjustment of the clock relationships of the whole design at the top level of the hierarchy. Figure 16-1 shows an example of synchronous blocks with registered outputs.

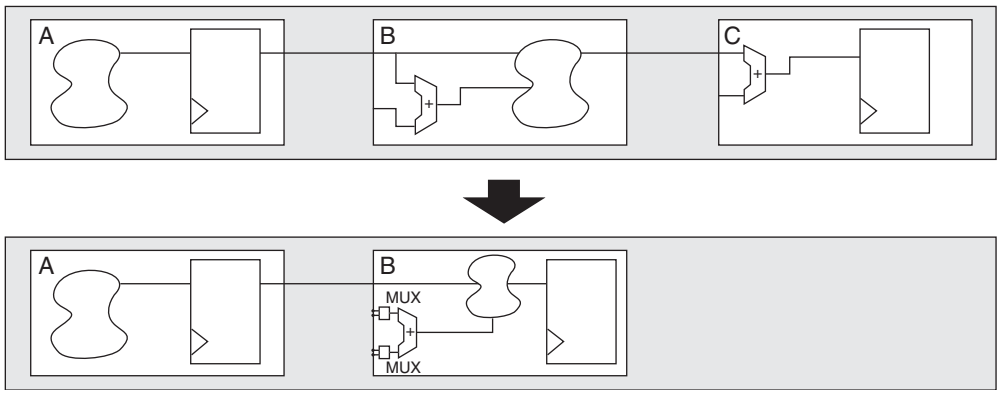
Figure 16-1. Synchronous Blocks with Registered Outputs



Keep Related Logic Together in the Same Block

Keeping related logic and sharable resources in the same block allows the sharing of common combinatorial terms and arithmetic functions within the block. It also allows the synthesis tools to optimize the entire critical path in a single operation. Since synthesis tools can only effectively handle optimization of certain amounts of logic, optimization of critical paths pending across the boundaries may not be optimal. Figure 16-2 shows an example of merging sharable resource in the same block.

Figure 16-2. Merge Sharable Resource in the Same Block



Separate Logic with Different Optimization Goals

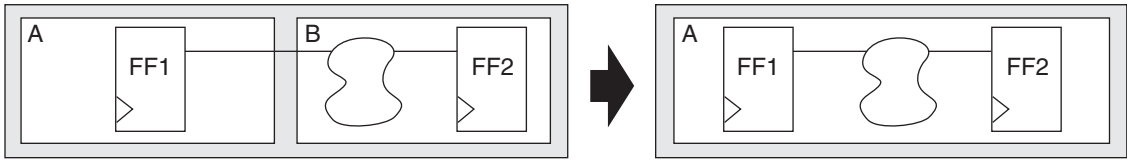
Separating critical paths from non-critical paths may achieve efficient synthesis results. At the beginning of the project, one should consider the design in terms of performance requirements and resource requirements. If there

are two portions of a block, one that needs to be optimized for area and a second that needs to be optimized for speed, they should be separated into two blocks. By doing this, different optimization strategies for each module can be applied without being limited by one another.

Keep Logic with the Same Relaxation Constraints in the Same Block

When a portion of the design does not require high performance, this portion can be applied with relaxed timing constraints such as “multicycle” to achieve high utilization of device area. Relaxation constraints help to reduce overall run time. They can also help to efficiently save resources, which can be used on critical paths. Figure 16-3 shows an example of grouping logic with the same relaxation constraint in one block.

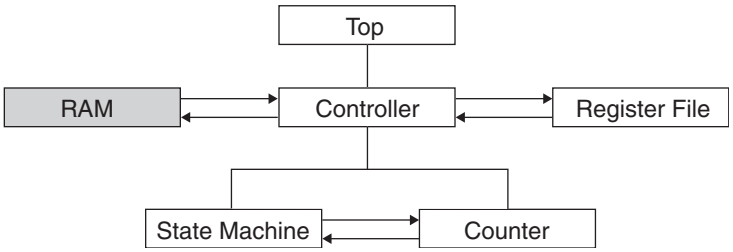
Figure 16-3. Logic with the Same Relaxation Constraint



Keep Instantiated Code in Separate Blocks

It is recommended that the RAM block in the hierarchy be left in a separate block (Figure 16-4). This allows for easy swapping between the RAM behavioral code for simulation, and the code for technology instantiation. In addition, this coding style facilitates the integration of the ispLEVER® IPexpress™ tool into the synthesis process.

Figure 16-4. Separate RAM Block



Keep the Number FPGA Gates at 30 to 80 PFU Per Block

This range varies based on the computer configuration, time required to complete each optimization run, and the targeted FPGA routing resources. Although a smaller block methodology allows more control, it may not produce the most efficient design since it does not provide the synthesis tool enough logic to apply “Resource Sharing” algorithms. On the other hand, having a large number of gates per block gives the synthesis tool too much to work on and causes changes that affect more logic than necessary in an incremental or multi-block design flow.

State Encoding Methodologies for State Machines

There are several ways to encode a state machine, including binary encoding, gray-code encoding and one-hot encoding. State machines with binary or gray-code encoded states have minimal numbers of flip-flops and wide combinatorial functions, which are typically favored for CPLD architectures. However, most FPGAs have many flip-flops and relatively narrow combinatorial function generators. Binary or gray-code encoding schemes can result in inefficient implementation in terms of speed and density for FPGAs. On the other hand, one-hot encoded state machine represents each state with one flip-flop. As a result, it decreases the width of combinatorial logic, which matches well with FPGA architectures. For large and complex state machines, one-hot encoding usually is the preferable method for FPGA architectures. For small state machines, binary encoding or gray-code encoding may be more efficient.

There are many ways to ensure the state machine encoding scheme for a design. One can hard code the states in the source code by specifying a numerical value for each state. This approach ensures the correct encoding of the state machine but is more restrictive in the coding style. The enumerated coding style leaves the flexibility of state machine encoding to the synthesis tools. Most synthesis tools allow users to define encoding styles either through

attributes in the source code or through the tool's Graphical User Interface (GUI). Each synthesis tool has its own synthesis attributes and syntax for choosing the encoding styles. Refer to the synthesis tools documentation for details about attributes syntax and values.

The following syntax defines an enumeration type in VHDL:

```
type type_name is (state1_name,state2_name,.....,stateN_name)
```

Here is a VHDL example of enumeration states:

```
type STATE_TYPE is (S0,S1,S2,S3,S4);
signal CURRENT_STATE, NEXT_STATE : STATE_TYPE;
```

The following are examples of Synplify® and LeonardoSpectrum® VHDL synthesis attributes.

Synplify:

```
attribute syn_encoding : string;
attribute syn_encoding of <signal_name> : type is "value ";
-- The syn_encoding attribute has 4 values : sequential, onehot, gray and safe.
```

LeonardoSpectrum:

```
-- Declare TYPE_ENCODING_STYLE attribute
-- Not needed if the exemplar_1164 package is used
type encoding_style is (BINARY, ONEHOT, GRAY, RANDOM, AUTO);
attribute TYPE_ENCODING_STYLE : encoding_style;
...
attribute TYPE_ENCODING_STYLE of <typename> : type is ONEHOT;
```

In Verilog, one must provide explicit state values for states. This can be done by using the bit pattern (e.g., 3'b001), or by defining a parameter and using it as the case item. The latter method is preferable. The following is an example using parameter for state values.

```
Parameter state1 = 2'h1, state2 = 2'h2;
...
current_state = state2; // setting current state to 2'h2
```

The attributes in the source code override the default encoding style assigned during synthesis. Since Verilog does not have predefined attributes for synthesis, attributes are usually attached to the appropriate objects in the source code as comments. The attributes and their values are case sensitive and usually appear in lower case. The following examples use attributes in Verilog source code to specify state machine encoding style.

Synplify:

```
Reg[2:0] state; /* synthesis syn_encoding = "value" */;
// The syn_encoding attribute has 4 values : sequential, onehot, gray and safe.
```

In LeonardoSpectrum, it is recommended to set the state machine variable to an enumeration type with enum pragma. Once this is set in the source code, encoding schemes can be selected in the LeonardoSpectrum GUI.

LeonardoSpectrum:

```
Parameter /* exemplar enum <type_name> */ s0 = 0, s1 = 1, s2 = 2, s3 = 3, S4 = 4;
Reg [2:0] /* exemplar enum <type_name> */ present_state, next_state ;
```

In general, synthesis tools will select the optimal encoding style that takes into account the target device architecture and size of the decode logic. One can always apply synthesis attributes to override the default encoding style if necessary.

Coding Styles for FSM

A finite state machine (FSM) is a hardware component that advances from the current state to the next state at the clock edge. As mentioned in the Encoding Methodologies for State Machines section, the preferable scheme for FPGA architectures is one-hot encoding. This section discusses some common issues encountered when constructing state machines, such as initialization and state coverage, and special case statements in Verilog.

General State Machine Description

Generally, there are two approaches to describe a state machine. One is to use one process/block to handle both state transitions and state outputs. The other is to separate the state transition and the state outputs into two different process/blocks. The latter approach is more straightforward because it separates the synchronous state registers from the decoding logic used in the computation of the next state and the outputs. This will make the code easier to read and modify, and makes the documentation more efficient. If the outputs of the state machine are combinatorial signals, the second approach is almost always necessary because it will prevent the accidental registering of the state machine outputs.

The following examples describe a simple state machine in VHDL and Verilog. In the VHDL example, a sequential process is separated from the combinatorial process. In Verilog code, two *always* blocks are used to describe the state machine in a similar way.

VHDL Example for State Machine

```
. . .
architecture lattice_fpga of dram_refresh is
   type state_typ is (s0, s1, s2, s3, s4);
   signal present_state, next_state : state_typ;
begin
   -- process to update the present state
   registers: process (clk, reset)
   begin
      if (reset='1') then
         present_state <= s0;
      elsif clk'event and clk='1' then
         present_state <= next_state;
      end if;
   end process registers;

   -- process to calculate the next state & output
   transitions: process (present_state, refresh, cs)
   begin
      ras <= '0'; cas <= '0'; ready <= '0';
      case present_state is
         when s0 =>
            ras <= '1'; cas <= '1'; ready <= '1';
            if (refresh = '1') then next_state <= s3;
            elsif (cs = '1') then next_state <= s1;
            else next_state <= s0;
            end if;
         when s1 =>
            ras <= '0'; cas <= '1'; ready <= '0';
            next_state <= s2;
         when s2 =>
            ras <= '0'; cas <= '0'; ready <= '0';
            if (cs = '0') then next_state <= s0;
            else next_state <= s2;
            end if;
         when s3 =>
            ras <= '1'; cas <= '0'; ready <= '0';
            next_state <= s4;
         when s4 =>
            ras <= '0'; cas <= '0'; ready <= '0';
            next_state <= s0;
         when others =>
            ras <= '0'; cas <= '0'; ready <= '0';
            next_state <= s0;
      end case;
   end process transitions;
. . .
```

Verilog Example for State Machine

```
. . .
parameter s0 = 0, s1 = 1, s2 = 2, s3 = 3, s4 = 4;

reg [2:0] present_state, next_state;
reg ras, cas, ready;

// always block to update the present state
always @ (posedge clk or posedge reset)
begin
   if (reset) present_state = s0;
   else present_state = next_state;
end

// always block to calculate the next state & outputs
always @ (present_state or refresh or cs)
begin
   next_state = s0;
   ras = 1'bX; cas = 1'bX; ready = 1'bX;
   case (present_state)
      s0 : if (refresh) begin
              next_state = s3;
              ras = 1'b1; cas = 1'b0; ready = 1'b0;
           end
           else if (cs) begin
              next_state = s1; ras = 1'b0; cas = 1'b1; ready = 1'b0;
           end
           else begin
              next_state = s0; ras = 1'b1; cas = 1'b1; ready = 1'b1;
           end
      s1 : begin
              next_state = s2; ras = 1'b0; cas = 1'b0; ready = 1'b0;
           end
      s2 : if (~cs) begin
              next_state = s0; ras = 1'b1; cas = 1'b1; ready = 1'b1;
           end
           else begin
              next_state = s2; ras = 1'b0; cas = 1'b0; ready = 1'b0;
           end
      s3 : begin
              next_state = s4; ras = 1'b1; cas = 1'b0; ready = 1'b0;
           end
      s4 : begin
              next_state = s0; ras = 1'b0; cas = 1'b0; ready = 1'b0;
           end
   endcase
end
. . .
```

Initialization and Default State

A state machine must be initialized to a valid state after power-up. This can be done at the device level during power up or by including a reset operation to bring it to a known state. For all Lattice Semiconductor FPGA devices, the Global Set/Reset (GSR) is pulsed at power-up, regardless of the function defined in the design source code. In the above example, an asynchronous reset can be used to bring the state machine to a valid initialization state. In the same manner, a state machine should have a default state to ensure the state machine will not go into an invalid state if not all the possible combinations are clearly defined in the design source code. VHDL and Verilog have different syntax for default state declaration. In VHDL, if a CASE statement is used to construct a state machine, "When Others" should be used as the last statement before the end of the statement, If an IF-THEN-ELSE statement is used, "Else" should be the last assignment for the state machine. In Verilog, use "default" as the last assignment for a CASE statement, and use "Else" for the IF-THEN-ELSE statement.

When Others in VHDL

```
...
architecture lattice_fpga of FSM1 is
   type state_typ is (deflt, idle, read, write);
   signal next_state : state_typ;
begin
process(clk, rst)
begin
   if (rst='1') then
      next_state <= idle; dout <= '0';
   elsif (clk'event and clk='1') then
      case next_state is
         when idle =>
            next_state <= read; dout <= din(0);
         when read =>
            next_state <= write; dout <= din(1);
         when write =>
            next_state <= idle; dout <= din(2);
         when others =>
            next_state <= deflt; dout <= '0';
      end case;
   end if;
end process;
...
```

Default Clause in Verilog

```
...
// Define state labels explicitly
parameter deflt=2'bxx;
parameter idle =2'b00;
parameter read =2'b01;
parameter write=2'b10;

reg [1:0] next_state;
reg dout;

always @(posedge clk or posedge rst)
   if (rst) begin
      next_state <= idle;
      dout <= 1'b0;
   end
   else begin
      case(next_state)
         idle: begin
                dout <= din[0]; next_state <= read;
               end
         read: begin
                dout <= din[1]; next_state <= write;
               end
         write: begin
                dout <= din[2]; next_state <= idle;
               end
         default: begin
                   dout <= 1'b0; next_state <= deflt;
                  end
```

Full Case and Parallel Case Specification in Verilog

Verilog has additional attributes to define the default states without writing it specifically in the code. One can use "full_case" to achieve the same performance as "default". The following examples show the equivalent representations of the same code in Synplify. LeonardoSpectrum allows users to apply Verilog-specific options in the GUI settings.

```
…
case (current_state)  // synthesis full_case
   2'b00 : next_state <= 2'b01;
   2'b01 : next_state <= 2'b11;
   2'b11 : next_state <= 2'b00;
…
```

```
…
case (current_state)
   2'b00 : next_state <= 2'b01;
   2'b01 : next_state <= 2'b11;
   2'b11 : next_state <= 2'b00;
   default : next_state <= 2bx;
```

"Parallel_case" makes sure that all the statements in a case statement are mutually exclusive. It is used to inform the synthesis tools that only one case can be true at a time. The syntax for this attribute in Synplify is as follows:

```
// synthesis parallel_case
```

Using Pipelines in the Designs

Pipelining can improve design performance by restructuring a long data path with several levels of logic and breaking it up over multiple clock cycles. This method allows a faster clock cycle by relaxing the clock-to-output and setup time requirements between the registers. It is usually an advantageous structure for creating faster data paths in register-rich FPGA devices. Knowledge of each FPGA architecture helps in planning pipelines at the

beginning of the design cycle. When the pipelining technique is applied, special care must be taken for the rest of the design to account for the additional data path latency. The following illustrates the same data path before (Figure 16-5) and after pipelining (Figure 16-6).

Figure 16-5. Before Pipelining

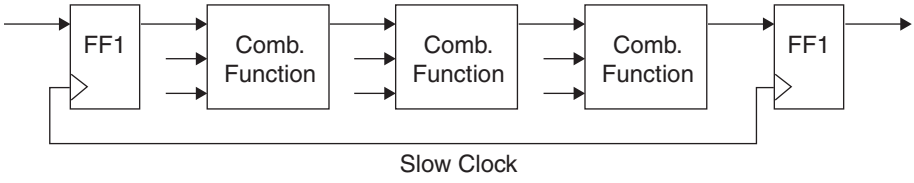
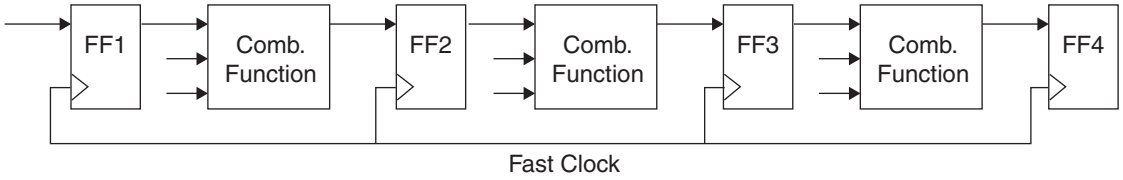


Figure 16-6. After Pipelining



Before pipelining, the clock speed is determined by the clock-to-out time of the source register, the logic delay through four levels of combinatorial logic, the associated routing delays, and the setup time of the destination register. After pipelining is applied, the clock speed is significantly improved by reducing the delay of four logic levels to one logic level and the associated routing delays, even though the rest of the timing requirements remain the same. It is recommended to check the Place and Route timing report to ensure that the pipelined design gives the desired performance.

Comparing IF statement and CASE statement

CASE and IF-THEN-ELSE statements are common for sequential logic in HDL designs. The IF-THEN-ELSE statement generally generates priority-encoded logic, whereas the CASE statement implements balanced logic. An IF-THEN-ELSE statement can contain a set of different expressions while a Case statement is evaluated against a common controlling expression. Both statements will give the same functional implementation if the decode conditions are mutually exclusive, as shown in the following VHDL codes.

```
-- Case Statement — mutually exclusive conditions
process (s, x, y, z)
begin
   O1 <= '0';
   O2 <= '0';
   O3 <= '0';
   case (s) is
      when "00" => O1 <= x;
      when "01" => O2 <= y;
      when "10" => O3 <= z;
   end case;
end process;
```

```
-- If-Then-Else — mutually exclusive conditions
process (s, x, y, z)
begin
   O1 <= '0';
   O2 <= '0';
   O3 <= '0';
   if s = "00" then O1 <= x;
   elsif s = "01" then O2 <= y;
   elsif s = "10" then O3 <= z;
   end if;
end process;
```

However, the use of If-Then-Else construct could be a key pitfall to make the design more complex than necessary, because extra logic are needed to build a priority tree. Consider the following examples:

```
--A: If-Then-Elese Statement: Complex O3 Equations
process(s1, s2, s3, x, y, z)
begin
   O1 <= '0';
   O2 <= '0';
   O3 <= '0';
   if s1 = '1' then
      O1 <= x;
   elsif s2 = '1' then
      O2 <= y;
   elsif s3 = '1' then
      O3 <= z;
   end if;
end process;
```

```
--B: If-Then-Else Statement: Simplified O3 Equation
process (s1, s2, s3, x, y, z)
begin
   O1 <= '0';
   O2 <= '0';
   O3 <= '0';
   if s1 = '1' then
      O1 <= x;
   end if;
   if s2 = '1' then
      O2 <= y;
   end if;
   if s3 <= '1' then
      O3 <= z;
   end if;
 end process;
```

If the decode conditions are not mutually exclusive, IF-THEN-ELSE construct will cause the last output to be dependent on all the control signals. The equation for O3 output in example A is:

```
O3 <= z and (s3) and (not (s1 and s2));
```

If the same code can be written as in example B, most of the synthesis tools will remove the priority tree and decode the output as:

```
O3 <= z and s3;
```

This reduces the logic requirement for the state machine decoder. If each output is indeed dependent of all of the inputs, it is better to use a CASE statement since CASE statements provide equal branches for each output.

Avoiding Non-intentional Latches

Synthesis tools infer latches from incomplete conditional expressions, such as an IF-THEN-ELSE statements without an Else clause. To avoid non-intentional latches, one should specify all conditions explicitly or specify a default assignment. Otherwise, latches will be inserted into the resulting RTL code, requiring additional resources in the device or introducing combinatorial feedback loops that create asynchronous timing problems. Non-intentional latches can be avoided by using clocked registers or by employing any of the following coding techniques:

- Assigning a default value at the beginning of a process
- Assigning outputs for all input conditions
- Using else, (when others) as the final clause

Another way to avoid non-intentional latches is to check the synthesis tool outputs. Most of the synthesis tools give warnings whenever there are latches in the design. Checking the warning list after synthesis will save a tremendous amount of effort in trying to determine why a design is so large later in the Place and Route stage.

HDL Design with Lattice Semiconductor FPGA Devices

The following section discusses the HDL coding techniques utilizing specific Lattice Semiconductor FPGA system features. This kind of architecture-specific coding style will further improve resource utilization and enhance the performance of designs.

Lattice Semiconductor FPGA Synthesis Library

The Lattice Semiconductor FPGA Synthesis Library includes a number of library elements to perform specific logic functions. These library elements are optimized for Lattice Semiconductor FPGAs and have high performance and utilization. The following are the classifications of the library elements in the Lattice Semiconductor FPGA Synthe-

sis Library. The definitions of these library elements can be found in the *Reference Manuals* section of the ispLEVER on-line help system.

- Logic gates and LUTs
- Comparators, adders, subtractors
- Counters
- Flip-flops and latches
- Memory, 4E-specific memory (block RAM function)
- Multiplexors
- Multipliers
- All I/O cells, including I/O flip-flops
- PIC cells
- Special cells, including PLL, GSR, boundary scan, etc.
- FPSC elements

IPepxress, a parameterized module complier optimized for Lattice FPGA devices, is available for more complex logic functions. IPexpress supports generation of library elements with a number of different options such as PLLs and creates parameterized logic functions such as PFU and EBR memory, multipliers, adders, subtractors, and counters. IPexpress accepts options that specify parameters for parameterized modules such as data path modules and memory modules, and produces a circuit description with Lattice Semiconductor FPGA library elements. Output from IPexpress can be written in EDIF, VHDL, or Verilog. In order to use synthesis tools to utilize the Lattice FPGA architectural features, it is strongly recommended to use IPexpress to generate modules for source code instantiation. The following are examples of Lattice Semiconductor FPGA modules supported by IPexpress:

- PLL
- Memory implemented in PFU:
  – Synchronous single-port RAM, synchronous dual-port RAM, synchronous ROM, synchronous FIFO
- Memory implemented with EBR:
  – Quad-port Block RAM, Dual-Port Block RAM, Single-Port Block RAM, ROM, FIFO
- Other EBR based Functions
  – Multiplier, CAM
- PFU based functions
  – Multiplier, adder, subtractor, adder/subtractor, linear feedback shifter, counter
- MPI/System Bus

IPexpress is especially efficient when generating high pin count modules as it saves time in manually cascading small library elements from the synthesis library. Detailed information about IPexpress and its user guide can be found in the ispLEVER help system.

Implementing Multiplexers

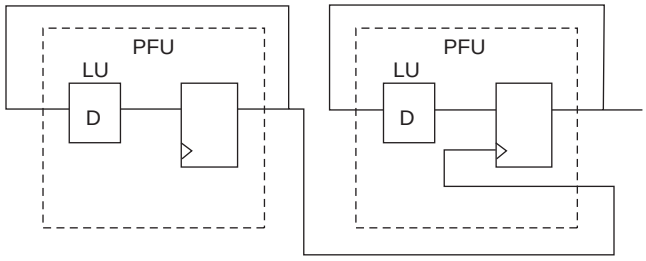
The flexible configurations of LUTs can realize any 4-, 5-, or 6-input logic function like 2-to-1, 3-to-1 or 4-to-1 multiplexers. Larger multiplexers can be efficiently created by programming multiple 4-input LUTs. Synthesis tools camn automatically infer Lattice FPGA optimized multiplexer library elements based on the behavioral description in the HDL source code. This provides the flexibility to the Mapper and Place and Route tools to configure the LUT mode and connections in the most optimum fashion.

```
16:1 MUX
…
process(sel, din)
   begin
      if    (sel="0000") then muxout <= din(0);
      elsif (sel="0001") then muxout <= din(1);
      elsif (sel="0010") then muxout <= din(2);
      elsif (sel="0011") then muxout <= din(3);
      elsif (sel="0100") then muxout <= din(4);
      elsif (sel="0101") then muxout <= din(5);
      elsif (sel="0110") then muxout <= din(6);
      elsif (sel="0111") then muxout <= din(7);
      elsif (sel="1000") then muxout <= din(8);
      elsif (sel="1001") then muxout <= din(9);
      elsif (sel="1010") then muxout <= din(10);
      elsif (sel="1011") then muxout <= din(11);
      elsif (sel="1100") then muxout <= din(12);
      elsif (sel="1101") then muxout <= din(13);
      elsif (sel="1110") then muxout <= din(14);
      elsif (sel="1111") then muxout <= din(15);
      else muxout <= '0';
      end if;
   end process;
…
```

Clock Dividers

There are two ways to implement clock dividers in Lattice Semiconductor FPGA devices. The first is to cascade the registers with asynchronous clocks. The register output feeds the clock pin of the next register (Figure 16-7). Since the clock number in each PFU is limited to two, any clock divider with more than two bits will require multiple PFU implementations. As a result, the asynchronous daisy chaining implementation of clock divider will be slower due to the inter-PLC routing delays. This kind of delays is usually ambiguous and inconsistent because of the nature of FPGA routing structures.

Figure 16-7. Daisy Chaining of Flip-flops



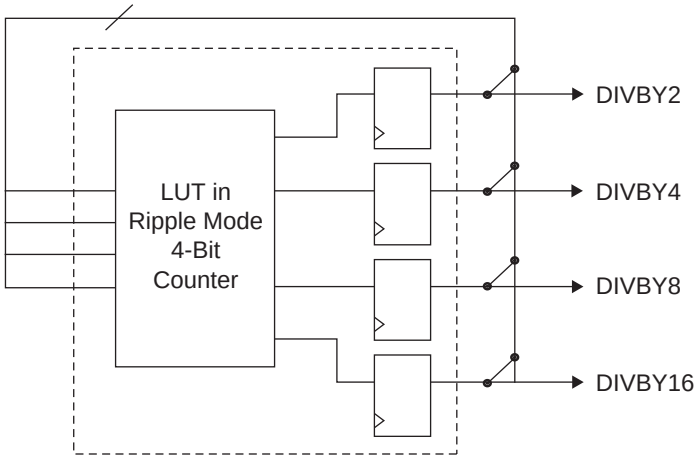
The following are the HDL representations of the design in Figure 16-7.

```
-- VHDL Example of Daisy Chaining FF
...
-- 1st FF to divide Clock in half
CLK_DIV1: process(CLK, RST)
begin
   if (RST='1') then
      clk1 <= '0';
   elsif (CLK'event and CLK='1') then
      clk1 <= not clk1;
   end if;
end process CLK_DIV1;

-- 2nd FF to divide clock in half
CLK_DIV2: process(clk1, RST)
begin
   if (RST='1') then
      clk2 <= '0';
   elsif (clk1'event and clk1='1') then
      clk2 <= not clk2;
   end if;
end process CLK_DIV2;
```

```
//Verilog Example of Daisy Chaining FF
...
always @(posedge CLK or posedge RST)
begin
   if (RST)
      clk1 = 1'b0;
   else
      clk1 = !clk1;
end

always @(posedge clk1 or posedge RST)
begin
   if (RST)
      clk2 = 1'b0;
   else
      clk2 = !clk2;
end

...
```

The preferable way is to fully employ the PLC's natural “Ripple-mode”. A single PFU can support up to 8-bit ripple functions with fast carry logic. Figure 16-8 is an example of 4-bit counter in PLC “Ripple Mode”. In Lattice Semiconductor FPGA architectures, an internal generated clock can get on the clock spine for small skew clock distribution, further enhancing the performance of the clock divider.

Figure 16-8. Use PLC “Ripple Mode”



Here are the HDL representations of the design in Figure 16-8.

```
-- VHDL : “RippleMode” Clock Divider
...

COUNT4: process(CLK, RST)
begin
   if (RST='1') then
      cnt <= (others=>'0');
   elsif (CLK'event and CLK='1') then
      cnt <= cnt + 1;
   end if;
end process COUNT4;

DIVBY4  <= cnt(1);
DIVBY16 <= cnt(3);
```

```
//Verilog : “RippleMode” Clock Divider
...
always @(posedge CLK or posedge RST)
begin
   if (RST)
      cnt = 4'b0;
   else
      cnt = cnt + 1'b1;
end

assign DIVBY4  = cnt[1];
assign DIVBY16 = cnt[3];

...
```

Register Control Signals

The general-purpose latches/FFs in the PFU are used in a variety of configurations depending on device family. For example, the Lattice EC, ECP, SC and XP family of devices clock, clock enable and LSR control can be applied to the registers on a slice basis. Each slice contains two LUT4 lookup tables feeding two registers (programmed asto be in FF or Latch mode), and some associated logic that allows the LUTs to be combined to perform functions such as LUT5, LUT6, LUT7 and LUT8. There is control logic to perform set/reset functions (prgorammable as synchronous/asynchronous), clock select, chip-select and wider RAM/ROM functions. The ORCA Series 4 family of devices clock, clock enable and LSR control can be applied to the registers on a nibble-wide basis. When writing design codes in HDL, keep the architecture in mind to avoid wasting resources in the device. Here are several points for consideration:

- If the register number is not a multiple of 2 or 4 (dependent on device family), try to code the registers in a way that all registers share the same clock, and in a way that all registers share the same control signals.
- Lattice Semiconductor FPGA devices have multiple dedicated Clock Enable signals per PFU. Try to code the asynchronous clocks as clock enables, so that PFU clock signals can be released to use global low-skew clocks.
- Try to code the registers with Local synchronous Set/Reset and Global asynchronous Set/Reset

For more detailed architecture information, refer to the Lattice Semiconductor FPGA data sheets.

Clock Enable

Figure 16-9 shows an example of gated clocking. Gating clock is not encouraged in digital designs because it may cause timing issues such as unexpected clock skews. The structure of the PFU makes the gating clock even more undesirable since it will use up all the clock resources in one PFU and sometimes waste the FF/ Latches resources in the PFU. By using the clock enable in the PFU, the same functionality can be achieved without worrying about timing issues as only one signal is controlling the clock. Since only one clock is used in the PFU, all related logic can be implemented in one block to achieve better performance. Figure 16-10 shows the design with clock enable signal being used.

Figure 16-9. Asynchronous: Gated Clocking

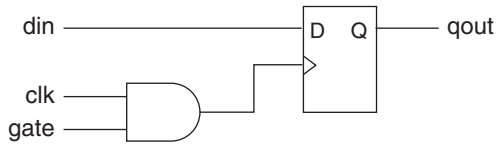
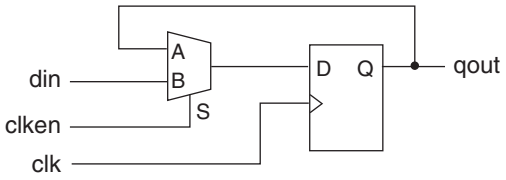


Figure 16-10. Synchronous: Clock Enabling



The VHDL and Verilog coding for Clock Enable are as shown in Figure 16-10.

```
-- VHDL example for Clock Enable

...
Clock_Enable: process(clk)
begin
   if (clk'event or clk='1') then
      if (clken='1') then
         qout <= din;
      end if;
   end if;
end process Clock_Enable;
```

```
// Verilog example for Clock Enable

...
always @(posedge clk)
   qout <= clken ? din : qout;

...
```

The following are guidelines for coding the Clock Enable in Lattice Semiconductor FPGAs:

- Clock Enable is only supported by FFs, not latches.
- Nibble wide FFs and slices inside a PLC share the same Clock Enable
- All flip-flops in the Lattice Semiconductor FPGA library have a positive clock enable signal
- In the ORCA Series 4 architecture, the Clock Enable signal has the higher priority over synchronous set/reset by default. However, it can be programmed to have the priority of synchronous LSR over the priority of Clock Enable. This can be achieved by instantiating the library element in the source code. For example, the library element FD1P3IX is a flip-flop that allows synchronous Clear to override Clock Enable. Users can also specify the priority of generic coding by setting the priority of the control signals differently. The following examples demonstrate coding methodologies to help the synthesis tools to set the higher priority of Clock Enable or synchronous LSR.

```
-- VHDL Example of CE over Sync. LSR
...
COUNT8: process(CLK, GRST)
begin
   if (GRST = '1') then
      cnt <= (others => '0');
   elsif (CLK'event and CLK='1') then
   -- CE Over LSR: Clock Enable has higher priority
      if (CKEN = '1') then
            cnt <= cnt + 1;
      elsif (LRST = '1') then
            cnt <= (others =>'0');
      end if;
   end if;
end process COUNT8;
```

```
// Verilog Example of CE over Sync. LSR
...
always @(posedge CLK or posedge GRST)
begin
   if (GRST)
      cnt = 4'b0;
   else
      if (CKEN)
         cnt = cnt + 1'b1;
      else if (LRST)
         cnt = 4'b0;
end...
```

```
-- VHDL Example of Sync. LSR Over CE
...
COUNT8: process(CLK, GRST)
begin
   if (GRST = '1') then
      cnt <= (others => '0');
   elsif (CLK'event and CLK='1') then
   -- LSR over CE: Sync. Set/Reset has higher priority
      if (LRST = '1') then
         cnt <= (others => '0');
      elsif (CKEN = '1') then
         cnt <= cnt + 1;
      end if;
```

```
// Verilog Example of Sync. LSR Over CE
...
always @(posedge CLK or posedge GRST)
begin
   if (GRST)
      cnt = 4'b0;
   else if (LRST)
      cnt = 4'b0;
   else if (CKEN)
      cnt = cnt + 1'b1;
end
...
```

SET / Reset

There are two types of set/reset functions in Lattice Semiconductor FPGAs: Global (GSR) and Local (LSR). The GSR signal is asynchronous and is used to initialize all registers during configuration. It can be activated either by an external dedicated pin or from internal logic after configuration. The local SET/Reset signal may be synchronous or asynchronous. GSR is pulsed at power up to either set or reset the registers depending on the configuration of the device. Since the GSR signal has dedicated routing resources that connect to the set and reset pin of the flip-flops, it saves general-purpose routing and buffering resources and improves overall performance. If asynchronous reset is used in the design, it is recommended to use the GSR for this function, if possible. The reset signal can be forced to be GSR by the instantiation library element. Synthesis tools will automatically infer GSR if all registers in

the design are asynchronously set or reset by the same wire. The following examples show the correct syntax for instantiating GSR in the VHDL and Verilog codes.

```
-- VHDL Example of GSR Instantiation
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity gsr_test is
          port (rst, clk: in std_logic;
          cntout : out std_logic_vector(1 downto 0));
end gsr_test;

architecture behave of gsr_test is
          signal cnt : std_logic_vector(1 downto 0);
begin

          u1: GSR port map (gsr=>rst);

          process(clk, rst)
          begin
                    if rst = '1' then
                              cnt <= "00";
                    elsif rising_edge (clk) then
                              cnt <= cnt + 1;
                    end if;
          end process;
          cntout <= cnt;
end behave;
```

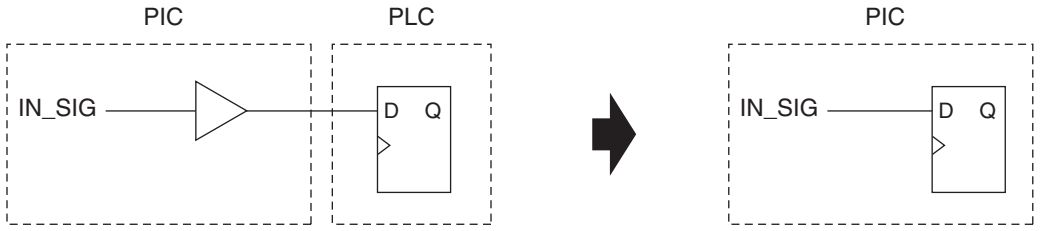
```
// Verilog Example of GSR Instantiation

module gsr_test(clk, rst, cntout);

input clk, rst;
output[1:0] cntout;

reg[1:0] cnt;

GSR u1 (.GSR(rst));

always @(posedge clk or negedge rst)
begin
          if (!rst)
                    cnt = 2'b0;
          else
                    cnt = cnt + 1;
end

assign cntout = cnt;

endmodule
```

Use PIC Features

Using I/O Registers/Latches in PIC

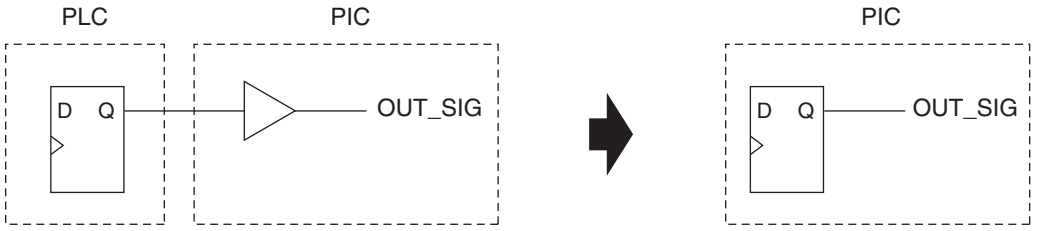
Moving registers or latches into Input/Output cells (PIC) may reduce the number of PLCs used and decrease routing congestion. In addition, it reduces setup time requirements for incoming data and clock-to-output delay for output data, as shown in Figure 16-11. Most synthesis tools will infer input registers or output registers in PIC if possible. Users can set synthesis attributes in the specific tools to turn off the auto-infer capability. Users can also instantiate library elements to control the implementation of PIC resource usage.

Figure 16-11. Moving FF into PIC Input Register



Before Using Input Register **After Using Input Register**

Figure 16-12. Moving FF into PIC Output Register



Before Using Output Register **After Using Output Register**

Inferring Bi-directional I/O

Users can either structurally instantiate the bi-directional I/O library elements, or behaviorally describe the I/O paths to infer bi-directional buffers. The following VHDL and Verilog examples show how to infer bi-directional I/O buffers.

```
-- Inferring Bi-directional I/O in VHDL

library ieee;
use ieee.std_logic_1164.all;

entity bidir_infer is
   port(A, B   : inout std_logic;
        dir    : in std_logic);
end bidir_infer;

architecture lattice_fpga of bidir_infer is
begin
   B <= A when (dir='1') else 'Z';
   A <= B when (dir='0') else 'Z';
end lattice_fpga
```

```
// Inferring Bi-directional I/O in Verilog

module bidir_infer (A, B, DIR);
inout A, B;
input DIR;

assign B = (DIR)  ? A : 1'bz;
assign A = (~DIR) ? B : 1'bz;

endmodule
```

Specifying I/O Types and Locations

Users can either assign I/O types and unique I/O locations in the Preference Editor or specify them as attributes in the VHDL or Verilog source code. The following examples show how to add attributes in the Synplify and LeonardoSpectrum synthesis tool sets. For a complete list of supported attributes, refer to the HDL Attributes section of the ispLEVER on-line help system.

-- VHDL example of specifying I/O type and location attributes for Synplify & Leonardo

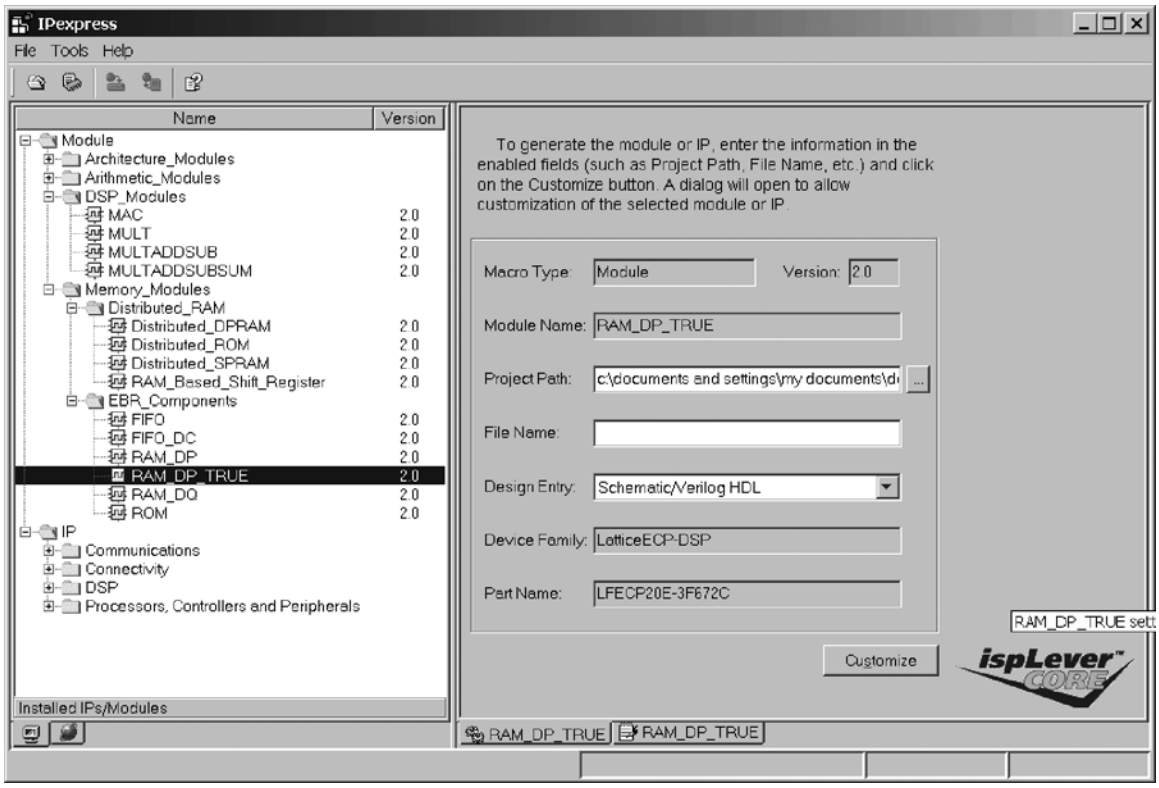
```
entity cnt is
     port(clk: in std_logic;
          res: out std_logic);
     attribute LEVELMODE: string:
     attribute LEVELMODE of clk : signal is “SSTL2”;
     attribute LOC of clk : signal is “V2”;
     attribute LEVELMODE of res : signal is “SSTL2”;
     attribute LOC of res : signal is “V3”;
end entity cnt;
```

-- Verilog example of specifying I/O type and location attributes for Synplify & Leonardo

```
module cnt(clk,res);

     input clk /* synthesis LEVELMODE=”SSTL2” LOC=”V2”*/;
     output res /* synthesis LEVELMODE=”SSTL2” LOC=”V3” */;

     ...

     // exemplar begin
          // exemplar attribute clk LEVELMODE SSTL2
          // exemplar attribute clk LOC V2
          // exemplar attribute res LEVELMODE SSTL2
          // exemplar attribute res LOC V3
     // exemplar end

endmodule
```

Implementation of Memories

Although an RTL description of RAM is portable and the coding is straightforward, it is not recommended because the structure of RAM blocks in every architecture is unique. Synthesis tools are not optimized to handle RAM implementation and thus generate inefficient netlists for device fitting. For Lattice Semiconductor FPGA devices, RAM blocks should be generated through IPexpress as shown in the following screen shot.



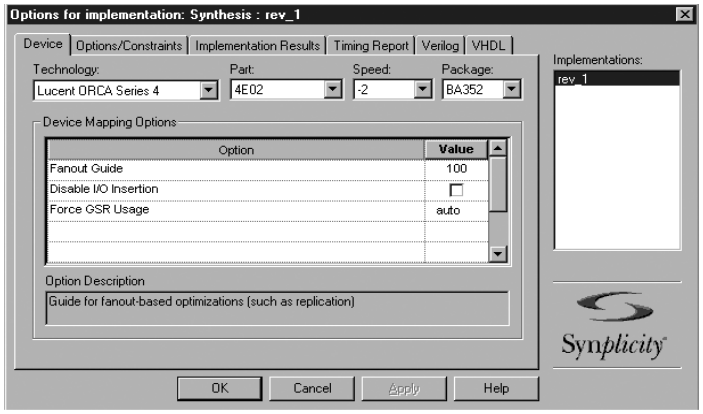
When implementing large memories in the design, it is recommended to construct the memory from the Enhanced Block RAM (EBR) components found in every Lattice Semiconductor FPGA device. When implementing small memories in the design, it is recommended to construct the memory from the resources in the PFU. The memory utilizing resources in the PFU can also be generated by IPexpress.

Lattice Semiconductor FPGAs support many different memory types including synchronous dual-port RAM, synchronous single-port RAM, synchronous FIFO and synchronous ROM. For more information on supported memory types per FPGA architecture, please consult the Lattice Semiconductor FPGA data sheets.

Preventing Logic Replication and Limited Fanout

Lattice Semiconductor FPGA device architectures are designed to handle high signal fanouts. When users make use of clock resources, there will be no hindrance on fanout problems. However, synthesis tools tend to replicate logic to reduce fanout during logic synthesis. For example, if the code implies Clock Enable and is synthesized with speed constraints, the synthesis tool may replicate the Clock Enable logic. This kind of logic replication occupies more resources in the devices and makes performance checking more difficult. It is recommended to control the logic replication in synthesis process by using attributes for high fanout limit.

In the Synplicity® project GUI, under the Implementation Options => Devices tab, users can set the Fanout Guide value to 1000 instead of using the default value of 100. This will guide the tool to allow high fanout signals without replicating the logic. In the LeonardoSpectrum tool project GUI, under Technology => Advanced Settings, users can set the Max Fanout to be any number instead of the default value “0”.



Use ispLEVER Project Navigator Results for Device Utilization and Performance

Many synthesis tools give usage reports at the end of a successful synthesis. These reports show the name and the number of library elements used in the design. The data in these reports do not represent the actual implementation of the design in the final Place and Route tool because the EDIF netlist will be further optimized during Mapping and Place and Route to achieve the best results. It is strongly recommended to use the MAP report and the PAR report in the ispLEVER Project Navigator tool to understand the actual resource utilization in the device. Although the synthesis report also provides a performance summary, the timing information is based on estimated logic delays only. The Place & Route TRACE Report in the ispLEVER Project Navigator gives accurate performance analysis of the design by including actual logic and routing delays in the paths.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Lattice Semiconductor Design Floorplanning

July 2004 Technical Note TN1010

Introduction

Lattice Semiconductor's ispLEVER® software, together with Lattice Semiconductor's catalog of programmable devices, provides options to help meet design timing and logic utilization requirements. Additionally, for those instances where objectives push the capabilities of the device architecture, ispLEVER provides the tools for meeting the most challenging requirements.

For the most aggressive design requirements, the designer should become familiar with a variety of timing constraints (called preferences) and Place And Route (PAR) techniques for providing the optimal PAR results as discussed in technical note number TN1018, Lattice Semiconductor Successful Place and Route.

If performance goals cannot be met with FPGA timing preferences and additional levels of the Place & Route design process, improved performance can be achieved by directing the physical layout of the circuit in the FPGA. This step, often referred to as floorplanning, is done by specifying FPGA location preferences.

This application note explains what floorplanning is, when it should be used, and how it is done with respect to Lattice Semiconductor FPGA designs. This document is divided into four major sections:

- Floorplanning definition, logical and physical
- When to floorplan, general versus specific reasons
- How to floorplan: grouping constraints, examples, and Floorplanner GUI presentation
- Special considerations, large and special groupings

Supported Architectures

Floorplanning can be done on all Lattice Semiconductor FPGA architectures.

Related Documentation

Designers are encouraged to reference the ispLEVER on-line documentation.

Floorplanning Definition

Floorplanning a digital logic design for implementation in a programmable device involves the physical or logical partitioning of the design elements which results in a change in the physical placement or implementation of the design elements. In other words, floorplanning is the grouping of design elements in a certain way to improve the performance of a design.

With Lattice Semiconductor FPGAs, floorplanning is an optional methodology to help designers improve the performance and density of a fully, automatically placed and routed design. Floorplanning is particularly useful in structured designs and data path logic. Design floorplanning is very powerful and provides a combination of automation and user control for design reuse and modular, hierarchical, and incremental design flows.

Complex FPGA Design Management

Lattice Semiconductor FPGAs can implement large system designs that contain millions of gates, hundreds of thousands of embedded memory bits, and intellectual property (IP) components. Design teams often work on large designs. The design complexity requires EDA tools to manage and optimize the design. Large design management is difficult, but performance optimization is even more difficult. Optimization requires many design iterations when adding or modifying components. Complex, large system designs require the following:

- The use of modular, hierarchical, or incremental design methods
- Software that makes management and optimization easier

- Use of IP blocks
- Reuse of previously optimized design elements

By controlling the placement of specified logic elements, design floorplanning methodologies help designers meet the requirements of large system design.

Floorplanning Design Flow

In both traditional and floorplanning FPGA design flows, the designer divides the system into modules. The modules can be individual circuits, parts of circuits, or parts of the design hierarchy. After module design and optimization, the designer integrates the modules into the system. Finally, the designer tests and optimizes the system.

In the traditional flow, the system may not meet performance requirements even if each module meets the requirements before integration. Even when timing requirements have been satisfied, changes to one module can affect the performance of others. Re-optimizing modules to meet system performance results in many design iterations.

Floorplanning methodologies assist in the design, testing, and optimization of each individual module while retaining the optimized characteristics of the individual modules. Module integration into the system requires only system optimization between modules. The floorplanning methodologies provide additional flexibility by allowing the ispLEVER software to automatically place defined modules, or allowing the user to control the placement of specific modules, which provide performance preservation and optimization.

When to Floorplan

Floorplanning methodologies are intended to assist users who require some degree of handcrafting for their designs. The designer must understand both the details of the device architectures and the ways floorplanning can be used to refine a design. Successful floorplanning is very much an iterative process and it can take time to develop a floorplan that outperforms an automatic, software processed design. Because of the nature of floorplanning and its interaction with the automatic MAP and PAR software tools, several prerequisites are necessary in order to floorplan a design successfully.

- Detailed knowledge of the specifics of the target architecture and device
- Detailed knowledge of the specifics of the design being implemented
- A design that lends itself to floorplanning
- A willingness to iterate a floorplan to achieve the desired results
- Realistic performance and density goals

For Lattice Semiconductor FPGAs, the general rule-of-thumb is that floorplanning should be considered when the performance needed cannot be met and routing delays account for over 60% of the critical path delays. That is, interacting components are too far apart in the FPGA array to achieve short routing delays. This has shown to be a problem especially with large designs in high density FPGAs because of the possibilities of long distance routes. As programmable logic design densities continue to escalate beyond 100,000 gates, traditional design flow — design entry to synthesis to place and route — will sometimes not yield predictable, timely, and optimized results.

Note that the guidelines discussed above only apply to designs that have been routed by the software for several routing iterations. The default number of routing iterations via the ispLEVER Project Navigator is variable depending on the Lattice Semiconductor FPGA device family chosen.

A note on delays: Path delays in programmable devices are made up of two parts: logical delays and routing delays. Logical delays in this context are delays through components, such as a a programmable function unit (PFU), a programmable input/output (PIO), a slice, or an embedded function (i.e. a block RAM, PLL, or embedded FPSC ASIC). The routing delay is the delay of the interconnect between components. Figures 1 and 2 show delay examples from timing wizard report files (.twr).

Figure 17-1. Floorplanning May be Able to Help Bring These Registers Closer

```
Logical Details:  Cell type  Pin type   Cell name  (clock net +/-)

 Source:          FF         Q          ibuf/reg_init_start (from clk_ib+)
 Destination:     FF         Data in    ibuf/sd/reg_new_state (to clk_ib +)

 Delay:               8.062ns  (18.2% logic, 81.8% route), 2 logic levels.
```

Figure 17-2. Floorplanning is Not Needed Here Because the Routing is Efficient

```
Logical Details:  Cell type  Pin type   Cell name  (clock net +/-)

 Source:          FF         Q          mem_if_tx_address_8 (from clk_c +)
 Destination:     FF         Data in    mem_if_tx_address_17 (to clk_c +)

 Delay:               7.358ns  (61.2% logic, 38.8% route), 4 logic levels.
```

Floorplan to Improve Design Performance

Properly applied, design floorplanning not only preserves, but improves design performance. Floorplanning methodologies can be used to place modules, entities, or any group of logic to regions in a device's floorplan. Because floorplanning assignments can be hierarchical, designers can have more control over the placement and performance of modules and groups of modules.

In addition to hierarchical blocks, like grouping an entire VHDL entity or Verilog module, designers can floorplan individual nodes (e.g., instantiate a library element for a function in the critical path and then group the library element). This technique is useful if the critical path spans multiple design blocks.

Note: Although floorplanning can increase performance, it may also degrade performance if it is not applied correctly within software limitations.

Floorplan to Preserve Module Performance

Floorplanning with design constraints maintains design performance by grouping the placement of nodes in a device, i.e., the relative placement of logic within a grouped region remains constant. The ispLEVER software then places the grouped region into the top-level design with these constraints. When placing logic in a region, the ispLEVER software does not preserve the routing information. This approach provides more flexibility when the software imports the region into the top-level design and helps fitting.

Floorplan for Design Reuse

Floorplanning facilitates design reuse by its ability to reproduce the performance of a module designed in a different project. For frequently used modules, designers can create a library of verified designs that can be incorporated into other larger designs. The library only has to contain the VHDL or Verilog source code along with grouping attributes and some comments detailing useful information to the user, such as performance and size. With a parameterized module, in-code assignments can specify the module's size and grouping assignments.

Targeting the same device used in the original design likely achieves the best results, although other devices in the same family will likely work well. When using a different device in the same family, the exact placement of the region may not be possible. Similar performance, however, may be possible by moving or floating regions. A floating region groups the logic together and guides the ispLEVER software toward achieving a placement that meets the performance requirements of the module. A similar approach can also be taken if exact placement of a module is not applicable because of multiple instantiations of a module in a top-level design.

How to Floorplan a Design

Design Performance Enhancement Strategies

Floorplanning methodologies improve the performance of designs that do not necessarily consist of individually optimized modules. The ability of specifying regions to group nodes together and provide relative placement enhances the usability of the ispLEVER place-and-route software tools. The design strategies for performance enhancement depend on the structure of a particular circuit. Strategies include:

- Defining regions based on design hierarchy if the hierarchy closely resembles the structure of the circuit. These designs typically consist of tightly integrated modules, where the logic for each module is self-contained and modules communicate through well-defined interfaces.
- Defining regions based on the critical path, if the critical path is long and spans multiple modules. Keeping the nodes in the critical path or the modules containing the critical path together may lead to improved performance.
- Defining regions based on connections by grouping nodes with high fan-outs and high fan-ins together to reduce delays in connections and wiring congestion in the device.

Note that designers may need to change existing design hierarchy and structure to make the design more amiable to floorplanning. This is especially applicable if modular hierarchy and structure was not considered at the beginning of design conception.

With the floorplaning methodologies, the user can choose to optimize modules either individually or after they have been integrated with the top-level design. The user can exercise varying amounts of control over the placement by using different types of regions. By using bounding boxes and location anchors selectively, the ispLEVER software can easily determine the best size and location for a region. Another approach is to optimize the top-level design without first optimizing the individual modules. This approach allows the ispLEVER software to place nodes within regions and move regions across the device. The user assigns modules to regions and then compiles the entire design. With this approach the user can place elements from different modules in a region.

Design Floorplanning Methodologies

There are several methods available to aid in the floorplanning of a logical design in a Lattice Semiconductor FPGA. This section illustrates these methods with examples of how to use the ispLEVER software tools to achieve performance goals. The three main floorplanning tools available include:

- The **PGROUP Physical Constraint** can be used as an attribute in VHDL and Verilog HDL source code or as a physical Preference in the .prf design constraint file. This can be used directly by the ispLEVER Placer software to bound and locate sections of a design for grouping in the FPGA array.
- The **UGROUP Logical Constraint** can be used as an attribute in VHDL and Verilog HDL source code to gather logical sections of a design for grouping in the FPGA array.
- The **Floorplanner Graphical User Interface (GUI)** can be used to interactively specify placement parameters in either the logical or physical domain for some of a design's modules (e.g. logic gates, registers, arithmetic functions) from one graphical user interface.

When to use PGROUP vs. UGROUP

UGROUPing differs from PGROUPing as follows:

- A PGROUP logical identifier, in EDIF, is prepended with text that describes the identifier's hierarchy.
- A UGROUP logical identifier, in EDIF, is not changed by prepending the hierarchy on the block instance identifier.

In other words, PGROUPing enforces strict hierarchical control while UGROUPing allows for a grouping of blocks in different hierarchies or a grouping of blocks with no hierarchy at all.

Also note that the PGROUP attribute can be placed on multiple instantiations of modules (e.g. VHDL generate statements) and each instantiation will have its own PGROUP. Using a UGROUP will not work in this case.

In Figures 3 and 4, the arrows represent control and data paths where there is interaction between different levels of hierarchy. The thick lined arrow represents the critical path where the design is failing to make performance.

Figure 17-3. PGROUP Same Hierarchy Example, PGROUP CONTROLLER

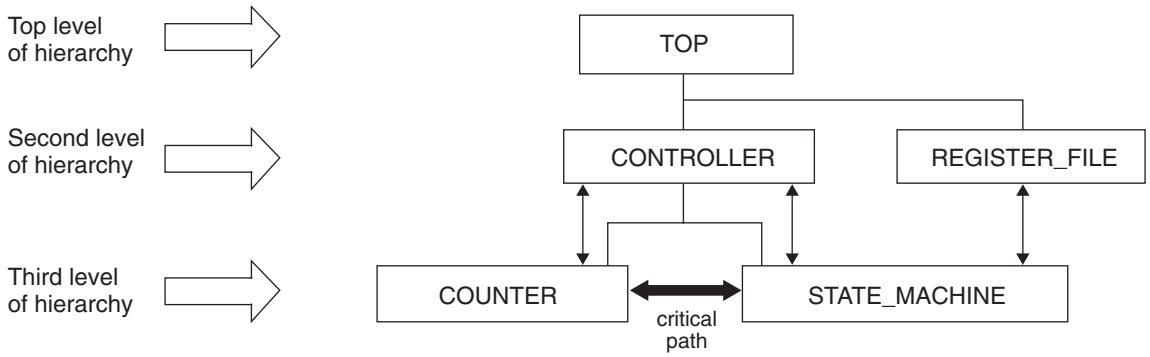


Figure 17-3 illustrates a design hierarchy where the failing paths are the connections between COUNTER and STATE_MACHINE design blocks. The easiest implementation for this example is to PGROUP the CONTROLLER, which is the module in which the COUNTER and STATE_MACHINE are instantiated within.

For example, if the following Synplify attribute is in the Verilog HDL file:

```
module CONTROLLER (<port_list>)/* synthesis pgroup="CONTROL_GROUP" */;
```

then the COUNTER and STATE_MACHINE will be grouped in the FPGA inside a boundary box. Now assume that the COUNTER is mapped into PFU_0 and PFU_1 and the STATE_MACHINE is mapped into PFU_2. The resulting preference generated by map in the `.prf` file will be:

```
PGROUP "TOP/CONTROLLER/CONTROL_GROUP"
   COMP "PFU_0"
   COMP "PFU_1"
   COMP "PFU_2";
```

Notice the `TOP/` hierarchy is prepended to the CONTROLLER PGROUP identifier.

Figure 17-4. UGROUP Different Hierarchy Example, UGROUP REGISTER_FILE and STATE MACHINE

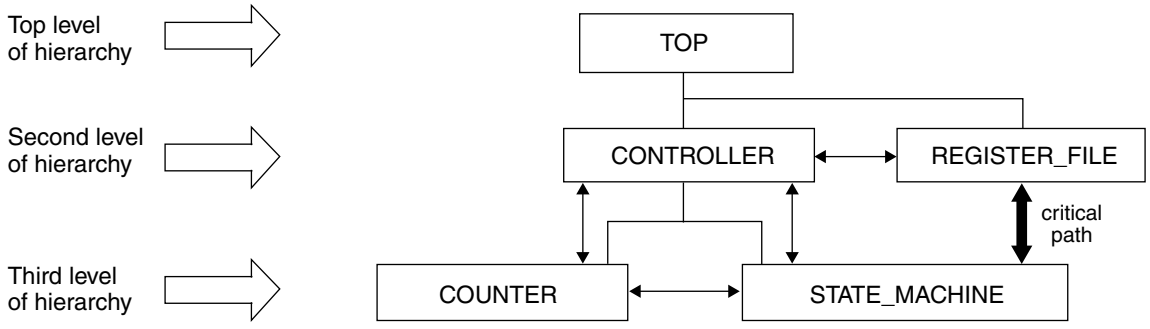


Figure 17-4 shows an example design hierarchy where the failing paths are the connections between REGISTER_FILE and STATE_MACHINE modules. The simplest thing to do here is to UGROUP the REGISTER_FILE and STATE_MACHINE together.

For example, if the following Synplify attributes are in the Verilog HDL file:

```
module REGISTER_FILE (<port_list>) /*synthesis ugroup="CRITICAL_GROUP" */;
```

and

```
module STATE_MACHINE (<port_list>) /*synthesis ugroup="CRITICAL_GROUP" */;
```

then the REGISTER_FILE and STATE_MACHINE will be grouped in the FPGA inside a default boundary box. Now assume that the REGISTER_FILE is mapped into PFU_4 and PFU_5 and the STATE_MACHINE is mapped into PFU_3. The resulting preference generated by map in the `.prf` file will be:

```
PGROUP "CRITICAL_GROUP"
   COMP "PFU_3"
   COMP "PFU_4"
   COMP "PFU_5";
```

Notice the `TOP/` hierarchy is not appended to the PGROUP identifier CRITICAL_GROUP. Also notice that UGROUP attributes result in PGROUP preferences. There is no UGROUP preference.

If PGROUP attributes instead of UGROUP attributes had been used for Figure 17-4:

```
module REGISTER_FILE (<port_list>) /*synthesis pgroup="CRITICAL_GROUP" */;
```

and

```
module STATE_MACHINE (<port_list>) /*synthesis pgroup="CRITICAL_GROUP" */;
```

then the resulting preference generated by map in the `.prf` file would be:

```
PGROUP "TOP/CONTROLLER/STATE_MACHINE/CRITICAL_GROUP"
   COMP "PFU_3"

PGROUP "TOP/REGISTER_FILE/CRITICAL_GROUP"
   COMP "PFU_4"
   COMP "PFU_5";
```

So, with PGROUP attributes, the STATE_MACHINE module would be grouped together in one bounding box and REGISTER_FILE module would be grouped together separately in another bounding box and the critical path shown in Figure 17-4 will not be optimized.

These examples do not utilize all the possible tools available for floorplanning. Please refer to ispLEVER On-line Help PGROUP section for many small syntax examples.

Floorplanner GUI Usage

Generally, the PGROUPs and UGROUPs are preferable to the Floorplanner GUI since they are easier to implement. For example, it is easier to type in a PGROUP attribute in the HDL code then to load the GUI with large netlists and find the desired block and perform add the PGROUP via mouse clicks. More importantly, the GUI does not allow the retention of floorplanning the way PGROUPing and UGROUPing does. Since the GUI does not back annotate the grouping attributes into the HDL, the GUI operations have to be redone every time there is a new design iteration.

The Floorplanner GUI can be useful for

- Viewing elements in a graphical environment to see a design's logical hierarchy.
- Viewing existing PGROUPs and UGROUPs.
- Resizing regions and boundary boxes (BBOXes).
- Graphically placing regions, PGROUPs, and UGROUPs and then running map, place, route, and trace to see the effects. This is usually an iterative process before finding an optimal solution.

If the Floorplanner GUI is used, it is strongly recommended that after finding the optimal grouping with the GUI, grouping attributes (PGROUP and/or UGROUPS) should be inserted into the HDL source code to preserve module performance over design revisions.

Special Floorplanning Considerations

Embedded Block RAM Placement

Block RAM placement can be done with simple LOCATE preferences. It is not always necessary to locate block RAMs. Do not use the PGROUPs, UGROUPs, or the Floorplanner GUI to group Block RAMs.

I/O Grouping

There is a complete set of physical constraints on PGROUPing I/O components. Please refer to ispLEVER On-line Help, Defining PIO Component Groups, for keyword explanations and syntax examples.

Large Module Grouping

It is strongly recommended that larger PGROUPs/UGROUPs (with many logical elements) be anchored and bounded by LOCATE and BBOX keywords. From the STATE_MACHINE example, we see that without anchoring the groups, the performance worsened compared to no floorplanning at all.

The BBOX should be strategically shaped and sized according to the module to be placed inside the BBOX. If the BBOX shape and size is not specified, the default BBOX size will be a square that is as small as possible. This is not the optimal BBOX for typical modules. The designer should shape the design with the datapath in mind and size the BBOX to be larger then needed so that the ispLEVER placer program can have more flexibility in placing logic elements inside the BBOX.

Carry Chains and Bus Grouping

Carry chains (used by ripple arithmetic functions like adders, counters, and multipliers) and logic modules connected by busses can easily be floorplanned inappropriately by a designer that is not aware of the internal routing resources available to optimize these carry chain and bus routes. As we saw from the multiplier example, certain groupings can reduce the performance of a design compared to no floorplanning at all. Great care should be used when floorplanning designs that use carry chains or busses so that these routes fall in optimal locations for optimal performance.

Broken Carry Chain Example

A 9-bit adder that is PGROUPed with no relative placement on the adder. Logic elements such as PFUs may give worse performance because the adder carry-chain is broken.

SLICs in Groups

For Lattice Semiconductor FPGA device families that contain Supplemental Logic and Interconnect Cells (SLICs), the SLICs are automatically removed from PGROUPs and UGROUPs by the ispLEVER software if they are not relatively placed. This is because SLICs are used by the tools for interconnects that are not foreseeable by designers. If SLIC placement has to be controlled for a design, the designer will need to instantiate and locate the SLICS in their preference or HDL files. It is recommended to allow the ispLEVER software to automatically place SLICs.

Summary

This application note defined floorplanning, discussed when it should be used, and detailed how floorplanning is done with respect to Lattice Semiconductor FPGA designs. Examples were used to illustrate and compare the different tools available to the designer for floorplanning.

Important items discussed:

- Floorplanning can improve timing for targeted critical paths.
- Improper floorplanning can make timing worse.

- Correct floorplanning can increase logic element count slightly.
- Completed floorplanning should be annotated into the HDL.
- Floorplanning enhances the reusability of designs by keeping placement directives in the HDL.
- Use PGROUPs for physical grouping in the same hierarchy.
- Use UGROUPs for logical grouping across hierarchies.

With Lattice Semiconductor FPGAs, floorplanning is an *optional* methodology to help designers improve performance and density of a fully, automatically placed and routed design. Floorplanning is particularly useful on structured designs and data path logic.

Design floorplanning is very powerful and provides a combination of automation and user control for design reuse and modular, hierarchical, and incremental design flows. It advances the design of large systems on FPGAs. It provides the designer with capabilities in the management and optimization of large systems. Its multifaceted capabilities result in shortened design cycles and faster time to market.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Lattice Semiconductor FPGA Successful Place and Route

July 2004 Technical Note TN1018

Introduction

Lattice Semiconductor's ispLEVER® software, together with Lattice Semiconductor's catalog of programmable devices, provides options to help meet design timing and logic utilization requirements. Additionally, for those instances where objectives push the capabilities of the device architecture, ispLEVER provides the tools for meeting the most challenging requirements.

For the most aggressive design requirements, the designer should become familiar with a variety of timing constraints (called preferences) and Place And Route (PAR) techniques for providing the optimal PAR results. This document describes these tips and techniques. Advanced techniques in floorplanning will not be discussed in this document. Instead they are covered in technical note number TN1010, *Lattice Semiconductor Design Floorplanning*.

ispLEVER Place and Route Software (PAR)

In the ispLEVER design flow, after a design has undergone the necessary translation to bring it into the mapped physical design (.ncd file) format, it is ready for placement and routing. This phase is handled by the timing-driven PAR software program. Designers can invoke PAR from the ispLEVER Project Navigator or from the command line.

PAR performs the following:

- Takes a mapped physical design (.ncd file) and a preference file (.prf) as input files.
- Places and routes the design, attempting to meet the timing preferences in the input .prf file.
- Creates a file which can then be processed by the ispLEVER design implementation tools.

Placement

The PAR process places the mapped physical design (.ncd file) in two stages: a constructive placement and an optimizing placement. PAR writes the physical design after each of these stages is complete.

During constructive placement, PAR places components into sites based on factors such as:

- Constraints specified in the input file (for example, certain components must be in certain locations).
- The length of connections.
- The available routing resources.
- Cost tables which assign random weighted values to each of the relevant factors. There are 100 possible cost tables.

Constructive placement continues until all components are placed. Optimizing placement is a fine-tuning of the results of the constructive placement.

Routing

Routing is also done in two stages: iterative routing and delay reduction routing (also called cleanup). PAR writes the physical design (.ncd file) only after iterations where the routing score has improved.

During iterative routing, the router performs an iterative procedure to converge on a solution that routes the design to completion or minimizes the number of unrouted nets.

During cleanup routing (also called delay reduction), the router takes the results of iterative routing and reroutes some connections to minimize the signal delays within the device. There are two types of cleanup routing that can be performed:

- A faster cost-based cleanup routing, which makes routing decisions by assigning weighted values to the factors (for example, the type of routing resources used) affecting delay times between sources and loads.
- A more CPU-intensive, delay-based cleanup routing, which makes routing decisions based on computed delay times between sources and loads on the routed nets.

Note that if PAR finds timing preferences in the preference file, timing-driven placement and routing is automatically invoked.

Timing Driven PAR Process

The ispLEVER software offers timing driven placement and routing through an integrated static timing analysis utility (i.e., it does not depend on input stimulus to the circuit). This means that placement and routing is executed according to timing constraints (preferences) that the designer specifies up front in the design process. PAR attempts to meet timing constraints in the preference file without exceeding the specified timing constraints.

To use timing-driven PAR, the designer simply writes timing preferences into a preference (.prf) file, which serves as input to the integrated static timing analysis utility. See the *Process Flows* section of the ispLEVER on-line help system for more information about the PAR software and ispLEVER design flow.

General Strategy Guidelines

Preferences should be inserted at the front end of a design flow. This prevents designers from having to change PAR physical preferences as net names may change with every synthesis run.

The tips below are general recommendations.

- Analyze Trace results in the integrated static timing analysis utility report (.twr) file carefully.
- Look at mapped frequency before you PAR a design to check for errors and warnings in the preference file and to check for logic depth. Logic depth is reported in .twr files as logic levels (components).
- Determine if design changes are required. A typical example design change is pipelining, or registering, the datapath. This technique may be the only way to achieve high internal frequencies if the designs logic levels are too deep.
- It is recommended to perform place and route early in the design phase with a preliminary preference file to gather information about the design.
- Tune up your preference file to include all I/O and internal timing paths as appropriate. The Translating Board Requirements into FPGA Preferences section of this document goes over an appropriate preference file example.
- Establish the pin-out in the preference file. Locating I/O can also be done in the HDL, as well as in synthesis constraint files.
- Push PAR when necessary by running multiple routing iterations and multiple placement iterations.
- Revise the preference file as appropriate, especially utilizing multicycle opportunities when possible.
- Floorplan the design if necessary (see technical note number TN1010, *Lattice Semiconductor Design Floorplanning*).
- For Lattice Semiconductor ORCA Series devices, use clock boosting as a last resort, remembering to run trace hold timing checks on the clock boosted design. Refer to the Clock Boosting section of this document for more information on clock boosting.

Typical Design Preferences

The full preference language includes many different design constraints from very global preferences to very specific preferences. To a new user this is a very large list to digest and utilize effectively. Listed here are the recommended preferences that should be applied to all designs. Refer to the *Constraints & Preferences* section of the ispLEVER on-line help system for more information on preferences.

- **Block Asynchronous Paths:** Prevents the timing tools from analyzing any paths from input pads to registers or from input pads to output pads.
- **Block RAM Reads during Write:** If using PFU based RAM, this will prevent timing analysis on a RAM read during a write on the same address in a single clock period.
- **Frequency/Period <net>:** Each clock net in the design should contain a frequency or period preference.
- **Input Setup:** Each synchronous input should have an input_setup preference.
- **Clock-to-Out:** Each synchronous output should have a clock_to_out preference.
- **Block <net>:** All asynchronous reset nets in the design should be blocked.
- **Multicycle:** The multicycle preference allows the designer to relax a frequency/period constraint on selected paths.

Proper Preferences

Providing proper preferences is key to a successful design. If the constraints of a preference file are tighter than the system requirements, the design will end up being over-constrained. As a consequence, PAR run times will be considerably longer. In addition, over-constraining non-critical paths will force PAR to waste unnecessary processing power trying to meet these constraints, hence creating possible conflicts with real critical paths that ought to be optimized first.

On the other hand, if a preference file is under-constrained compared to real system requirements, real timing issues not previously seen during dynamic timing simulations and static timing analysis could be observed on a test board, or during production.

Common causes of over-constrained timing preferences include:

- Multicycle paths not specified.
- Multiple paths to/from I/Os with different specifications.
- Attempt to fool the PAR tool with tighter than necessary specifications.

Note that over-constrained designs will also need a significantly larger amount of processing power and computing resources. As a result, it might be necessary to increase some of the allocated system resources (as in increasing your PC virtual memory paging size).

Common causes of under-constrained timing preferences include:

- I/O specifications not defined.
- Asynchronous logic without MAXDELAY preferences.
- Internally generated or unintentional clocks not specified in preference file.
- Blocking critical paths.

In general, to make sure that no critical paths were left out due to under-constraining, it is recommended to check for path coverage at the end of a Trace report file (.twr).

An example of such an output is shown in Figure 18-1.

Figure 18-1. Trace Report (.twr) Timing Summary Example

```
Timing summary:
---------------

Timing errors: 4096  Score: 25326584

Constraints cover 36575 paths, 6 nets, and 8635 connections (99.0% coverage)
```

This particular example shows a 99.0% coverage. The way to find unconstrained paths is to run Trace with the "Check Unconstrained Paths" checkbox selected. This will give a list of all of the signals that are not covered under timing analysis. In some designs, many of these signals are a common ground net that indeed does not need to be constrained. Designers should understand this point and use Trace (the ispLEVER static timing analysis tool) to check unconstrained paths to make sure they are not missing any design paths that are timing critical.

Also, note the timing score shown in Figure 18-1. The timing score shows the total amount of error (in picoseconds) for all timing preferences constraining the design. PAR attempts to minimize the timing score, PAR does not attempt to maximize frequency.

The above discussion can be summarized by the following single equality:

Quality of Preference File = Quality of PAR Results

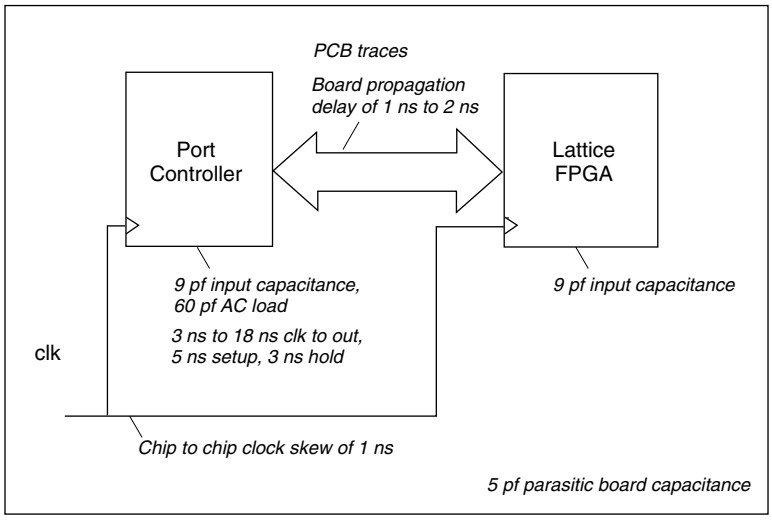
Translating Board Requirements into FPGA Preferences

Understanding the system board level timing and design constraints is the primary requirement for producing a complete preference file. As a result, the major requirements such as clock frequency, I/O timing and loads can be translated into the appropriate preference statements in a constraint file.

The following exercise will provide an example on how to extract preferences from system conditions.

Figure 18-2 shows an example system involving the interface between a port controller and a Lattice Semiconductor FPGA.

Figure 18-2. Interface Timing Example



In the system above, several parameters have already been provided:

- System clock frequency: period (P): 30 ns.
- Port controller maximum output propagation delay (PDMAXp): 18ns.
- Port controller minimum output propagation delay (PDMINp): 3 ns.
- Port controller input setup specification (TSp): 5 ns.
- Port controller input hold specification (THp): 3 ns.
- Max board propagation delay (PDMAXb): 6 ns.
- Min board propagation delay (PDMINb): 1 ns.
- Port controller to FPGA device clock skew and vice versa (Tskew): 1 ns.

- Board trace AC loading (Cbac): 60 pf.
- Board trace parasitic capacitance (Cb): 5 pf.
- Port controller input capacitance (Cp) :9 pf.
- FPGA device input capacitance (Co): 9 pf.

The above information was specified under the following environmental conditions:

- Maximum ambient temperature (Ta): 70 (C.
- Estimated Power Consumption (Q): 2 W.
- 680 PBGAM Package Thermal resistance (Φj) at 0 feet per minute (fpm) airflow: 13.4 °C/W.

The goal of this exercise is to compute the following device I/O constraints:

1. Input setup specification.
2. Input hold specification.
3. Maximum output propagation delay.
4. Minimum output propagation delay.
5. Output loading.
6. Temperature.

The only parameter which can be obtained from the above is the device junction temperature:

$$\begin{aligned} Tj &= \Phi j * Q - Ta \\ &= 13.4 * 2 + 70 \\ &= 96.8\ °C \end{aligned}$$

The required constraints can be computed as follows:

1. Input setup specification
$$\begin{aligned} &= P - PDMAXp - PDMAXb - Tskew \\ &= 30 - 18 - 2 - 1 \\ &= 9\ ns \end{aligned}$$

2. Input hold specification
$$\begin{aligned} &= PDMINp + PDMINb - Tskew \\ &= 3 + 1 - 1 \\ &= 3\ ns \end{aligned}$$

3. Output maximum propagation delay requirement
$$\begin{aligned} &= P - TSp - PDMAXb - Tskew \\ &= 30 - 5 - 6 - 1 \\ &= 18\ ns \end{aligned}$$

4. Output minimum propagation delay requirement
$$\begin{aligned} &= Thp - PDMINb + Tskew \\ &= 3 - 1 + 1 \\ &= 3\ ns \end{aligned}$$

5. Output loading
$$\begin{aligned} &= Cbac + Cb + Cp \\ &= 60 + 5 + 9 \\ &= 74\ pf \end{aligned}$$

The preference file to use for this example is shown in Figure 18-3. For more preference language syntax and examples, refer to the Constraints & Preferences section of the ispLEVER on-line help system.

Figure 18-3. Interface Timing Preference File Example

```
PERIOD PORT "clk" 30 NS ;
INPUT_SETUP "port_controller*" 9 NS HOLD 3 NS CLKNET "clk";
CLOCK_TO_OUT "port_controller*" 18 NS MIN 3 NS CLKNET "clk";
OUTPUT PORT "port_controller*" LOAD 74 PF ;
TEMPERATURE 96.8 C ;
```

Analyzing Timing Reports

This section describes two examples of actual Trace reports (.twr report file from Trace). The purpose is to analyze both examples and understand each section of the reports given the design paths constrained.

Example 1. Multicycle Between Two Different Clocks

In this first example, CLKA and CLKB were assigned 104 MHz and 66 MHz frequencies respectively.

In addition, a multicycle constraint was specified as per the preference file:

```
FREQUENCY  NET "CLKA" 104 MHZ ;
FREQUENCY NET "CLKB" 66 MHZ ;
MULTICYCLE "M2" START CLKNET "CLKA" END CLKNET "CLKB" 2.000000 X ;
```

See Figure 18-4 for the block diagram and waveform for this example. The resulting Trace report is shown in Figure 18-5.

Figure 18-4. Multicycle Clock Domains Block Diagram and Waveform

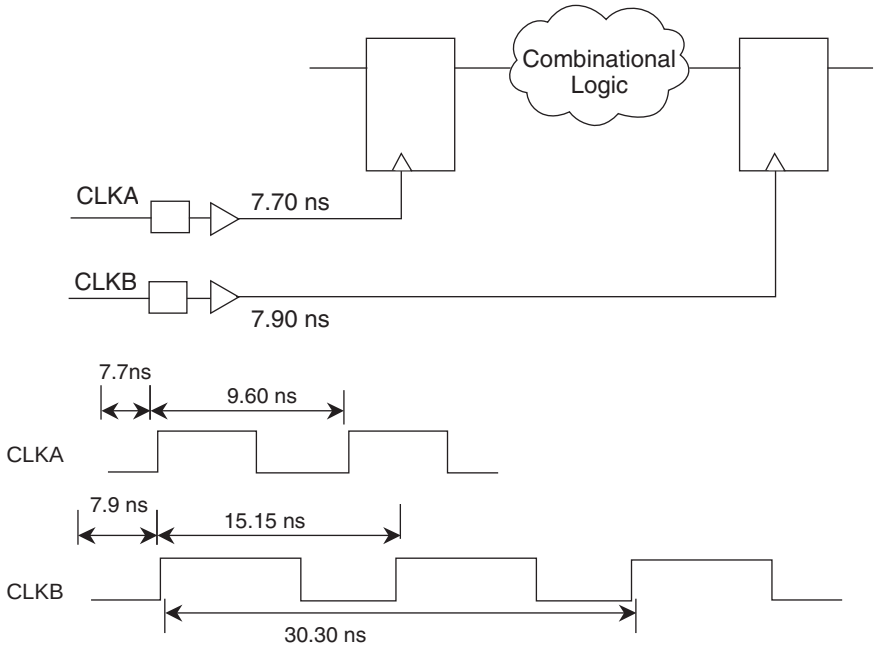


Figure 18-5. Trace Report for Multicycle Clock Domains Example

```
================================================================================
Preference: MULTICYCLE "M2" START CLKNET "CLKA"  END CLKNET "CLKB"  2.000000 X ;
            40 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------
WARNING - trce: Clock skew between net 'CLKA' and net 'CLKB' not
          computed: nets may not be related
--------------------------------------------------------------------------------


Passed:  The following path meets requirements by 27.945ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)

   Source:         FF         Q              v_fifo_bank_1_stfifo0_wr_count_2  (from CLKA +)
   Destination:    FF         Data in        v_fifo_bank_1_stfifo0_wr_count_r_2  (to CLKB +)

   Delay:               2.456ns  (37.3% logic, 62.7% route), 1 logic levels.

 Constraint Details:

      2.456ns physical path delay PFU_155 to PFU_156 meets
     30.302ns delay constraint less
     -0.099ns DIN_SET requirement (totaling 30.401ns) by 27.945ns

 Physical Path Details:

   Name    Fanout   Delay (ns)          Site               Resource
REG_DEL     ---     0.917    R22C16.CLK0 to      R22C16.Q2 PFU_155 (from CLKA)
ROUTE         1     1.539      R22C16.Q2 to    R23C17.DIN2 v_fifo_bank_1_stfifo0_wr_countZ0Z_2 (to CLKB)
                  --------
                    2.456   (37.3% logic, 62.7% route), 1 logic levels.

 Clock Skew Details:

 Source Clock Path:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     1.192       AM17.PAD to      AM17.INDD ip_CLKA
ROUTE         1     2.989      AM17.INDD to   LLPPLL.CLKIN ip_CLKA_c
MCLK_DEL    ---     0.424   LLPPLL.CLKIN to    LLPPLL.MCLK v_io_ppl3_tx4_1_mtppll_rsp_rsppll_0_0
ROUTE       177     3.094    LLPPLL.MCLK to    R22C16.CLK0 CLKA
                  --------
                    7.699   (21.0% logic, 79.0% route), 2 logic levels.

 Destination Clock Path:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     1.192        C17.PAD to       C17.INDD ip_CLKB
ROUTE         1     3.091       C17.INDD to   ULPPLL.CLKIN ip_CLKB_c
MCLK_DEL    ---     0.424   ULPPLL.CLKIN to    ULPPLL.MCLK v_io_ppl3_tx4_1_mtppll_mac_macpll_0_0
ROUTE       263     3.182    ULPPLL.MCLK to    R23C17.CLK0 CLKB
                  --------
                    7.889   (20.5% logic, 79.5% route), 2 logic levels.
```

In Figure 18-5, notice how the path is described in terms of “Logical Details.”

This section shows both the source and destination registers using their unmapped names from the EDIF (Electronic Data Interchange Format) file. This is a feature that allows the user to recognize the type of logic being analyzed.

Based on the declared frequencies for both clocks, we already know the following:

- CLKA period = 9.6 ns.
- CLKB period = 15.15 ns.

- No relative phase information exists between both clocks. As a result, Trace does not factor in the skews on either clock.

As a consequence, we know that, ignoring everything else (clock skews, registers library setups, etc.), a single cycle positive edge to positive edge setup available from CLKA to CLKB is: 15.15ns (refer to waveforms in Figure 18-4). Hence, with 2X multicycle, the resulting setup would be twice that number, or:

Ts = 30.3 ns
(shows up as delay constraint under Constraint Details section of Trace report)

Having computed this, the available setup margin is known to be as follows:

M = (Ts - Td) - Ds

Where:

- Td = path delay from clock pin of source register to D pin of destination=2.456 ns. Shown in the Physical Path Details section of Trace report.
- Ds = destination cell library setup requirement= -0.099 ns. This matches DIN_SET under Constraint Details section of the .twr Trace report.

There is no phase relationship between CLKA and CLKB as indicated by the warnings in Figure 18-5. Hence, the following skews were correctly ignored:

- TSB = delay or skew on destination clock CLKB = 7.889 ns. Shown in the Clock Skews detail section of Trace report.
- TSA = delay or skew on source clock CLKA = 7.699 ns. Shown in the Clock Skews detail section of Trace report.

Hence:

- M = (30.3 - 2.46) - (-0.099) = 27.9 ns. This matches the number in the “PASSED” section at the top of the Trace report.

Example 2. CLOCK_TO_OUT with PLL Feedback

In this example, ip_macclk_c is assigned to 66 MHZ and the clock to out propagation delays are constrained in the preference file:

```
FREQUENCY NET "ip_macclk_c" 66 MHZ;
CLOCK_TO_OUT ALLPORTS 7.000000 ns CLKPORT "ip_macclk" ;
```

See Figure 18-6 for the block diagram for this example. The resulting Trace report is shown in Figure 18-7.

Figure 18-6. CLOCK_TO_OUT with PLL

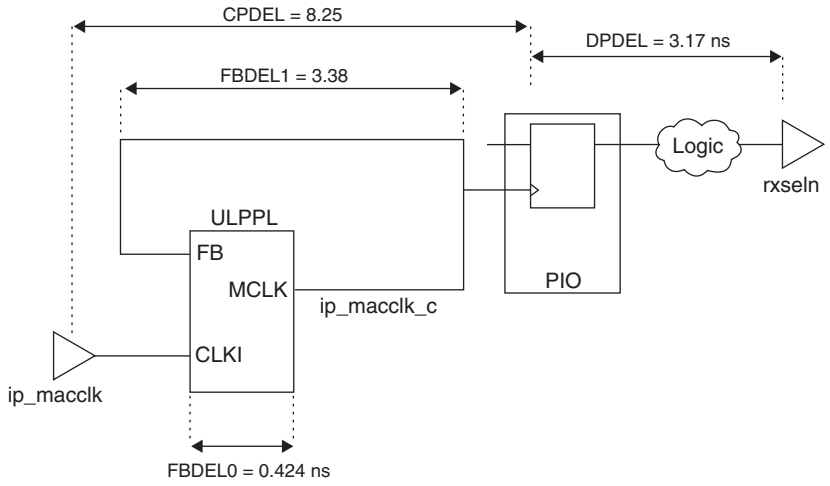


Figure 18-7. Trace Report for CLOCK_TO_OUT with PLL

```
================================================================================
Preference: CLOCK_TO_OUT ALLPORTS 7.000000 ns CLKPORT "ip_macclk" ;
            2 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------

Passed:  The following path meets requirements by 0.681ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)
   Source:         IO-FF Out  Q              ppl3_rx5_1_rxselnio  (from macclk +)
   Destination:    Port       Pad            rxseln
   Data Path Delay:     3.164ns  (100.0% logic, 0.0% route), 1 logic levels.
   Clock Path Delay:    8.249ns  (19.6% logic, 80.4% route), 2 logic levels.

 Constraint Details:
      8.249ns delay ip_macclk to rxseln less
      5.094ns feedback compensation
      3.164ns delay rxseln to rxseln (totaling 6.319ns) meets
      7.000ns offset ip_macclk to rxseln by 0.681ns

 Physical Path Details:

      Clock path ip_macclk to rxseln:
   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     1.192        C17.PAD to       C17.INDD ip_macclk
ROUTE         1     3.235       C17.INDD to   ULPPLL.CLKIN ip_macclk_c
MCLK_DEL    ---     0.424   ULPPLL.CLKIN to    ULPPLL.MCLK v_io_ppl3_tx4_1/mtppll_mac/macpll_0_0
ROUTE       141     3.398    ULPPLL.MCLK to         F32.SC macclk
                  --------
                    8.249   (19.6% logic, 80.4% route), 2 logic levels.

      Data path rxseln to rxseln:
   Name    Fanout   Delay (ns)          Site               Resource
OUTREGF_DE  ---     3.164         F32.SC to        F32.PAD rxseln (from macclk)
                  --------
(100.0% logic, 0.0% route), 1 logic levels.

      Feedback path:

   Name    Fanout   Delay (ns)          Site               Resource
MCLK_DEL    ---     0.424   ULPPLL.CLKIN to    ULPPLL.MCLK v_io_ppl3_tx4_1/mtppll_mac/macpll_0_0
ROUTE       141     3.380    ULPPLL.MCLK to      ULPPLL.FB macclk
                  --------
                    3.804   (11.1% logic, 88.9% route), 1 logic levels.

Report:    6.319ns is the minimum offset for this preference.
```

The different path measurements were obtained from the Trace report shown in Figure 18-7 as follows:

- DPDEL = Data Path Delay = 3.16 ns. Shown under Physical Path Details-> Data path in the timing report.
- FBDEL0 = Feedback cell delay across PLL = 0.42 ns, which is the first entry value under Feedback Path.
- FBDEL1 = Feedback routing delay from PLL output to PLL FB pin = 3.38 ns, which is the second entry value under Feedback Path.

The full feedback delay includes both FBDEL0 and FBDEL1 (0.42 + 3.38 = 3.80) under Feedback Path, in addition to any internal PLL delay added after the FB pin. Such a delay is a programmable attribute defined as FB_PDEL. This programmable value can be set to any of one of 4 values (DEL0, DEL1, DEL2 or DEL3; DEL0 being 0 delay) in either the HDL file input to synthesis, or in the graphical Editor for Programmable Integrated Circuits (EPIC) software tool included with the ispLEVER software.

Therefore, the total feedback delay would be:

$$FBDEL = FBDEL0 + FBDEL1 + FB\_PDEL = 3.80 + FB\_PDEL$$

Under “Constraint Details” of the report file, the feedback compensation (FBDEL) is shown to be 5.09 ns. Since this value is different from 3.804, we conclude that a non-zero value of FB_PDEL was applied (5.10 - 3.80 = 1.29 ns). This value corresponds to FB_PDEL = DEL2 in an OR4E4-2 device.

Now, let's verify the available margin on this CLOCK_TO_OUT preference:

$$\begin{aligned} M &= CKOUT - (CPDEL + DPDEL - FBDEL) \\ &= 7.000 - (8.249 + 3.164 - 5.094) = 0.681 \text{ ns} \end{aligned}$$

This value matches the one at the top of the report file (“Passed” section). It also matches the final value under “Constraints Details”.

ispLEVER Controlled Place and Route

Extensive benchmark experiments have been performed in order to determine the most optimum per device default settings for all PAR options. At times, improved timing results can be obtained on a design by design basis by trying different variations of the PAR options. This section describes the techniques that can be used within the ispLEVER graphical user interface (GUI) to improve timing results from Trace on placed and routed designs.

Running Multiple Routing Passes

Improved timing results can be obtained by increasing the number of routing passes during the Routing phase of PAR.

The PAR options window in Figure 18-8 can be launched by the following steps:

1. In the Project Navigator Source window, select the **target FPGA device**.
2. In the Processes window, right-click the **Place & Route Design** process and select **Properties** to open the dialog box.

In the example screen shot shown in Figure 18-8, the router will route the design for five routing iterations, or until all the timing preferences are met, whichever comes first. For example, PAR will stop after the second routing iteration if it hits a timing score of zero on the second routing iteration.

The highest selection in Placement Effort level will result in longer PAR run times but may give better design timing results. A lower Placement Effort will result in shorter PAR run times but will likely give less than optimal design timing results.

Figure 18-8. PAR Options Window

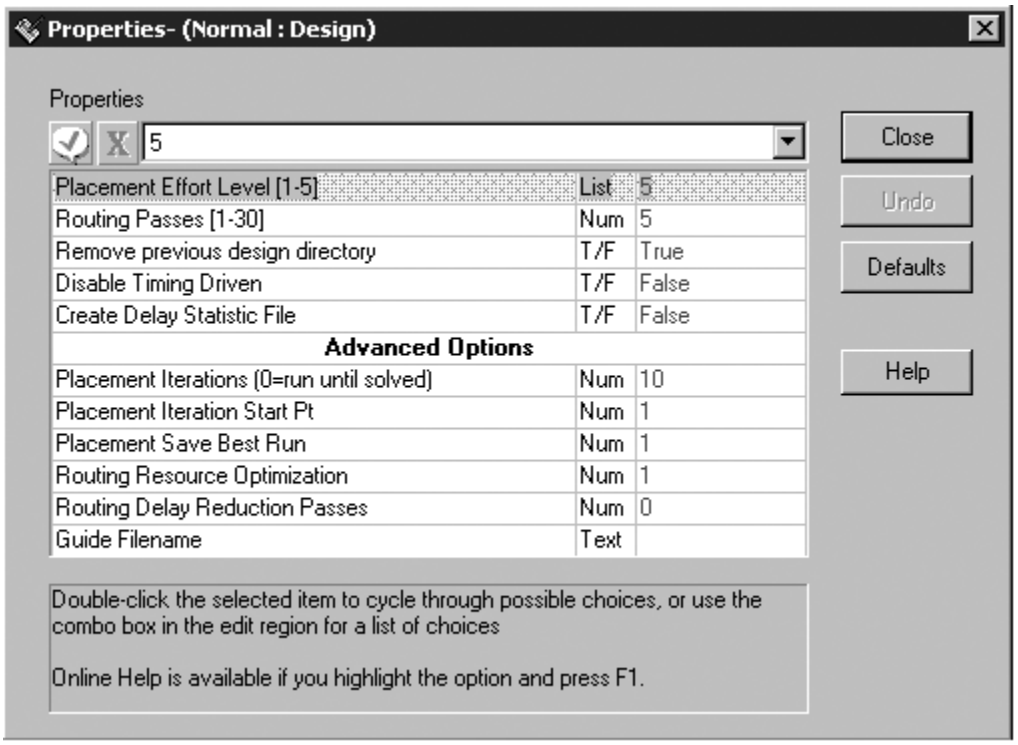


Figure 18-9. Example PAR report (.par) File, Routing Section

```
0 connections routed; 26590 unrouted.
Starting router resource preassignment
Completed router resource preassignment. Real time: 11 mins 31 secs
Starting iterative routing.
End of iteration 1
26590 successful; 0 unrouted; (151840) real time: 14 mins 29 secs
Dumping design to file
d:\ip\design.ncd.
End of iteration 2
26590 successful; 0 unrouted; (577) real time: 16 mins 23 secs
Dumping design to file
d:\ip\design.ncd.
End of iteration 3
26590 successful; 0 unrouted; (0) real time: 17 mins 39 secs
Dumping design to file
```

The place and route (.par) report file contains execution information about the PAR command run. The report also shows the steps taken as the program converges on a placement and routing solution. In the routing convergence example text in Figure 18-9, the number in parenthesis is the timing score after each iteration. In this example, timing was met after three routing iterations, as can be seen from the (0) timing score.

Using Multiple Placement Iterations (Cost Tables)

Using multiple placement iterations can be achieved by selecting the Advanced Options in Figure 18-8.

As shown in the Advanced Options of Figure 18-8, the number of iterations is set to 10 and the placement start point is set to iteration 1 (cost table 1). Only the best NCD file is to be saved as per the following line. Once PAR runs, the tool will loop back through the place and route flow until the number of iterations has reached 10. In this

example, the NCD file with the best timing score would be saved. The tool keeps track of the timing and routing performance for every iteration in a file called the multiple par report (.par). Such a file is shown in Figure 18-10.

Figure 18-10. Multiple PAR Report (.par)

```
Level/          Number      Timing      Run         NCD
Cost [ncd]      Unrouted    Score       Time        Status
----------      --------    -------     -----       --------
5_4   *         0           0           01:58       Complete
5_6             0           25          02:01       Complete
5_2             0           102         01:45       Complete
5_7             0           158         02:15       Complete
5_3             0           186         01:54       Complete
5_10            0           318         02:39       Complete
5_1             0           470         01:51       Complete
5_8             0           562         02:25       Complete
5_5             0           732         02:00       Complete
5_9             0           844         02:27       Complete

* : Design saved.
```

Figure 18-10 indicates that:

- The “5_” under the Level/Cost column means that the Placement Effort level was set to 5. The Placement Effort level can range from 1 (lowest) to 5 (highest).
- 10 different iterations ran (10 cost tables).
- Timing scores are expressed in total picoseconds (ps) by which the design is missing constraints on all preferences.
- Iteration number 4 (cost table 4) achieved a 0 timing score and hence was the design saved. More than one .ncd file can be saved. This is user-controlled via the “Placement Save Best Runs” value box shown in Figure 18-8.
- Each iteration routed completely.

Note that, in Figure 18-8, if “Placement Iterations (0=run until solved)” is set to 0, the tool will run indefinitely through multiple iterations until a 0 timing score is reached. In a design that is known to have large timing violations, a 0 timing score will never be reached. As a consequence, the user must intervene and stop the flow at a given point in time.

In general, multiple placement iterations can help placement but can also use many CPU cycles. Multiple placement iterations should be used carefully due to system limitations and the uncertainty of results. It is better to fix the root cause of timing problems in the design stage.

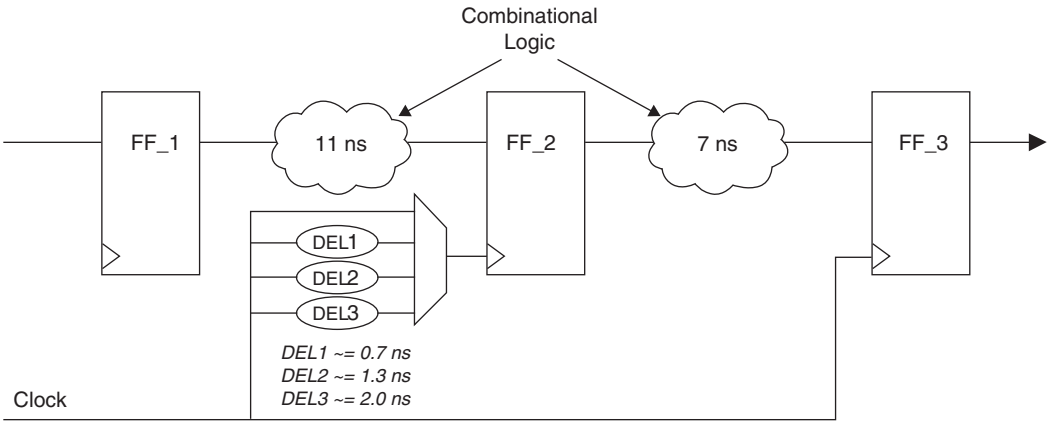
Clock Boosting

Clock boosting, supported in Lattice Semiconductor’s ORCA Series device family, is the deliberate introduction of clock skew on a target flop to increase the setup margin. Every programmable flip-flop in the device has programmable delay elements before clock inputs for this purpose. The automated clock boosting tool will attempt to meet setup constraints by introducing delays to as many target registers as needed to meet timing, in effect, borrow register hold margins to meet register set-up timing. The following bullets summarize how clock boosting is accomplished in Lattice Semiconductor ORCA Series device family.

- A 4-tap delay cell structure in front of the clock port of every flip-flop in the device (includes I/O flip-flops)
- Ability to borrow clock cycle time from one easily-met path and give this time to a difficult-to-meet path

Clock boosting is typically most useful in designs that are only missing timing on a few paths for one or two preferences. If the design is missing timing by over a few nanoseconds on any given path, clock boosting will not be able to schedule skew in a way that will eliminate enough timing to make the critical preference. Clock boosting run times can be shortened by using a preference file with only the failing preferences in it.

Figure 18-11. Clock Boosting Example



Target Performance: 10 ns period (100 MHz)

The example illustrated in Figure 18-11 shows two register-to-register transfers that both need to meet the 10 ns period constraint. By using delay cell DEL2 to delay the clock input on flip-flop FF_2, the first register transfer will make its period constraint with a new minimum period of ~9.7 ns and the second register transfer will make its period constraint by ~8.3 ns.

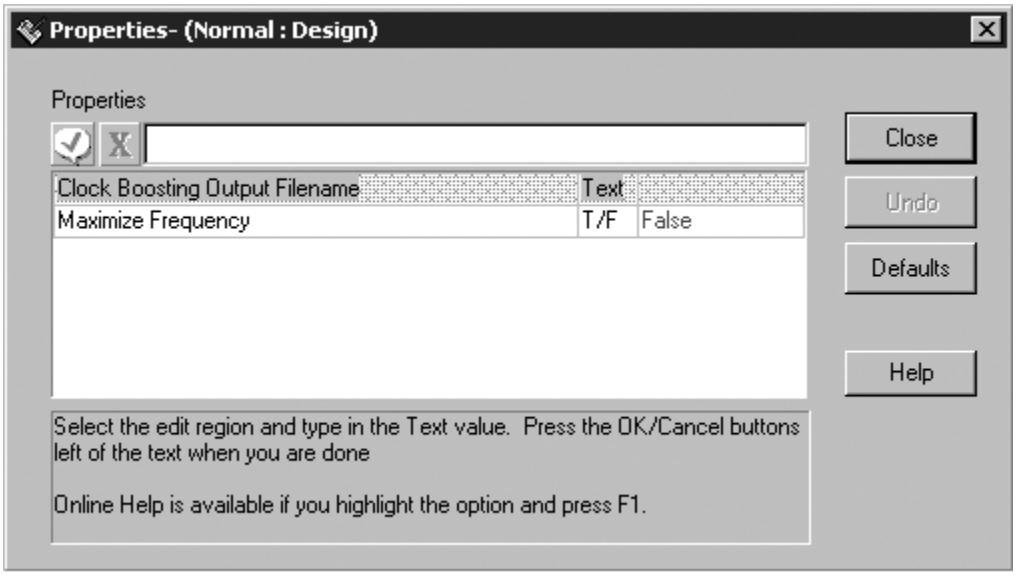
The D1, D2, and D3 delays shown in Figure 18-11 are variable depending on the speed grade and Lattice Semiconductor FPGA device family. For complete timing information, reference the software generated timing data sheet, included with ispLEVER, for the desired Lattice Semiconductor FPGA device family.

To Perform Clock Boosting in the Project Navigator

1. In the Project Navigator Sources window, select the **target device**.
2. In the Processes window, right-click the **Clock Boosting** under **Place & Route Design** process, and then select **Properties** to open the Properties dialog box.
3. Select the **Clock Boosting Output Filename** property from the property list and type the name of the output file name in the edit region (<file_name>.ncd).
4. Click **Close** to close the dialog box.

As shown in Figure 18-12, the original .ncd and .prf files as well as the output .ncd file are typed into the corresponding entries. Checking “Maximize Frequency” will push the tool to improve the frequency beyond the input preference requirement. This is generally only useful for bench marking.

Figure 18-12. Clock Boosting Window



Other important considerations on the practicality of using clock boosting:

- Some circuits show big improvement, others have no gain. Clock boosting results are very design-dependent.
- Clock boosting uses minimum delay values which have not yet been validated at the system level.
- Automatic clock boosting identifies skew and hold time issues. However, after clock boosting is performed, designers are strongly recommended to run Trace twice, once with regular, maximum delay analysis, and again with minimum delays. The designer should then read over both resultant .twr timing reports to make sure there are no timing errors. The minimum delay analysis is done by checking the “Check Hold Times” checkbox in the Trace Options GUI window.

Guided Map and PAR

To decrease PAR runtimes after minor changes to a logical design, guided mapping uses a previously generated .ncd file to “guide” the mapping of the new logical design. Guided mapping can be performed from the Guide Filename property in the Project Navigator Map Design process, or specified using the command line **-g** option with the file name of the guide file. In general, guided MAP should only be used in conjunction with guided PAR.

To Perform Guided Mapping in the Project Navigator

1. In the Project Navigator Sources window, select the **target device**.
2. In the Processes window, right-click the **Map Design** process, and then select **Properties** to open the Properties dialog box.
3. Select the **Guide Filename** property from the property list and type the name of the guide file name in the edit region (<file_name>.ncd).
4. Click **Close** to close the dialog box.

The Map operation will use the guide file to generate the new design file.

To Perform Guided PAR in the Project Navigator

1. In the Project Navigator Sources window, select the **target device**.
2. In the Processes window, right-click the **Place & Route Design** process and select **Properties** to open the dialog box.
3. Under Advanced Options, select the **Guide Filename** property and type the name of the file in the text field.

4. Click **Close** to close the dialog box.
5. Double-click the **Place & Route Design** process. The ispLEVER software runs the process using the specified guide file.

Notes on Guided Mapping

All guidance criteria is based on signal name matching. Topology of combinatorial logic is considered when Softwire LUTs (SWLs) exist in the guided file.

Register elements are mapped in two passes. In the first pass, register control signals are matched by name exactly. In the second pass, the control signals names are not matched. This methodology provides a greater chance of matching for registers since control signal names have a tendency to change from successive synthesis runs. Other matching considerations are as follows:

- For combinatorial logic, new SWLs are matched from SWLs extracted from the guide design.
- All unmatched logic are mapped through the regular mapping process.
- The performance of the guided mapped design can be no better than the original.
- A guide report, <design_name>.gpr, gives details of the success guided map had in matching with the guide file.

Notes on Guided PAR

To decrease PAR runtimes after minor changes to the physical design file (.ncd), guided PAR uses a previously placed and/or routed .ncd file to “guide” the placement and routing of the new .ncd file. Guided PAR can be performed from the Project Navigator or specified using the command line **-g** option with the file name of the guide file.

For PAR to use a guide file for design, PAR first tries to find a guiding object (i.e., nets, components, and/or macros) in the guide file that corresponds to an object in the new .ncd file. A guiding object is an object in the guide file of the same name, type, and connectivity as an object in the new .ncd file. A guided object is an object in the new .ncd file that has a corresponding guiding object in the guide file.

After PAR compares the objects in each file, it places and routes each object of the new .ncd file based on the placement/routing of its guiding object. If PAR fails to find a guiding object for a component, for example, PAR will try to find one based on the connectivity. PAR appends the names of all objects which do not have a guiding object in the guide file to .gpr (Guided PAR Report) file. The matching factor specifies the percentage of the same connectivity that guiding and guided objects must have. It can only be specified using the **-mf** option in the command line. The matching factor option applies to nets and components only. When matching factor is 100 (the default), a guiding object must have exactly the same connectivity as the object it is guiding. When a matching factor is specified, the value specified is taken as the minimum percentage of the same connectivity that a guided object and its guiding object have. Note that the matching factor is always 100 when the guided PAR is performed from the Project Navigator.

After all guided objects are placed and routed, PAR locks down the locations of all guided components and macros and then proceeds with its normal operation. Guided PAR supports the following preferences: USE SPINE, USE PRIMARY, USE SECONDARY, USE LONGLINE, USE HALFLINE, LOCATE COMP, LOCATE MACRO, and hard-placed PGROUPs.

Conclusion

In general, different designs respond better to different strategies. The processes outlined in this application note may not be optimal for all cases. For a design's first place and route, run PAR at the low placer effort level and with a low number of routing iterations. There is no point in running 100 cost tables if the design's logic depth is too high. The techniques discussed within this document, like interpreting static timing reports and using proper preferences, will guide the user to better PAR results.

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Board Timing Guidelines for the DDR SDRAM Controller IP Core

June 2004 Technical Note TN1071

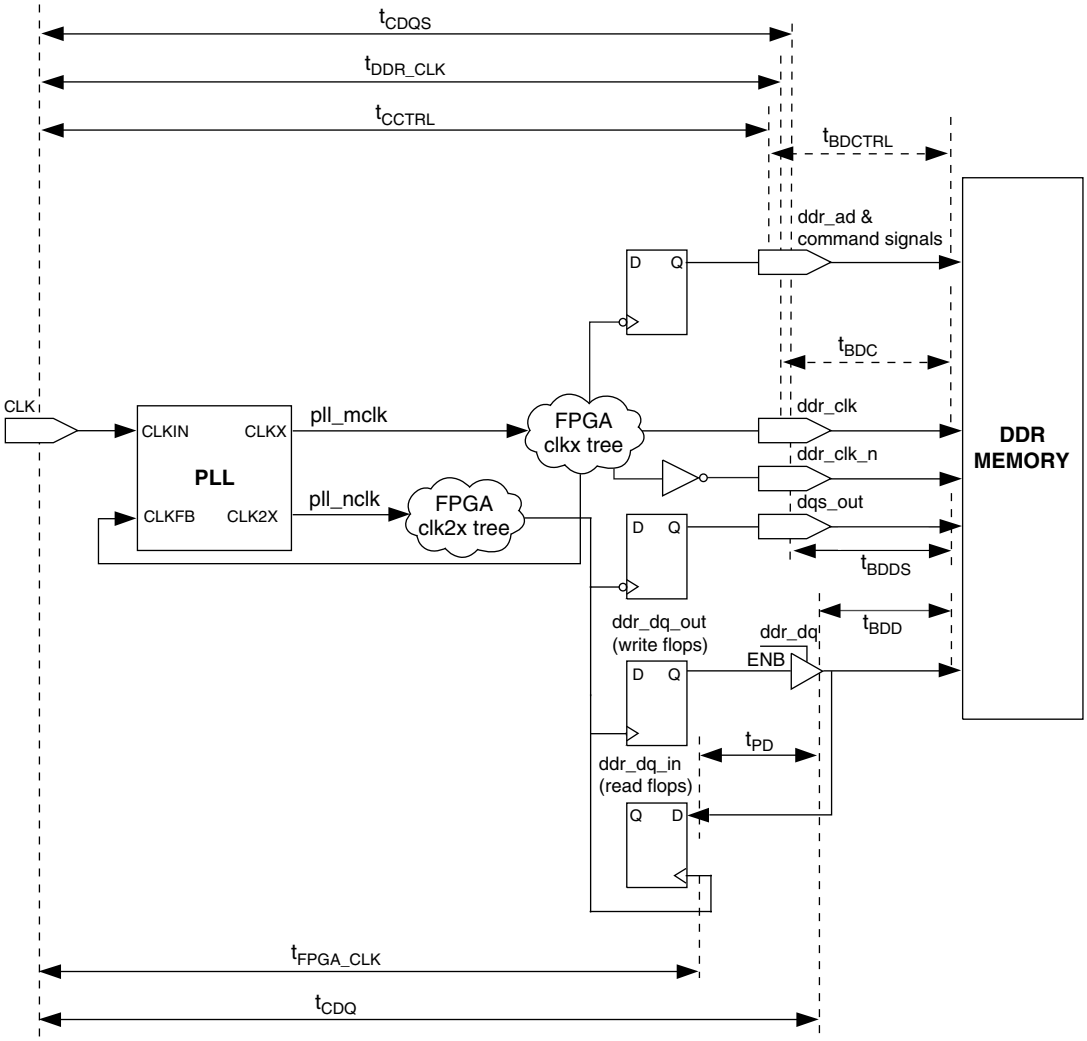
Introduction

This document describes how to meet board timing requirements for DDR signals. The Lattice DDR SDRAM Controller IP core, non-pipelined version (DDR-NP) is used as an example.

Figure 19-1 describes the timing diagram for the DDR signals. A total of five clocks are used in the DDR board design using the Lattice DDR IP core. The following is the clock description:

`clk`:	Input clock for PLL (max. frequency of 133MHz for DDR NP)
`ddr_clk`:	Output clock going to DDR (max. frequency of 133MHz for DDR NP)
`ddr_clk_n`:	Negated version of `ddr_clk`
`pll_mclk (clkx)`:	Same as `ddr_clk`, used inside the FPGA only.
`pll_nclk (clk2x)`:	A 266MHz clock for DDR NP, used inside the FPGA only.

Figure 19-1. DDR Signal Timing Diagram



As shown in Figure 19-1, input to PLL is CLK (133MHz for DDR NP). The PLL generates `pll_mclk` (133MHz) and `pll_nclk` (266MHz). The clocks `ddr_clk` and `ddr_clk_n` go to DDR memory and are delayed by I/O pad delay with respect to `pll_mclk`. The clocks `pll_mclk` and `pll_nclk` are internal to the FPGA. Command and address signals are clocked by a negative edge of `pll_mclk`. The signal `dqs_out` acts as a clock for DDR write and is generated by negative edge of `pll_nclk`. The signal `ddr_dq_out` is the DDR write data bus and generated by positive edge of `pll_nclk`. The flops `ddr_dq_*` latch the read data and are clocked by positive edge of `pll_nclk`.

Read Operation

Figure 19-2 shows the timing of the DDR read operation. Table 19-1 describes the timing arcs of the read operation.

Figure 19-2. Read Timing Diagram

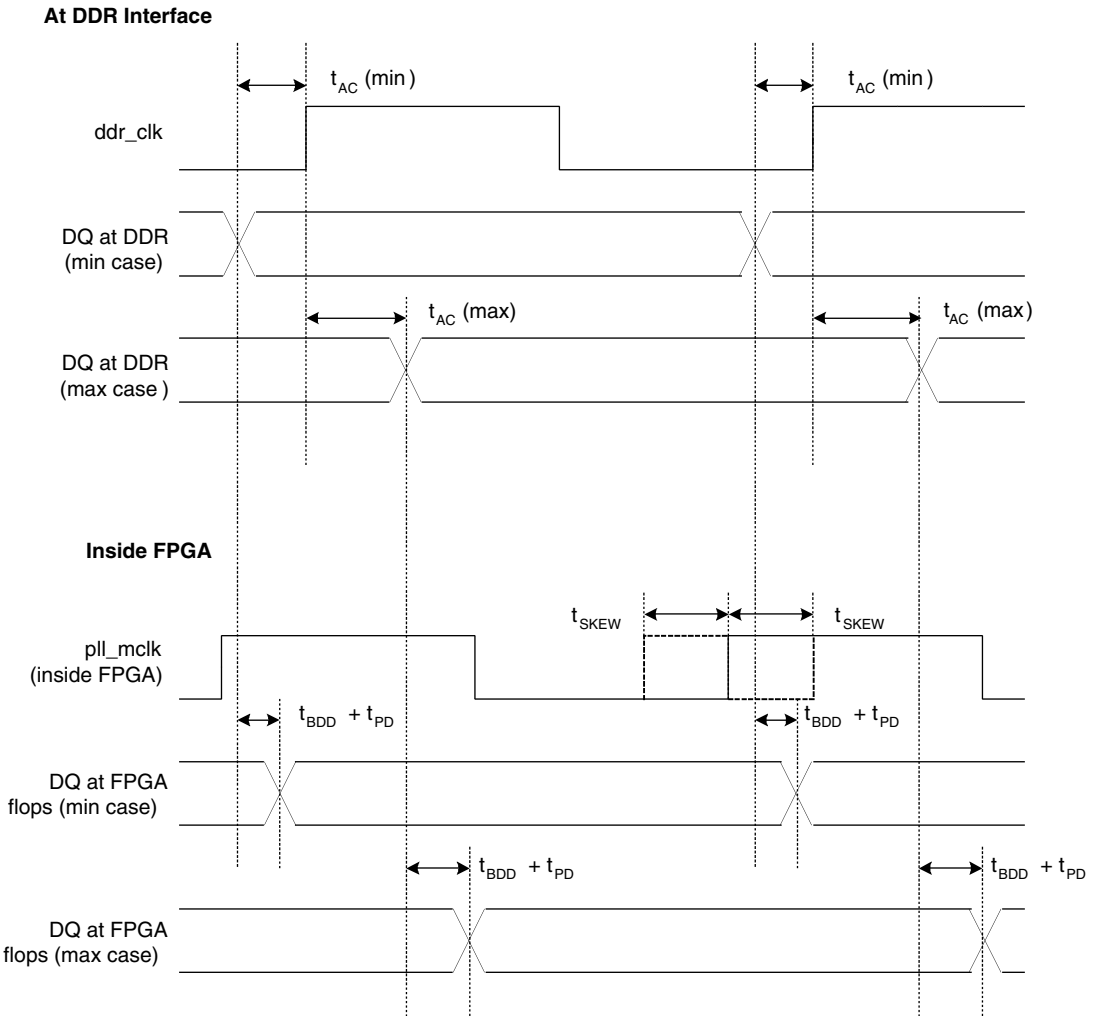


Table 19-1. Read Operation Timing Arcs

Symbol	Description	Example: DDR-NP on ORCA 4
t_{CK}	Clock period of `ddr_clk`	7.5ns
t_{DDR_CLK} (max)	Delay from the CLK input of the FPGA to the `ddr_clk` pad including Feedback compensation (Clock Path Delay - Feedback Path).	2.47[1]
t_{DDR_CLK} (min)	Delay from the CLK input of the FPGA to the `ddr_clk` pad including Feedback compensation (Clock Path Delay - Feedback Path).	1.138[1]
t_{BDC}	Board delay of `ddr_clk` from FPGA to DDR SDRAM.	—
$t_{AC(MAX)}$	Time from the rising edge of `ddr_clk` after which the data is available at DDR output pins (max.).	0.75ns
$t_{AC(MIN)}$	Time from the rising edge of `ddr_clk` after which the data is available at DDR output pins (min.).	-0.75ns
t_{BDD}	Board delay from DDR SDRAM data pad to the FPGA `ddr_dq` pad.	—
t_{PD}	Propagation delay from FPGA input pad to the `ddr_dq_in` flip-flop input pin (Data Path Delay).	0.0ns[1]
t_{FDS}	Set-up time required by the `ddr_dq_in` flip-flop (`INREG_SET`).	3.195ns[1]
t_{FDH}	Hold time required by the `ddr_dq_in` flip-flop (`INREG_HLD`).	-1.609ns[1]
t_{SKEW}	Skew of the PLL.	0.3ns
t_{FPGA_CLK} (max)	Delay from the CLK input of the FPGA to the `ddr_dq_in` flip-flop clock input including feedback compensation (Clock Out Path Delay - Feedback Path).	2.935ns[1]
t_{FPGA_CLK} (min)	Delay from the CLK input of the FPGA to the `ddr_dq_in` flip-flop clock input including feedback compensation (Clock Out Path Delay - Feedback Path).	1.239ns[1]

1. t_{FPGA_CLK}, t_{DDR_CLK}, t_{PD} and t_{FDS} can be easily obtained from the PNR time reports.

Set-up Time Calculation for the Data Input (Max. Case)

The DDR Controller IP core uses the positive edge of `pll_nclk` to latch in the data.

Table 19-1 timing arcs are used to calculate the following:

Max. delay of clock to `ddr_dq_in` flops = t_{FPGA_CLK} (max) + (t_{CK} * 1/2) - t_{SKEW} - t_{FDS}

Max. delay of DDR read data to `ddr_dq_in` flops = t_{DDR_CLK} (max) + t_{BDC} + t_{AC} (max) + t_{BDD} + t_{PD}

To meet set-up time at `ddr_dq_in` flops, Clock Delay - Data Delay > 0

Therefore:

t_{FPGA_CLK} (max) + (t_{CK} * 1/2) - t_{SKEW} - t_{FDS} - t_{DDR_CLK} (max) - t_{BDC} - t_{AC} (max) - t_{BDD} - t_{PD} > 0

Isolating the board delays, we get:

(t_{BDD} + t_{BDC}) < t_{FPGA_CLK} (max) + (t_{CK} * 1/2) - t_{SKEW} - t_{FDS} - t_{DDR_CLK} (max) - t_{AC} (max) - t_{PD}

(t_{BDD} + t_{BDC}) < 3.75 - 0.3 - 3.195 - 2.47 + 2.935 - 0.75 - 0.0

(t_{BDD} + t_{BDC}) < -0.03 ns

Hold Time Calculation for the Data Input (Min. Case)

As shown in Figure 19-2, the min data is available at DDR output pins after t_{AC} (min) time from the rising edge of `ddr_clk`. Since t_{AC} (min) is generally a negative number, data appears before the rising edge. This data will incur board delay (t_{BDD}) and propagation delay from FPGA input pad to the flip-flop input pin (t_{PD}).

Min. Delay of DDR read Data = t_{DDR_CLK} (min) + t_{BDC} + t_{AC} (min) + t_{BDD} + t_{PD}

Min. Delay of Clock to ddr_dq_in flops = t_{FPGA_CLK} (min) + t_{SKEW} + t_{FDH}

To meet hold time at ddr_dq_in flops, Data Delay - Clock Delay > 0

Therefore:

t_{DDR_CLK} (min) + t_{BDC} + t_{AC} (min) + t_{BDD} + t_{PD} - t_{FPGA_CLK} (min) - t_{SKEW} - t_{FDH} > 0

Isolating the board delays, we get:

(t_{BDD} + t_{BDC}) > t_{FPGA_CLK} (min) + t_{SKEW} + t_{FDH} - t_{DDR_CLK} (min) - t_{AC} (min) - t_{PD}

(t_{BDD} + t_{BDC}) > (1.239) ns + 0.3 + (-1.609ns) - (1.138) - (-0.75) - 0

(t_{BDD} + t_{BDC}) > -0.458 ns

Conclusion: To meet read set-up and hold timing, board delay for ddr_dq, ddr_clk and ddr_clk_n should be:

-0.458ns < (t_{BDD} + t_{BDC}) < -0.03ns

Write Operation

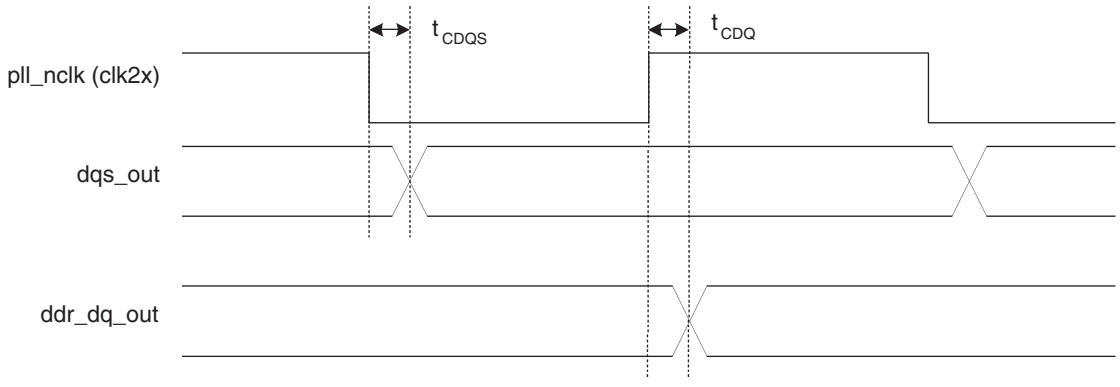
For a proper write operation, data (ddr_dq) should meet set-up (t_{DS}) and hold (t_{DH}) time requirements of DDR SDRAM with respect to ddr_dqs signal. The ddr_dqs signal is generated with respect to negative edge of pll_nclk and data ddr_dq out is generated with respect to positive edge of pll_nclk as shown in Figure 19-3. As a result, 1/2 clk2x (3.75ns/2) is provided as set-up and hold for ddr_dq_out with respect to dqs_out.

For maximum set-up and hold margin, the ddr_dqs and ddr_dq traces on the board should be matched.

Table 19-2. Write Operation Timing Arcs

Symbol	Description	ORCA 4
t_{DS}	Set-up time required by the DQ with respect to DQS for DDR SDRAM.	0.75ns
t_{DH}	Hold time required by the DQ with respect to DQS for DDR SDRAM.	0.75 ns
t_{CDQ}	Clock-to-out timing for ddr_dq with respect to pll_nclk.	—
t_{CDQS}	Clock-to-out timing for ddr_dqs with respect to pll_nclk.	—
t_{BDDS}	Board delay of ddr_dqs from FPGA to DDR SDRAM pins.	—

Figure 19-3. Write Timing Diagram



Write Set-up

Clock Delay = t_{CDQS} + 1/2 clk2x - t_{DS} + t_{BDDS}

Data Delay = t_{CDQ} + t_{BDD}

Clock Delay - Data Delay > 0

Therefore:

t_{CDQS} + 1/2 clk2x - t_{DS} + t_{BDDS} - t_{CDQ} - t_{BDD} > 0

Assumptions for write set-up and hold equations:

1. t_{BDDS} and t_{BDD} are equal (board delays are same both for dqs_out and ddr_dq_out).

2. t_{CDQ} and t_{CDQS} are equal (both are output delays from I/O flop).

Therefore:

1/2 clk2x - t_{DS} > 0

3.75/2 - 0.75 > 0

1.125 > 0

Write Hold

Data Delay = t_{CDQ} + t_{BDD}

Clock Delay = t_{CDQS} + 1/2 clk2x + t_{DH} + t_{BDDS}

Data Delay - Clock Delay > 0

Therefore:

t_{CDQS} + 1/2 clk2x - t_{DH} + t_{BDDS} - t_{CDQ} - t_{BDD} > 0

Assumptions for write set-up and hold equations:

1. t_{BDDS} and t_{BDD} are equal (board delays are same both for dqs_out and ddr_dq_out).

2. t_{CDQ} and t_{CDQS} are equal (both are output delays from I/O flop).

Therefore:

1/2 clk2x - t_{DH} > 0

3.75/2 - 0.75 > 0

1.125 > 0

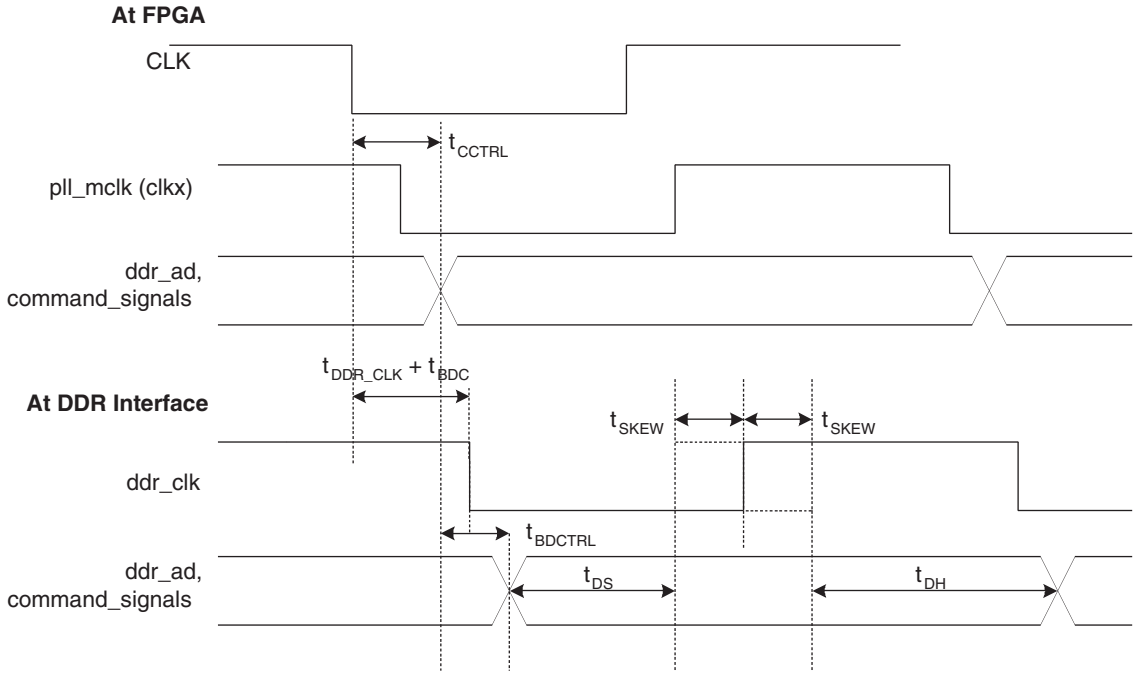
Address and Command Signals

Address (ddr_ad) and command signals (ddr_cas, ddr_ras, ddr_we) should meet set-up (t_{DS}) and hold (t_{DH}) timings at DDR interface with respect to positive edge of ddr_clk. Address and command signals are clocked using negative edge of pll_mclk inside the FPGA as shown below. The ddr_clk signal is a delayed by pad delay and board delay at DDR interface compared to pll_mclk inside the FPGA. As a result, 1/2clkx of set-up and hold is provided by design.

Table 19-3. Timing Arcs for Address and Command Signals

Symbol	Description	ORCA4
t_{CCTRL} (max)	Is the clock-to-out time for `ddr_ad` and command signals. (Clock Path Delay - Feedback Path) + Data Path Delay	4.834 ns
t_{CCTRL} (min)	Is the clock-to-out time for `ddr_ad` and command signals. (Clock Path Delay - Feedback Path) + Data Path Delay	2.147 ns
t_{BDCTRL}	Is the board delay of `ddr_ad` and command signals from FPGA pins to DDR SDRAM pins.	—

Figure 19-4. Timing Diagram for Address and Command Signals



Set-up Calculation

Max Delay of Clock to DDR = t_{DDR_CLK} (max) + t_{BDC} + t_{CK} * 1/2 - t_{SKEW} - t_{DS}

Max Delays of command signals Data to DDR = t_{CCTRL} (max) + t_{BDCTRL}

To meet set up time at DDR memory, Clock Delay - Data Delay > 0

Therefore:

t_{DDR_CLK} (max) + t_{BDC} + t_{CK} * 1/2 - t_{SKEW} - t_{DS} - t_{CCTRL} (max) - t_{BDCTRL} > 0

Isolating the board delays, we get:

t_{BDCTRL} - t_{BDC} < t_{DDR_CLK} (max) + t_{CK} * 1/2 - t_{SKEW} - t_{DS} - t_{CCTRL} (max)

t_{BDCTRL} - t_{BDC} < 2.47 + 3.75 - 0.3 - 0.75 - 4.834

t_{BDCTRL} - t_{BDC} < 0.336 ns

Hold Calculation

Min Delay of command signals Data to DDR = t_{CCTRL} (min) + t_{BDCTRL} + t_{CK} * 1/2

Min Delay of Clock to DDR = t_{DDR_CLK} (min) + t_{BDC} + t_{SKEW} + t_{DH}

To meet hold time at DDR memory, Data Delay - Clock Delay > 0

Therefore:

t_{CCTRL} (min) + t_{BDCTRL} + t_{CK} * 1/2 - t_{DDR_CLK} (min) - t_{BDC} - t_{SKEW} - t_{DH} > 0

Isolating the board delays, we get:

t_{BDCTRL} - t_{BDC} > - t_{CCTRL} (min) - t_{CK} * 1/2 + t_{DDR_CLK} (min) + t_{SKEW} + t_{DH}

t_{BDCTRL} - t_{BDC} > -2.147 - 3.75 + (1.138) + 0.3 + 0.75

t_{BDCTRL} - t_{BDC} > -3.709

t_{BDCTRL} - t_{BDC} > -3.709 ns

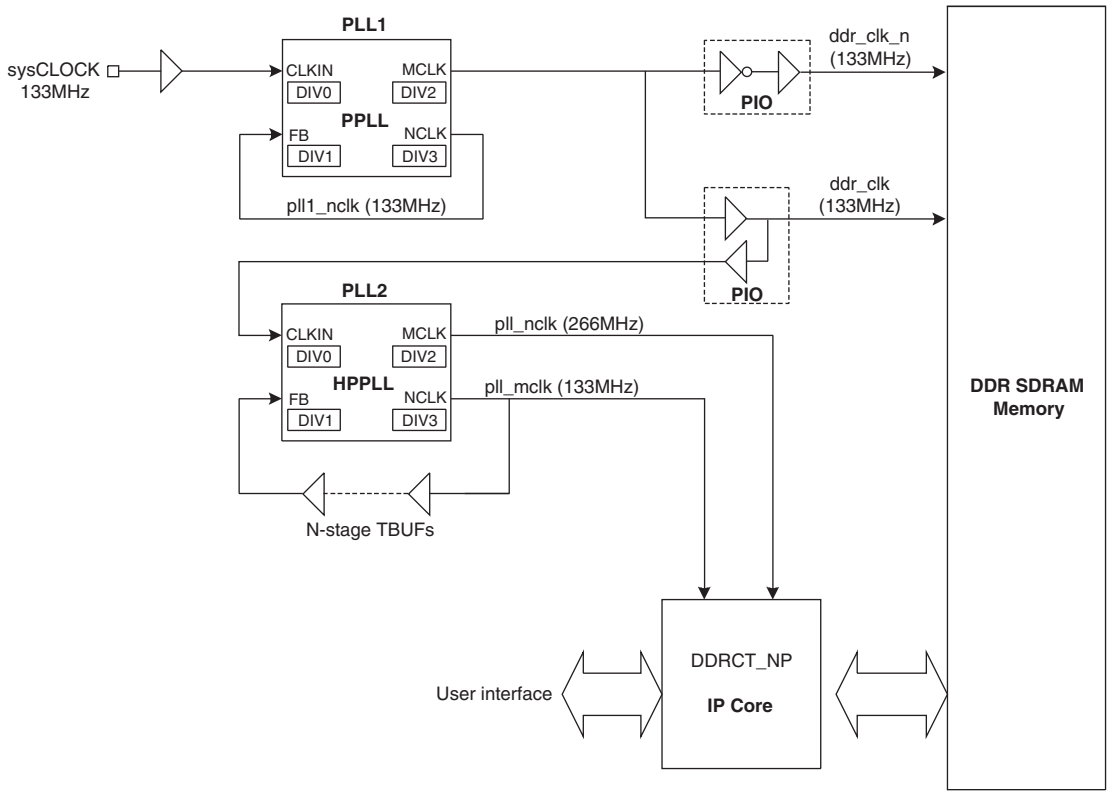
Conclusion: To meet set-up and hold timings of command signals, board delay of command signals `ddr_clk` and `ddr_clk_n` should be:

-3.709 ns < (t_{BDCTRL} - t_{BDC}) < 0.336 ns

Board Design Guidelines

- The `ddr_clk` and `ddr_clk_n` pads should be placed adjacent to each other in the FPGA to get similar internal FPGA delays.
- The `ddr_clk` and `ddr_clk_n` trace delays on the board should be matched.
- The DQ trace delays can be calculated using the following formula, for memory reads:
  t_{SKEW} + t_{FDH} - t_{AC} (min) - t_{PD} - t_{DDR_CLK} + t_{FPGA_CLK} < (t_{BDD} + t_{BDC}) < (t_{CK} * 1/2) - t_{SKEW} - t_{FDS} - t_{AC} (max) - t_{PD} - t_{DDR_CLK} + t_{FPGA_CLK}
- The DQ and DQS trace lengths should be balanced and matching to get maximum set-up/hold time during memory writes.
- The address and control signals for the DDR SDRAM are generated on the negative edge of the FPGA clock. The trace lengths for address and control lines are calculated using following equation:
  -t_{CCTRL} - t_{CK} * 1/2 + t_{DDR_CLK} + t_{SKEW} + t_{DH} < (t_{BDCTRL} - t_{BDC}) < t_{DDR_CLK} + t_{CK} * 1/2 - t_{SKEW} - t_{DS} - t_{CCTRL} + t_{BDC}
- As shown in Figure 19-1, both FPGA internal clock and `ddr_clk` are generated by a single PLL. It may be difficult to meet read data Set-up and hold timing with a single PLL. As shown in Figure 19-5, a two-PLL clocking scheme is proposed to meet read data set-up and hold timing. Adjusting feedback delay of PLL2 can control delay of `pll_mclk`. Increasing delay on `pll_mclk` can increase the read set-up margin but it also decreases the hold margin. To get better timing, skew between `ddr_clk` and `pll_mclk` has to be minimized.

Figure 19-5. Two PLL Clocking Scheme



Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Appendix A. Example Extractions of Delays from Timing Reports

From the Set-up Report below, which was run for MAX conditions:

- t_{PD} = 0.0 ns
- t_{FDS} = 3.195 ns
- t_{FPGA_CLK} (max) = 6.206 - 3.271 = 2.935 ns

```
================================================================
Preference: INPUT_SETUP PORT "ddr_dq_*" 2.000000 ns CLKNET "pll_nclk" ;
            32 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------------------------------
-----------------------------
Passed:  The following path meets requirements by 1.740ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)

   Source:            Port       Pad            ddr_dq_23
   Destination: O-FF In   Data in        U1_ddrct_np_o4_1_008/U3_databusif/ddr_dqoeZ0Z_23  (to
pll_nclk +)

   Data Path Delay:     0.000ns  (0.0% logic, 0.0% route), 0 logic levels.

   Clock Path Delay:    6.206ns  (29.3% logic, 70.7% route), 2 logic levels.

 Constraint Details:

      0.000ns delay ddr_dq_23 to ddr_dq_23 less
      2.000ns offset ddr_dq_23 to clk (totaling -2.000ns) meets
      6.206ns delay clk to ddr_dq_23 less
      3.271ns feedback compensation less
      3.195ns INREG_SET requirement (totaling -0.260ns) by 1.740ns

 Physical Path Details:

      Data path ddr_dq_23 to ddr_dq_23:

   Name    Fanout   Delay (ns)          Site               Resource
                  --------
                    0.000   (0.0% logic, 0.0% route), 0 logic levels.

      Clock path clk to ddr_dq_23:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     1.431        AB4.PAD to       AB4.INCK clk
ROUTE         1     0.816       AB4.INCK to  LLHPPLL.CLKIN clk_c
NCLK_DEL    ---     0.385  LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       136     3.574   LLHPPLL.NCLK to         N24.SC pll_nclk
                  --------
                    6.206   (29.3% logic, 70.7% route), 2 logic levels.
```

```
      Feedback path:

   Name    Fanout   Delay (ns)          Site               Resource
NCLK_DEL     ---     0.385  LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE        136     2.886   LLHPPLL.NCLK to     LLHPPLL.FB pll_nclk
                  --------
                    3.271   (11.8% logic, 88.2% route), 1 logic levels.

Report:    0.260ns is the minimum offset for this preference.
```

From the Hold Report below, which was run for MIN conditions:

- t_{PD} = 0.0 ns
- t_{FDH} = -1.609 ns
- t_{FPGA_CLK} (min) = 3.144 - 1.905 = 1.239 ns

```
================================================================
Preference: INPUT_SETUP PORT "ddr_dq_*" 2.000000 ns CLKNET "pll_nclk" ;
            32 items scored, 0 timing errors detected.
--------------------------------------------------------------------------------------------------------
--------


Passed:  The following path meets requirements by 0.370ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)

   Source:         Port       Pad            ddr_dq_31
    Destination:     IO-FF In   Data in         U1_ddrct_np_o4_1_008/U3_databusif/ddr_dqoeZ0Z_31   (to
pll_nclk +)

   Data Path Delay:     0.000ns  (0.0% logic, 0.0% route), 0 logic levels.

   Clock Path Delay:    3.144ns  (25.7% logic, 74.3% route), 2 logic levels.

 Constraint Details:

      0.000ns delay ddr_dq_31 to ddr_dq_31 plus
      0.000ns hold offset ddr_dq_31 to clk (totaling 0.000ns) meets
      3.144ns delay clk to ddr_dq_31 plus
      1.905ns feedback compensation less
     -1.609ns INREG_HLD requirement (totaling -0.370ns) by 0.370ns

 Physical Path Details:

      Data path ddr_dq_31 to ddr_dq_31:

   Name    Fanout   Delay (ns)          Site               Resource
                  --------
                    0.000   (0.0% logic, 0.0% route), 0 logic levels.
```

```
      Clock path clk to ddr_dq_31:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     0.576        AB4.PAD to       AB4.INCK clk
ROUTE         1     0.507       AB4.INCK to  LLHPPLL.CLKIN clk_c
NCLK_DEL    ---     0.231  LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       136     1.830   LLHPPLL.NCLK to         C25.SC pll_nclk
                  --------
                    3.144   (25.7% logic, 74.3% route), 2 logic levels.

      Feedback path:

   Name    Fanout   Delay (ns)          Site               Resource
NCLK_DEL    ---     0.231  LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       136     1.674   LLHPPLL.NCLK to     LLHPPLL.FB pll_nclk
                  --------
                    1.905   (12.1% logic, 87.9% route), 1 logic levels.

Report: There is no minimum offset greater than zero for this preference.
```

From the Set-up Report below, which was run for MAX conditions:

- t_{DDR_CLK} (max) = 5.741 - 3.271 = 2.47 ns

```
================================================================================
Preference: CLOCK_TO_OUT PORT "ddr_cas_n" MAX 5.500000 ns CLKPORT "clk" CLKOUT PORT "ddr_clk" ;
            1 item scored, 0 timing errors detected.
--------------------------------------------------------------------------------------------------------
----------------------------


Passed:  The following path meets requirements by 3.182ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)

   Source:         Unknown    Q             U1_ddrct_np_o4_1_008/U1_cmdexe/ddr_cas_nZ0  (from ddr_clk_c
-)
   Destination:    Port       Pad           ddr_cas_n

   Data Path Delay:     1.713ns  (100.0% logic, 0.0% route), 1 logic levels.

   Clock Path Delay:    6.346ns  (28.6% logic, 71.4% route), 2 logic levels.

 Constraint Details:

      6.346ns delay clk to ddr_cas_n less
      3.271ns feedback compensation
      1.713ns delay ddr_cas_n to ddr_cas_n less
      2.470ns delay clk to ddr_clk (totaling 2.318ns) meets
      5.500ns offset clk to ddr_cas_n by 3.182ns

 Physical Path Details:

      Clock path clk to ddr_cas_n:

   Name    Fanout   Delay (ns)          Site               Resource
```

```
IN_DEL      ---     1.431        AB4.PAD to       AB4.INCK clk
ROUTE         1     0.816       AB4.INCK to  LLHPPLL.CLKIN clk_c
MCLK_DEL    ---     0.385  LLHPPLL.CLKIN to   LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       449     3.714   LLHPPLL.MCLK to        AE15.SC ddr_clk_c
                  --------
                    6.346   (28.6% logic, 71.4% route), 2 logic levels.

      Data path ddr_cas_n to ddr_cas_n:

   Name    Fanout   Delay (ns)          Site               Resource
OUTREG_DEL  ---     1.713        AE15.SC to       AE15.PAD ddr_cas_n (from ddr_clk_c)
                  --------
                    1.713   (100.0% logic, 0.0% route), 1 logic levels.

      Clock out path:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     1.431        AB4.PAD to       AB4.INCK clk
ROUTE         1     0.816       AB4.INCK to  LLHPPLL.CLKIN clk_c
MCLK_DEL    ---     0.385  LLHPPLL.CLKIN to   LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       449     1.191   LLHPPLL.MCLK to      AF3.OUTDD ddr_clk_c
OUTDD_DEL   ---     1.918      AF3.OUTDD to        AF3.PAD ddr_clk
                  --------
                    5.741   (65.0% logic, 35.0% route), 3 logic levels.

      Feedback path:

   Name    Fanout   Delay (ns)          Site               Resource
NCLK_DEL    ---     0.385  LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       136     2.886   LLHPPLL.NCLK to     LLHPPLL.FB pll_nclk
                  --------
                    3.271   (11.8% logic, 88.2% route), 1 logic levels.

Report:    2.318ns is the minimum offset for this preference.
```

From the Hold Report below, which was run for MIN conditions:

- t_{DDR_CLK} (min) = 3.043 - 1.905 = 1.138 ns

```
================================================================================
Preference: CLOCK_TO_OUT PORT "ddr_cas_n" MAX 5.500000 ns CLKPORT "clk" CLKOUT PORT "ddr_clk" ;
            1 item scored, 0 timing errors detected.
--------------------------------------------------------------------------------------------------------
----------------------------


Passed:  The following path meets requirements by 1.056ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)

   Source:         Unknown    Q            U1_ddrct_np_o4_1_008/U1_cmdexe/ddr_cas_nZ0  (from ddr_clk_c
-)
   Destination:    Port       Pad            ddr_cas_n

   Data Path Delay:     0.928ns  (100.0% logic, 0.0% route), 1 logic levels.
```

```
   Clock Path Delay:    3.171ns  (25.4% logic, 74.6% route), 2 logic levels.

 Constraint Details:

      3.171ns delay clk to ddr_cas_n less
      1.905ns feedback compensation
      0.928ns delay ddr_cas_n to ddr_cas_n less
      1.138ns delay clk to ddr_clk (totaling 1.056ns) meets
      0.000ns hold offset clk to ddr_cas_n by 1.056ns

 Physical Path Details:

      Clock path clk to ddr_cas_n:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     0.576        AB4.PAD to       AB4.INCK clk
ROUTE         1     0.507       AB4.INCK to  LLHPPLL.CLKIN clk_c
MCLK_DEL    ---     0.231  LLHPPLL.CLKIN to   LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       449     1.857   LLHPPLL.MCLK to        AE15.SC ddr_clk_c
                  --------
                    3.171   (25.4% logic, 74.6% route), 2 logic levels.

      Data path ddr_cas_n to ddr_cas_n:

   Name    Fanout   Delay (ns)          Site               Resource
OUTREG_DEL  ---     0.928        AE15.SC to       AE15.PAD ddr_cas_n (from ddr_clk_c)
                  --------
                    0.928   (100.0% logic, 0.0% route), 1 logic levels.

      Clock out path:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     0.576        AB4.PAD to       AB4.INCK clk
ROUTE         1     0.507       AB4.INCK to  LLHPPLL.CLKIN clk_c
MCLK_DEL    ---     0.231  LLHPPLL.CLKIN to   LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       449     0.778   LLHPPLL.MCLK to      AF3.OUTDD ddr_clk_c
OUTDD_DEL   ---     0.951      AF3.OUTDD to        AF3.PAD ddr_clk
                  --------
                    3.043   (57.8% logic, 42.2% route), 3 logic levels.

      Feedback path:

   Name    Fanout   Delay (ns)          Site               Resource
NCLK_DEL    ---     0.231  LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       136     1.674   LLHPPLL.NCLK to     LLHPPLL.FB pll_nclk
                  --------
                    1.905   (12.1% logic, 87.9% route), 1 logic levels.

Report:    1.056ns is the maximum offset for this preference.
================================================================================
```

From the Set-up Report below, which was run for MAX conditions. The report shown here is for ddr_ad.

- t_{CCTRL} (max) = (6.392-3.271) + 1.713 = 4.834 ns

Find delays similarly for ddr_ras_n, ddr_cas_n, ddr_we_n, ddr_ba, ddr_cs_n and ddr_cke signals. Then take the max of those delays as t_{CCTRL} (max).

```
================================================================================
Preference: CLOCK_TO_OUT PORT "ddr_ad_*" 5.500000 ns CLKNET "ddr_clk_c" ;
            12 items scored, 0 timing errors detected.

Passed:  The following path meets requirements by 0.666ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)

   Source:         Unknown    Q             U1_ddrct_np_o4_1_008/U1_cmdexe/ddr_adZ0Z_6  (from ddr_clk_c
-)
   Destination:    Port       Pad           ddr_ad_6

   Data Path Delay:     1.713ns  (100.0% logic, 0.0% route), 1 logic levels.

   Clock Path Delay:    6.392ns  (28.4% logic, 71.6% route), 2 logic levels.

 Constraint Details:

      6.392ns delay clk to ddr_ad_6 less
      3.271ns feedback compensation
      1.713ns delay ddr_ad_6 to ddr_ad_6 (totaling 4.834ns) meets
      5.500ns offset clk to ddr_ad_6 by 0.666ns

 Physical Path Details:

      Clock path clk to ddr_ad_6:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     1.431        AB4.PAD to       AB4.INCK clk
ROUTE         1     0.816       AB4.INCK to  LLHPPLL.CLKIN clk_c
MCLK_DEL    ---     0.385  LLHPPLL.CLKIN to   LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       449     3.760   LLHPPLL.MCLK to        AE14.SC ddr_clk_c
                  --------
                    6.392   (28.4% logic, 71.6% route), 2 logic levels.

      Data path ddr_ad_6 to ddr_ad_6:

   Name    Fanout   Delay (ns)          Site               Resource
OUTREG_DEL  ---     1.713        AE14.SC to       AE14.PAD ddr_ad_6 (from ddr_clk_c)
                  --------
                    1.713   (100.0% logic, 0.0% route), 1 logic levels.

      Feedback path:

   Name    Fanout   Delay (ns)          Site               Resource
NCLK_DEL    ---     0.385  LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       136     2.886   LLHPPLL.NCLK to     LLHPPLL.FB pll_nclk
                  --------
                    3.271   (11.8% logic, 88.2% route), 1 logic levels.

Report:    4.834ns is the minimum offset for this preference.
```

From the Hold Report below, which was run for MIN conditions. The report shown here is for ddr_ad* only.

- t_{CCTRL} (min) = (3.124-1.905) + 0.928 = 2.147 ns

Find delays similarly for ddr_ras_n, ddr_cas_n, ddr_we_n, ddr_ba, ddr_cs_n and ddr_cke signals. Then take the min of those delays as t_{CCTRL} (min).

```
================================================================================
Preference: CLOCK_TO_OUT PORT "ddr_ad_*" 5.500000 ns CLKNET "ddr_clk_c" ;
            12 items scored, 0 timing errors detected.

Passed:  The following path meets requirements by 2.147ns

 Logical Details:  Cell type  Pin type       Cell name  (clock net +/-)

   Source:         Unknown    Q             U1_ddrct_np_o4_1_008/U1_cmdexe/ddr_adZ0Z_4  (from ddr_clk_c
-)
   Destination:    Port       Pad           ddr_ad_4

   Data Path Delay:     0.928ns  (100.0% logic, 0.0% route), 1 logic levels.

   Clock Path Delay:    3.124ns  (25.8% logic, 74.2% route), 2 logic levels.

 Constraint Details:

      3.124ns delay clk to ddr_ad_4 less
      1.905ns feedback compensation
      0.928ns delay ddr_ad_4 to ddr_ad_4 (totaling 2.147ns) meets
      0.000ns hold offset clk to ddr_ad_4 by 2.147ns

 Physical Path Details:

      Clock path clk to ddr_ad_4:

   Name    Fanout   Delay (ns)          Site               Resource
IN_DEL      ---     0.576        AB4.PAD to       AB4.INCK clk
ROUTE         1     0.507       AB4.INCK to  LLHPPLL.CLKIN clk_c
MCLK_DEL    ---     0.231  LLHPPLL.CLKIN to   LLHPPLL.MCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       449     1.810   LLHPPLL.MCLK to         T26.SC ddr_clk_c
                  --------
                    3.124   (25.8% logic, 74.2% route), 2 logic levels.

      Data path ddr_ad_4 to ddr_ad_4:

   Name    Fanout   Delay (ns)          Site               Resource
OUTREG_DEL  ---     0.928         T26.SC to        T26.PAD ddr_ad_4 (from ddr_clk_c)
                  --------
                    0.928   (100.0% logic, 0.0% route), 1 logic levels.

      Feedback path:

   Name    Fanout   Delay (ns)          Site               Resource
NCLK_DEL    ---     0.231  LLHPPLL.CLKIN to   LLHPPLL.NCLK U2_ddr_pll_orca/ddr_pll_0_0
ROUTE       136     1.674   LLHPPLL.NCLK to     LLHPPLL.FB pll_nclk
                  --------
                    1.905   (12.1% logic, 87.9% route), 1 logic levels.

Report:    2.220ns is the maximum offset for this preference.
```

PCB Layout Recommendations for BGA Packages

September 2006 Technical Note TN1074

Introduction

As Ball Grid Array (BGA) packages become increasingly popular, it is important to understand how they are affected by various board layout techniques. This document provides a brief overview of PCB layout considerations when working with BGA packages. It outlines some of the most common problems and provides tips for avoiding them at the design stage.

Advantages and Disadvantages of BGA Packaging

One of the greatest advantages of BGA packaging over other new technologies is that it can be supported with existing placement and assembly equipment. BGAs also offer significantly more misalignment tolerance, less susceptibility to coplanarity issues and easier PCB signal routing under a BGA package (see Figure 20-1).

The primary drawback of BGA packaging is the inability to access the solder joints for testing and inspection (a later section in this document provides layout recommendations for testing). At best, only the outermost row of balls can be seen, and board size and other components often restrict even that view. The best option available for a complete inspection of the device is X-ray imaging. By this means, the user can visually assess shorted connections, missing balls, filled vias, and in some cases, opens (see Figure 20-2). Opens and partial opens (where the solder did not wet the entire pad) are more difficult to see and may require higher resolution equipment.

Figure 20-1. Misalignment of BGA Balls vs. QFP Leads

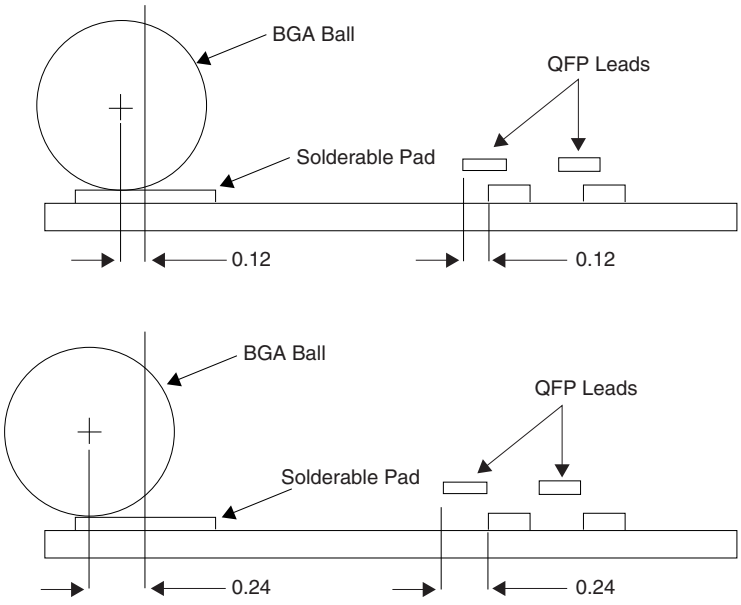
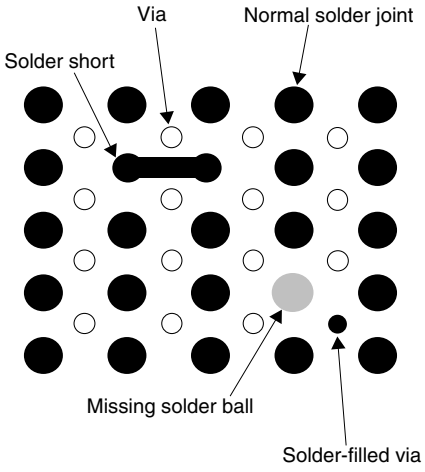


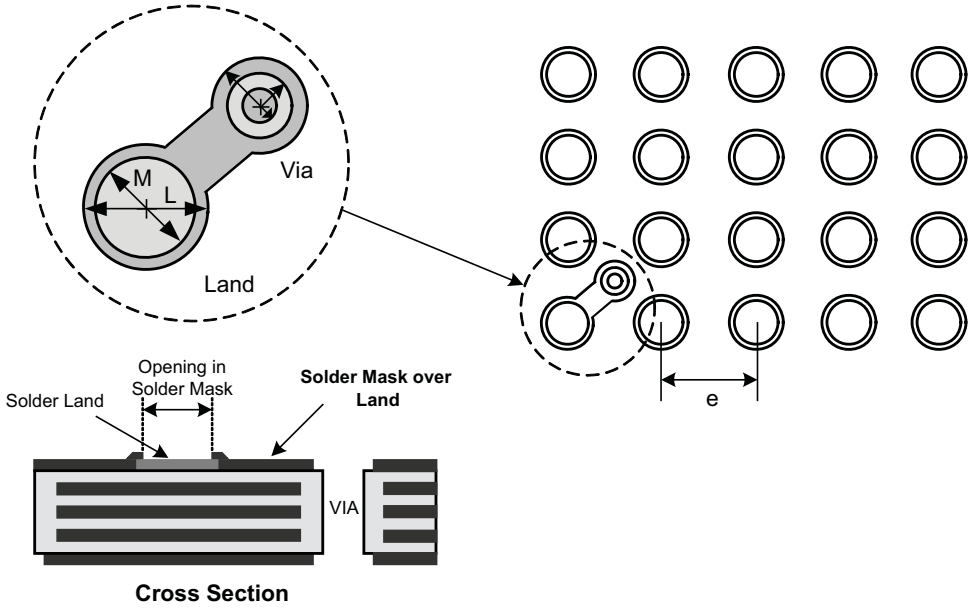
Figure 20-2. Example of How Defects May Appear in an X-Ray Image



PCB Layout

All Lattice BGA packages utilize Solder Mask Defined (SMD) pads. For optimized solder joint formation, the PCB pads should match the BGA solder pads (see Figure 20-3). For example, if the BGA has solder pads with 0.60 mm openings, so should the corresponding site on the PCB.

Figure 20-3. SMD Pad with Example Dimensions



Plated Through Hole (Via) Placement

Probably the most critical aspect of BGA PCB layout is the consideration for Plated Through Hole (PTH) placement. If the pad is too close or on top of the hole, or if there is no solder mask covering the via, then it is possible for the ball solder and paste to melt and be wicked into the hole. If enough solder is lost into the hole, the result could be an open for that lead. While this type of defect can usually be detected in an X-ray image, it is best avoided at layout (see Figure 20-2).

BGA Board Layout Recommendations

	Pitch 0.5mm csBGA	Pitch 0.8mm caBGA	Pitch 1.00mm (fpBGA, ftBGA, fpSBGA, fcBGA)				Pitch 1.27mm PBGA, SBGA
			All Other fpBGA, ftBGA, fpSBGA	100 fpBGA, 256 ftBGA	Organic fcBGA	Ceramic fcBGA	
Solder Land Diameter (L)	0.43	0.53	0.66	0.53	0.80	0.70	0.80
Opening in Solder Mask (M)	0.30	0.40	0.45	0.40	0.60	0.50	0.63
Solder Ball Land Pitch (e)	0.50	0.80	1.00	1.00	1.00	1.00	1.27

Note: The numbers in this table are intended to be used as an example only. The actual numbers are dependent on the PCB manufacturing tolerance.

BGA Package Types

Package Type	Description
PBGA	**Plastic BGA** with 1.27 mm solder ball pitch. Die up configuration.
fpBGA	**Fine Pitch BGA** – Plastic BGA with 1.0 mm solder ball pitch. Die up configuration.
ftBGA	**Fine Pitch Thin BGA** – Thin plastic BGA with 1.0 mm solder ball pitch. Die up configuration.
caBGA	**Chip Array BGA** – Plastic BGA with 0.8 mm solder ball pitch. Die up configuration.
csBGA	**Chip Scale BGA** – Plastic BGA with 0.5 mm solder ball pitch. Die up configuration.
fcBGA	**Flip-Chip BGA** with 1.0 mm solder ball pitch. Die down configuration. May have a ceramic or plastic substrate.
SBGA	**Super BGA** – Similar to PBGA, but with an integrated heatsink plate. This package has 1.27 mm solder ball pitch and die down configuration. SBGA packages offer enhanced thermal dissipation capability.
fpSBGA	**Fine Pitch SBGA** – Super BGA with 1.0 mm solder ball pitch. Die down configuration.

Further Information

For additional information, please visit the web sites listed below.

www.amkor.com/Products/all_products

www.latticesemi.com/lit/docs/package/amkor_bga_appnote.pdf

Technical Support Assistance

Hotline: 1-800-LATTICE (North America)
+1-503-268-8001 (Outside North America)
e-mail: techsupport@latticesemi.com
Internet: www.latticesemi.com

Revision History

Date	Version	Change Summary
January 2005	01.0	Initial release.
November 2005	01.1	Figures updated.
June 2006	01.2	Removed NSMD content.
September 2006	01.3	Added note to BGA Board Layout Recommendations table.
		Reformatted BGA Package Types section in tabular format.

Section III. LatticeXP Family Handbook Revision History

LatticeXP Family Handbook Revision History

April 2007 **Handbook HB1001**

Revision History

Date	Handbook Revison Number	Change Summary
February 2005	01.0	Initial release.
April 2005	01.1	LatticeXP Family Data Sheet updated to version 01.1.
		Technical note TN1051 updated to version 01.1.
May 2005	01.2	LatticeXP Family Data Sheet updated to version 01.2.
July 2005	01.3	LatticeXP Family Data Sheet updated to version 02.0.
		Technical note TN1052 updated to version 02.1.
July 2005	01.4	LatticeXP Family Data Sheet updated to version 02.1.
		Technical note TN1056 updated to version 03.1.
August 2005	01.5	LatticeXP Family Data Sheet updated to version 02.2.
September 2005	01.6	LatticeXP Family Data Sheet updated to version 03.0.
September 2005	01.7	
October 2005	01.8	LatticeXP Family Data Sheet updated to version 03.1.
		Technical note TN1056 updated to version 03.2.
		Technical note TN1051 updated to version 01.5.
		Technical note TN1050 updated to version 03.1.
		Technical note TN1049 updated to version 04.0.
		Technical note TN1082 updated to version 01.3.
		Technical note TN1054 updated to version 01.1.
		Technical note TN1008 updated to version 02.1.
		Technical note TN1074 updated to version 01.1.
December 2005	02.0	LatticeXP Family Data Sheet updated to version 04.0.
February 2006	02.1	LatticeXP Family Data Sheet updated to version 04.1.
		Technical note TN1051 updated to version 01.6.
		Technical note TN1082 updated to version 01.4.
		Technical note TN1054 updated to version 01.2.
March 2006	02.2	LatticeXP Family Data Sheet updated to version 04.2.
March 2006	02.3	LatticeXP Family Data Sheet updated to version 04.3.
		Technical note TN1056 updated to version 03.3.
April 2006	02.4	LatticeXP Family Data Sheet updated to version 04.4.
May 2006	02.5	LatticeXP Family Data Sheet updated to version 04.5.
		Technical note TN1051 updated to version 01.7.
October 2006	02.6	LatticeXP Family Data Sheet updated to version 04.7.
		Technical note TN1074 updated to version 01.3.
		Technical note TN1051 updated to version 01.8.
		Technical note TN1049 updated to version 04.1.
December 2006	02.7	LatticeXP Family Data Sheet updated to version 04.8.
		Technical note TN1052 updated to version 02.2.
February 2007	02.8	LatticeXP Family Data Sheet updated to version 04.9.

Date	Handbook Revison Number	Change Summary
April 2007	02.9	Technical note TN1050 updated to version 03.2.

Note: For detailed revision changes, please refer to the revision history for each document.