iCE40™ LP/HX Family Handbook

HB1011 Version 01.0, October 2012

iCE40 LP/HX Family Handbook Table of Contents

October 2012

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Section I. iCE40 LP/HX Family Overview

Advance DS1040A Version 01.0, October 2012

iCE40 LP/HX Family Overview Introduction

October 2012 Advance Data Sheet DS1040A

Features

- **Flexible Logic Architecture**
  - Four devices with 384 to 7,680 LUT4s and 21 to 206 I/Os
- **Ultra Low Power Devices**
  - Advanced 40 nm low power process
  - As low as 10 µW standby power
  - Programmable low swing differential I/Os
  - Stand-by mode power saving option
- **Embedded and Distributed Memory**
  - Up to 128 Kbits sysMEM™ Embedded Block RAM
- **Pre-Engineered Source Synchronous I/O**
  - DDR registers in I/O cells
- **High Performance, Flexible I/O Buffer**
  - Programmable sysIO™ buffer supports wide range of interfaces:
    - LVCMOS 3.3/2.5/1.8/1.5
    - LVDS25E, Sub-LVDS
    - Schmitt trigger inputs, up to 50 mV hysteresis
  - Programmable pull-up mode
- **Flexible On-Chip Clocking**
  - Eight low-skew global clock resources
  - Up to two analog PLLs per device
- **Flexible Device Configuration**
  - Nonvolatile Configuration Memory (NVCM)
    - Internal, lowest-cost, secure, Non-volatile Configuration Memory
    - Single-chip, secure solution
  - Configurable through standard SPI interface or CPU interface
- **Broad Range of Package Options**
  - QFN, VQFP, TQFP, and csBGA package options
  - Small footprint package options
    - As small as 2.5x2.5mm
  - Advanced halogen-free packaging

Table 1-1. iCE40 Family Selection Guide

Part Number		LP384	LP1K	LP4K	LP8K	HX1K	HX4K	HX8K
Logic Cells (LUT + Flip-Flop)		384	1,280	3,520	7,680	1,280	3,520	7,680
RAM4K Memory Blocks		0	16	20	32	16	20	32
RAM4K RAM bits		0	64K	80K	128K	64K	80K	128K
Phase-Locked Loops (PLLs)		0	1[2]	2[3]	2	1[2]	2	2
Configuration bits (maximum)		63 Kb	273 Kb	546 Kb	1,092 Kb	273 Kb	546 Kb	1,092 Kb
Core Operating Current at 0 KHz[1]		10 µA	52 µA	107 µA	217 µA	93 µA	231 µA	493 µA
Maximum Programmable I/O Pins		63	95	167	178	95	95	206
Maximum Differential Input Pairs		8	12	20	23	11	12	26
Package	**Code**	**Programmable I/O: Max I/O (LVDS25E)**						
32 QFN (5 x 5mm, 0.5mm)	QN32	21(4)						
36 ucBGA (2.5 x 2.5mm, 0.4mm)	CM36[4]		25(3)[2]					
	CM36A	25(3)	25(3)[2]					
49 ucBGA (3 x 3mm, 0.4mm)	CM49	37(6)	35(5)					
81 ucBGA (4 x 4mm, 0.4mm)	CM81	55(3)	63(8)	63(9)[3]				
81 csBGA (5 x 5mm, 0.5mm)	CB81		62(8)					
84 QFN (7 x 7mm, 0.5mm)	QN84		67(7)[2]					

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Table 1-1. iCE40 Family Selection Guide (Cont.)

Package	Code	LP384	LP1K	LP4K	LP8K	HX1K	HX4K	HX8K
		Programmable I/O: Max I/O (LVDS25E)						
100 VQFP (14 x 14mm, 0.5mm)	VQ100					72(9)[2]		
121 ucBGA (5 x 5mm, 0.4mm)	CM121		95 (12)	93 (13)	93 (13)			
121 csBGA (6 x 6mm, 0.5mm)	CB121		92 (12)					
132 csBGA (8 x 8mm, 0.5mm)	CB132					95(11)	95(12)	95(12)
144 TQFP (20 x 20mm, 0.5mm)	TQ144					96(12)	107(14)	
225 ucBGA (7 x 7mm, 0.4mm)	CM225			167 (20)	178 (23)			178(23)
256-ball caBGA (14 x 14mm, 0.8mm)	CT256							206(26)

1. Typical Icc with LP devices at 1.0V Vcc and HX at 1.2V Vcc.
2. No PLL available on the CM36, CM36A, QN84 and VQ100 packages.
3. Only one PLL available on the CM81 package.
4. New designs should use the CM36A package.

Introduction

The iCE40 family of ultra-low power, non-volatile FPGAs has four devices with densities ranging from 384 to 7680 Look-Up Tables (LUTs). In addition to LUT-based, low-cost programmable logic, these devices feature Embedded Block RAM (EBR), Non-volatile Configuration Memory (NVCM) and Phase Locked Loops (PLLs). These features allow the devices to be used in low-cost, high-volume consumer and system applications.

The iCE40 devices are fabricated on a 40 nm CMOS low power process. The device architecture has several features such as programmable low-swing differential I/Os and the ability to turn off on-chip PLLs dynamically. These features help manage static and dynamic power consumption, resulting in low static power for all members of the family. The iCE40 devices are available in two versions – ultra low power (LP) and high performance (HX) devices.

The iCE40 FPGAs are available in a broad range of advanced halogen-free packages ranging from the space saving 2.5x2.5 mm chip-scale BGA to the PCB-friendly 20x20 mm TQFP. Table 1-1 shows the LUT densities, package and I/O options, along with other key parameters.

The iCE40 devices offer enhanced I/O features such as pull-up resistors, open drain outputs. Pull-up features are controllable on a “per-pin” basis.

The iCE40 devices also provide flexible, reliable and secure configuration from on-chip NVCM. These devices can also configure themselves from external SPI Flash or be configured by an external master such as a CPU.

Lattice provides a variety of design tools that allow complex designs to be efficiently implemented using the iCE40 family of devices. Popular logic synthesis tools provide synthesis library support for iCE40. Lattice design tools use the synthesis tool output along with the user-specified preferences and constraints to place and route the design in the iCE40 device. These tools extract the timing from the routing and back-annotate it into the design for timing verification.

Lattice provides many pre-engineered IP (Intellectual Property) modules, including a number of reference designs, licensed free of charge, optimized for the iCE40 FPGA family. By using these configurable soft core IP cores as standardized blocks, users are free to concentrate on the unique aspects of their design, increasing their productivity.

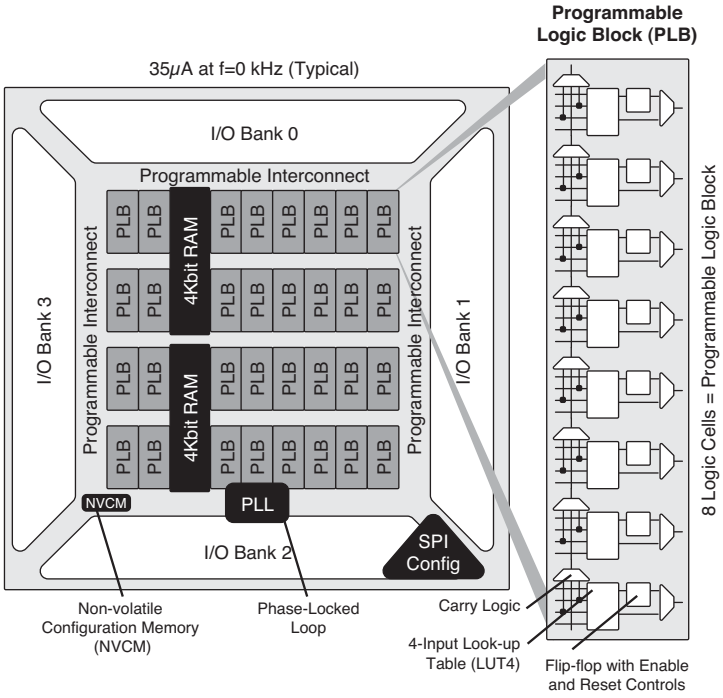
iCE40 LP/HX Family Data Overview Architecture

October 2012 Advance Data Sheet DS1040

Architecture Overview

The iCE40 family architecture contains an array of Programmable Logic Blocks (PLB), sysCLOCK™ PLLs, Non-volatile Programmable Configuration Memory (NVCM) and blocks of sysMEM™ Embedded Block RAM (EBR) surrounded by Programmable I/O (PIO). Figure 2-1 shows the block diagram of the iCE40-1K device.

Figure 2-1. iCE40-1K Device, Top View



The logic blocks, Programmable Logic Blocks (PLB) and sysMEM EBR blocks, are arranged in a two-dimensional grid with rows and columns. Each row has either logic blocks or EBR blocks. The PIO cells are located at the periphery of the device, arranged in banks. The PLB contains the building blocks for logic, arithmetic, and register functions. The PIOs utilize a flexible I/O buffer referred to as a sysIO buffer that supports operation with a variety of interface standards. The blocks are connected with many vertical and horizontal routing channel resources. The place and route software tool automatically allocates these routing resources.

In the iCE40 family, there are four sysIO banks. There are different types of I/O buffers on the different banks. Refer to the details in later sections of this document. The sysMEM EBRs are large 4 Kbit, dedicated fast memory blocks. These blocks can be configured as RAM, ROM or FIFO.

The iCE40 architecture also provides up to two sysCLOCK Phase Locked Loop (PLL) blocks. The PLLs have multiply, divide, and phase shifting capabilities that are used to manage the frequency and phase relationships of the clocks.

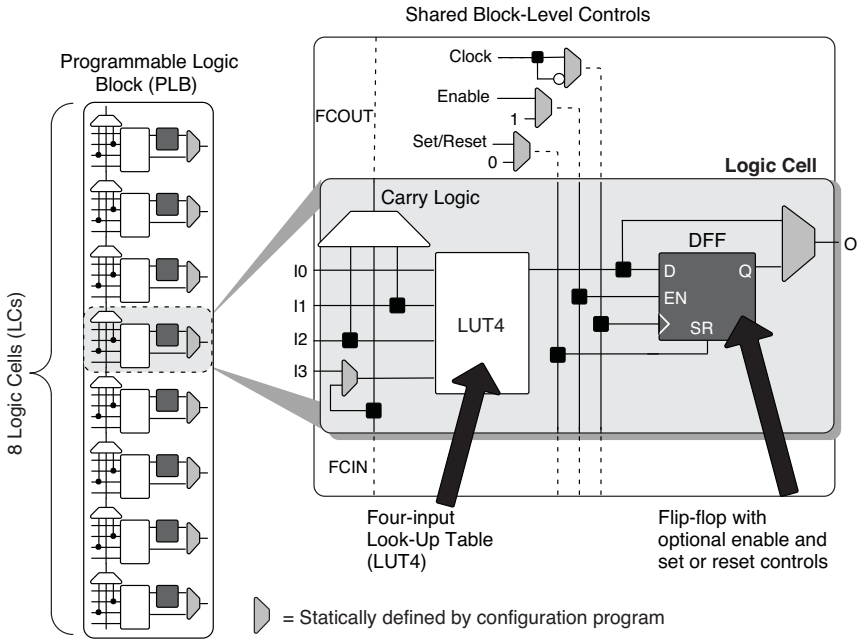
Every device in the family has a SPI port that supports programming and configuration of the device. The iCE40 includes on-chip, Nonvolatile Configuration Memory (NVCM).

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

PLB Blocks

The core of the iCE40 device consists of Programmable Logic Blocks (PLB) which can be programmed to perform logic and arithmetic functions. Each PLB consists of eight interconnected Logic Cells (LC) as shown in Figure 2-2. Each LC contains one LUT and one register.

Figure 2-2. PLB Block Diagram



Logic Cells

Each Logic Cell includes three primary logic elements shown in Figure 2-2.

- A four-input Look-Up Table (LUT4) builds any combinational logic function, of any complexity, requiring up to four inputs. Similarly, the LUT4 element behaves as a 16x1 Read-Only Memory (ROM). Combine and cascade multiple LUT4s to create wider logic functions.
- A 'D'-style Flip-Flop (DFF), with an optional clock-enable and reset control input, builds sequential logic functions. Each DFF also connects to a global reset signal that is automatically asserted immediately following device configuration.
- Carry Logic boosts the logic efficiency and performance of arithmetic functions, including adders, subtracters, comparators, binary counters and some wide, cascaded logic functions.

Table 2-1. Logic Cell Signal Descriptions[2]

Function	Type	Signal Names	Description
Input	Data signal	I0, I1, I2, I3	Inputs to LUT4
Input	Control signal	Enable	Clock enable shared by all LCs in the PLB
Input	Control signal	SR[1]	Asynchronous or synchronous local set/reset shared by all LCs in the PLB.
Input	Control signal	Clock	Clock one of the eight Global Buffers, or from the general-purpose interconnects fabric shared by all LCs in the PLB
Input	Inter-PLB signal	FCIN	Fast carry in[1]
Output	Data signals	O	LUT4 output or register bypass signals
Output	Inter-PFU signal	FCO	Fast carry out[1]

1. If SR is not used, then the flip-flop is never set/reset, except when cleared immediately after configuration or by the Global Reset signal.

Modes of Operation
Each LC has up to three potential modes of operation: Logic, Ripple, or ROM.

Logic Mode: In this mode, the LCs in each PLB are configured as 4-input combinatorial look-up tables. A LUT4 can have 16 possible input combinations. Any four-input logic functions can be generated by programming this lookup table.

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

- Addition 1-bit
- Subtraction 1-bit
- Add/subtract 1-bit using dynamic control
- Up counter 1-bit
- Down counter 1-bit
- Up/down counter with asynchronous clear
- Up/down counter with preload (sync)
- Ripple mode multiplier building block
- Multiplier support
- Comparator functions of A and B inputs
  - A greater-than-or-equal-to B
  - A not-equal-to B
  - A less-than-or-equal-to B

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

ROM Mode: ROM mode uses the LUT logic. Preloading is accomplished through the programming interface during PLB configuration.

Routing

There are many resources provided in the iCE40 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-PLB connections are made with three different types of routing resources: x1 (spans two PLBs), x4 (spans three PLBs) and x12 (spans seven PLBs). The x1, x4 and x12 connections provide fast and efficient connections in the horizontal and vertical directions.

The design tool take 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/Control Distribution Network

Each iCE40 device has eight global inputs, two pins on each side of the device. These can be used as clock, reset or enable signals. The dedicated global pins are identified as GBIN[7..0] and the global buffers are identified as-GBUF[7..0]. These eight inputs may be used as general purpose I/O if they are not used to drive the clock nets. Global buffer GBUF7 in I/O Bank 3 also provides an optional direct LVDS25E or Sub-LVDS differential clock input.

Table 2-2 lists the connections between a specific global buffer and the inputs on a PLB. All global buffers optionally connect to all clock inputs. Any four of the eight global buffers can drive logic inputs to a PLB. Even-numbered

global buffers optionally drive the Reset input to a PLB. Similarly, odd-numbered buffers optionally drive the PLB clock-enable input.

Table 2-2. Global Buffer (GBUF) Connections to Programmable Logic Blocks

Global Buffer	LUT Inputs	Clock	Clock Enable	Reset
GBUF0	Yes, any 4 of 8 GBUF Inputs	✓	✓	
GBUF1		✓		✓
GBUF2		✓	✓	
GBUF3		✓		✓
GBUF4		✓	✓	
GBUF5		✓		✓
GBUF6		✓	✓	
GBUF7		✓		✓

The maximum frequency for the global buffers are shown in the iCE40 External Switching Characteristics tables later in this document.

Global Hi-Z Control

The global high-impedance control signal, GHIZ, connects to all I/O pins on the iCE40 device. This GHIZ signal is automatically asserted throughout the configuration process, forcing all user I/O pins into their high-impedance state. Similarly, the PIO pins can be forced into their high-impedance state via the Scan Test controller.

Global Reset Control

The global reset control signal connects to all PLB and PIO flip-flops on the iCE40 device. The global reset signal is automatically asserted throughout the configuration process, forcing all flip-flops to their defined wake-up state. For PLB flip-flops, the wake-up state is always reset, regardless of the PLB flip-flop primitive used in the application. The PIO flip-flops are always reset during configuration, although the output flip-flop can be inverted before leaving the iCE40 device.

sysCLOCK Phase Locked Loops (PLLs)

The sysCLOCK PLLs provide the ability to synthesize clock frequencies. The iCE40 devices have one or more sysCLOCK PLLs. REFERENCECLK is the reference frequency input to the PLL and its source can come from an external I/O pin or from internal routing. EXTFEEDBACK is the feedback signal to the PLL which can come from internal routing or an external I/O pin. The feedback divider is used to multiply the reference frequency and thus synthesize a higher frequency clock output.

The PLLOUT output has an output divider, thus allowing the PLL to generate different frequencies for each output. The output divider can have a value from 1 to 8. The PLLOUT outputs can all be used to drive the iCE40 global clock network directly or general purpose routing resources can be used.

The LOCK signal is asserted when the PLL determines it has achieved lock and de-asserted if a loss of lock is detected. A block diagram of the PLL is shown in Figure 2-3.

The setup and hold times of the device can be improved by programming a phase shift into the PLLOUT output clock which will advance or delay the output clock with reference to the REFERENCECLK clock. This phase shift can be either programmed during configuration or can be adjusted dynamically. In dynamic mode, the PLL may lose lock after a phase adjustment on the output used as the feedback source and not relock until the t_{LOCK} parameter has been satisfied.

For more details on the PLL, see TN1251, iCE40 sysCLOCK PLL Design and Usage Guide.

Figure 2-3. PLL Diagram

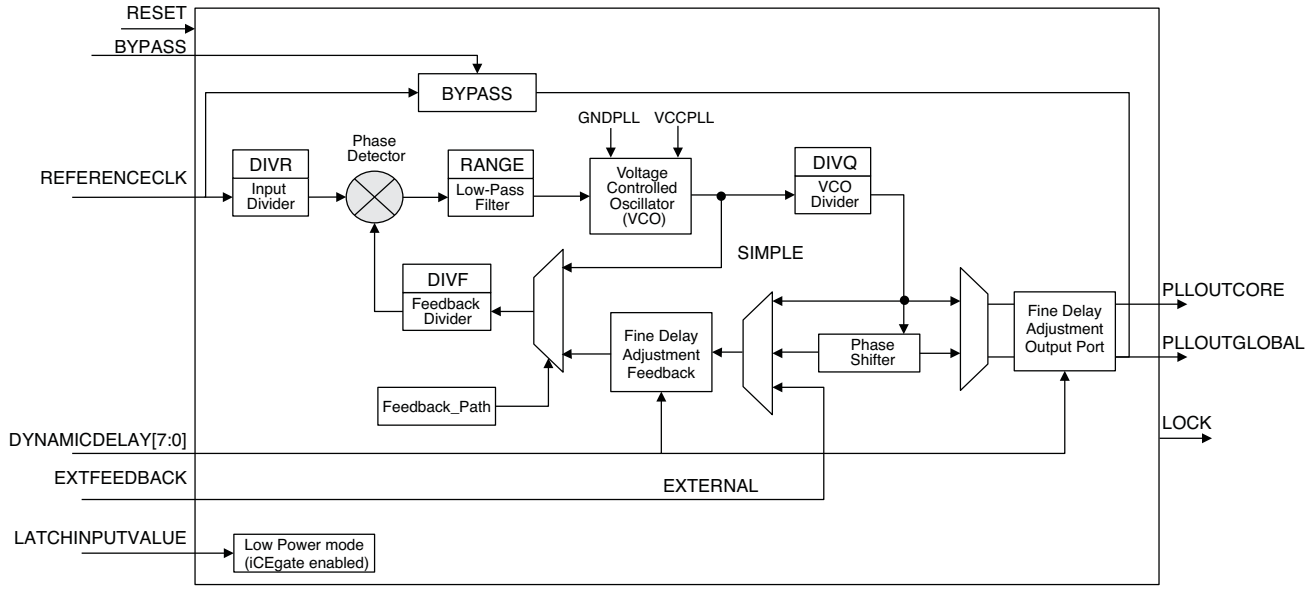


Table 2-3 provides signal descriptions of the PLL block.

Table 2-3. PLL Signal Descriptions

Port Name	I/O	Description
REFERENCECLK	I	Input clock to PLL
EXTFEEDBACK	I	When FEEDBACK_PATH is set to SIMPLE, the BYPASS control selects which clock signal connects to the PLLOUT output. 0 = PLL generated signal 1 = REFERENCECLK
DYNAMIC_DELAY[7:0]	I	Fine delay adjustment control inputs. Enabled when DELAY_ADJUSTMENT_MODE is set to DYNAMIC.
PLLOUT	O	Output from the Phase-Locked Loop (PLL). Connects to programmable interconnect and has optimal connections to global clock buffers GBUF4 and GBUF5.
LOCK	O	When High, indicates that the PLL output is phase aligned or locked to the input reference clock.
LATCHINPUTVALUE	I	When enabled, forces the PLL into low-power mode; PLL output held static at last input clock value. Set ENABLE ICEGATE_PORTA and PORTB to ‘1’ to enable.
RESET	I	Active high reset.

sysMEM Embedded Block RAM Memory

Each iCE40 device includes multiple high-speed synchronous sysMEM Embedded Block RAMs (EBRs), each 4 Kbit in size. This memory can be used for a wide variety of purposes including data buffering, PROM for the soft processor and FIFO.

sysMEM Memory Block

The sysMEM block can implement single port, dual port, pseudo dual port, or FIFO memories with programmable logic resources. Each block can be used in a variety of depths and widths as shown in Table 2-4.

Table 2-4. sysMEM Block Configurations

Block RAM Configuration	Block RAM Configuration and Size	WADDR Port Size (Bits)	WDATA Port Size (Bits)	RADDR Port Size (Bits)	RDATA Port Size (Bits)	MASK Port Size (Bits)
SB_RAM256x16 SB_RAM256x16NR SB_RAM256x16NW SB_RAM256x16NRNW	256x16 (4K)	8 [7:0]	16 [15:0]	8 [7:0]	16 [15:0]	16 [15:0]
SB_RAM512x8 SB_RAM512x8NR SB_RAM512x8NW SB_RAM512x8NRNW	512x8 (4K)	9 [8:0]	8 [7:0]	9 [7:0]	8 [7:0]	No Mask Port
SB_RAM1024x4 SB_RAM1024x4NR SB_RAM1024x4NW SB_RAM1024x4NRNW	1024x4 (4K)	10 [9:0]	4 [7:0]	10 [7:0]	4 [3:0]	No Mask Port
SB_RAM2048x2 SB_RAM2048x2NR SB_RAM2048x2NW SB_RAM2048x2NRNW	2048x2 (4K)	11 [10:0]	2 [7:0]	11 [7:0]	2 [1:0]	No Mask Port

RAM Initialization and ROM Operation

If desired, the contents of the RAM can be pre-loaded during device configuration.

By preloading the RAM block during the chip configuration cycle and disabling the write controls, the sysMEM block can also be utilized as a ROM.

Memory Cascading

Larger and deeper blocks of RAM can be created using EBR sysMEM Blocks.

RAM4k Block

Figure 2-4 shows the 256x16 memory configurations and their input/output names. In all the sysMEM RAM modes, the input data and addresses for the ports are registered at the input of the memory array. The output data of the memory is registered at the memory array output.

Figure 2-4. sysMEM Memory Primitives

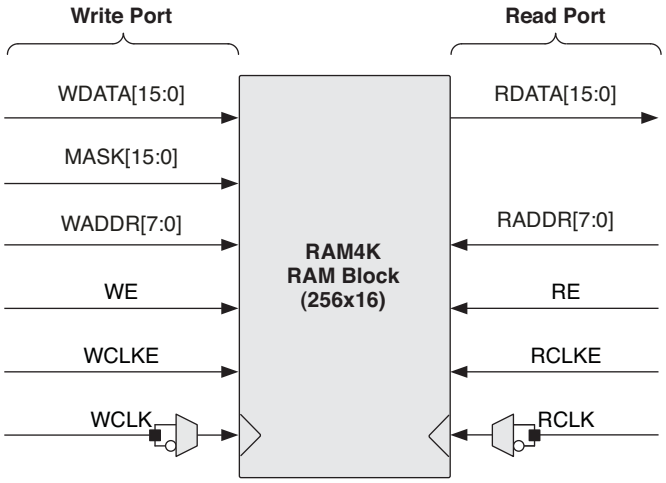


Table 2-5. EBR Signal Descriptions

Signal Name	Direction	Description
WDATA[15:0]	Input	Write Data input.
MASK[15:0]	Input	Masks write operations for individual data bit-lines. 0 = write bit; 1 = don't write bit
WADDR[7:0]	Input	Write Address input. Selects one of 256 possible RAM locations.
WE	Input	Write Enable input.
WCLKE	Input	Write Clock Enable input.
WCLK	Input	Write Clock input. Default rising-edge, but with falling-edge option.
RDATA[15:0]	Output	Read Data output.
RADDR[7:0]	Input	Read Address input. Selects one of 256 possible RAM locations.
RE	Input	Read Enable input.
RCLKE	Input	Read Clock Enable input.
RCLK	Input	Read Clock input. Default rising-edge, but with falling-edge option.

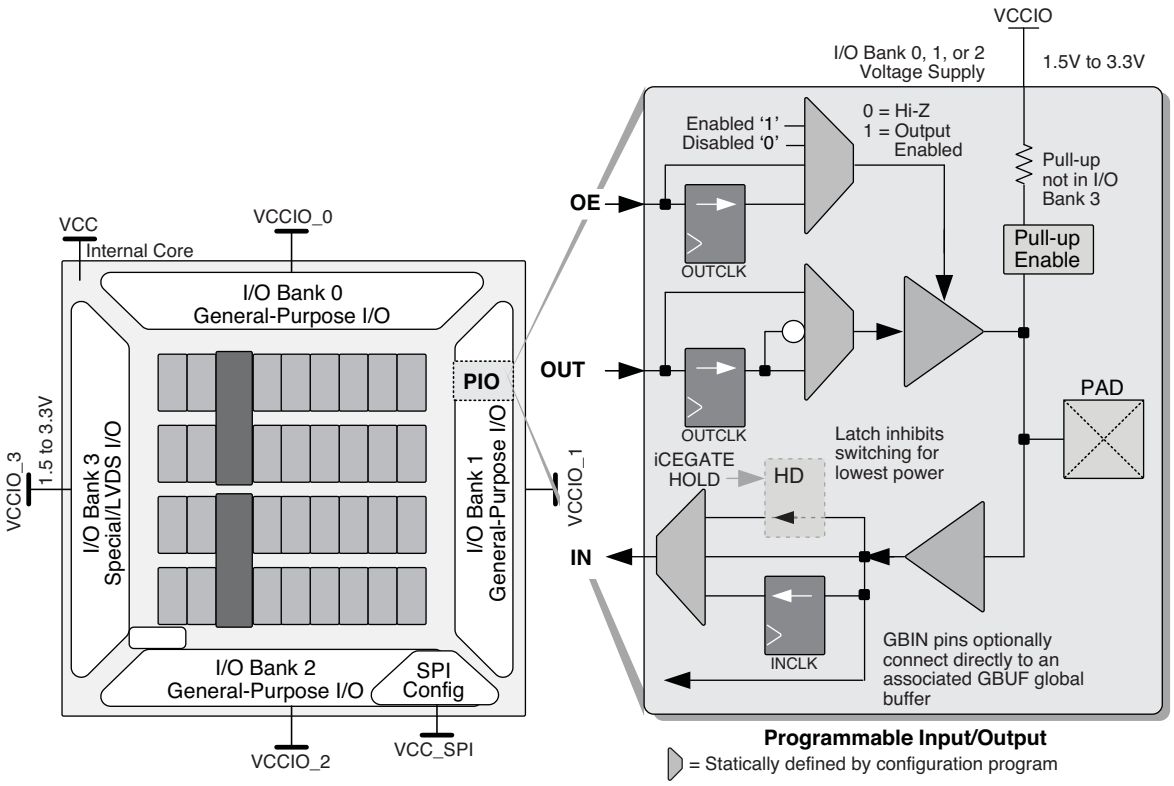
For further information on the sysMEM EBR block, please refer to TN1250, Memory Usage Guide for iCE40 Devices.

Programmable I/O (PIO)

The programmable logic associated with an I/O is called a PIO. The individual PIO are connected to their respective sysIO buffers and pads. The PIOs are placed on all four sides of the device.

I/O Bank 3 has additional capabilities, including LVDS25E differential I/O and the ability to interface to Mobile DDR memories.

Figure 2-5. I/O Bank and Programmable I/O Cell



The PIO contains three blocks: an input register block, output register block iCEgate™ and tri-state register block. To save power, the optional iCEgate latch can selectively freeze the state of individual, non-registered inputs within an I/O bank. These blocks contain registers for operating in a variety of modes along with the necessary clock and selection logic.

Table 2-6. PIO Signal List

Pin Name	I/O Type	Description
OUTCLK	Input	Output register clock
IOENA	Input	Output register clock enable
INCLK	Input	Input register clock enable
OE	Input	Output enable
OUT	Input	Output data from the core
IN	Output	Input data to the core

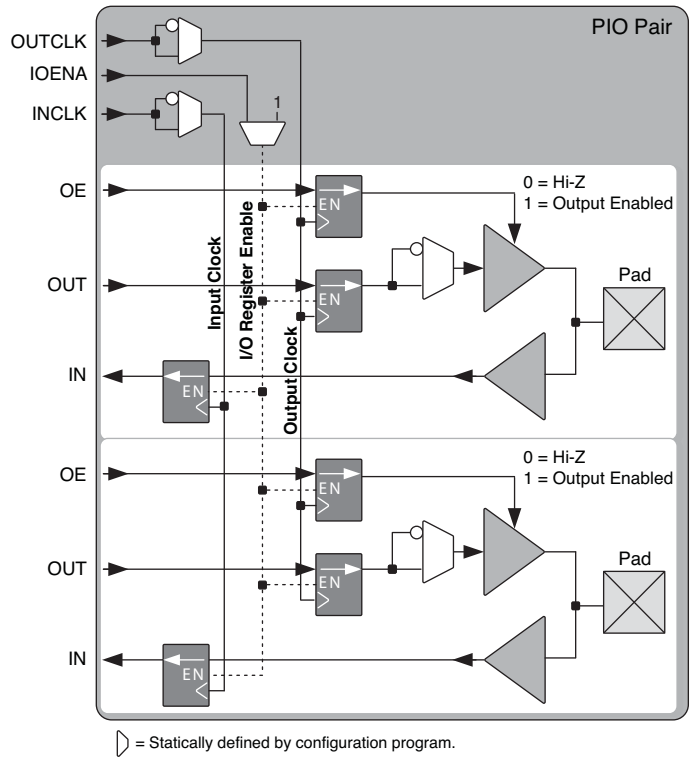
1. Available in PIO on right edge only.

Input Register Block

The input register blocks for the PIOs on all edges contain registers that can be used to condition high-speed interface signals before they are passed to the device core.

Figure 2-6 shows the input register block for the PIOs.

Figure 2-6. iCE I/O Register Block Diagram



sysIO Buffer

Each I/O is associated with a flexible buffer referred to as a sysIO buffer. These buffers are arranged around the periphery of the device in groups referred to as banks. The sysIO buffers allow users to implement a wide variety of standards that are found in today's systems including LVCMOS and LVDS25E.

Each bank is capable of supporting multiple I/O standards. In the iCE40 devices, single-ended LVCMOS buffers, and Bank 3 additionally support differential LVDS25E buffers. Each sysIO bank has its own VCCIO.

Typical I/O Behavior During Power-up

The internal power-on-reset (POR) signal is deactivated when VCC, VCCIO_2, VPP_2V5, and VCC_SPI have reached the level defined in the Power-On-Reset Voltage table in the DC and Switching Characteristics section of this data sheet. After the POR signal is deactivated, the FPGA core logic becomes active. It is the user's responsibility to ensure that all 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-stated with a weak pull-up to V_{CCIO}. The I/O pins will maintain the blank configuration until V_{CC} and V_{CCIO} (for I/O banks containing configuration I/Os) have reached levels, at which time the I/Os will take on the user-configured settings only after a proper download/configuration.

Supported Standards

The iCE40 sysIO buffer supports both single-ended and Bank 3 supports differential standards. The single-ended standard supported is LVCMOS. The buffer supports the LVCMOS 1.5, 1.8, 2.5, and 3.3V standards. The buffer has individually configurable options for bus maintenance (weak pull-up or none) in Banks 0, 1, 2 and open drain support in all banks. Differential receivers for LVDS25E are supported on Bank 3 of iCE40 devices.

Figure 2-7 show the I/O standards (together with their supply and reference voltages) supported by the iCE40 devices.

Table 2-7. Supported Input Standards

Input Standard	VCCIO (Typical)			
	3.3V	2.5V	1.8V	1.5
Single-Ended Interfaces				
LVCMOS33	✓			
LVCMOS25		✓		
LVCMOS18			✓	
LVCMOS15				✓[2]
Differential Interfaces[1]				
LVDS25E		✓	✓	

1. Bank 3 only.
2. Not supported by the Configuration Bank VCC_SPI.

Table 2-8. Supported Output Standards

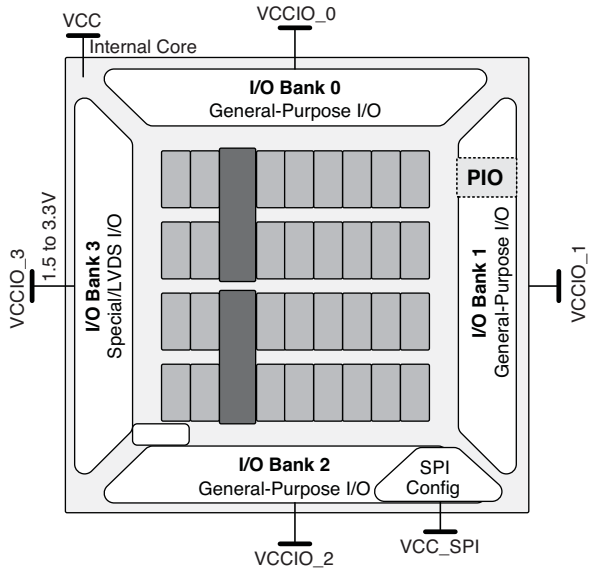
Output Standard	V_{CCIO} (Typical)
Single-Ended Interfaces	
LVCMOS33	3.3
LVCMOS25	2.5
LVCMOS18	1.8
LVCMOS15[3]	1.5
LVCMOS33, Open Drain	—
LVCMOS25, Open Drain	—
LVCMOS18, Open Drain	—
LVCMOS15, Open Drain	—
Differential Interfaces	
LVDS25E[1, 2]	2.5

1. Bank 3 only.
2. These interfaces can be emulated with external resistors in all devices.
3. Not supported by the Configuration Bank VCC_SPI.

sysIO Buffer Banks

iCE40 devices have four I/O banks with an additional configuration bank VCC_SPI for the SPI I/Os. Figure 2-7 shows the sysIO banks and their associated supplies for all devices.

Figure 2-7. iCE40 Banks



Configuration Frequencies

The typical configuration frequency ranges from 8.5 MHz to 42.5 MHz. The software default value of the Master Clock (MCLK) is nominally 8.5 MHz.

Table 2-9 lists all the available MCLK frequencies.

Table 2-9. Available MCLK Frequencies

Symbol	Oscillator Mode	Frequency (MHz)		Description
		Min.	Max.	
f_{OSCD}	Default	7	10	Default oscillator frequency. Slow enough to safely operate with any SPI serial PROM.
f_{OSCL}	Low Frequency	21	30	Supported by most SPI serial Flash PROMs.
f_{OSCH}	High Frequency	35	50	Supported by some high-speed SPI serial Flash PROMs.
	Off	0	0	Oscillator turned off by default after configuration to save power.

Non-Volatile Configuration Memory

All iCE40 devices provide a Non-Volatile Configuration Memory (NVCM) block which can be used to configure the device.

For more information on the NVCM, please refer to TN1248, iCE40 Programming and Configuration Usage Guide.

Power On Reset

iCE40 devices have power-on reset circuitry to monitor V_{CC} VCCIO_2, VPP_2V5, and VCC_SPI voltage levels during power-up and operation. At power-up, the POR circuitry monitors V_{CC} VCCIO_2, VPP_2V5, and VCC_SPI (controls configuration) voltage levels. It then triggers download from the on-chip configuration Flash memory after reaching the power-up levels specified in the Power-On-Reset Voltage table in the DC and Switching Characteristics section of this data sheet. Before and during configuration, the I/Os are held in tri-state. I/Os are released to user functionality once the device has finished configuration.

Programming, Configuration and Testing

This section describes the programming, configuration and testing features of the iCE40 family.

Device Programming
The NVCM memory can be programmed through the SPI port or be factory programmed.

Device Configuration
There are various ways to configure an iCE40 device including:

1. Internal NVCM Download
2. Standard Serial Peripheral Interface (Master SPI mode) – interface to boot PROM memory
3. System microprocessor to drive a Serial Slave SPI port (SSPI mode)

The test access port consists of dedicated I/Os: SI, SO, SCK, SS_B. The test access port shares its power supply with VCCIO Bank 5 and can operate with LVCMOS3.3, 2.5, and 1.8 standards.

For more details on boundary scan test, see TN1248, iCE40 Programming and Configuration Usage Guide.

Standby Mode and Power Saving Options

iCE40 devices are available in two options for maximum flexibility: LP and HX devices. The LP devices have ultra low static and dynamic power consumption. These devices can use a 1.0V core voltage that further reduces power consumption. The HX devices are designed to provide high performance. The HX devices operate at 1.2V V_{CC}.

iCE40 devices have been designed with features that allow users to meet the static and dynamic power requirements of their applications by controlling various device subsystems such as iCEGate, PLLs, etc. In order to maximize power savings, iCE40 devices support an ultra low power Stand-by mode. While most of these features are available in both device types, these features are mainly intended for use with iCE40 LP devices to manage power consumption.

In stand-by mode, the iCE40 devices are powered on and configured. Internal logic, I/Os and memories are switched on and remain operational, as the user logic waits for an external input. Various subsystems in the device such as the PLLs and iCEGate can be configured such that they are automatically turned “off” or go into a low power consumption state to save power when the device enters this state.

Table 2-10. iCE40 Power Saving Features Description

Device Subsystem	Feature Description
PLL	When LATCHINPUTVALUE is enabled, forces the PLL into low-power mode; PLL output held static at last input clock value.
iCEGate	To save power, the optional iCEgate latch can selectively freeze the state of individual, non-registered inputs within an I/O bank. Registered inputs are effectively frozen by their associated clock or clock-enable control.

iCE40 LP/HX Family Overview Ordering Information

October 2012 Advance Data Sheet DS1040A

iCE40 Part Number Description

Ultra Low Power (LP) Devices

iCE40LPXXX – XXXXX

Device Family
iCE40 mobileFPGA

Series
LP = Low Power Series

Logic Cells
384 = 384 Logic Cells
1K = 1,280 Logic Cells
4K = 3,520 Logic Cells
8K = 7,680 Logic Cells

Package
CM36 = 36-Ball ucBGA (0.4 mm Pitch)
CM49 = 49-Ball ucBGA (0.4 mm Pitch)
CM81 = 81-Ball ucBGA (0.4 mm Pitch)
CB81 = 81-Ball csBGA (0.5 mm Pitch)
CM121 = 121-Ball ucBGA (0.4 mm Pitch)
CB121 = 121-Ball csBGA (0.5 mm Pitch)
CM225 = 225-Ball csBGA (0.4 mm Pitch)
QN32 = 32-Pin QFN (0.5 mm Pitch)
QN84 = 84-Pin QFN (0.5 mm Pitch)

High Performance (HX) Devices

iCE40HXXX – XXXXX

Device Family
iCE40 mobileFPGA

Series
HX = High-Performance Series

Logic Cells
1K = 1,280 Logic Cells
4K = 3,520 Logic Cells
8K = 7,680 Logic Cells

Package
CB132 = 132-Ball csBGA (0.5 mm Pitch)
CM225 = 225-Ball ucBGA (0.4 mm Pitch)
CT256 = 256-Ball caBGA (0.8 mm Pitch)
TQ144 = 144-Pin TQFP (0.5 mm Pitch)
VQ100 = 100-Pin VQFP (0.5 mm Pitch)

Ordering Information

iCE40 devices have top-side markings for the industrial grade, as shown below:

Industrial

Note: Markings are abbreviated for small packages.

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Ultra Low Power Industrial Grade Devices, Halogen Free (RoHS) Packaging

Part Number	LUTs	Supply Voltage	Package	Leads	Temp.
iCE40LP384-CM49	384	1.0V / 1.2V	Halogen-Free ucBGA	49	IND
iCE40LP384-CM81	384	1.0V / 1.2V	Halogen-Free ucBGA	81	IND
iCE40LP384-QN32	384	1.0V / 1.2V	Halogen-Free QFN	32	IND
iCE40LP1K-CM36	1280	1.0V / 1.2V	Halogen-Free ucBGA	36	IND
iCE40LP1K-CM49	1280	1.0V / 1.2V	Halogen-Free ucBGA	49	IND
iCE40LP1K-CM81	1280	1.0V / 1.2V	Halogen-Free ucBGA	81	IND
iCE40LP1K-CB81	1280	1.0V / 1.2V	Halogen-Free csBGA	81	IND
iCE40LP1K-CM121	1280	1.0V / 1.2V	Halogen-Free ucBGA	121	IND
iCE40LP1K-CB121	1280	1.0V / 1.2V	Halogen-Free csBGA	121	IND
iCE40LP1K-QN84	1280	1.0V / 1.2V	Halogen-Free QFN	84	IND
iCE40LP4K-CM81	3520	1.0V / 1.2V	Halogen-Free ucBGA	81	IND
iCE40LP4K-CM121	3520	1.0V / 1.2V	Halogen-Free ucBGA	121	IND
iCE40LP4K-CM225	3520	1.0V / 1.2V	Halogen-Free ucBGA	225	IND
iCE40LP8K-CM121	7680	1.0V / 1.2V	Halogen-Free ucBGA	121	IND
iCE40LP8K-CM225	7680	1.0V / 1.2V	Halogen-Free ucBGA	225	IND

High-Performance Industrial Grade Devices, Halogen Free (RoHS) Packaging

Part Number	LUTs	Supply Voltage	Package	Leads	Temp.
iCE40HX1K-CB132	1280	1.2V	Halogen-Free csBGA	132	IND
iCE40HX1K-VQ100	1280	1.2V	Halogen-Free QFP	100	IND
iCE40HX1K-TQ144	1280	1.2V	Halogen-Free TQFP	144	IND
iCE40HX4K-CB132	3520	1.2V	Halogen-Free csBGA	132	IND
iCE40HX4K-TQ144	3520	1.2V	Halogen-Free TQFP	144	IND
iCE40HX8K-CB132	7680	1.2V	Halogen-Free csBGA	132	IND
iCE40HX8K-CM225	7680	1.2V	Halogen-Free ucBGA	225	IND
iCE40HX8K-CT256	7680	1.2V	Halogen-Free caBGA	256	IND

iCE40 LP/HX Family Overview Revision History

October 2012 **Advance Data Sheet DS1040A**

Date	Version	Section	Change Summary
October 2012	01.0	—	Initial release

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Section II. iCE40 LP/HX Family Technical Notes

iCE40 Programming and Configuration

September 2012 Technical Note TN1248

Introduction

The iCE40™ devices are SRAM-based FPGAs, with an on-chip, one-time programmable NVCM (Non-volatile Configuration Memory) to store configuration data. The SRAM memory cells are volatile, meaning that once power is removed from the device, its configuration is lost, and must be reloaded on the next power-up. This behavior has the advantage of being re-programmable in the field which provides flexibility for products already deployed to the field. But it also requires that the configuration information be stored in a non-volatile device and loaded each time power is applied to the device. The on-chip NVCM allows the device to configure instantly and greatly enhances the design security by eliminating the need to use an external memory device. The configuration data can also be stored in an external SPI Flash from which the FPGA can configure itself upon power-up. This is useful for prototyping the FPGA or in situations where re-configurability is required. Additionally, the device can be configured by a processor in an embedded environment.

The key programming and configuration features are summarized below:

- Instant on
- Single-chip, secure solution (one-time programmable)
- Multiple configuration image support

Configuration Overview

The iCE40 devices contain two types of memory, SRAM and NVCM (one-time programmable). SRAM memory contains the active configuration. The NVCM and the external SPI Flash provides a non-volatile storage for the configuration data. Additionally, the iCE40 configuration data can be downloaded from an external processor, microcontroller, or DSP processor using the SPI interface. In this document, the term “programming” refers to the programming of the NVCM and the term “configuration” refers to the configuration of SRAM memory. For both programming and configuration, the iCE40 FPGA utilizes the SPI configuration interface.

As described in Table 5-1, iCE40 components are configured for a specific application by loading a binary configuration bitstream image, generated by the Lattice development system. For high-volume applications, the bitstream image is usually permanently programmed in the on-chip Nonvolatile Configuration Memory (NVCM). However, the bitstream image can also be stored externally in a standard, low-cost commodity SPI serial Flash PROM. The iCE40 component can automatically load the image using the SPI Master Configuration Interface. Similarly, the iCE40 configuration data can be downloaded from an external processor, microcontroller, or DSP processor using a SPI-like serial interface.

Table 5-1. iCE40 Device Configuration Modes

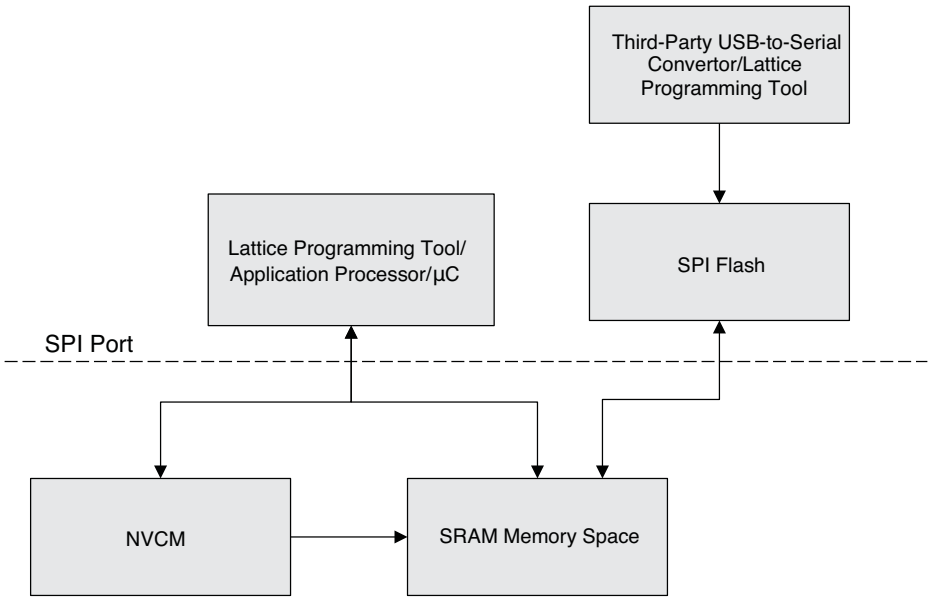
Mode	Analogy	Configuration Data Source
NVCM	ASIC	Internal, lowest-cost, secure, one-time programmable Nonvolatile Configuration Memory (NVCM).
SPI Flash	Microprocessor	External, low-cost, commodity, SPI serial Flash PROM.
SPI Peripheral	Processor Peripheral	Configured by external device, such as a processor, microcontroller, or DSP using practically any data source, such as system Flash, a disk image, or over a network connection.

Figure 5-1 provides an overview of the configuration and programming of the iCE40 FPGA. For configuration and programming, the device can be accessed using the SPI interface/protocol described in later sections of this technical note. The SRAM can configure itself (device in master mode) from the on-chip NVCM, external SPI Flash or

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Lattice programming hardware. An external processor or programming hardware can also configure the SRAM with the FPGA in slave SPI mode. The NVCM can be programmed using the Lattice Diamond® Programmer (version 2.0.1 or later) or an external processor.

Figure 5-1. Configuring and Programming the iCE40 Device

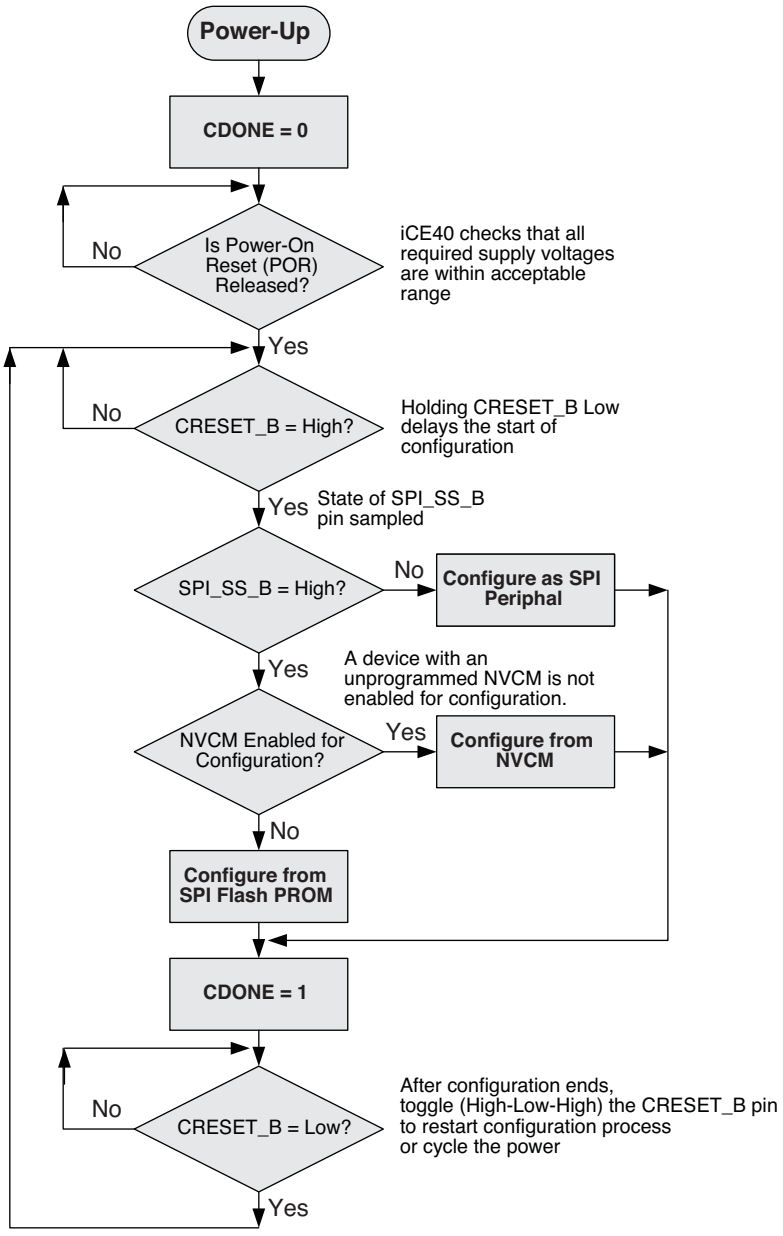


Configuration Mode Selection

The iCE40 configuration mode is selected according to the following priority described below and illustrated in Figure 5-2.

- After exiting the Power-On Reset (POR) state or when CRESET_B returns High after being held Low, the iCE40 FPGA samples the logical value on its SPI_SS_B pin. Like other programmable I/O pins, the SPI_SS_B pin has an internal pull-up resistor. Refer to the iCE40 Family Data Sheet for the minimum pulse width requirement of CRESET_B.
- If the SPI_SS_B pin is sampled as a logic '1' (High), then …
  - Check if the iCE40 is enabled to configure from the Nonvolatile Configuration Memory (NVCM). If the NVCM is programmed, the iCE40 device will configure from NVCM.
    - If enabled to configure from NVCM, the iCE40 device configures itself using the Nonvolatile Configuration Memory (NVCM).
    - If not enabled to configure from NVCM, then the iCE40 FPGA configures using the SPI Master Configuration Interface.
- If the SPI_SS_B pin is sampled as a logic '0' (Low), then the iCE40 device waits to be configured from an external controller or from another iCE40 device in SPI Master Configuration Mode using an SPI-like interface.

Figure 5-2. Device Configuration Control Flow



Nonvolatile Configuration Memory (NVCM)

All standard iCE40 devices have an internal, nonvolatile configuration memory (NVCM). The NVCM is large enough to program a complete iCE40 device, including initializing all Embedded Block RAM. The NVCM memory also has very high programming yield due to extensive error checking and correction (ECC) circuitry.

The NVCM is ideal for cost-sensitive, high-volume production applications, saving the cost and board space associated with an external configuration PROM. Furthermore, the NVCM provides exceptional design security, protecting critical intellectual property (IP). The NVCM contents are entirely contained within the iCE40 device and are not readable once protected by the one-time programmable Security bits. Furthermore, there is no observable difference between a programmed or un-programmed memory cell using optical or electron microscopy.

The NVCM memory has a programming interface similar to a 25-series SPI serial Flash PROM. Consequently, it can be programmed using Diamond Programmer (version 2.0.1 or later) before or after circuit board assembly or

programmed in-system from a microprocessor or other intelligent controller. The NVCM can also be pre-programmed at the factory. Contact Lattice Technical Support or your local Lattice sales office for assistance.

Configuration Control Signals

The iCE40 configuration process is self-timed and controlled by a few internal signals and device I/O pins, as described in Table 5-2.

Table 5-2. iCE40 Configuration Control Signals

Signal Name	Direction	Description
POR	Internal Control	Internal Power-On Reset (POR) circuit.
OSC	Internal Control	Internal configuration oscillator.
CRESET_B	Input	Configuration Reset input. Active-Low. No internal pull-up resistor.
CDONE	Open-drain Output	Configuration Done output. Permanent, weak pull-up resistor to VCCIO_2.

The Power-On Reset circuit, POR, automatically resets the iCE40 component to a known state during power-up (cold boot). The POR circuit monitors the relevant voltage supply inputs, as shown in Figure 5-4. Once all supplies exceed their minimum thresholds, the configuration controller can start the configuration process.

The configuration controller begins configuring the iCE40 device, clocked by the Internal Oscillator, OSC. The OSC oscillator continues controlling configuration unless the iCE40 device is configured using the SPI Peripheral Configuration Interface.

Figure 5-3. iCE40 Configuration Control Pins

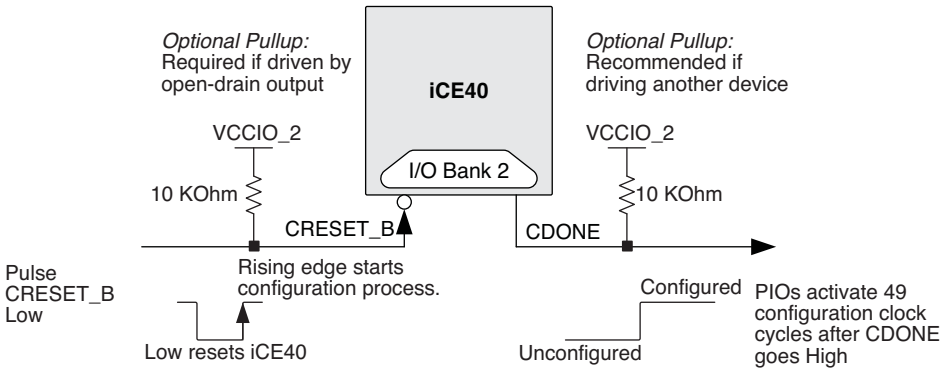


Figure 5-3 shows the two iCE40 configuration control pins, CRESET_B and CDONE. When driven Low, the dedicated Configuration Reset input, CRESET_B, resets the iCE40 device. When CRESET_B returns High, the iCE40 FPGA restarts the configuration process from its power-on conditions (Cold Boot). The CRESET_B pin is a pure input with no internal pull-up resistor. If driven by open-drain driver or un-driven, then connect the CRESET_B pin to a 10 KOhm pull-up resistor connected to the VCCIO_2 supply.

The iCE40 device signals the end of the configuration process by actively turning off the internal pull-down transistor on the Configuration Done output pin, CDONE. The pin has a permanent, weak internal pull-up resistor to the VCCIO_2 rail. If the iCE40 device drives other devices, then optionally connect the CDONE pin to a 10 KOhm pull-up resistor connected to the VCCIO_2 supply.

The PIO pins activate according to their configured function after 49 configuration clock cycles. The internal oscillator is the configuration clock source for the SPI Master Configuration Interface and when configuring from Nonvolatile Configuration Memory (NVCM). When using the SPI Peripheral Configuration Interface, the configuration clock source is the SPI_SCK clock input pin.

Internal Oscillator

During SPI Master or NVCM configuration mode, the controlling clock signal is generated from an internal oscillator. The oscillator starts operating at the default frequency. During the configuration process, however, bit settings within the configuration bitstream can specify a higher-frequency mode in order to maximize SPI bandwidth and reduce overall configuration time. See data sheet for the specified oscillator frequency range.

Using the SPI Master Configuration Interface, internal oscillator controls all the interface timing and clocks the SPI serial Flash PROM via the SPI_SCK clock output pin.

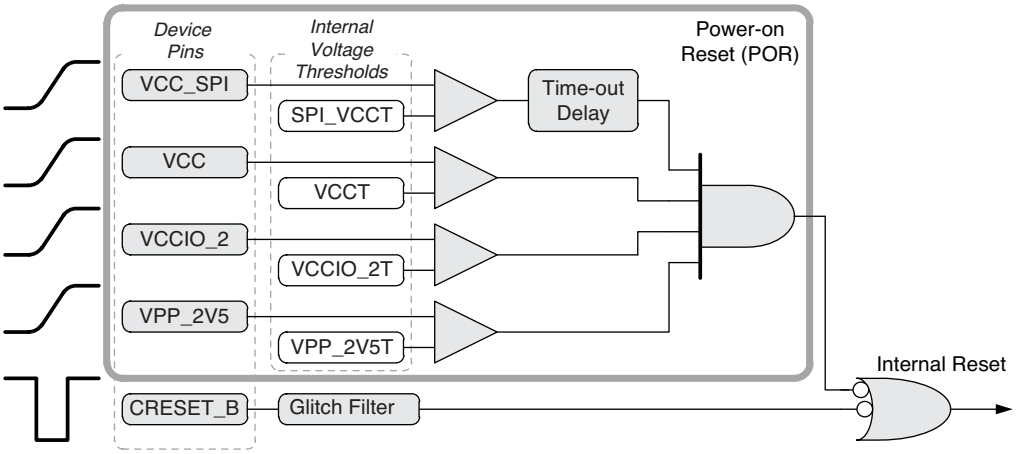
The oscillator output, which also supplies the SPI SCK clock output during the SPI Flash configuration process, has a 50% duty cycle.

Internal Device Reset

Figure 5-4 presents the various signals that internally reset the iCE40 internal logic.

- Power-On Reset (POR)
- CRESET_B Pin

Figure 5-4. iCE40 Internal Reset Circuitry



Power-On Reset (POR)

The Power-on Reset (POR) circuit monitors specific voltage supply inputs and holds the device in reset until all the relevant supplies exceed the internal voltage thresholds. The VCC_SPI supply also has an additional time-out delay to allow an attached SPI serial PROM to power up properly. Table 5-3 shows the POR supply inputs. The Nonvolatile Configuration Memory (NVCM) requires that the VPP_2V5 supply be connected, even if the application does not use the NVCM.

Table 5-3. Power-on Reset (POR) Voltage Resources

Supply Rail	iCE40 Production Devices
VCC	Yes
VCC_SPI	Yes
VCCIO_2	Yes
VPP_2V5	Yes

CRESET_B Pin

The CRESET_B pin resets the iCE40 internal logic when Low.

sysCONFIG Port

The sysCONFIG port is used to program and configure the iCE40 FPGA. The device has a SPI configuration interface as the sysCONFIG port which can be used to configure the device.

Table 5-4. sysCONFIG Ports

Interface	Port	Description
sysCONFIG	SPI Master Configuration interface	In this mode, the FPGA configures itself from an external SPI Flash or Diamond Programmer (version 2.0.1 or later). The FPGA behaves as master, generates internal clock and drives the clock to the external SPI Flash.
	SPI Peripheral Configuration interface	In this mode, the FPGA behaves as a Slave device. An external Application Processor, µC or Diamond Programmer (version 2.0.1 or later) configures or programs the device.

sysCONFIG Pins

The iCE40 FPGA has a set of sysCONFIG pins that are used to program and configure the device. The sysCONFIG pins are grouped together to create the sysCONFIG port, as discussed above. The sysCONFIG pins are dual-function, meaning they can be recovered as user I/O after configuration is complete. Table 5-5 shows the sysCONFIG pins.

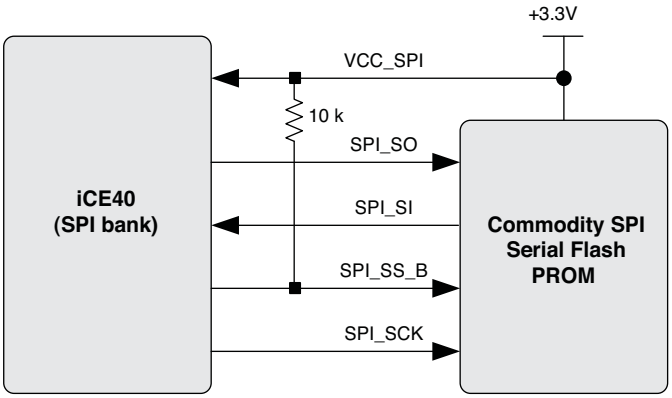
Table 5-5. sysCONFIG Pins

Pin Name	Associated sysCONFIG Port	Pin Direction	Description
CRESET_B	—	Input	Configuration Reset input, active-low. No internal pull-up resistor.
CDONE	—	Output	Configuration Done output. Permanent, weak pull-up resistor to VCCIO_2.
SPI_SS_B	SPI Master/Peripheral configuration interface	Input/Output	An important dual-function, active-low slave select pin. After the device exits POR or CRESET_B is toggled (High-Low-High), it samples the SPI_SS_B to select the configuration mode (an output in Master mode and an input in Slave mode).
SPI_SI	SPI Master/Peripheral configuration interface	Input	A dual-function, serial input pin in both configuration modes.
SPI_SO	SPI Master/Peripheral configuration interface	Output	A dual-function, serial output pin in both configuration modes.
SPI_SCK	SPI Master/Peripheral configuration interface	Input/Output	A dual-function clock signal. An output in Master mode and input in Slave mode.

SPI Master Configuration Interface

All iCE40 devices can be configured from an external, commodity SPI serial Flash PROM, as shown in Figure 5-5. The SPI configuration interface is essentially its own independent I/O bank, powered by the VCC_SPI supply input. Presently, most commercially-available SPI serial Flash PROMs require a 3.3V supply.

Figure 5-5. iCE40 SPI Master Configuration Interface



The SPI configuration interface is used primarily during development before mass production, where the configuration is then permanently programmed in the NVCM configuration memory. However, the SPI interface can also be the primary configuration interface allowing easy in-system upgrades and support for multiple configuration images.

The SPI control signals are defined in Table 5-6.

Table 5-6. SPI Master Configuration Interface Pins (SPI_SS_B High Before Configuration)

Signal Name	Direction	Description
VCC_SPI	Supply	SPI Flash PROM voltage supply input.
SPI_SO	Output	SPI Serial Output from the iCE40 device.
SPI_SI	Input	SPI Serial Input to the iCE40 device, driven by the select SPI serial Flash PROM.
SPI_SS_B	Output	SPI Slave Select output from the iCE40 device. Active Low.
SPI_SCK	Output	SPI Slave Clock output from the iCE40 device.

After configuration, the SPI port pins are available to the user-application as additional PIO pins, supplied by the VCC_SPI input voltage.

SPI PROM Requirements

The iCE40 SPI Flash configuration interface supports a variety of SPI Flash memory vendors and product families. However, Lattice does not specifically test, qualify, or otherwise endorse any specific SPI Flash vendor or product family. The iCE40 SPI interface supports SPI PROMs that they meet the following requirements.

- The PROM must operate at 3.3V or 2.5V in order to trigger the iCE40 FPGA's power-on reset circuit.
- The PROM must support the 0x0B Fast Read command, using a 24-bit start address and has 8 dummy bits before the PROM provides first data (see Figure 5-7).
- The PROM must have enough bits to program the iCE40 device.
- The PROM must support data operations at the upper frequency range for the selected iCE40 internal oscillator frequency (see data sheet). The oscillator frequency is selectable when creating the FPGA bitstream image.
- For lowest possible power consumption after configuration, the PROM should also support the 0xB9 Deep Power Down command and the 0xAB Release from Deep Power-down Command (see Figure 5-6 and Figure 5-8). The low-power mode is optional.

- The PROM must be ready to accept commands 10 µs after meeting its power-on conditions. In the PROM data sheet, this may be specified as t_{VSL} or t_{VCSL}. It is possible to use slower PROMs by holding the CRESET_B input Low until the PROM is ready, then releasing CRESET_B, either under program control or using an external power-on reset circuit.

The iCEblink40™ development board and associated programming software uses an ST Micro/Numonyx M25Pxx SPI serial Flash PROM. Table 5-7 gives the bitstream sizes for different densities of the iCE40 FPGA that can be used to select a SPI Flash.

Table 5-7. Bitstream Sizes for Different iCE40 FPGA Densities Used to Select a SPI Flash

Device	Bytes	Bits
iCE40-LP384	7872	62,976
iCE40-LP/HX1K	34,112	272,896
iCE40-LP/HX4K	68,224	545,792
iCE40-LP/HX8K	136,448	1,091,584

Enabling SPI Configuration Interface

To enable the SPI configuration mode, the SPI_SS_B pin must be allowed to float High. The SPI_SS_B pin has an internal pull-up resistor. If SPI_SS_B is Low, then the iCE40 component defaults to the SPI Slave configuration mode.

SPI Master Configuration Process

The iCE40 SPI Master Configuration Interface supports a variety of modern, high-density, low-cost SPI serial Flash PROMs. Most modern SPI PROMs include a power-saving Deep Power-down mode. The iCE40 component exploits this mode for additional system power savings.

The iCE40 SPI interface starts by driving SPI_SS_B Low, and then sends a Release from Power-down command to the SPI PROM, hexadecimal command code 0xAB. Figure 5-6 provides an example waveform. This initial command wakes up the SPI PROM if it is already in Deep Power-down mode. If the PROM is not in Deep Power-down mode, the extra command has no adverse affect other than that it requires a few additional microseconds during the configuration process. The iCE40 device transmits data on the SPI_SO output, on the falling edge of the SPI_SCK output. The SPI PROM does not provide any data to the iCE40 device's SPI_SI input. After sending the last command bit, the iCE40 device de-asserts SPI_SS_B High, completing the command. The iCE40 device then waits a minimum of 10 µS before sending the next SPI PROM command.

Figure 5-6. SPI Release from Deep Power-down Command

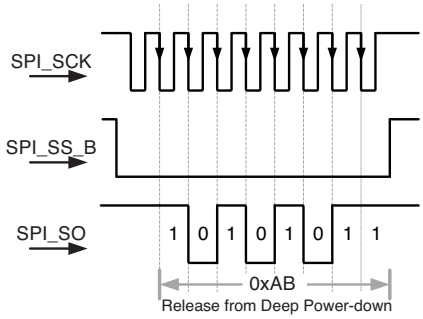
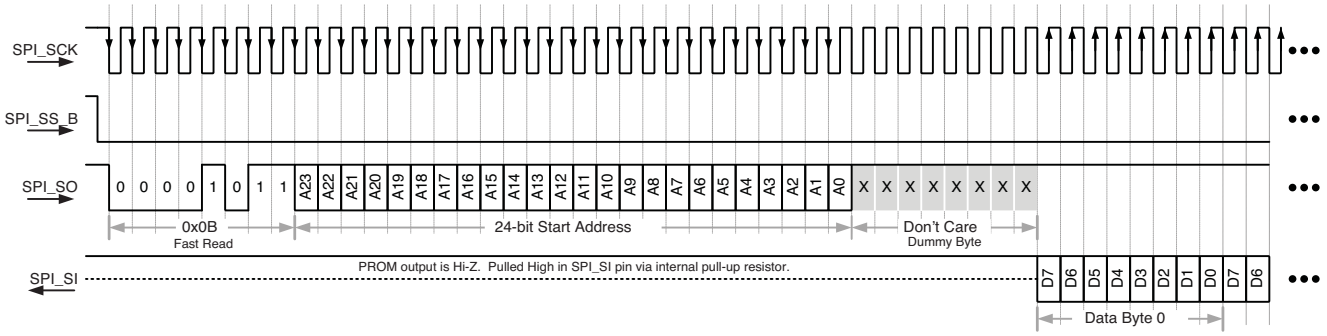


Figure 5-7 illustrates the next command issued by the iCE40 device. The iCE40 SPI interface again drives SPI_SS_B Low, followed by a Fast Read command, hexadecimal command code 0x0B, followed by a 24-bit start address, transmitted on the SPI_SO output. The iCE40 device provides data on the falling edge of SPI_SS_B. Upon initial power-up, the start address is always 0x00_0000. After waiting eight additional clock cycles, the iCE40

device begins reading serial data from the SPI PROM. Before presenting data, the SPI PROM's serial data output is high-impedance. The SPI_SI input pin has an internal pull-up resistor and sees high-impedance as logic '1'.

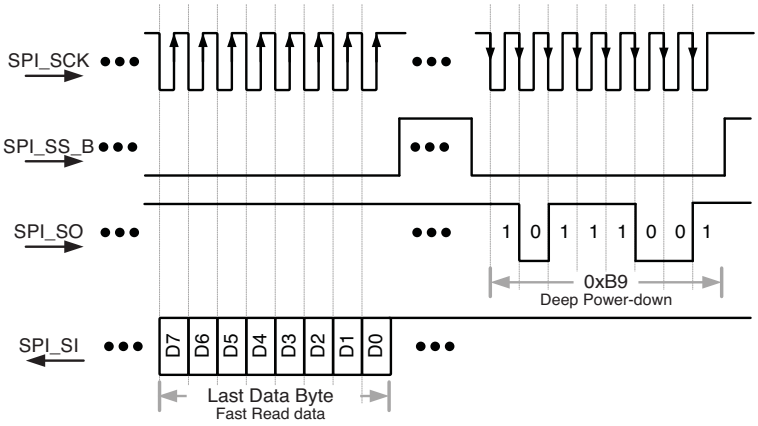
Figure 5-7. SPI Fast Read Command



The external SPI PROM supplies data on the falling edge of the iCE40 device's SPI_SCK clock output. The iCE40 device captures each PROM data value on the SPI_SI input, using the rising edge of the SPI_SCK clock signal. The SPI PROM data starts at the 24-bit address presented by the iCE40 device. PROM data is serially output, byte by byte, with most-significant bit, D7, presented first. The PROM automatically increments an internal byte counter as long as the PROM is selected and clocked.

After transferring the required number configuration data bits, the iCE40 device ends the Fast Read command by de-asserting its SPI_SS_B PROM select output, as shown in Figure 5-8. To conserve power, the iCE40 device then optionally issues a final Deep Power-down command, hexadecimal command code 0xB9. After de-asserting the SPI_SS_B output, the SPI PROM enters its Deep Power-down mode. The final power-down step is optional; the application may wish to use the SPI PROM and can skip this step, controlled by a configuration option.

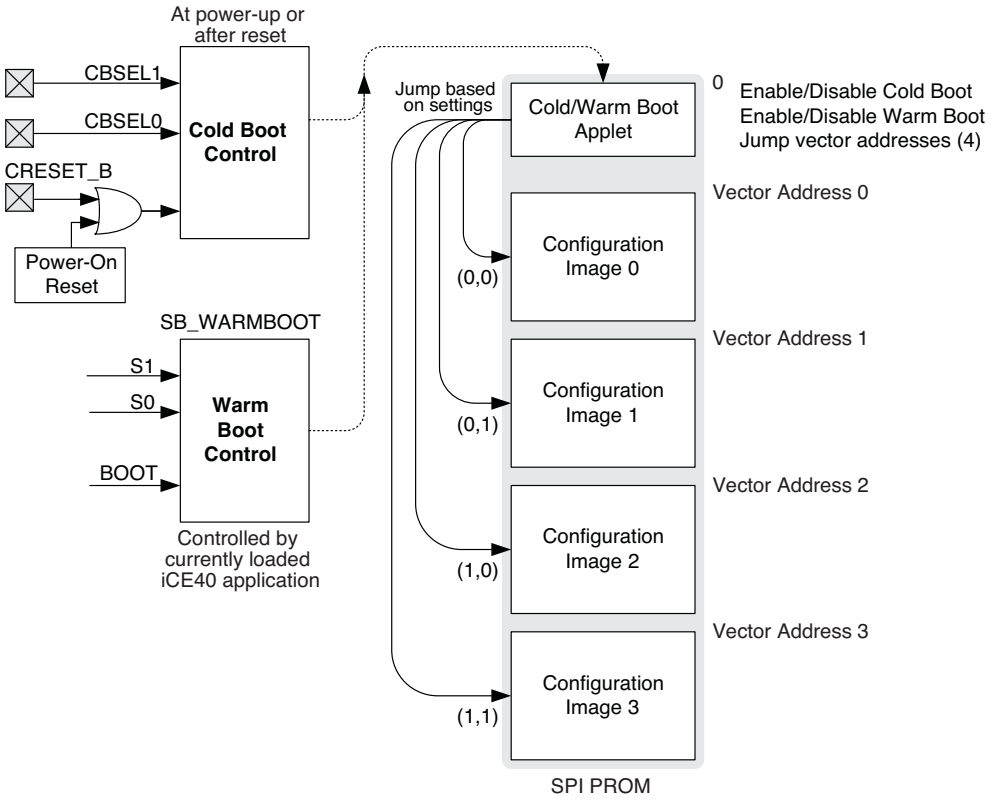
Figure 5-8. Final Configuration Data, SPI Deep Power-down Command



Cold Boot Configuration Option

By default, the iCE40 FPGA is programmed with a single configuration image, either from internal NVCM memory, from an external SPI Flash PROM, or externally from a processor or microcontroller.

Figure 5-9. Cold Boot and Warm Boot Configuration



When self-loading from NVCM or from an SPI Flash PROM, the FPGA supports an additional configuration option called Cold Boot mode. When this option is enabled in the configuration bitstream, the iCE40 FPGA boots normally from power-on or a master reset (CRESET_B = Low pulse), but monitors the value on two PIO pins that are borrowed during configuration, as shown in Figure 5-9. These pins, labeled PIO2/CBSEL0 and PIO2/CBSEL1, tell the FPGA which of the four possible SPI configurations to load into the device.

- Load from initial location, either from NVCM or from address 0 in SPI Flash PROM. For Cold Boot or Warm Boot applications, the initial configuration image contains the cold boot/warm boot applet.
- Check if Cold Boot configuration feature is enabled in the bitstream.
  - If not enabled, FPGA configures normally.
  - If Cold Boot is enabled, then the FPGA reads the logic values on pins CBSEL[1:0]. The FPGA uses the value as a vector and then reads from the indicated vector address.
  - At the selected CBSEL[1:0] vector address, there is a starting address for the selected configuration image.
    - For SPI Flash PROMs, the new address is a 24-bit start address in Flash.
    - If the selected bitstream is in NVCM, then the address points to the internal NVCM.
- Using the new start address, the FPGA restarts reading configuration memory from the new location.

When creating the initial configuration image, the Lattice development software loads the start address for up to four configuration images in the bitstream. The value on the CBSEL[1:0] pins tell the configuration controller to read a specific start address, then to load the configuration image stored at the selected address. The multiple bitstreams are stored either in the SPI Flash or in the internal NVCM.

After configuration, the CBSEL[1:0] pins become normal PIO pins available to the application.

The Cold Boot feature allows the iCE40 to be reprogrammed for special application requirements such as the following.

- A normal operating mode and a self-test or diagnostics mode.
- Different applications based on switch settings.
- Different applications based on a card-slot ID number.

Warm Boot Configuration Option

The Warm Boot configuration is similar to the Cold Boot feature, but is completely under the control of the FPGA application.

A special design primitive, SB_WARMBOOT, allows an FPGA application to choose between four configuration images using two internal signal ports, S1 and S0, as shown in Figure 5-9. These internal signal ports connect to programmable interconnect, which in turn can connect to PLB logic and/or PIO pins.

After selecting the desired configuration image, the application then asserts the internal signal BOOT port High to force the FPGA to restart the configuration process from the specified vector address stored in PROM.

Time-Out and Retry

When configuring from external SPI Flash, the iCE40 device looks for a synchronization word. If the device does not find a synchronization word within its timeout period, the device automatically attempts to restart the configuration process from the very beginning. This feature is designed to address any potential power-sequencing issues that may occur between the iCE40 device and the external PROM.

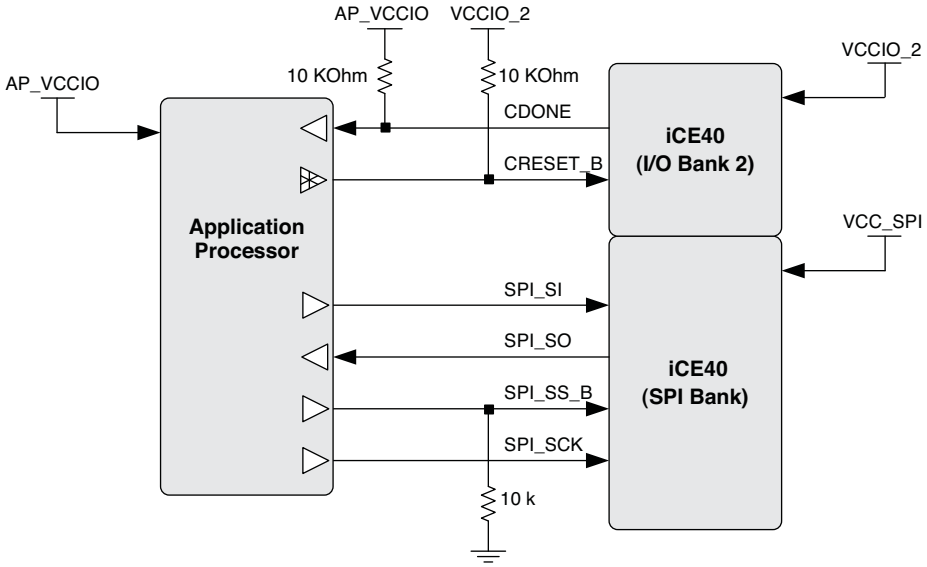
The iCE40 device attempts to reconfigure six times. If not successful after six attempts, the iCE40 FPGA automatically goes into low-power mode.

SPI Peripheral Configuration Interface

Using the SPI peripheral configuration interface, an application processor (AP) serially writes a configuration image to an iCE40 FPGA using the iCE40's SPI interface, as shown in Figure 5-5. The iCE40's SPI configuration interface is a separate, independent I/O bank, powered by the VCC_SPI supply input. Typically, VCC_SPI is the same voltage as the application processor's I/O. The configuration control signals, CDONE and CRESET_B, are supplied by the separate I/O Bank 2 voltage input, VCCIO_2.

This same SPI peripheral interface supports programming for the iCE40's Nonvolatile Configuration Memory (NVCM).

Figure 5-10. iCE40 SPI Peripheral Configuration Interface



The SPI control signals are defined in Table 5.

Table 5-8. SPI Peripheral Configuration Interface Pins (SPI_SS_B Low when CRESET_B Released)

Signal Name	Direction	iCE40 I/O Supply	Description
CDONE	Output	VCCIO_2	Configuration Done output from iCE40. Connect to a 10 KOhm pull-up resistor to the application processor I/O voltage, AP_VCC.
CRESET_B	Input		Configuration Reset input on iCE40. Typically driven by AP using an open-drain driver, which also requires a 10 KOhm pull-up resistor to VCCIO_2.
VCC_SPI	Supply	VCC_SPI	SPI Flash PROM voltage supply input.
SPI_SI	Input		SPI Serial Input to the iCE40 FPGA, driven by the application processor.
SPI_SO	Output		SPI Serial Output from CE65 device to the application processor. Not actually used during SPI peripheral mode configuration but required if the SPI interface is also used to program the NVCM.
SPI_SS_B	Input		SPI Slave Select output from the application processor. Active Low. Optionally hold Low prior to configuration using a 10 KOhm pull-down resistor to ground.
SPI_SCK	Input		SPI Slave Clock output from the application processor.

After configuration, the SPI port pins are available to the user-application as additional PIO pins, supplied by the VCC_SPI input voltage.

Enabling SPI Configuration Interface

The optional 10 KOhm pull-down resistor on the SPI_SS_B signal ensures that the iCE40 FPGA powers up in the SPI peripheral mode. Optionally, the application processor drives the SPI_SS_B pin Low when CRESET_B is released, forcing the iCE40 FPGA into SPI peripheral mode.

SPI Peripheral Configuration Process

Figure 5-11 illustrates the interface timing for the SPI peripheral mode and Figure 5-12 outlines the resulting configuration process. The actual timing specifications appear in the data sheet. The application processor (AP) begins by driving the iCE40 CRESET_B pin Low, resetting the iCE40 FPGA. Similarly, the AP holds the iCE40's SPI_SS_B pin Low. The AP must hold the CRESET_B pin Low for at least 200 ns. Ultimately, the AP either

releases the CRESET_B pin and allows it to float High via the 10 KOhm pull-up resistor to VCCIO_2 or drives CRESET_B High. The iCE40 FPGA enters SPI peripheral mode when the CRESET_B pin returns High while the SPI_SS_B pin is Low,

After driving CRESET_B High or allowing it to float High, the AP must wait a minimum of 300 µs, allowing the iCE40 FPGA to clear its internal configuration memory.

After waiting for the configuration memory to clear, the AP sends the configuration image generated by the Diamond Programmer (version 2.0.1 or later). An SPI peripheral mode configuration image must not use the Cold Boot or Warm Boot options. Send the entire configuration image, without interruption, serially to the iCE40's SPI_SI input on the falling edge of the SPI_SCK clock input. Once the AP sends the 0x7EAA997E synchronization pattern, the generated SPI_SCK clock frequency must be within the specified 1 MHz to 25 MHz range (40 ns to 1 µs clock period) while sending the configuration image. Send each byte of the configuration image with most-significant bit (msb) first. The AP sends data to the iCE40 FPGA on the falling edge of the SPI_SCK clock. The iCE40 FPGA internally captures each incoming SPI_SI data bit on the rising edge of the SPI_SCK clock. The iCE40's SPI_SO output pin is not used during SPI peripheral mode but must connect to the AP if the AP also programs the iCE40's Nonvolatile Configuration Memory (NVCM).

*Note: The iCE40 configuration image must be sent as one contiguous stream **without interruption**. The SPI_SCK clock period must be between 40 ns to 1 µs (1 MHz to 25 MHz).*

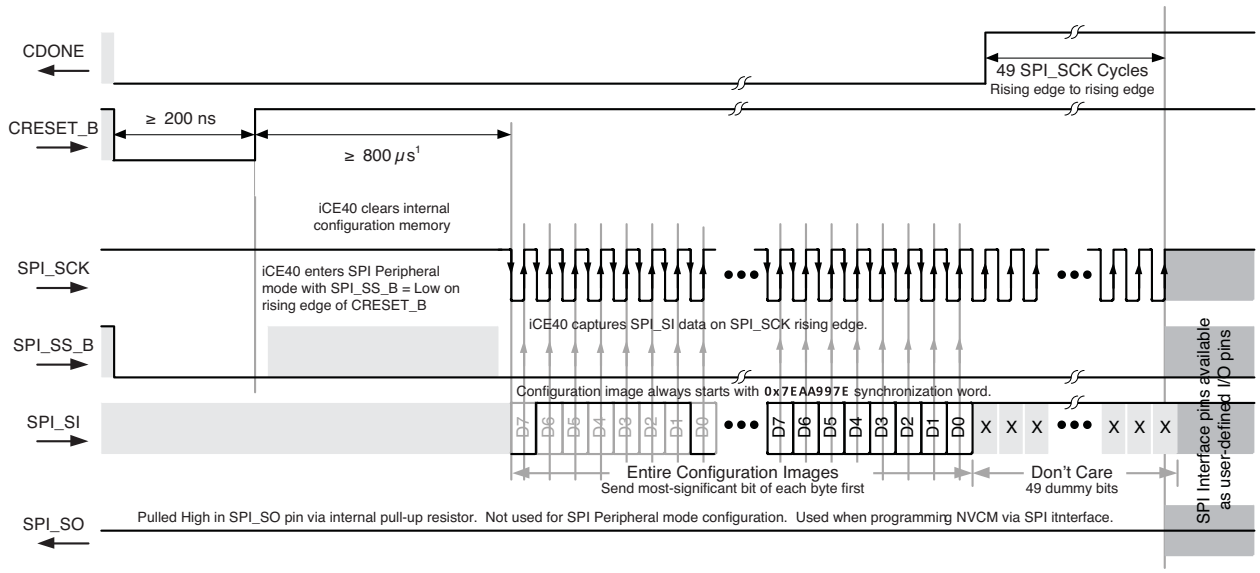
After sending the entire image, the iCE40 FPGA releases the CDONE output allowing it to float High via the 10 KOhm pull-up resistor to AP_VCC. If the CDONE pin remains Low, then an error occurred during configuration and the AP should handle the error accordingly for the application.

After the CDONE output pin goes High, send at least 49 additional dummy bits, effectively 49 additional SPI_SCK clock cycles measured from rising-edge to rising-edge.

After the additional SPI_CLK cycles, the SPI interface pins then become available to the user application loaded in FPGA.

To reconfigure the iCE40 FPGA or to load a different configuration image, merely restart the configuration process by pulsing CRESET_B Low or power-cycling the FPGA.

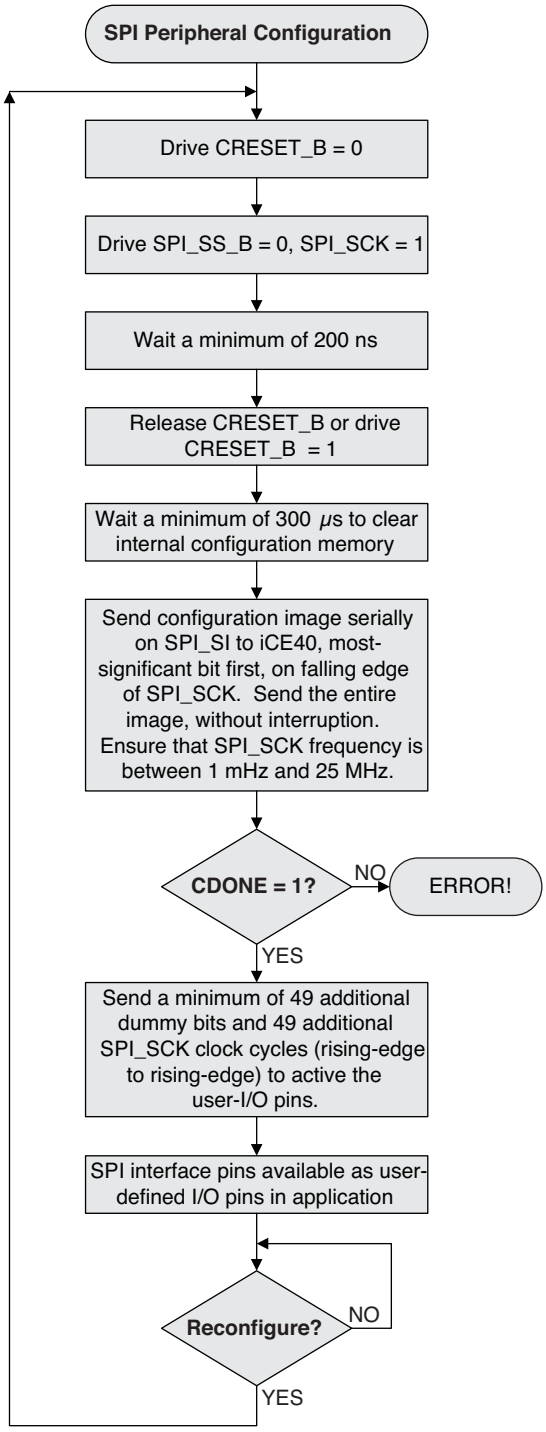
Figure 5-11. Application Processor Waveforms for SPI Peripheral Mode Configuration Process



1. Refer to Appendix A for timing based on density.

*Note: The iCE40 configuration image must be sent as one contiguous stream **without interruption**. The SPI_SCK clock period must be between 40 ns to 1 µs (1 MHz to 25 MHz).*

Figure 5-12. SPI Peripheral Configuration Process



Refer to Appendix A for the SPI peripheral configuration procedure.

Voltage Compatibility

As shown in Figure 5-5, there are potentially three different supply voltages involved in the SPI Peripheral interface, described in Table 5-9.

Table 5-9. SPI Peripheral Mode Supply Voltages

Supply Voltage	Description
AP_VCCIO	I/O supply to the Application Processor (AP)
VCC_SPI	Voltage supply for the iCE40 SPI interface
VCCIO_2	Supply voltage for the iCE40 I/O Bank 2

Table 5-10 describes how to maintain voltage compatibility for two interface scenarios. The easiest interface is when the Application Processor's (AP) I/O supply rail and the iCE40's SPI and VCCIO_2 bank supply rails all connect to the same voltage. The second scenario is when the AP's I/O supply voltage is greater than the iCE40's VCCIO_2 supply voltage.

Table 5-10. CRESET_B and CDONE Voltage Compatibility

Condition	CRESET_B			CDONE Pull-up	Requirement
	Direct	Open-Drain	Pull-up		
VCCIO_AP = VCC_SPI VCCIO_AP = VCCIO_2	OK	OK with pull-up	Required if using open-drain output	Recommended	AP can directly drive CRESET_B High and Low although an open-drain output recommended is if multiple devices control CRESET_B. If using an open-drain driver, the CRESET_B input must include a 10 KOhm pull-up resistor to VCCIO_2. The 10 KOhm pull-up resistor to AP_VCCIO is also recommended.
AP_VCCIO > VCCIO_2	N/A	Required, requires pull-up	Required	Required	The AP must control CRESET_B with an open-drain output, which requires a 10 KOhm pull-up resistor to VCCIO_2. The 10 KOhm pull-up resistor to AP_VCCIO is required.

NVCM Programming

The NVCM can be programmed in the following ways:

- Diamond Programmer
  - Programming using the Diamond Programmer (Diamond 2.0.1 or later) is recommended for prototyping. Programming is supported using the Lattice programming cable. For more information refer to the Diamond Programmer Online Help and UG48, Lattice ispDOWNLOAD Cable User's Guide.
- Factory Programming
  - The Lattice factory offers NVCM programming. For more information contact your local Lattice sales office.
- Embedded Programming
  - The NVCM can be programmed using a processor. For more information contact your local Lattice sales office.

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
March 2012	01.3	Initial release.
June 2012	01.4	Updated document with new corporate style.
September 2012	01.5	Updated based on latest iCE40 information: - Included configuration and programming information - Updated bit file sizes - Updated Table 3. Power-on Reset (POR) Voltage Resources - Removed JTAG references - Include configuration algorithm in Appendix A

Appendix A. SPI Peripheral Configuration Procedure

CPU Configuration Procedure

The sequence for configuring the iCE40 SRAM follows.

Table 5-11. iCE40 SRAM Configuration Sequence

Index	Step	Action	Description
1	Power up the iCE40 FPGA or toggle its CRESET signal low for 200ns and toggle back to high with SPI_SS_B = 0, forcing the device to enter Slave mode. Keep SPI_SS_B low until step 4.	Hold SPI_SS_B =0 and toggle CRESET	See Figure 5-13 below.
2	Wait > = 800us to 1200us until the iCE40 FPGA completes internal housekeeping work and is ready to receive CPU bitmap data and instructions.	Wait >= 1200us for L8K	
3	From this point on, the iCE40 FPGA requires CPU provides operation clock through the SPI_SCK pin of the iCE40 FPGA until configuration is complete.	Feed clock to the SPI_CLK pin	
4	Read and start sending the FPGA bitmap to the iCE40 device. Data goes to the SPI_SDI pin.	Tx bitmap data from user memory	Each clock shifts one bit of data at the clock falling edge. Hold previous SPI_SS_B state, no toggle. Complete sending all bits in bitmap file. Important: Continuous clock is required.
5	Wait for 100 clocks.	Shifting 100 clocks	Configuration complete after 100 clocks.
6	Monitor the CDONE pin. It should go to high. Otherwise, the configuration fails and stops.	Check CDONE Hi	Device in operation when CDONE = 1; device fails when CDONE = 0.

Configuration Waveforms

iCE40 Reset Waveforms

The reset timing waveforms for initiating NVCM programming are shown below.

Figure 5-13. iCE40 Reset Waveform 1

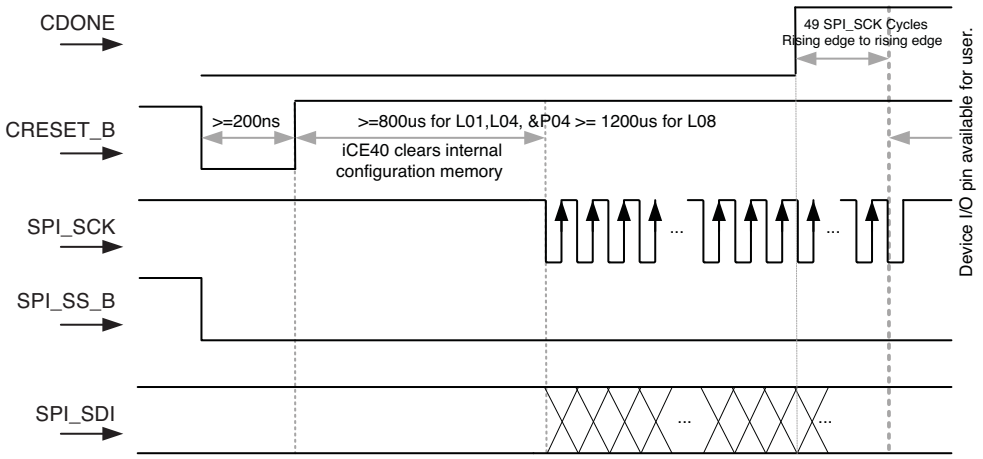


Figure 5-14. iCE40 Timing Waveform 2

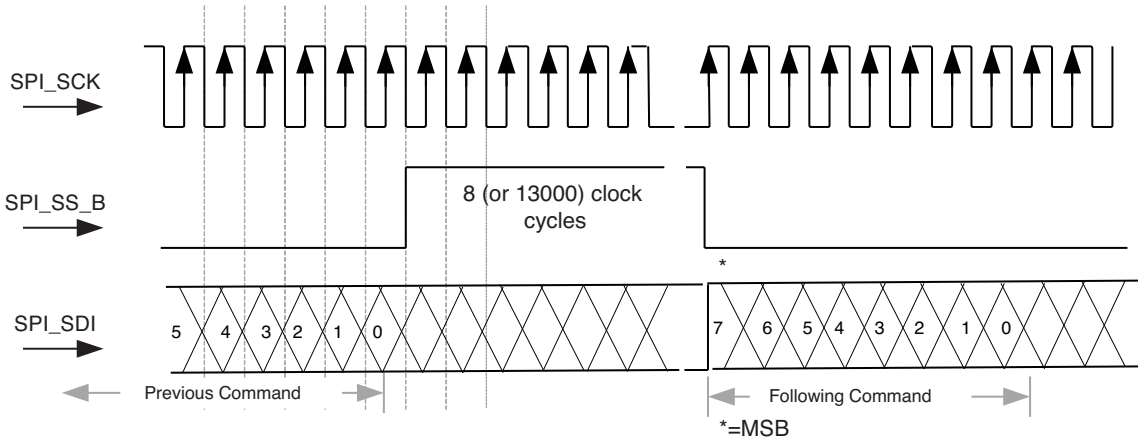


Figure 5-15. iCE40 Timing Waveform 3

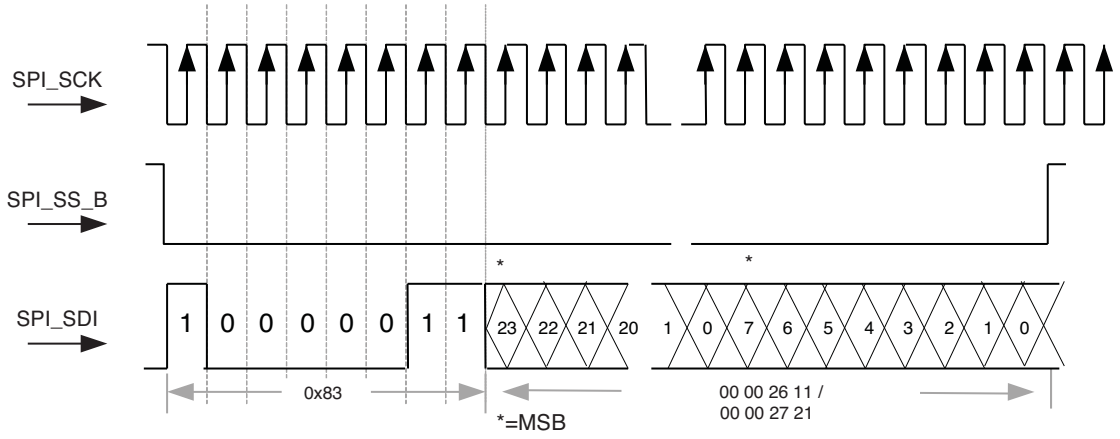
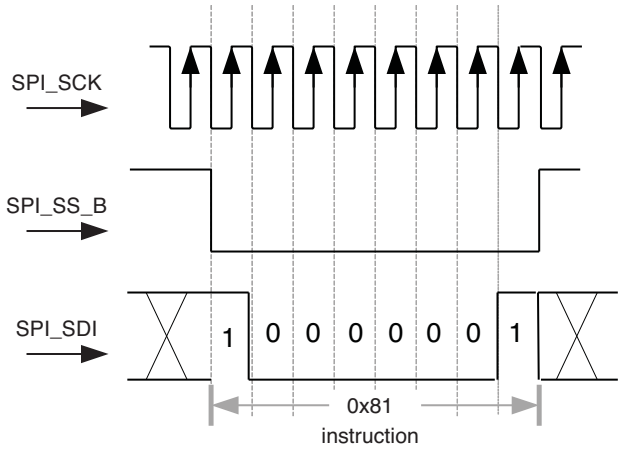


Figure 5-16. iCE40 Timing Waveform 4



Pseudo Code

Configuration

The pseudo code included below will configure an iCE40 device. It assumes a raw binary file generated from iCEcube will be used (*.bin). Alternatively, a hex file is also generated from iCEcube and can be used as well. The implementation should only make the necessary text-to-binary conversions.

```
//
//  iCE40 Configuration Pseudo-code
//
void Config_iCE40 (type, file)
{
   //
   // Open Hex File early to avoid clock delay later
   //
   file_pointer = fopen(file);// Open bin file

   //
   // Reset the iCE40 Device
   //
   Set_Port(SPI_SS_B, false);// Set SPI_SS_B low
   Set_Port(CRESET, false);// Set CRESET low
   Set_Port(SPI_CLK, true);// Set SPI_CLK high
   nSec_Delay(200);// Delay 200 nsec
   Set_Port(CRESET, true);// Set CRESET high
   if (type == L1K or L1K)
     uSec_Delay(800);// Delay 800 usec if L1K,L4K
   else if (type == L1K)
     uSec_Delay(1200);// Delay 1200 usec for L8K
   Set_Port(SPI_SS_B, true);// Set SPI_SS_B high
   Send_Clocks (8);// Send 8 clocks
   //
   // Send data from bin file
   //
   Send_File(file_pointer);// Send bin file
   Send_Clocks (100);// Send 100 clocks
   //
   // Verify successful configuration
   //
   if (Get_Port(CDONE))
     Return PASS;// PASS if CDONE is true
   else
     Return FAIL;// FAIL if CDONE is false
}

//
//  Clock Generation 10MHz
//
void Send_Clocks(num_clocks)
{
   for {i = 0; i < num_clocks; i++}
   {
     Set_Port(SPI_CLOCK, false);// Set SPI_CLK low
     nSec_Delay(50);// Delay 50 nsec
```

```
      Set_Port(SPI_CLOCK, true);// Set SPI_CLK high
      nSec_Delay(50);// Delay 50 nsec
   }
}

//
//  Send Data from file
//
void Send_File(file_pointer)
{
   byte = getc(file_pointer);// Read first byte from file
   while (byte != EOF)
   {
      Send_Byte (byte);// Send data byte
      byte = getc(file_pointer);// read next byte from file
   }
}
```

Memory Usage Guide for iCE40 Devices

September 2012 **Technical Note TN1250**

Introduction

This technical note discusses memory usage for the iCE40™ device family. It is intended to be used as a guide to the high-speed synchronous RAM Blocks and the iCE40 sysMEM™ Embedded Block RAM (EBR). The EBR is the embedded block RAM of the device, each 4Kbit in size.

The iCE40 device architecture provides resources for memory-intensive applications. Single-Port RAM, Dual-Port RAM and FIFO can be constructed using the EBRs. The EBRs can be utilized by instantiating software primitives as described later in this document. Apart from primitive instantiation, the iCECube2™ design software infers generic codes as EBRs.

Memories in iCE40 Devices

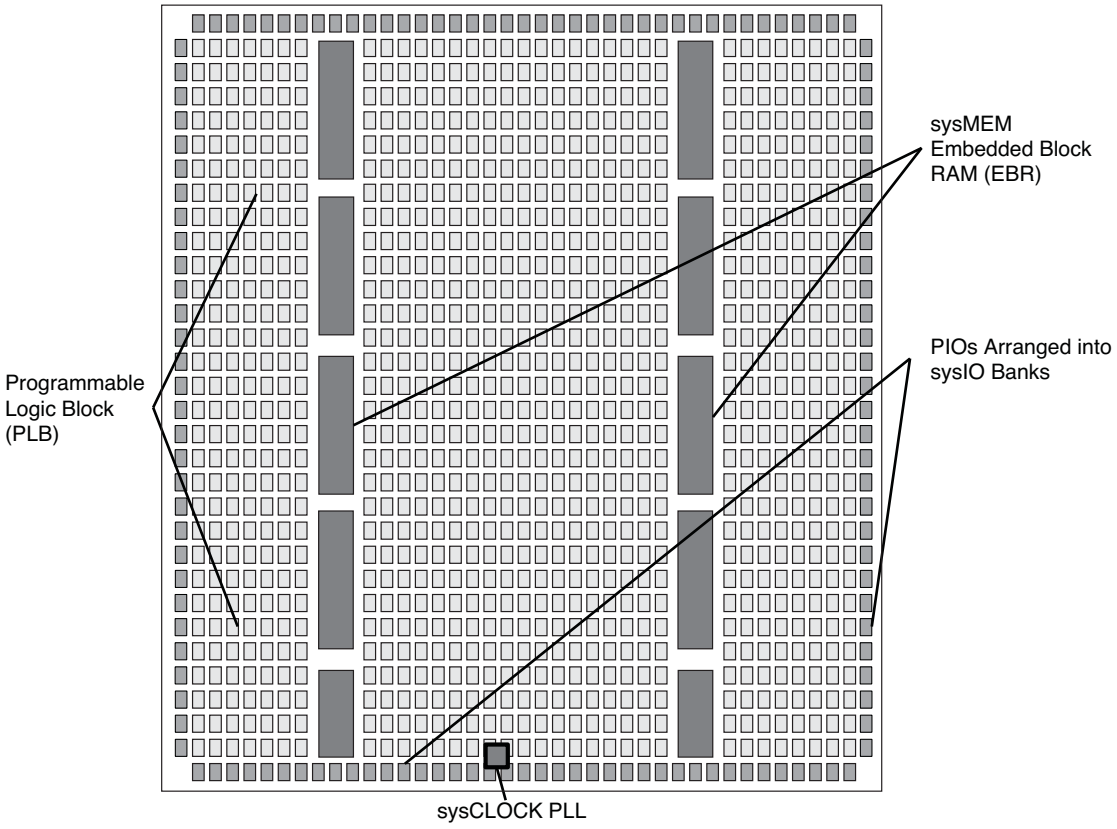
iCE40 devices contain an array of EBRs. Table 6-1 lists the number of EBRs in the device family.

Table 6-1. iCE40 Family EBRs

Device	LP384	LP1K	LP4K	LP8K	HX1K	HX4K	HX8K
Number of EBRs	0	16	20	32	16	20	32

Figure 6-1 shows the placement of EBRs in a typical iCE40 device (does not represent true numbers of design elements).

Figure 6-1. Typical Layout of an iCE40 Device

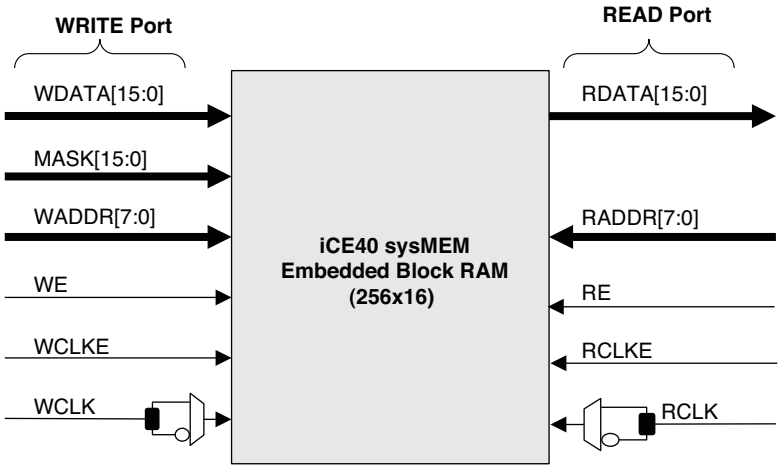


© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

iCE40 sysMEM Embedded Block RAM

Each iCE40 device includes multiple high-speed synchronous EBRs, each 4Kbit in size. A single iCE40 device integrates between eight and 32 such blocks. Each EBR is a 256-word deep by 16-bit wide, two-port register file, as illustrated in Figure 6-2. The input and output connections to and from an EBR feed into the programmable interconnect resources.

Figure 6-2. sysMEM Embedded Block RAM



Using programmable logic resources, an EBR implements a variety of logic functions, each with configurable input and output data widths.

- Random-access memory (RAM)
  - Single-port RAM with a common address, enable, and clock control lines
  - Two-port RAM with separate read and write control lines, address inputs, and enable
- Register file and scratchpad RAM
- First-In, First-Out (FIFO) memory for data buffering applications
- 256-deep by 16-wide ROM with registered outputs; contents loaded during configuration
- Counters, sequencers

As shown in Figure 6-2, an EBR has separate write and read ports, each with independent control signals. Table 6-2 lists the signals for both ports. Additionally, the write port has an active-low bit-line write-enable control; optionally mask write operations on individual bits. By default, input and output data is 16 bits wide, although the data width is configurable using programmable logic and, if needed, multiple EBRs.

The WCLK and RCLK inputs optionally connect to one of the following clock sources:

- The output from any one of the eight Global Buffers, or
- A connection from the general-purpose interconnect fabric

Signals

Table 6-2 lists the signal names, direction, and function of each connection to the EBR block.

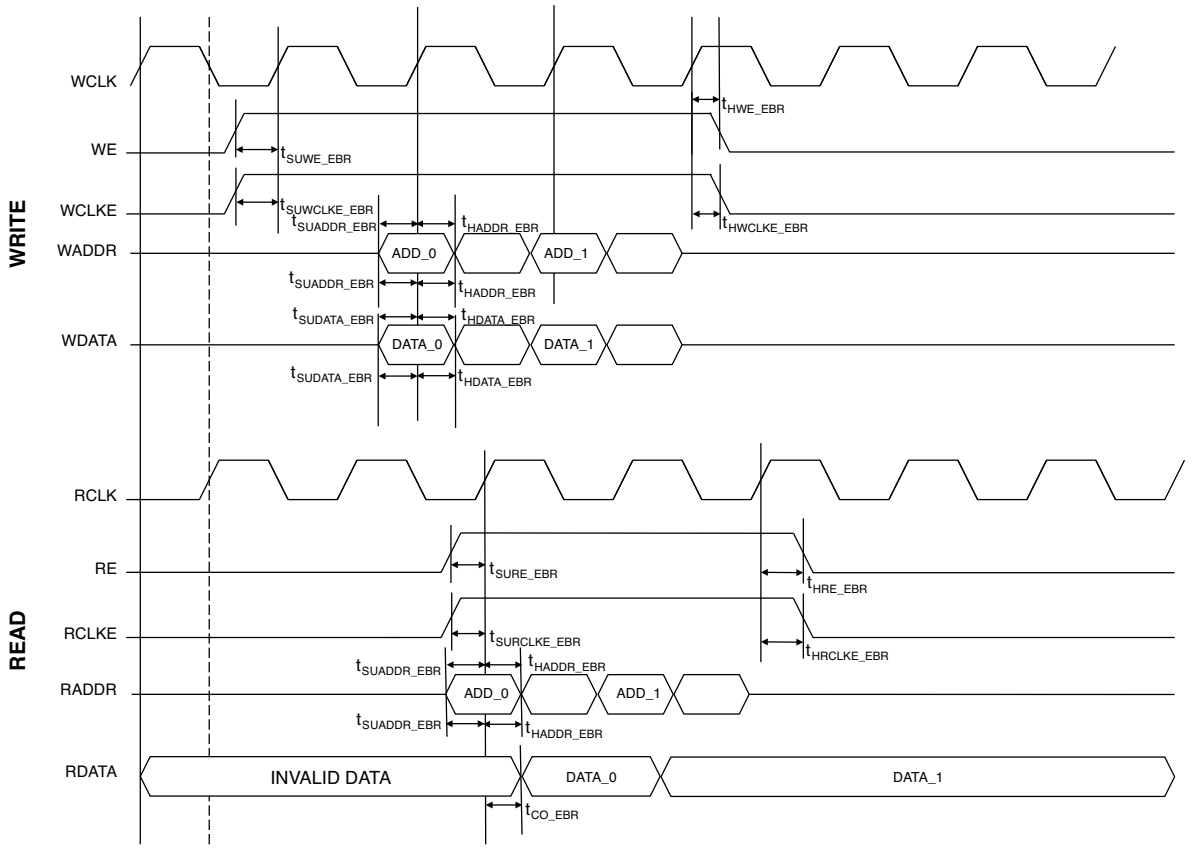
Table 6-2. EBR Signal Descriptions

Signal Name	Direction	Description
WDATA[15:0]	Input	Write Data input.
MASK[15:0]	Input	Masks write operations for individual data bit-lines. 0 = write bit; 1 = don't write bit
WADDR[7:0]	Input	Write Address input. Selects one of 256 possible RAM locations.
WE	Input	Write Enable input.
WCLKE	Input	Write Clock Enable input.
WCLK	Input	Write Clock input. Default rising-edge, but with falling-edge option.
RDATA[15:0]	Output	Read Data output.
RADDR[7:0]	Input	Read Address input. Selects one of 256 possible RAM locations.
RE	Input	Read Enable input.
RCLKE	Input	Read Clock Enable input.
RCLK	Input	Read Clock input. Default rising-edge, but with falling-edge option.

Timing Diagram

Figure 6-3 shows the timing diagram for the EBR memory module.

Figure 6-3. EBR Module Timing Diagram[1]



1. Internal timing values are considered in the iCEcube2 software's place and route.

Write Operations

By default, all EBR write operations are synchronized to the rising edge of WCLK although the clock is invertible as shown in Figure 6-2.When the WCLKE signal is low, the clock to the EBR block is disabled, keeping the EBR in its lowest power mode.

To write data into the EBR block, perform the following operations:

- Supply a valid address on the WADDR[7:0] address input port
- Supply valid data on the WDATA[15:0] data input port
- To write or mask selected data bits, set the associated MASK input port accordingly. For example, write operations on data bit D[i] are controlled by the associated MASK[i] input.
  - MASK[i] = 0: Write operations are enabled for data line WDATA[i]
  - MASK[i] = 1: Mask write operations are disabled for data line WDATA[i]
- Enable the EBR write port (WE = 1)
- Enable the EBR write clock (WCLKE = 1)
- Apply a rising clock edge on WCLK (assuming that the clock is not inverted)

Read Operations

By default, all EBR read operations are synchronized to the rising edge of RCLK although the clock is invertible as shown in Figure 6-2.

To read data from the EBR block, perform the following operations:

- Supply a valid address on the RADDR[7:0] address input port
- Enable the EBR read port (RE = 1)
- Enable the EBR read clock (RCLKE = 1)
- Apply a rising clock edge on RCLK
- After the clock edge, the EBR contents located at the specified address (RADDR) appear on the RDATA output port

EBR Considerations

Read Data Register Undefined Immediately After Configuration

Unlike the flip-flops in the Programmable Logic Blocks and Programmable I/O pins, the RDATA port is not automatically reset after configuration. Consequently, immediately following configuration and before the first valid Read Data operation, the initial RDATA read value is undefined.

Pre-loading EBR Data

The data contents for an EBR block can be optionally pre-loaded during iCE40 configuration. If not pre-loaded during configuration, then the EBR contents must be initialized by the iCE40 application before the EBR contents are valid. EBR initialization data can be done in the RTL code. Pre-loading the EBR data in the configuration bitstream increases the size of the configuration image accordingly.

EBR Contents Preserved During Configuration

EBR contents are preserved (write protected) during configuration, assuming that voltage supplies are maintained throughout. Consequently, data can be passed between multiple iCE40 configurations by leaving it in an EBR block and then skipping pre-loading during the subsequent reconfiguration.

Low-Power Setting

To place an EBR block in its lowest power mode, keep WCLKE = 0 and RCLKE = 0. In other words, when not actively using an EBR block, disable the clock inputs.

iCE40 sysMEM Embedded Block RAM Memory Primitives

This section lists the iCE40 sysMEM EBR software primitives that can be instantiated in the RTL. Different EBRs are used in different configurations. Each EBR has separate write and read ports, each with independent control signals. Each EBR can be configured into a RAM block of size 256x16, 512x8, 1024x4 or 2048x2. The data contents of the EBR can optionally be pre-loaded during iCE40 device configuration by specifying the initialization data in the primitive instantiation.

Table 6-3 lists the supported dual port synchronous RAM configurations, each of 4Kbits in size. The RAM blocks can be directly instantiated in the top module and taken through the iCube2 software flow.

Table 6-3. EBR Configurations and Primitive Names

Block RAM Configuration	Block RAM Configuration and Size	WADDR Port Size (Bits)	WDATA Port Size (Bits)	RADDR Port Size (Bits)	RDATA Port Size (Bits)	MASK Port Size (Bits)
SB_RAM256x16 SB_RAM256x16NR SB_RAM256x16NW SB_RAM256x16NRNW	256x16 (4K)	8 [7:0]	16 [15:0]	8 [7:0]	16 [15:0]	16 [15:0]
SB_RAM512x8 SB_RAM512x8NR SB_RAM512x8NW SB_RAM512x8NRNW	512x8 (4K)	9 [8:0]	8 [7:0]	9 [8:0]	8 [7:0]	No Mask Port
SB_RAM1024x4 SB_RAM1024x4NR SB_RAM1024x4NW SB_RAM1024x4NRNW	1024x4 (4K)	10 [9:0]	4 [3:0]	10 [9:0]	4 [3:0]	No Mask Port
SB_RAM2048x2 SB_RAM2048x2NR SB_RAM2048x2NW SB_RAM2048x2NRNW	2048x2 (4K)	11 [10:0]	2 [1:0]	11 [10:0]	2 [1:0]	No Mask Port

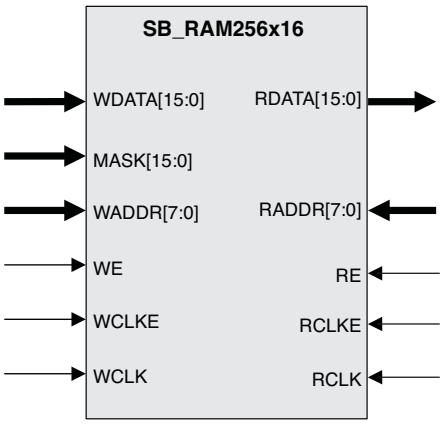
For iCE40 EBR primitives with a negative-edged read or write clock, the base primitive name is appended with a ‘N’ and a ‘R’ or ‘W’ depending on the clock that is affected (see Table 6-4 for the 256x16 RAM block configuration).

Table 6-4. Naming Convention for RAM Primitives

RAM Primitive Name	Description
SB_RAM256x16	Positive-edged read clock, positive-edged write clock
SB_RAM256x16NR	Negative-edged read clock, positive-edged write clock
SB_RAM256x16NW	Positive-edged read clock, negative-edged write clock
SB_RAM256x16NRNW	Negative-edged read clock, negative-edged write clock

SB_RAM256x16

Figure 6-4. SB_RAM256x16 Primitive



Verilog Instantiation

```
SB_RAM256x16 ram256X16_inst (
    .RDATA(RDATA_c[15:0]),
    .RADDR(RADDR_c[7:0]),
    .RCLK(RCLK_c),
    .RCLKE(RCLKE_c),
    .RE(RE_c),
    .WADDR(WADDR_c[7:0]),
    .WCLK(WCLK_c),
    .WCLKE(WCLKE_c),
    .WDATA(WDATA_c[15:0]),
    .WE(WE_c),
    .MASK(MASK_c[15:0])
);

defparam ram256x16_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

```
defparam ram256x16_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram256x16_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation

```
ram256X16_inst : SB_RAM256x16
generic map (
INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)

port map (
   RDATA => RDATA_c,
   RADDR => RADDR_c,
   RCLK => RCLK_c,
   RCLKE => RCLKE_c,
   RE => RE_c,
   WADDR => WADDR_c,
   WCLK=> WCLK_c,
   WCLKE => WCLKE_c,
   WDATA => WDATA_c,
   MASK => MASK_c,
   WE => WE_c
);
```

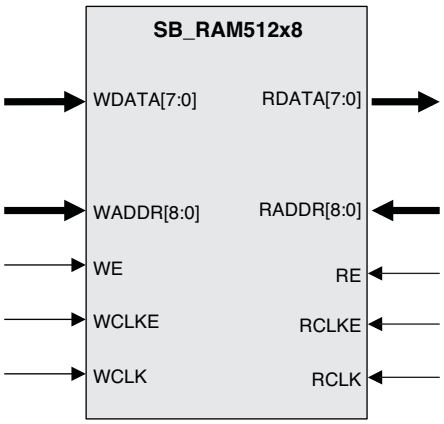
Table 6-5 is a complete list of SB_RAM256x16 based primitives.

Table 6-5. SB_RAM256x16 Based Primitives

Primitive	Description
SB_RAM256x16	SB_RAM256x16 //Positive edged clock RCLK WCLK (RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM256x16NR	SB_RAM256x16NR // Negative edged Read Clock – i.e. RCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM256x16NW	SB_RAM256x16NW // Negative edged Write Clock – i.e. WCLKN (RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM256x16NRNW	SB_RAM256x16NRNW // Negative edged Read and Write – i.e. RCLKN WRCKLN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

SB_RAM512x8

Figure 6-5. SB_RAM512x8 Primitive



Verilog Instantiation

```
SB_RAM512x8 ram512X8_inst (
    .RDATA(RDATA_c[7:0]),
    .RADDR(RADDR_c[8:0]),
    .RCLK(RCLK_c),
    .RCLKE(RCLKE_c),
    .RE(RE_c),
    .WADDR(WADDR_c[8:0]),
    .WCLK(WCLK_c),
    .WCLKE(WCLKE_c),
    .WDATA(WDATA_c[7:0]),
    .WE(WE_c)
);

defparam ram512x8_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

```
defparam ram512x8_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram512x8_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation

```
ram512X8_inst : SB_RAM512x8
generic map (
INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)

port map (
   RDATA => RDATA_c,
   RADDR => RADDR_c,
   RCLK => RCLK_c,
   RCLKE => RCLKE_c,
   RE => RE_c,
   WADDR => WADDR_c,
   WCLK=> WCLK_c,
   WCLKE => WCLKE_c,
   WDATA => WDATA_c,
```

```
   WE => WE_c
);
WE => WE_c
);
```

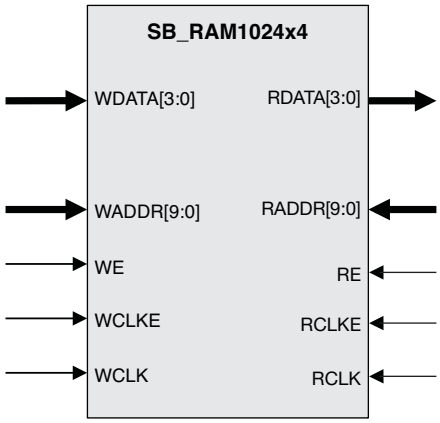
Table 6-6 is a complete list of SB_RAM512x8 based primitives.

Table 6-6. SB_RAM512x8 Based Primitives

Primitive	Description
SB_RAM512x8	SB_RAM512x8 //Positive edged clock RCLK WCLK (RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM512x8NR	SB_RAM512x8NR // Negative edged Read Clock – i.e. RCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM512x8NW	SB_RAM512x8NW // Negative edged Write Clock – i.e. WCLKN (RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM512x8NRNW	SB_RAM512x8NRNW // Negative edged Read and Write – i.e. RCLKN WRCKLN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

SB_RAM1024x4

Figure 6-6. SB_RAM1024x4 Primitive



Verilog Instantiation

```
SB_RAM1024x4 ram1024x4_inst (
   .RDATA(RDATA_c[3:0]),
   .RADDR(RADDR_c[9:0]),
   .RCLK(RCLK_c),
   .RCLKE(RCLKE_c),
   .RE(RE_c),
   .WADDR(WADDR_c[3:0]),
   .WCLK(WCLK_c),
   .WCLKE(WCLKE_c),
   .WDATA(WDATA_c[9:0]),
   .WE(WE_c)
);

defparam ram1024x4_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

```
defparam ram1024x4_inst.INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram1024x4_inst.INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation

```
Ram1024X4_inst : SB_RAM1024x4
generic map (
INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)

port map (
RDATA => RDATA_c,
RADDR => RADDR_c,
RCLK => RCLK_c,
```

```
RCLKE => RCLKE_c,
RE => RE_c,
WADDR => WADDR_c,
WCLK=> WCLK_c,
WCLKE => WCLKE_c,
WDATA => WDATA_c,
WE => WE_c
);
```

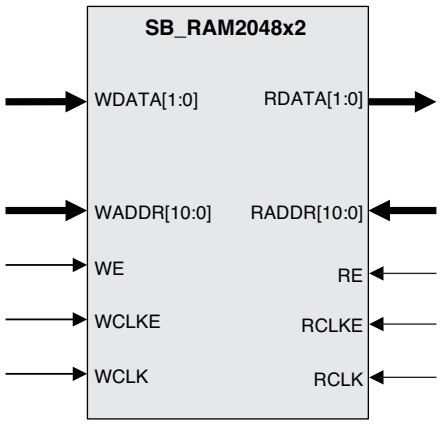
Table 6-7 is a complete list of SB_RAM1024x4 based primitives.

Table 6-7. SB_RAM1024x4 Based Primitives

Primitive	Description
SB_RAM1024x4	SB_RAM1024x4 //Positive edged clock RCLK WCLK (RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM1024x4NR	SB_RAM1024x4NR // Negative edged Read Clock – i.e. RCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM1024x4NW	SB_RAM1024x4NW // Negative edged Write Clock – i.e. WCLKN (RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM1024x4NRNW	SB_RAM1024x4NRNW // Negative edged Read and Write – i.e. RCLKN WRCKLN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

SB_RAM2048x2

Figure 6-7. SB_RAM2048x2



Verilog Instantiation

```
SB_RAM2048x2 ram2048x2_inst (
    .RDATA(RDATA_c[2:0]),
    .RADDR(RADDR_c[10:0]),
    .RCLK(RCLK_c),
    .RCLKE(RCLKE_c),
    .RE(RE_c),
    .WADDR(WADDR_c[2:0]),
    .WCLK(WCLK_c),
    .WCLKE(WCLKE_c),
    .WDATA(WDATA_c[10:0]),
    .WE(WE_c)
);
```

```
defparam ram2048x2_inst.INIT_0 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_1 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_2 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_3 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_4 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_5 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_6 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_7 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_8 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_9 =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_A =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_B =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_C =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_D =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_E =
256'h0000000000000000000000000000000000000000000000000000000000000000;
defparam ram2048x2_inst .INIT_F =
256'h0000000000000000000000000000000000000000000000000000000000000000;
```

VHDL Instantiation

```
Ram2048x2_inst : SB_RAM2048x2
generic map (
INIT_0 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_1 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_2 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_3 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_4 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_5 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_6 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_7 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_8 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_9 => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_A => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_B => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_C => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_D => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_E => X"0000000000000000000000000000000000000000000000000000000000000000",
INIT_F => X"0000000000000000000000000000000000000000000000000000000000000000"
)
```

```
port map (
   RDATA => RDATA_c,
   RADDR => RADDR_c,
   RCLK => RCLK_c,
   RCLKE => RCLKE_c,
   RE => RE_c,
   WADDR => WADDR_c,
   WCLK=> WCLK_c,
   WCLKE => WCLKE_c,
   WDATA => WDATA_c,
   WE => WE_c
);
```

Table 6-8 is a complete list of the SB_RAM2048x2 based primitives.

Table 6-8. SB_RAM2048x2 Based Primitives

Primitive	Description
SB_RAM2048x2	SB_RAM2048x2 //Positive edged clock RCLK WCLK (RDATA, RCLK, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM2048x2NR	SB_RAM2048x2NR // Negative edged Read Clock – i.e. RCLKN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLK, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM2048x2NW	SB_RAM2048x2NW // Negative edged Write Clock – i.e. WCLKN (RDATA, RCLK, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);
SB_RAM2048x2NRNW	SB_RAM2048x2NRNW // Negative edged Read and Write – i.e. RCLKN WRCKLN (RDATA, RCLKN, RCLKE, RE, RADDR, WCLKN, WCLKE, WE, WADDR, MASK, WDATA);

SB_RAM40_4K

SB_RAM40_4K is the basic physical RAM primitive which can be instantiated and configured to different depths and data ports. The SB_RAM40_4K block has a size of 4 Kbits with separate write and read ports, each with independent control signals. By default, input and output data is 16 bits wide, although the data width is configurable using the READ_MODE and WRITE_MODE parameters. The data contents of the SB_RAM40_4K block are optionally pre-loaded during iCE device configuration.

Table 6-9. SB_RAM40_4K Naming Convention Rules

RAM Primitive Name	Description
SB_RAM40_4K	Positive-edged read clock, positive-edged write clock
SB_RAM40_4KNR	Negative-edged read clock, positive-edged write clock
SB_RAM40_4KNW	Positive-edged read clock, negative-edged write clock
SB_RAM40_4KNRNW	Negative-edged clock, negative-edged write clock

Figure 6-8. SB_RAM40_4K

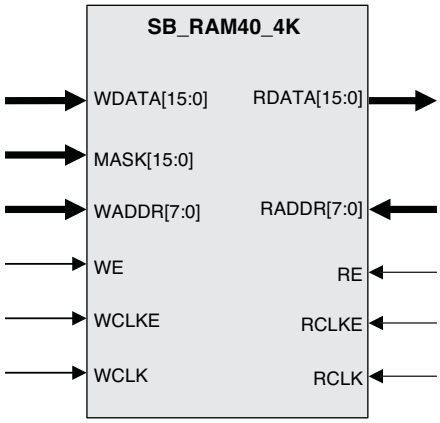


Table 6-10 lists the signals for both ports.

Table 6-10. SB_RAM40_4K Signal Descriptions

Signal Name	Direction	Description
WDATA[15:0]	Input	Write Data input.
MASK[15:0]	Input	Bit-line Write Enable input, active low. Applicable only when WRITE_MODE parameter is set to '0'.
WADDR[7:0]	Input	Write Address input. Selects up to 256 possible locations.
WE	Input	Write Enable input, active high.
WCLK	Input	Write Clock input, rising-edge active.
WCLKE	Input	Write Clock Enable input.
RDATA[15:0]	Output	Read Data output.
RADDR[7:0]	Input	Read Address input. Selects one of 256 possible locations.
RE	Input	Read Enable input, active high.
RCLK	Input	Read Clock input, rising-edge active.
RCLKE	Input	Read Clock Enable input.

Table 6-11 describes the parameter values to infer the desired RAM configuration.

Table 6-11. SB_RAM40_4K Primitive Parameter Descriptions

Parameter Name	Description	Parameter Value	Configuration
INIT_0, … …,INIT_F	RAM Initialization Data. Passed using 16 parameter strings, each comprising 256 bits. (16x256=4096 total bits)	INIT_0 to INIT_F	Initialize the RAM with predefined value
WRITE_MODE	Sets the RAM block write port configuration	0	256x16
		1	512x18
		2	1024x4
		3	2048x2
READ_MODE		0	256x16
		1	512x8
		2	1024x4
		3	2048x2

Verilog Instantiation

```
// Physical RAM Instance without Pre Initialization
SB_RAM40_4K ram40_4kinst_physical (
   .RDATA(RDATA),
   .RADDR(RADDR),
   .WADDR(WADDR),
   .MASK(MASK),
   .WDATA(WDATA)
   .RCLKE(RCLKE),
   .RCLK(RCLK),
   .RE(RE),
   .WCLKE(WCLKE),
   .WCLK(WCLK),
   .WE(WE)
);

defparam ram40_4kinst_physical.READ_MODE=0;
defparam ram40_4kinst_physical.WRITE_MODE=0;
```

VHDL Instantiation

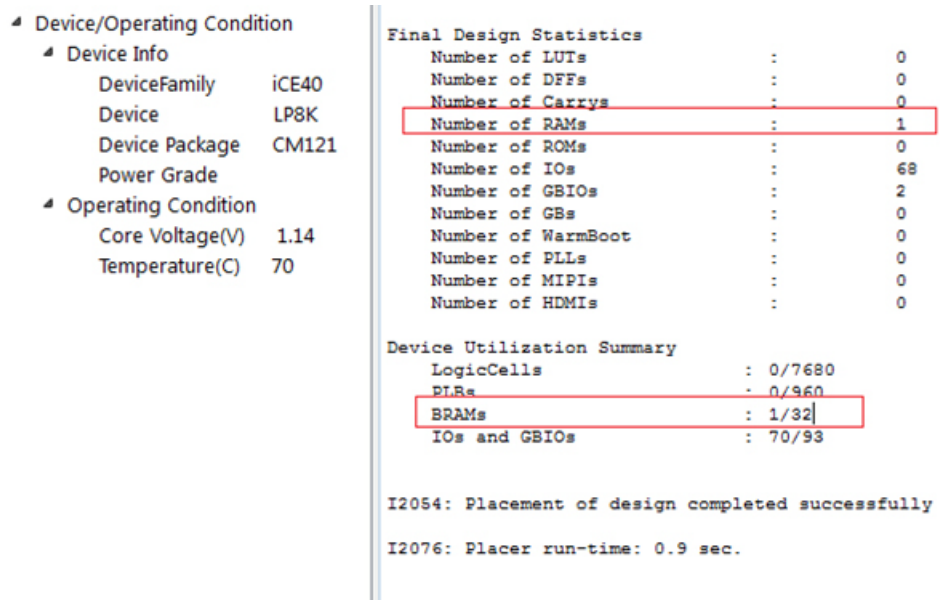
```
-- Physical RAM Instance without Pre Initialization
ram40_4kinst_physical : SB_RAM40_4K
generic map (
   READ_MODE => 0,
   WRITE_MODE= >0
)

port map (
   RDATA=>RDATA,
   RADDR=>RADDR,
   WADDR=>WADDR,
   MASK=>MASK,
   WDATA=>WDATA,
   RCLKE=>RCLKE,
   RCLK=>RCLK,
   RE=>RE,
   WCLKE=>WCLKE,
   WCLK=>WCLK,
   WE=>WE
);
```

EBR Utilization Summary in iCEcube2 Design Software

The placer.log file in the iCEcube2 design software shows the device utilization summary. The Final Design Statistics and Device Utilization Summary sections show the number of EBRs (or RAMs) used against the total number. Figure 6-9 shows the EBR usage when one SB_RAM256x16 was instantiated in the design.

Figure 6-9. iCEcube2 Design Software Report File



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
September 2012	01.0	Initial release.

Appendix A. Standard HDL Code References

This appendix contains standard HDL (Verilog and VHDL) codes for popular memory elements, which can be used to infer a sysMEM EBR automatically. Standard HDL coding techniques do not require you to know the details of the Block RAMs of the device and are inferred automatically.

Single-Port RAM

Verilog

```
module ram (din, addr, write_en, clk, dout);// 512x8
  parameter addr_width = 9;
  parameter data_width = 8;
  input [addr_width-1:0] addr;
  input [data_width-1:0] din;
  input write_en, clk;
  output [data_width-1:0] dout;
  reg [data_width-1:0] dout; // Register for output.
  reg [data_width-1:0] mem [(1<<addr_width)-1:0];

  always @(posedge clk)
  begin
    if (write_en)
      mem[(addr)] <= din;
    dout = mem[addr]; // Output register controlled by clock.
  end
endmodule
```

VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity ram is
generic (
  addr_width : natural := 9;--512x8
  data_width : natural := 8);
port (
  addr : in  std_logic_vector (addr_width - 1 downto 0);
  write_en : in  std_logic;
  clk : in  std_logic;
  din : in  std_logic_vector (data_width - 1 downto 0);
  dout : out std_logic_vector (data_width - 1 downto 0));
end ram;

architecture rtl of ram is
  type mem_type is array ((2** addr_width) - 1 downto 0) of
    std_logic_vector(data_width - 1 downto 0);
  signal mem : mem_type;
begin
  process (clk)
  begin
    if (clk'event and clk = '1') then
      if (write_en = '1') then
        mem(conv_integer(addr)) <= din;
      end if;
```

```
            dout <= mem(conv_integer(addr));
    end if;

      -- Output register controlled by clock.
  end process;
end rtl;
```

Dual Port Ram

Verilog

```
module ram (din, write_en, waddr, wclk, raddr, rclk, dout);//512x8
  parameter addr_width = 9;
  parameter data_width = 8;
  input [addr_width-1:0] waddr, raddr;
  input [data_width-1:0] din;
  input write_en, wclk, rclk;
  output reg [data_width-1:0] dout;
  reg [data_width-1:0] mem [(1<<addr_width)-1:0]
     ;

  always @(posedge wclk) // Write memory.
  begin
    if (write_en)
      mem[waddr] <= din; // Using write address bus.
  end
  always @(posedge rclk) // Read memory.
  begin
    dout <= mem[raddr]; // Using read address bus.
  end
endmodule
```

VHDL

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity ram is
generic (
  addr_width : natural := 9;--512x8
  data_width : natural := 8);
port (
  write_en : in  std_logic;
  waddr : in  std_logic_vector (addr_width - 1 downto 0);
  wclk : in  std_logic;
  raddr : in  std_logic_vector (addr_width - 1 downto 0);
  rclk : in  std_logic;
  din : in  std_logic_vector (data_width - 1 downto 0);
  dout : out std_logic_vector (data_width - 1 downto 0));
end ram;

architecture rtl of ram is
  type mem_type is array ((2** addr_width) - 1 downto 0) of
    std_logic_vector(data_width - 1 downto 0);
  signal mem : mem_type;
```

```
begin
  process (wclk)
  -- Write memory.
  begin
    if (wclk'event and wclk = '1') then
      if (write_en = '1') then
        mem(conv_integer(waddr)) <= din;
          -- Using write address bus.
      end if;
    end if;
  end process;
  process (rclk) -- Read memory.
  begin
    if (rclk'event and rclk = '1') then
      dout <= mem(conv_integer(raddr));
        -- Using read address bus.
    end if;
  end process;
end rtl;
```

iCE40 sysCLOCK PLL Design and Usage Guide

September 2012 Technical Note TN1251

Introduction

This technical note discusses the clock resources available in the iCE40™ device. Details are provided for global buffers and sysCLOCK™ PLLs. The iCE40 devices include an ultra-low power Phase Locked Loop (PLL) to support a variety of display, imaging and memory interface applications. Table 7-1 lists the number of sysCLOCK PLLs and global buffers in the iCE40 device family.

Table 7-1. Number of PLLs in the iCE40 Device Family

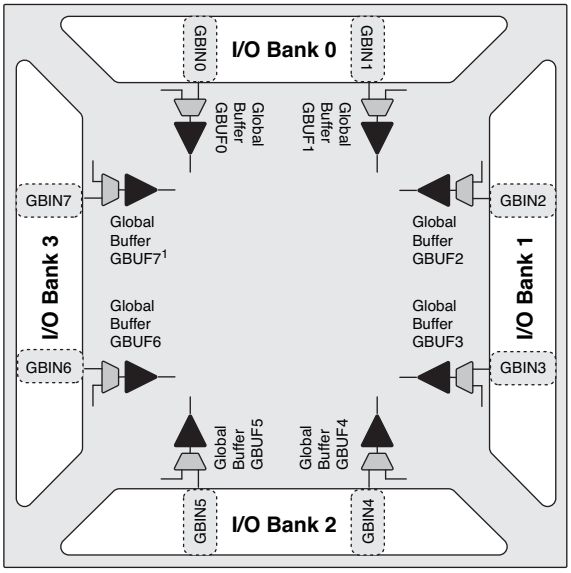
Parameter	LP384	LP1K	LP4K	LP8K	HX1K	HX4K	HX8K
Number of PLLs	0	1	2	2	1	2	2
Number of Global Buffers	8	8	8	8	8	8	8

Note: The following device packages do not have PLLs: CM36, CM36A, QN84 and VQ100. The CM81 package has one PLL.

Global Routing Resources

The iCE40 device has eight high drive buffers called global buffers (GBUFx). These are connected to eight low-skew global lines, designed primarily for clock distribution but are also useful for other high-fanout signals such as set/reset and enable signals.

Figure 7-1. High-drive, Low-skew, High-fanout Global Buffer Routing Resources



1. GBUF7 and its associted PIO are best for direct differential clock inputs.

The input (sources) to the GBUFx can be:

- Global buffer inputs (GBINx)
- Programmable interconnect[1]
- PLL output[1]
- Programmable input/output block (PIO)[1]

1. To use a global buffer along with a user interface or PIO, use the SB_GB primitive if it is not inferred automatically.

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

The associated GBINx pin represents the best pin to drive a global buffer from an external source.

Verilog Instantiation

```
SB_GB My_Global_Buffer_i (// required for a user's internally generated FPGA signal that is
heavily loaded and requires global buffering. For example, a user's logic-generated clock.

.USER_SIGNAL_TO_GLOBAL_BUFFER (Users_internal_Clk),

.GLOBAL_BUFFER_OUTPUT ( Global_Buffered_User_Signal) );
```

VHDL Instantiation

```
component SB_GB
   port (
   USER_SIGNAL_TO_GLOBAL_BUFFER  :     input std_logic;
   GLOBAL_BUFFER_OUTPUT          :     output std_logic);
end component;

My_Global_Buffer_i: SB_GB
   port map (
   USER_SIGNAL_TO_GLOBAL_BUFFER  =>    Users_internal_Clk,
   BUFFER                        =>    Global_Buffered_User_Signal);
```

Refer to the iCE Technology Library for more details on device primitives

If not used in an application, individual global buffers are turned off to save power.

Table 7-2 lists the connections between a specific global buffer and the inputs on a Programmable Logic Block (PLB). Refer to the Architecture section of the iCE40 Family Data Sheet for more information on PLBs. All global buffers optionally connect to all clock inputs. Any four of the eight global buffers can drive logic inputs to a PLB. Even-numbered global buffers optionally drive the reset input to a PLB. Similarly, odd-numbered buffers optionally drive the PLB clock-enabled input.

Table 7-2. Global Buffer Connections to a Programmable Logic Block

Global Buffer	LUT Inputs	Clock	Clock Enable	Reset
GBUF0	Yes, any 4 of 8 GBUF Inputs	✓	✓	
GBUF1		✓		✓
GBUF2		✓	✓	
GBUF3		✓		✓
GBUF4		✓	✓	
GBUF5		✓		✓
GBUF6		✓	✓	
GBUF7		✓		✓

Table 7-3 lists the connections between a specific global buffer and the inputs on a Programmable I/O (PIO) pair. Although there is no direct connection between a global buffer and a PIO output, such a connection is possible by first connecting through a PLB LUT4 function. Again, all global buffers optionally drive all clock inputs. However, even-numbered global buffers optionally drive the clock-enable input on a PIO pair.

Table 7-3. Global Buffer Connections to Programmable I/O Pair

Global Buffer	Output Connections	Input Clock	Output Clock	Clock Enable
GBUF0	None (connect through PLB LUT)	✓	✓	✓
GBUF1		✓	✓	
GBUF2		✓	✓	✓
GBUF3		✓	✓	
GBUF4		✓	✓	✓
GBUF5		✓	✓	
GBUF6		✓	✓	✓
GBUF7		✓	✓	

iCE40 sysCLOCK PLL

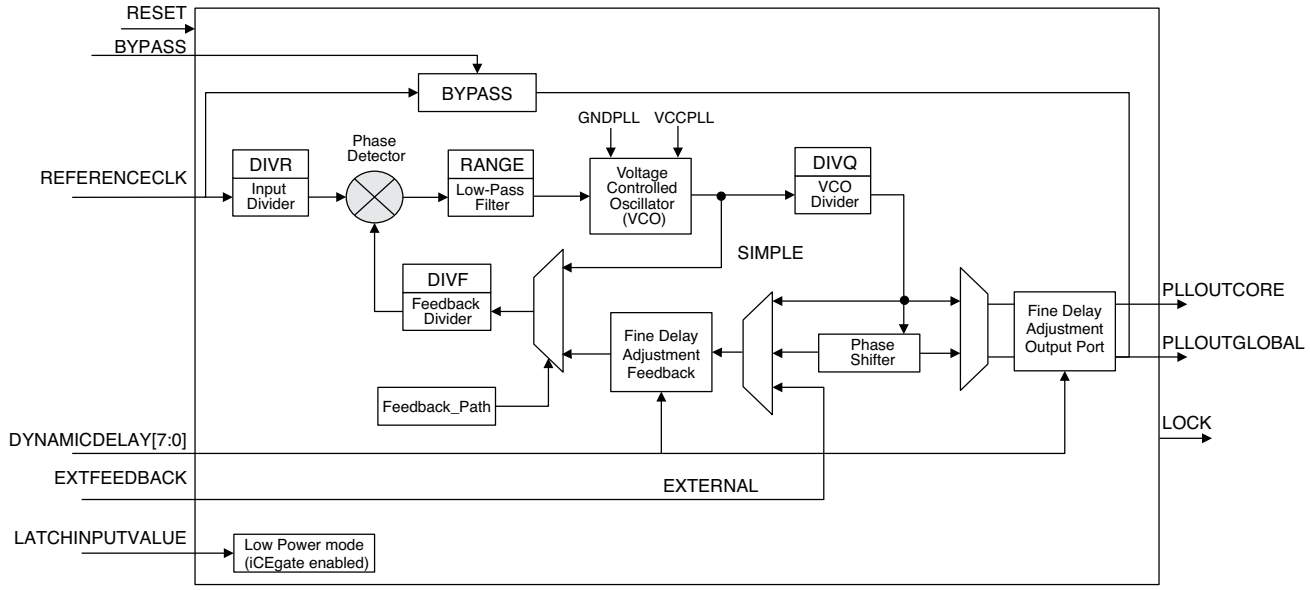
The iCE40 Phase Locked Loop (PLL) provides a variety of user-synthesizable clock frequencies, along with custom phase delays.The PLL in the iCE40 device can be configured and utilized with the help of software macros or the PLL Module Generator. The PLL Module Generator utility helps users to quickly configure the desired settings with the help of a GUI and generate Verilog code which configures the PLL macros. Figure 7-2 shows the iCE40 sysCLOCK PLL block diagram.

iCE40 sysCLOCK PLL Features

The PLL provides the following functions in iCE40 applications:

- Generates a new output clock frequency
  - Clock multiplication
  - Clock division
- De-skews or phase-aligns an output clock to the input reference clock
  - Faster input set-up time
  - Faster clock-to-output time
- Corrects output clock to have nearly a 50% duty cycle, which is important for Double Data Rate (DDR) applications
- Optionally phase shifts the output clock relative to the input reference clock
  - Optimal data sampling within the available bit period
  - Fixed quadrant phase shifting at 0°, 90°
  - Optional fine delay adjustments of up to 2.5 ns (typical) in 150 ps increments (typical)

Figure 7-2. iCE40 Phase Locked Loop (sysCLOCK PLL) Block Diagram



Signals

Table 7-4 lists the signal names, direction, and function of each connection to the PLL. Some of the signals have an associated attribute or property, as listed in Table 7-4. Table 7-4 lists the attributes or properties associated with the PLL and the allowable settings for each attribute.

Note: Signals and attribute settings of PLL primitives are for reference only. It is recommend to generate a PLL module with the GUI-based PLL Module Generator as explained in "Generating iCE40 PLL Using PLL Module Generator in iceCube2 Design Software" on page 8.

Table 7-4. PLL Signals

Signal Name	Direction	Description
REFERENCECLK	Input	Input reference clock
RESET	Input	Reset
BYPASS	Input	When FEEDBACK_PATH is set to SIMPLE, the BYPASS control selects which clock signal connects to the PLLOUT output. 0 = PLL generated signal 1 = REFERENCECLK
EXTFEEDBACK	Input	External feedback input to PLL. Enabled when the FEEDBACK_PATH attribute is set to EXTERNAL.
DYNAMICDELAY[3:0]	Input	Fine delay adjustment control inputs. Enabled when DELAY_ADJUSTMENT_MODE is set to DYNAMIC.
LATCHINPUTVALUE	Input	When enabled, forces the PLL into low-power mode; PLL output is held static at the last input clock value. Set ENABLE ICEGATE_PORTA and PORTB to '1' to enable.
PLLOUTGLOBAL	Output	Output from the Phase-Locked Loop (PLL). Drives a global clock network on the FPGA. The port has optimal connections to global clock buffers GBUF4 and GBUF5.
PLLOUTCORE	Output	Output clock generated by the PLL, drives regular FPGA routing. The frequency generated on this output is the same as the frequency of the clock signal generated on the PLLOUTLGOBAL port.
LOCK	Output	When High, indicates that the PLL output is phase aligned or locked to the input reference clock.

Table 7-5. PLL Attributes and Settings in PLL Macro [1]

Parameter Name	Description	Parameter Value	Description
FEEDBACK_PATH	Selects the feedback path to the PLL	SIMPLE	Feedback is internal to the PLL, directly from VCO.
		DELAY	Feedback is internal to the PLL, through the Fine Delay Adjust Block.
		PHASE_AND_DELAY	Feedback is internal to the PLL, through the Phase Shifter and the Fine Delay Adjust Block.
		EXTERNAL	Feedback path is external to the PLL and connects to the EXTFEEDBACK pin. Also uses the Fine Delay Adjust Block.
DELAY_ADJUSTMENT_MODE_FEEDBACK	Selects the mode for the Fine Delay Adjust block in the feedback path	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_FEEDBACK parameter setting.
		DYNAMIC	Delay of Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[3:0] pins.
FDA_FEEDBACK	Sets a constant value for the Fine Delay Adjust Block in the feedback path	0, 1,…,15	The PLLOUTGLOBAL and PLLOUTCORE signals are delay compensated by (n+1)*150 ps, where n = FDA_FEEDBACK only if DELAY_ADJUSTMENT_MODE_FEEDBACK is set to FIXED.
DELAY_ADJUSTMENT_MODE_RELATIVE	Selects the mode for the Fine Delay Adjust block	FIXED	Delay of the Fine Delay Adjust Block is fixed, the value is specified by the FDA_RELATIVE parameter setting.
		DYNAMIC	Delay of the Fine Delay Adjust Block is determined by the signal value at the DYNAMICDELAY[7:4] pins.
FDA_RELATIVE	Sets a constant value for the Fine Delay Adjust Block	0, 1,…,15	The PLLOUTGLOBALA and PLLOUTCOREA signals are additionally delayed by (n+1)*150 ps, where n = FDA_RELATIVE. Used if DELAY_ADJUSTMENT_MODE_RELATIVE is set to FIXED.
SHIFTREG_DIV_MODE	Selects shift register configuration	0,1	Used when FEEDBACK_PATH is set to PHASE_AND_DELAY. 0 = Divide by 4 1 = Divide by 7

Table 7-5. PLL Attributes and Settings in PLL Macro (Continued)[1]

Parameter Name	Description	Parameter Value	Description
PLLOUT_SELECT	Selects the signal to be output at the PLLOUTCORE and PLLOUTGLOBAL ports	SHIFTREG_0deg	0° phase shift only if the setting of FEEDBACK_PATH is set to PHASE_AND_DELAY.
		SHIFTREG_90deg	90° phase shift only if the setting of FEEDBACK_PATH is PHASE_AND_DELAY and SHIFTREG_DIV_MODE = 0.
		GENCLK	The internally generated PLL frequency will be output without any phase shift.
		GENCLK_HALF	The internally generated PLL frequency will be divided by 2 and then output. No phase shift.
DIVR	REFERENCECLK divider	0,1,2,…,15	These parameters are used to control the output frequency, depending on the FEEDBACK_PATH setting.
DIVF	Feedback divider	0,1,..,63	
DIVQ	VCO divider	0,1,…,7	
FILTER_RANGE	PLL filter range	0,1,…,7	
EXTERNAL_DIVIDE_FACTOR	Divide-by factor of a divider in external feedback path	User specified value. Default = 1	Specified only when there is a user-implemented divider in the external feedback path.
ENABLE_ICEGATE	Enables the PLL power-down control	0	Power-down control disabled.
		1	Power-down controlled by the LATCHINPUTVALUE input.

1. The attributes are automatically configured through the PLL Module Generator.

Clock Input Requirements

Proper operation requires the following considerations:

- A stable monotonic (single frequency) reference clock input
- The reference clock input must be within the input clock frequency range, F_{REF}, specified in the data sheet
- The reference clock must have a duty cycle that meets the requirement specified in the data sheet
- The jitter on the reference input clock must not exceed the limits specified in the data sheet

PLL Output Requirements

The PLL output clock, PLLOUT, has the following restrictions:

- The PLLOUT output frequency must be within the limits specified in the data sheet
- The PLLOUT output is not valid or stable until the PLL LOCK output remains high

Functional Description

The PLL optionally multiplies and/or divides the input reference clock to generate a PLLOUT output clock of another frequency. The output frequency depends on the frequency of the REFERENCLK input clock and the settings for the DIVR, DIVF, DIVQ, RANGE, and FEEDBACK_PATH attribute settings, as indicated in Figure 7-2.

The PLL's phase detector and Voltage Controlled Oscillator (VCO) synthesize a new output clock frequency based on the attribute settings. The VCO is an analog circuit and has independent voltage supply and ground connections labeled VCCPLL and GNDPLL.

PLLOUT Frequency for All Modes Except FEEDBACK_PATH = SIMPLE

For all the FEEDBACK_PATH modes, except SIMPLE, the PLLOUT frequency calculated as per the equation below.

$$F_{PLLOUT} = \frac{F_{REFERENCECLK} \times (\mathrm{DIVF} + 1)}{\mathrm{DIVR} + 1}$$

PLLOUT Frequency for FEEDBACK_PATH = SIMPLE

In the SIMPLE feedback mode, the PLL feedback signal taps directly from the output of the VCO, before the final divider stage. Consequently, the PLL output frequency has an additional divider step, DIVQ, contributed by the final divider step as shown in equation below. (DIVF, DIVQ and DIVR are binary).

$$F_{PLLOUT} = \frac{F_{REFERENCECLK} \times (\mathrm{DIVF} + 1)}{2^{(DIVQ)} \times (DIVR + 1)}$$

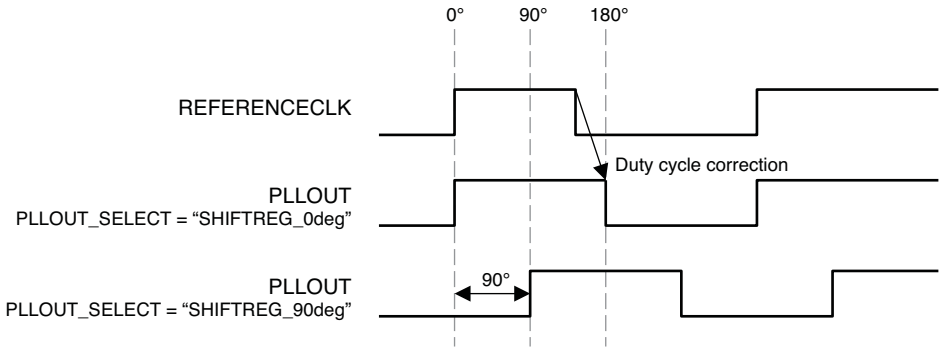
Fixed Quadrant Phase Shift

The PLL optional phase feature shifts the PLLOUT output by a specified quadrant or quarter clock cycle as shown in Figure 7-3 and Table 7-6. The quadrant phase shift option is only available when the FEEDBACK_PATH attribute is set to PHASE_AND_DELAY.

Table 7-6. PLL Phase Shift Options

PLLOUT_SELECT	Duty Cycle Correction	Phase Shift	Fraction Clock Cycle
SHIFTREG_0deg	Yes	0º	None
SHIFTREG_90deg	Yes	90º	Quarter Cycle

Figure 7-3. Fixed Quadrant Phase Shift Control



Unlike the Fine Delay Adjustment, the quadrant phase shifter always shifts by a fixed phase angle. The resulting phase shift, measured in delay, depends on the clock period and the PLLOUT_PHASE phase shift setting, as shown in the equation below.

$$Delay = \frac{\text{Phase Shift}}{360°} \times \text{Clock\_Period}$$

Fine Delay Adjustment (FDA)

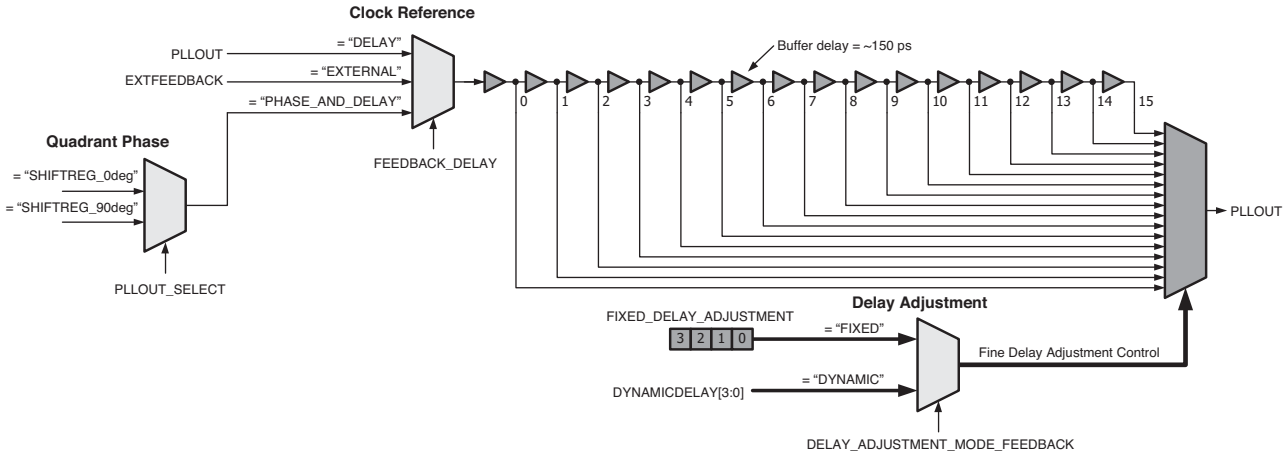
The PLL provides two optional fine delay adjustment blocks that control the delay of the PLLOUT output relative to the input reference clock, to an external feedback signal, or relative to the selected quadrant phase shifted clock. One FDA is placed in the feedback path, while the other FDA provides delay on the output port directly. If a two-port PLL is used, this additional delay is applied only on Port A. Unlike the Feedback FDA, the output port FDA is not dependent on FEEDBACK_PATH, and can be used even if FEEDBACK_PATH = Simple. The PLL Module Generator provides easy selection of the two fine delay adjust blocks. Figure 7-4 shows the typical first fine delay adjust control block.

The delay is adjusted by selecting one or more of the 16 delay taps inside the fine delay adjustment block. Each tap is approximately 150 ps.

Fine Delay Adjustment (nominal) = (n+1) * 150ps; 0 ≤ n ≤ 15, where 'n' is the number of delay taps.

The number of taps can be selected statically (by providing the value within the PLL Module Generator) or dynamically by setting the values in DYNAMICDELAY [7:0]. DYNAMICDELAY [3:0] sets the tap numbers for the feedback path fine delay adjustment block while DYNAMICDELAY [7:0] sets values for the output port FDA. Refer to parameters DELAY_ADJUSTMENT_MODE_FEEDBACK and DELAY_ADJUSTMENT_MODE_RELATIVE in Table 7-2 for more details.

Figure 7-4. Fine Delay Adjust Control



Phase Angle Equivalent

The fine delay adjustment feature injects an actual delay value, rather than a fixed phase angle like the Fixed Quadrant Phase Shift feature. Use the equation below to convert the fine adjustment delay to a resulting phase angle.

$$\text{Phase Shift} = \frac{\text{Fine\_Delay\_Adustment}}{\text{Clock Period}} \times 360°$$

Low Power Mode

The iCE40 sysCLOCK PLL has low operating power by default. The PLL can be dynamically disabled to further reduce power. The low-power mode must first be enabled by setting the ENABLE_ICEGATE attribute to '1'. Once enabled, use the LATCHINPUTVALUE to control the PLL's operation, as shown in Table 7-7. The PLL must reacquire the input clock and LOCK when the LATCHINPUTVALUE returns from '1' to '0', external feedback is used and path goes out into the fabric.

Table 7-7. PLL LATCHINPUTVALUE Control

ENABLE_ICEGATE Attribute	LATCHINPUTVALUE Input	Function
0	Don't Care	PLL is always enabled.
1	0	PLL is enabled and operating.
	1	PLL is in low-power mode; PLLOUT output holds last clock state.

Generating iCE40 PLL Using PLL Module Generator in iceCube2 Design Software

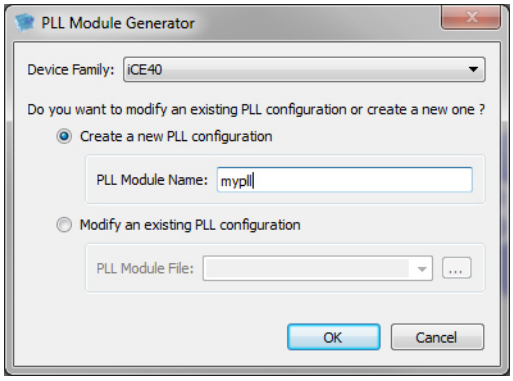
A GUI-based PLL Configuration tool is provided in iceCube2™ design software which configures the iCE40 PLL software macros based on the inputs in the GUI. The resultant HDL code can be used for synthesis.

Figure 7-5 shows the iceCube2 design software. The PLL module generator GUI can be invoked from the Tool menu as shown.

Figure 7-5. iceCube2 Design Software

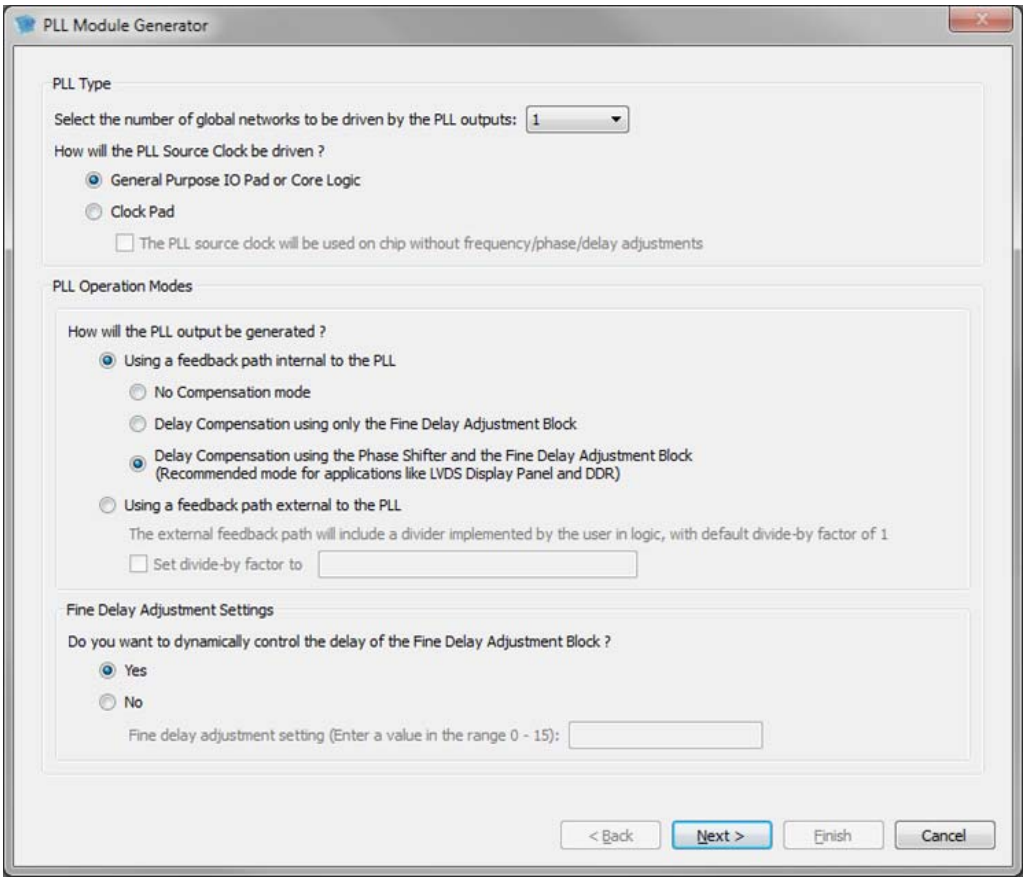
Figure 7-6 shows the PLL configuration GUI. Select the device (iCE40 in this case) and other desired operations.

Figure 7-6. PLL Module Generator/Modify Existing PLL Configuration



Click **OK** and the PLL frequency settings window will open up, as shown in Figure 7-7.

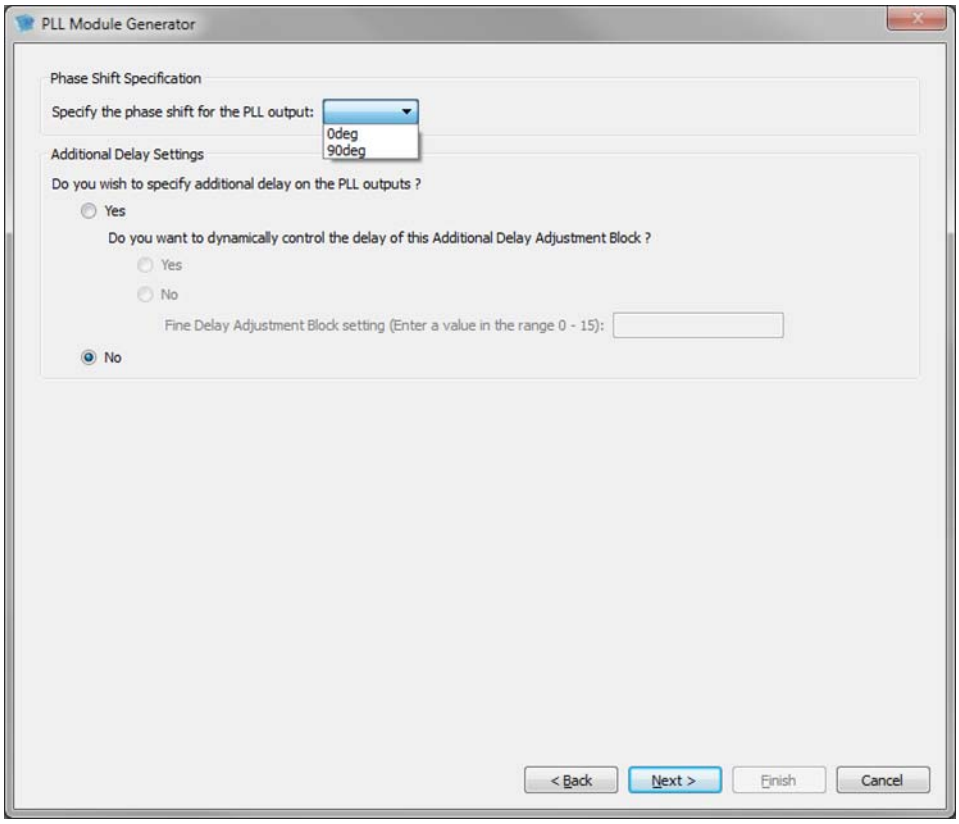
Figure 7-7. Settings Window 1



Refer to Table 7-8 for details on user options. Select desired settings which are self-explanatory. Note that some of the options are only activated when other required selections are made. These settings directly modify the PLL signals and attributes of the PLL software macro, as explained in Tables 7-4 and 7-5.

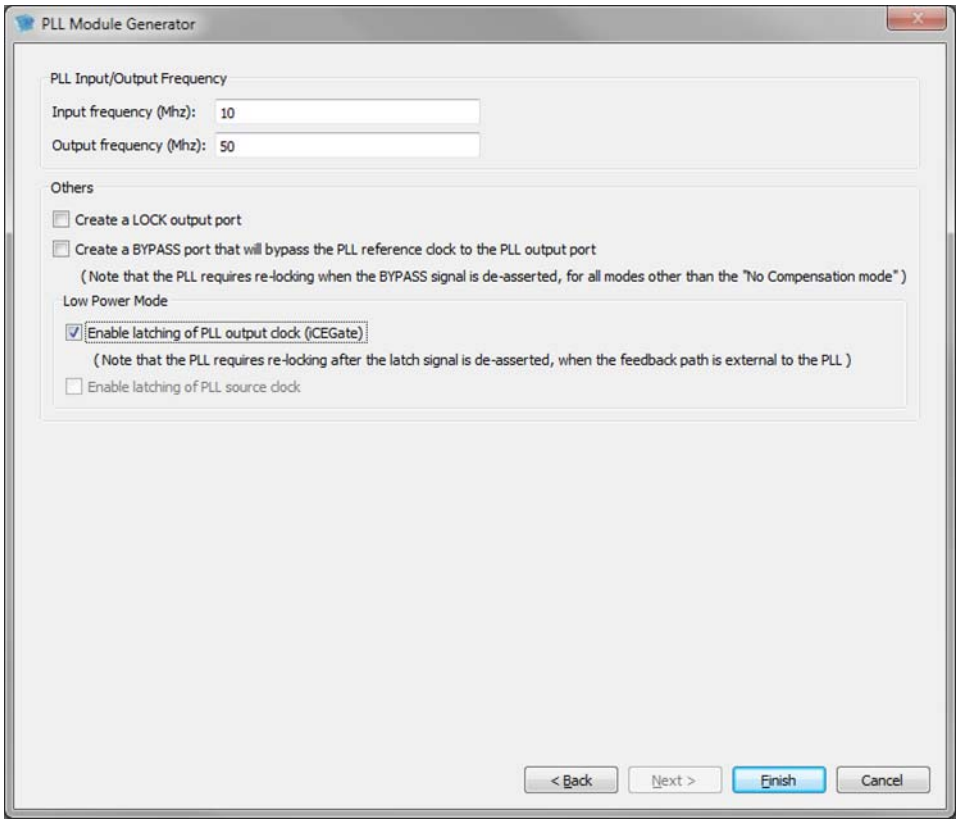
Figure 7-8 shows the next setting window.

Figure 7-8. Settings Window 2



Select desired values and click **Next**. The last of the settings windows will open, as shown in Figure 7-9.

Figure 7-9. Settings Window 3



After selecting the desired values, the final window the PLL Module Generator opens, as shown in Figure 7-10. This window shows all the values of the attributes and parameters that were discussed in the previous sections and in Tables 7-4 and 7-5. It also shows which PLL Macro type has been selected. The PLL Macro Type used in this example is SB_PLL40_Core.

Figure 7-10. PLL Configuration – Final Settings

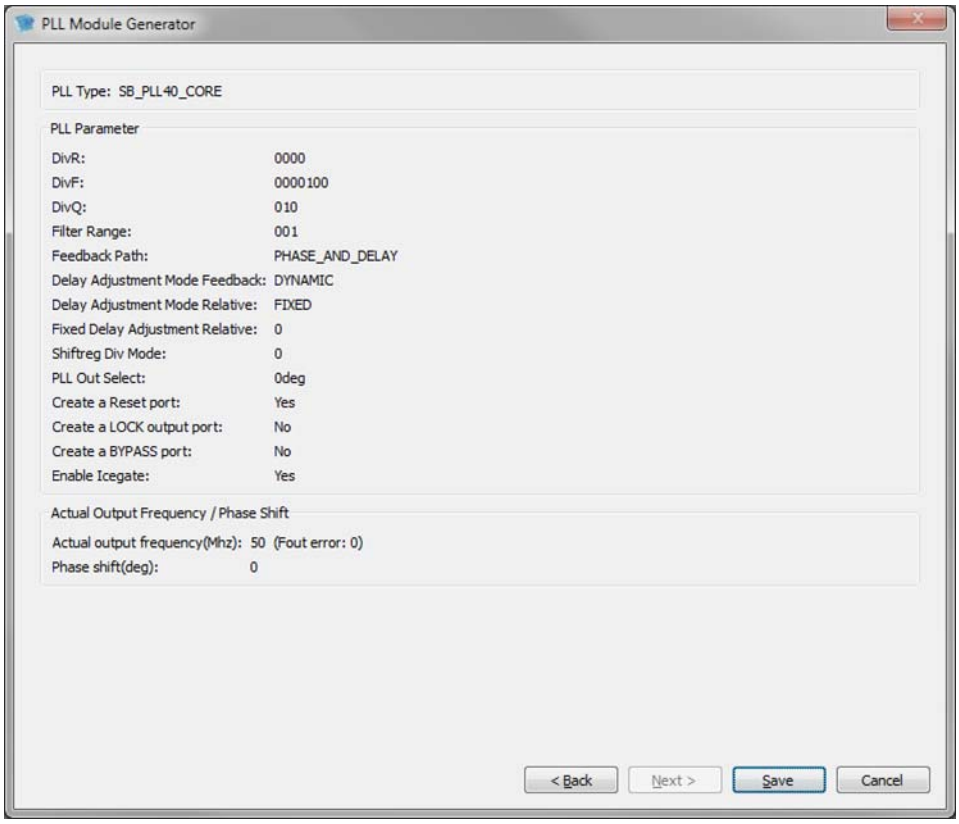


Table 7-8. PLL Configuration Tool User Parameters

User Parameter	Description	Range	Description
PLL Type			
Select the number of global networks to be driven by the PLL outputs	Setting the value to '1' generates a PLL which drives a single global clock network, as well as regular routing. Setting the value to '2' generates a PLL which drives two global clock networks as well as two regular routing resources.	1	
		2	
How will the PLL Source Clock be driven?		General Purpose I/O Pad or Core Logic	In this scenario, the PLL input (source clock) is driven by a signal from the FPGA fabric. This signal can either be generated on the FPGA core, or it can be an external signal that was brought onto the FPGA using a General Purpose I/O pad.
		Clock Pad	The PLL input clock (source) is driven by a dedicated clock pad located in I/O Bank 2 (bottom bank) or I/O Bank 0 (top bank). If the number of global networks is two, the source clock of the PLL can be used as is (i.e. without any frequency, delay compensation or phase adjustments). It is recommended that if the source clock is required on-chip, this option should not be selected.

Table 7-8. PLL Configuration Tool User Parameters (Continued)

User Parameter	Description	Range	Description
PLL Operation Modes			
How will the PLL output be generated?	The PLL can be configured to operate in one of multiple modes. An Operation Mode determines the feedback path of the PLL and enables phase alignment of the generated clock with regard to the source clock.	Using a feedback path internal to the PLL	This option is related to the phase delay introduced in the feedback path. The options are: **No Compensation mode** – There is no phase delay in the feedback path. **Delay compensation using only the Fine Delay Adjustment (FDA) Block** – The feedback path traverses through the FDA block, as explained in the section "Fine Delay Adjustment (FDA)" on page 7. **Delay compensation using the phase shifter and the Fine Delay Adjustment (FDA) Block** – For single-port PLL types, the Phase Shifter provides two outputs corresponding to a phase shift of 0º and 90º. For two-port PLL types, the Phase Shifter has two modes: Divide-by-4 mode and Divide-by-7 mode. In Divide-by-4 mode, the output of the B port can be shifted either 0º or 90º with regard to A port outputs. In Divide-by-7 mode, the B port output frequency can be set to have a frequency ratio of 3.5:1 or 7:1 with regard to the port A output frequency.
		Using a feedback path external to the PLL	The feedback path traverses through FPGA routing (external to the PLL) followed by the FDA block. In effect, two delay controls are available – the external path for coarse adjustment and the FDA block for fine delay adjustment.
Fine Delay Adjustment Settings			
Do you want to dynamically control the delay of the Fine Delay Adjustment block?	Enabled only when Compensation mode or external Feedback mode is selected. The delay contributed by the FDA block can be fixed or controlled dynamically during FPGA operation. If fixed, it is necessary to provide a number (n) in the range of 0-15 to specify the delay contributed to the feedback path. For further details, see "Fine Delay Adjustment (FDA)" on page 7.	Yes	
		No	Enter a value in the range 0-15.
Phase Shift Specification			
Specify the phase shift for the PLL output	Enabled only when delay compensation using the phase shifter is selected. Gives a phase shift of 0º and 90º to the output clock.	0º	
		90º	
Target Application[1]			
LVDS Display Panel	Two different frequencies can be observed on different ports (A and B).	3.5:1	The frequency of port B is 3.5x of Port A.
		7:1	The frequency of Port B is 7x of Port A.
DDR Application	The signal on Port B can be phase shifted by 90º with respect to signal A.	0º	
		90º	

Table 7-8. PLL Configuration Tool User Parameters (Continued)

User Parameter	Description	Range	Description
Additional Delay Settings			
Do you wish to specify additional delay on the PLL outputs?	In addition to Fine Delay Adjustment in the feedback path, the user can specify additional delay on the PLL output ports.	Yes No	The delay contributed by the delay block can be fixed or controlled dynamically during FPGA operation. If fixed, it is necessary to provide a number (n) in the range 0-15 to specify the delay contributed to the feedback path. The delay for a setting 'n' is calculated as follows : FDA delay = (n+1)*0.15 ps, range of n = 0 to 15. *Note: This additional delay is applied on the output of single-port PLL and port A of two-port PLL Types.*
PLL Input/output Frequency			
Input Frequency (MHz)	Specify input frequency. Refer to the iCE40 Family Data Sheet for the input range.		
Output Frequency (MHz)	Specify desired output frequency. Refer the iCE40 Family Data Sheet for the output frequency range.		
Others			
Create a LOCK output port	A lock signal is provided to indicate that the PLL has locked on to the incoming signal. Lock asserts High to indicate that the PLL has achieved frequency lock with a good phase lock.		
Create a BYPASS port that will bypass the PLL reference clock to the PLL output port	A BYPASS signal is provided which both powers-down the PLL core and bypasses it such that the PLL output tracks the input reference frequency.		
Low Power Mode	A control is provided to dynamically put the PLL into a lower power mode through the iCEGate feature. The iCEGate feature latches the PLL output signal and prevents unnecessary toggling.	Enable latching of PLL clock (iCEGate)	Dynamically controls power by enabling the signal LATCHINPUTVALUE. Refer to the section "Low Power Mode" on page 8.
		Enable latching of PLL source clock	Default setting

1. Enabled when the Number of Global Networks to be Driven by the PLL Outputs option is set to '2'.

PLL Module Generator Output

The PLL module generator generates two HDL files:

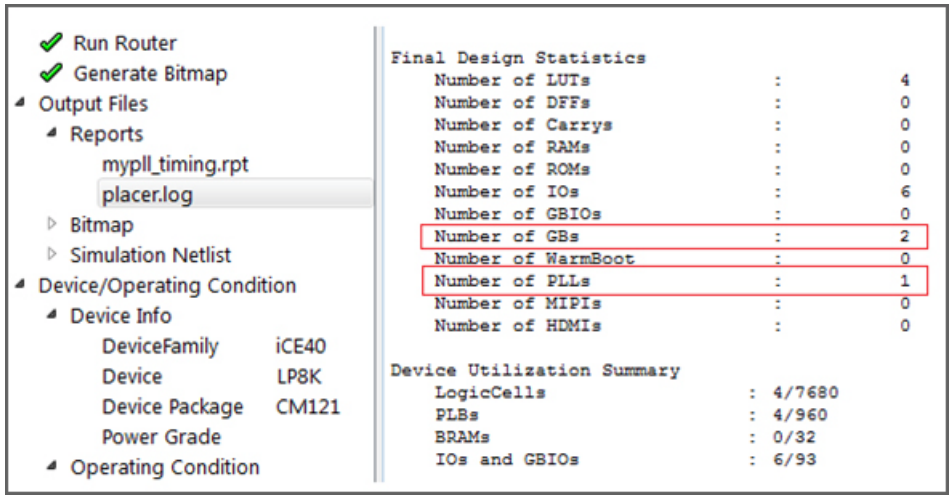
- <module_name>_inst.v
- <module_name>.v

The <module_name>_inst.v is the instantiation template to be used in the custom top level design. The <module_name>.v contains the PLL software macro with the required attributes and signal values, calculated based on the inputs to the GUI.

iCEcube2 Design Software Report File

The placer.log file (Final Design Statistics section) shows the use of PLLs and GBUFs (global buffers) along with other design elements of the iCE40 device. Note that when GBUF is instantiated in the RTL to connect to a PIO, an extra slice will be utilized for connection.

Figure 7-11. Report File Showing the Use of PLLs and Global Buffers



Hardware Design Considerations

Analog Power Supply Filter for PLL

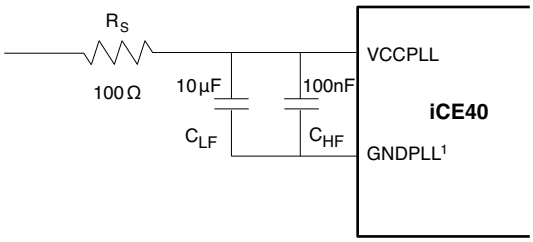
The iCE40 sysCLOCK PLL contains some analog blocks, so the PLL requires a separate power and ground that is quiet and stable to reduce the output clock jitter of the PLL.

Typically, an R-C filter as shown in Figure 7-12 is used as a power supply filter on the PLL power and ground pins. The series resistor (R_S) limits the voltage drop across the filter. A high frequency non-electrolytic capacitor (C_{HF}) is placed in parallel with a lower frequency electrolytic capacitor (C_{LF}). C_{HF} is used to attenuate high frequency components while C_{LF} is used for low frequency cut-off.

Board layout around the high frequency capacitor and the path to the pads is critical. The PLL power (V_{CCPLL}) path must be a single wire from the FPGA pin to the high frequency capacitor (C_{HF}), then to the low frequency capacitor (C_{LF}), through the series resistor (R_S) and then to board power $V_{CC.}$ The distance from the FPGA pin to the high frequency capacitor should be as short as possible. Similarly, the PLL Ground (GNDPLL) path should be from the FPGA pin to the high frequency capacitor (C_{HF}) and then to the low frequency capacitor (C_{LF}), with the distance from the FPGA pin to the C_{HF} being as short as possible.

The sysCLOCK PLL has the DC ground connection made on the FPGA, so the external PLL ground connection (GNDPLL) must NOT be connected to the board's ground. Figure 7-12 also includes sample values for the components that make up the PLL power supply filter.

Figure 7-12. Power Supply Filter for VCCPLL and GNDPLL



1. GNDPLL should not beconnected to the board's ground

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
September 2012	01.0	Initial release.

iCE40 Hardware Checklist

September 2012 Technical Note TN1252

Introduction

When designing complex hardware using the iCE40™ device, designers must pay special attention to critical hardware configuration requirements. This technical note steps through these critical hardware requirements related to the iCE40 device. This document does not provide detailed step-by-step instructions but gives a high-level summary checklist to assist in the design process.

The iCE40 ultra-low power, non-volatile devices are available in two versions – LP series for low power applications and HX series for high performance applications.

This technical note assumes that the reader is familiar with the iCE40 device features as described in the iCE40 Family Data Sheet and the technical notes included in the iCE40 Handbook. The critical hardware areas covered in this technical note include:

- Power supplies as they relate to the supply rails and how to connect them to the PCB and the associated system
- Configuration and how to connect the configuration mode selection
- Device I/O interface and critical signals

Power Supply

The VCC (core supply voltage) VCCIO_2, SPI_VCC and VPP_2V5 determine the iCE40 device's stable condition. These supplies need to be at a valid and stable level before the device can become operational. Refer to the iCE40 Family Data Sheet for voltage requirements.

Table 8-1. Power Supply Description and Voltage Levels

Supply	Voltage (Nominal Value)	Description
VCC	1.20V (LP devices 1.0V or 1.2V)	Core supply voltage
VCCIO_X	1.5V to 3.3V	Power supply for I/O banks
VPP_2V5	2.5V	NVCM programming and operating supply voltage
VPP_FAST	Leave unconnected	Optional fast NVCM programming supply
SPI_VCC	1.8V to 3.3V	SPI interface supply voltage
VCCPLL[1, 2]	1.2V	Analog voltage supply to Phase Locked Loop (PLL)

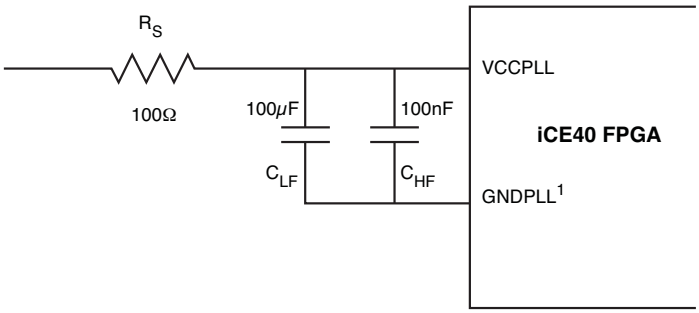
1. VCCPLL must be tied to VCC when PLL is not used.
2. External power supply filter required for VCCPLL and GNDPLL.

Analog Power Supply Filter for PLL

The iCE40 sysCLOCK™ PLL contains analog blocks, so the PLL requires a separate power and ground that is quiet and stable to reduce the output clock jitter of the PLL. The sysCLOCK PLL has the DC ground connection made on the FPGA, so the external PLL ground connection (GNDPLL) must NOT be connected to the board's ground. Figure 8-1 also includes sample values for the components that make up the PLL power supply filter.

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Figure 8-1. Isolating PLL Supplies



1. Note that GNDPLL should not be connected to the board's ground.

Configuration Considerations

The iCE40 device contains two types of memory, CRAM (Configuration RAM) and NVCM (Non-volatile Configuration Memory). CRAM memory contains the active configuration. The NVCM provides on-chip storage of configuration data. It is one-time programmable and is recommended for mass-production.

For more information, refer to TN1248, iCE40 Programming and Configuration.

The configuration and programming of the iCE40 device is a SPI port, both in Master and Slave modes. In Master SPI mode, the device configures its CRAM from an external SPI Flash connected to it. In Slave mode, the device can be configured or programmed using the Lattice Diamond® Programmer or embedded processor.

The SPI_SS_B determines if the iCE40 CRAM is configured from an external SPI (SPI_SS_B=0) or from the NVCM (SPI_SS_B=1). This pin is sampled after Power-on-Reset (POR) is released or CRESET_B is held low or toggled (High-Low-High).

Table 8-2. Configuration Pins

Pin Name	Function	Direction	External Termination	Notes
CRESET_B	Configuration Reset input, active low.	Input	10 KOhm pull-up to VCCIO_2.	A low on CRESET_B delay's configuration.
CDONE	Configuration Done output from iCE40.	Output	Optional 10 KOhm pull-up to VCCIO_2.	
SPI_VCC	SPI interface supply voltage.	Supply		
SPI_SI	SPI serial input to the iCE40, in both Master and Slave modes.	Input		Released to user I/O after configuration.
SPI_SO	SPI serial output from the iCE40, in both Master and Slave modes.	Output		Released to user I/O after configuration.
SPI_SCK	SPI clock	Input/Output	10 KOhm pull-up to VCCIO_2 recommended.	Direction based on Master or Slave modes. Released to user I/O after configuration.

Table 8-2. Configuration Pins (Continued)

Pin Name	Function	Direction	External Termination	Notes
SPI_SS_B	Chip select	Input (Slave mode)/ Output (Master mode)	10 KOhm pull-up to VCCIO_2 in Master mode and a 10 KOhm pull-down in Slave mode.	Refer to TN1248, iCE40 Programming and Configuration, for more details.
TRST_B	Scan Test operation	Input	Can be connected to GND if Scan Test is not used.	Keep Low during normal operation; High for Scan Test operation.

SPI Flash Requirement in Master SPI Mode

Users are free to select any industry standard SPI Flash. The SPI Flash must support the 0x0B Fast Read command, using a 24-bit start address with eight dummy bits before the PROM provides first data. Refer to TN1248, iCE40 Programming and Configuration, for additional information.

LVDS Pin Assignments

The differential inputs are supported only by Bank 3; however, differential outputs are supported in all banks.

Checklist

Table 8-3. iCE40 Hardware Checklist

	iCE40 Hardware Checklist Item	OK	N/A
1	Power Supply		
1.1	Core supply VCC at 1.2V		
1.2	I/O power supply VCCIO 0-3 at 1.5V to 3.3V		
1.3	SPI_VCC at 1.8V to 3.3V		
1.4	VCCPLL pulled to VCC even if PLL not used		
1.5	Power supply filter for VCCPLL and GNDPLL		
1.6	GNDPLL must NOT be connected to the board		
2	Power-on-Reset (POR) inputs		
2.1	VCC		
2.2	SPI_VCC		
2.3	VCCIO_0-3		
2.4	VPP_2V5		
	VPP_FAST		
3	Configuration		
3.1	Configuration mode based on SPI_SS_B		
3.2	Pull-up on CRESET_B,CDONE pin		
3.3	TRST_B is kept low for normal operation		
4	I/O pin assignment		
4.1	LVDS pin assignment considerations		

Technical Support Assistance

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

Revision History

Date	Version	Change Summary
September 2012	01.0	Initial release.
September 2012	01.1	LVDS Pin Assignments text section – corrected description of differential input and output support.

Using Differential I/O (LVDS, Sub-LVDS) in iCE40 Devices

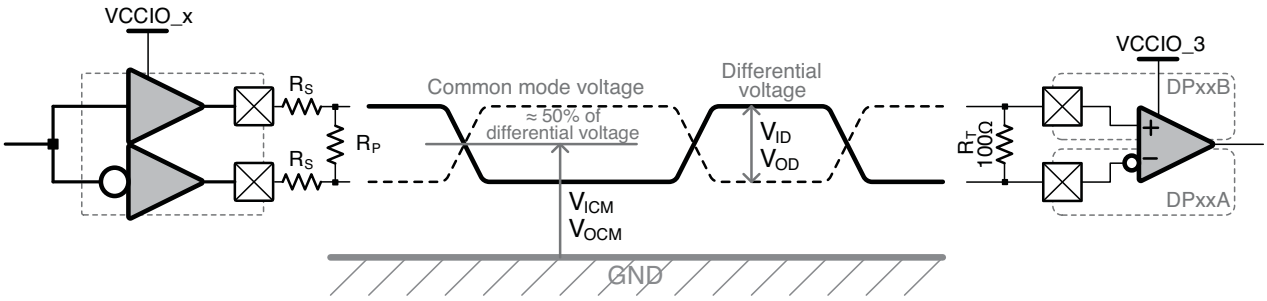
September 2012 **Technical Note TN1253**

Introduction

Differential I/O standards are popular in a variety of consumer applications, especially those that require high-speed data transfers such as graphic display drivers and camera interfaces. In these systems, multiple signals are typically combined onto a smaller number of time-division-multiplexed high-speed, differential serial channels.

Differential signals require two Programmable I/O (PIO) pins, working as a pair or a channel, as shown in Figure 9-1. One side of the pair represents the true polarity of the signal while the other side of the pair represents the opposite polarity. The resulting logic value is the difference between the two sides of the signal pair.

Figure 9-1. Differential Signaling Electrical Parameters



The key electrical parameters are the common mode voltage and the differential voltage. For iCE40™ applications, the common mode voltage is essentially half the I/O Bank supply voltage. The differential voltage depends on the values of the external compensation resistors, discussed in "LVDS and Sub-LVDS Termination" on page 3.

Table 9-1. Representative Electrical Characteristics of Various Differential I/O Standards

		LVDS FPD-Link	Sub-LVDS FlatLink3G	Units
iCE40 Support		Inputs and Outputs	Inputs and Outputs	—
Output Frequency	Max.	525	TBD	Mbps
Input Frequency	Max.	480	TBD	Mbps
VCCIO	Nom./Typ.	2.5	1.8	V
V_{OD}	Min.	250	100	mV
	Nom./Typ.	350	150	
	Max.	450	200	
V_{OS}, V_{CM}	Min.	1.125	0.8	V
	Nom.	1.25	0.9	
	Max.	1.375	1.0	
V_{IDTH}	Min.	250	70	mV
	Max.	450	200	
V_{ICM}	Min.		0.6	V
	Max.		1.2	

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.

Differential signaling provides many advantages. In the examples discussed here, all the differential I/O standards have reduced voltage swing, which allows faster switching speeds and potentially higher bandwidth. Reduced voltage swings also mean less dynamic power consumption and reduced electromagnetic interference (EMI).

Differential switching provides **improved noise immunity** and **reduces duty-cycle distortion** caused the differences in rise- and fall-time by the output driver.

The higher potential switching speeds of differential I/O allows data to be multiplexed onto a **reduced number of wires** at a much higher data rate per line. The reduced number of wires reduces system cost and in some cases simplifies the system design. The internal phase-locked loop (PLL) available in iCE40 FPGAs provides convenient on-chip clock multiplication or division to support such applications.

Differential Outputs

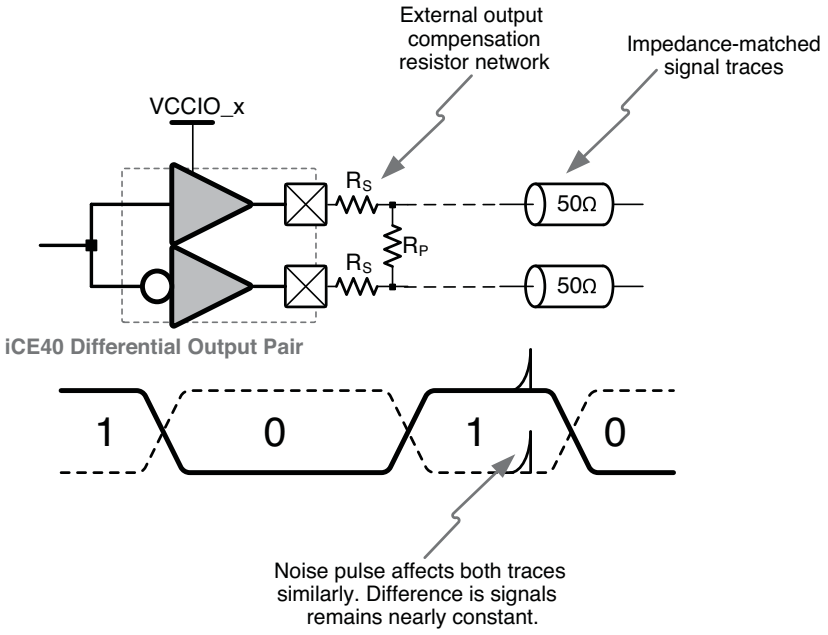
For some differential I/O standards, such as LVDS, the output driver is actually a current source. On iCE40 FPGAs, however, differential outputs are constructed using a pair of single-ended PIO pins as shown in Figure 9-3, and an external resistor network consisting of three resistors. Because differential outputs are built from two single-ended LVCMOS outputs, differential outputs are available in any I/O bank.

The two FPGA outputs must be part of the same I/O tile as indicated in the iCE40 data sheet. The pair choice also depends on the chosen device package as not all I/O tile pairs are bonded out in all packages. Consult the package pinout sections of the iCE40 device data sheets for additional information.

PIO outputs are available in all I/O banks and support data rates up to 525 Mbps.

Each differential I/O output pair requires a three-resistor termination network to adjust output characteristics to match those for the specific differential I/O standard. The output characteristics depend on the values of the parallel resistor (R_P) and series resistors (R_S). These resistors should be surface mounted as close as possible to the FPGA output pins.

Figure 9-2. Differential Output Pair



Differential Inputs

Differential inputs are only supported in I/O Bank 3. The maximum number of differential input pairs per device is shown in Table 9-1 but the actual number available depends on the specific package used.

Table 9-2. Maximum Differential Input Pairs Available on iCE40 FPGAs

	iCE40HX1K	iCE40LP1K	iCE40HX4K	iCE40LP4K	iCE40LP8K	iCE40HX8K
Maximum Differential Input Pairs	11	12	12	20	23	26

Differential inputs require specific PIO pin pairs as listed in the iCE40 data sheet. Each differential input pair consists of one pin labeled DPxxA and another labeled DPxxB, where “xx” represents the differential pair number. Both pins must be in the same differential pair.

Connect the positive or true polarity side of the differential pair to the DPxxA input and the negative or complementary side of the pair to the DPxxB input. If it is easier to route the differential pair, the input pins can be swapped, which produces an inverted input value. The inverted input value can subsequently be inverted by logic within the FPGA.

An input termination resistor must be connected between the DPxxA and DPxxB pins to generate the differential signal. The resistor’s value must be twice the trace impedance, as described in the following section.

Typically, the resulting signal pair is routed on the printed circuit board (PCB) using controlled impedance and delay matching.

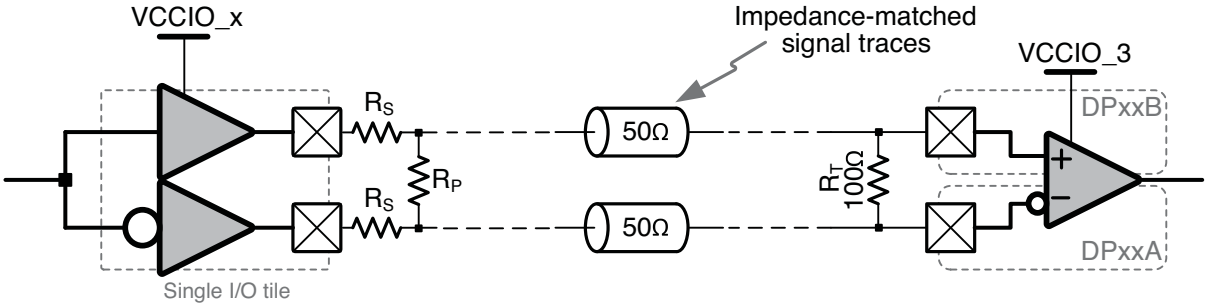
LVDS and Sub-LVDS Termination

LVDS and Sub-LVDS inputs require external compensation and termination resistors for proper operation, as shown in Figure 9-4. A termination resistor, R_T, between the positive and negative inputs at the receiver forms a current loop. The current across this resistor generates the voltage detected by the receiver’s differential input comparator.

Similarly, iCE40 LVDS and Sub-VLDS outputs require an external resistor network, consisting of two series resistors, R_S, and a parallel resistor, R_P. This resistor network adjusts the FPGA’s output driver to provide the necessary current and voltage characteristic s required by the specification.

The signals are routed with matched trace impedance, Z_0, on the printed circuit board, typically with 50 Ω impedance.

Figure 9-3. iCE40 LVDS or Sub-LVDS Differential I/O Channel



The resistor values for the compensation network are described below. These equations are also provided in the Differential I/O spreadsheet. The variables are defined and described in Table 9-3.

Input Termination Resistor (R_T)

$$R_T = 2 \times Z_0 \quad (1)$$

Output Parallel Resistor (R_P)

$$R_P = 2 \times \left(\frac{Z_0 \times VCCIO}{VCCIO - (2 \times V_{OD})} \right) \quad (2)$$

Output Series Resistor (R_S)

$$R_S = \left(\frac{Z_0 \times \frac{R_P}{2}}{\frac{R_P}{2} - Z_0} \right) - R_{OUTPUT} \quad (3)$$

Table 9-3. Compensation Resistor Equation Variables

Variable	Description
Z_0	Characteristic impedance of the printed circuit board trace; data sheet values assume 50 Ω traces
VCCIO	I/O Bank supply voltage, nominal, volts
V_{OD}	Differential output voltage swing, nominal, volts
R_{OUTPUT}	iCE40 output source resistance, 30 Ω
R_P	LVDS/Sub-LVDS output source compensation parallel resistor
R_S	LVDS/Sub-LVDS output source compensation series resistor
R_T	LVDS/Sub-LVDS input termination resistor

Using the Companion iCE40 Differential I/O Calculator Spreadsheet

The iCE40 data sheet recommends specific values for LVDS and Sub-LVDS differential outputs but also assumes that the differential signals are routed with 50 Ω characteristic impedance (Z_0). This may not be possible in all applications. Likewise, the system may require some other slightly different I/O standard.

This technical note includes a companion spreadsheet, available for download from the Lattice website and shown in Figure 9-4, to calculate resistor values for non-standard conditions. The values in Figure 9-4 are the default conditions for the LVDS I/O standard. For other standards, simply modify the VCCIO voltage, the differential output voltage, V_{OD}, and the characteristic impedance of the printed circuit board traces, Z_0.

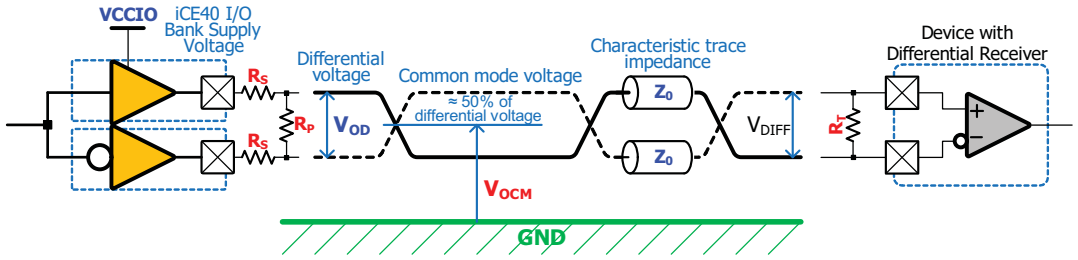
Figure 9-4. iCE40 Differential I/O Calculator Spreadsheet

iCE40™ Differential Output Calculator (Version 2.3, 28-AUG-2012)

	I/O Bank Supply	Diff. Output Voltage	Trace Impedance	Common Mode Output Voltage	Source Series Resistor	Source Parallel Resistor	Termination Resistor	Resistor Network Current	Termination Current	Actual Differential Voltage
Symbol	VCCIO	V_{OD}	Z_0	V_{OCM}	R_S	R_P	R_T	I_{RES}	I_{RT}	V_{DIFF}
Value	2.5	0.35	50	1.25	150	140	100	6.0	3.5	0.349
Units	volts	volts	ohm	volts	ohm	ohm	ohm	mA	mA	volts

Companion to Technical Note **TN1253: Using Differential I/O (LVDS, SubLVDS) in iCE40 mobileFPGAs**

Directions: Enter the values for **VCCIO**, V_{OD}, and Z_0. The resulting resistor values appear under R_S, R_P, and R_T. The V_{OCM} voltage is aslo calculated.



1.) The R_S and R_P resistors should be surface mounted as close to the iCE40 mobileFPGA output balls/pins as possible.
2.) The termination resistor, R_T, should be as close to the recieving device's differential inputs as possible.
3.) The actual differential voltage, V_{DIFF}, may vary slightly from differential output voltage due to rounding of resistor values.

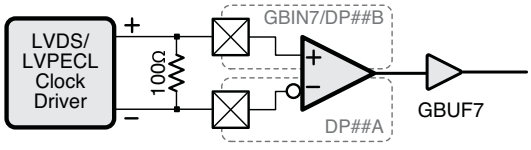
The spreadsheet automatically calculates the common mode output voltage, V_{OCM}, which is half of the VCCIO supply voltage. The spreadsheet also automatically calculates the values for the resistor network.

Finally, the spreadsheet also calculates the current draw through the resistor network and for the termination resistor. The spreadsheet rounds the resistor values to the nearest 10 Ω. Consequently, the spreadsheet also calculates the actual differential voltage based on the specified resistor values.

Differential Clock Input

iCE40 FPGAs have eight global buffers for distributing clocks or other high fanout signals. Global buffer GBUF7, shown in Figure 9-5, is specifically designed to accept a differential clock input on the associated GBIN7/DPxxB, DPxxA differential input pair, which is part of I/O Bank 3. Connect an external 100 Ω termination resistor across the input pair. When VCCIO_3 is 2.5V, this global buffer input accepts either LVDS or LVPECL clock inputs.

Figure 9-5. LVDS or LVPECL Clock Input



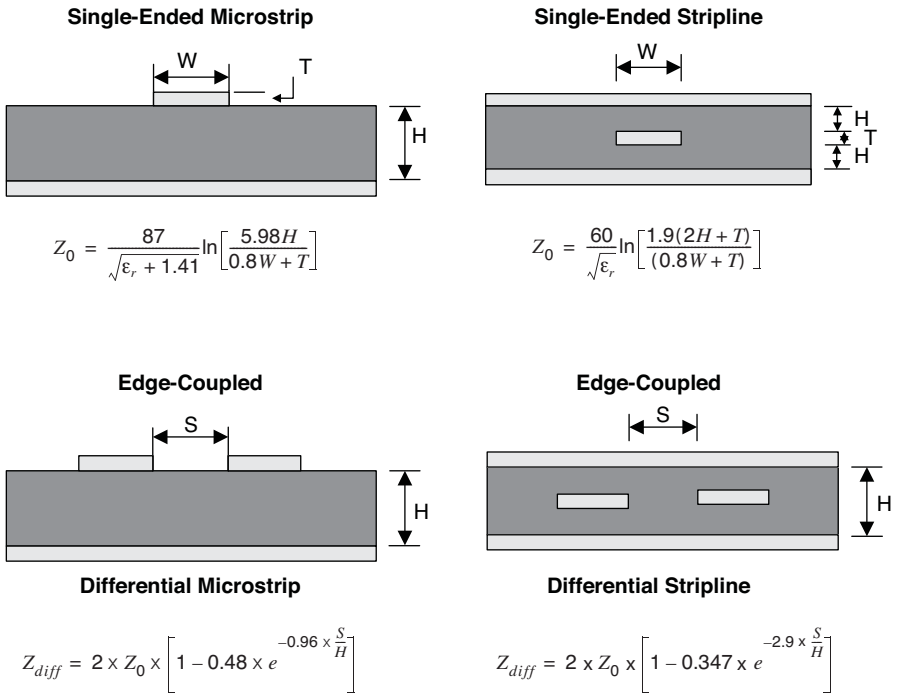
iCE40 Differential Signaling Board Layout Requirements

Figure 9-6 depicts several transmission line structures commonly used in printed circuit boards (PCBs). Each structure consists of a signal line and a return path with uniform cross-section along its length. The structure's dimensions along with the dielectric material properties determine the characteristic impedance of the transmission line. Figure 9-6 shows examples of edge-coupled microstrips, and edge-coupled or broad-side-coupled striplines.

To maintain constant differential impedance along the length, maintain uniform trace width and spacing, including good symmetry between the two lines.

For differential outputs, place the surface-mounted R_P and R_S resistors as close to the package balls as possible. Similarly, place the 100 Ω termination resistor, R_T, as close as possible to the differential input pair.

Figure 9-6. Controlled Impedance Transmission Lines



Typical PC board trace impedance is Z_0 = 50 Ohms. For a single-ended microstrip, the trace impedance is calculated by using the following equation:

Single-ended microstrip trace impedance

$$Z_0 = \frac{87}{\sqrt{\varepsilon_r + 1.41}} \ln\left[\frac{5.98H}{0.8W + T}\right] \quad (4)$$

Single-ended stripline trace impedance

$$Z_0 = \frac{60}{\sqrt{\varepsilon_r}} \ln\left[\frac{1.9(2H + T)}{(0.8W + T)}\right] \quad (5)$$

For an LVDS pair, differential impedance can be determined by using the following equations for differential microstrip and differential stripline:

Differential impedance for differential microstrip

$$Z_{diff} = 2 \times Z_0 \times \left[1 - 0.48 \times e^{-0.96 \times \frac{S}{H}}\right] \quad (6)$$

Differential impedance for differential stripline

$$Z_{diff} = 2 \times Z_0 \times \left[1 - 0.347 \times e^{-2.9 \times \frac{S}{H}}\right] \quad (7)$$

Because the coupling of two traces can lower the effective impedance, use 60 Ω design rules to achieve a differential impedance of approximately 100 ohms.

Layout Recommendations to Minimize Reflection

Skew delay is introduced if the trace lengths between the signals in the differential pair are not similar. With differing trace lengths, the signals on each side of the differential pair arrive at slightly different times and reflect off the receiver termination, creating a common-mode noise source on the transmission line.

Common-mode noise degrades the receiver's eye diagram, reduces signal integrity, and creates crosstalk between neighboring signals on the board. To minimize reflections due to unmatched trace lengths, consider the following guidelines:

- Match the length of each signal within the differential signal pair to within 20 mils.
- Minimize turns and vias or feed-throughs. Route differential pairs as straight as possible from point-to-point. Do not use 90-degree turns when routing differential pairs. Instead, use 45-degree bevels or rounded curves.
- Minimize vias on or near differential trace lines as these may create additional impedance discontinuities that will increase reflections at the receiver. If vias are required, place them as close to the receiver as possible.
- Use controlled impedance PCB traces. That is, control trace spacing, width, and thickness using stripline or microstrip layout techniques.

EMI and Noise Cancellation

Differential signaling offers tremendous advantages over single-ended signaling because it is less susceptible to noise. In differential signaling, two current carrying conductors are routed together, with the current in one conductor always equal in magnitude but opposite in direction to the other conductor. The result is that both electromagnetic fields cancel each other.

Common-mode noise rejection is another advantage of differential signaling. The receiver ignores any noise that couples equally on both sides of the differential signal, as shown in Figure 9-8.

Figure 9-7. Single-ended Input Retains Signal Noise

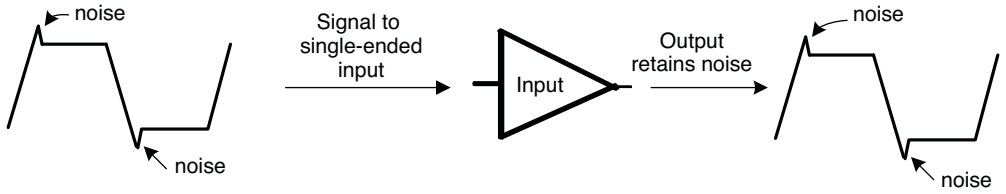
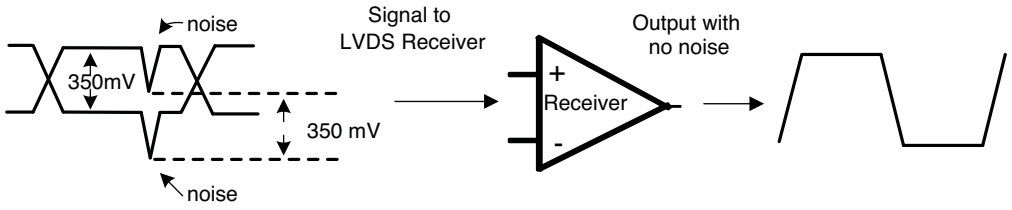


Figure 9-8. Differential Input Eliminates Common-Mode Noise



Reducing EMI Noise

Any skew between differential signals can generate EMI noise on the board. The following are some guidelines to reduce EMI emission onboard:

- Match edge rates and signal skew between differential signals as closely as possible. Different signal rise and fall times and skew between signal pairs create common-mode noise, which generates EMI noise.
- Maintain spacing between differential signal pairs that is less than twice the PCB trace width.

Defining Differential I/O

Single-ended I/O connections are automatically inferred from the top-level VHDL or Verilog circuit description during logic synthesis. However, differential I/O connections must be specifically instantiated using design primitives from the Lattice technology library for iCE40 FPGAs. Table 9-4 lists the specific I/O primitive required by differential I/O type. Documentation on the design primitives is located under the iCEcube2™ installation directory:

C:\SbtTools\doc\SBT_ICE_Technology_Library.pdf

Table 9-4. Differential I/O Types and Required SiliconBlue I/O Primitive

Differential I/O Type	iCE40 Design Primitive	PIN_TYPE	IO_STANDARD
Differential clock input	SB_GB_IO	See "PIN_TYPE Parameter" on page 9.	SB_LVDS_INPUT
Differential input	SB_IO		SB_LVDS_INPUT
Differential output	SB_IO		SB_LVCMOS

The SB_IO and SB_GB_IO primitives also require specific parameter settings. Differential inputs always require that IO_STANDARD be set to SB_LVDS_INPUT. Differential outputs typically require that the IO_STANDARD parameter be set to SB_LVCMOS.

SB_IO Primitive

Table 9-5 lists the signal ports for the SB_IO primitive, which describes one of the Programmable I/O (PIO) pins on an iCE40 FPGA. The table also shows the signal direction for each port, relative to the PIO pin (the SB_IO primitive). These same signals also appear on the SB_GB_IO primitive, which describes a global buffer input.

Table 9-5. Port Names, Signal Direction, and Description for SB_IO (SB_GB_IO) Primitive

Port Name	Direction	Description
PACKAGE_PIN	I/O	Connection to top-level input, output, or bidirectional signal port.
LATCH_INPUT_VALUE	Input	iCEgate latch input. When High, hold the last pad value. Used for power reduction in some PIN_TYPE modes. There is one control input per I/O Bank. 0 = Input data flows freely 1 = Last data value on pad held constant to save power
CLOCK_ENABLE	Input	Clock enable input, shared connection to all flip-flops within the SB_IO primitive. If this port is left unconnected, automatically tied High. 0 = Flip-flops hold their current value 1 = Flip-flops accept new data on the active clock edge
INPUT_CLOCK	Input	Clock for all input flip-flops. If this port is left connected, automatically tied Low.
OUTPUT_CLOCK	Input	Clock for all output flip-flops. If this port is left connected, automatically tied Low.
OUTPUT_ENABLE	Input	Enables the output buffer. 0 = Output disabled, pad is high-impedance (Hi-Z) 1 = Output enabled, actively driving
D_OUT_0	Input	Data output. For DDR output modes, this is the value clocked out on the rising edge of the OUTPUT_CLOCK.
D_OUT_1	Input	Data output used in DDR output modes. This is the value clocked out on the falling edge of the OUTPUT_CLOCK.
D_IN_0	Output	Data input. For DDR input modes, this is the value clocked into the device on the rising edge of the INPUT_CLOCK.
D_IN_1	Output	For DDR input modes, this is the value clocked into the device on the falling edge of the INPUT_CLOCK.

SB_GB_IO Primitive

Global buffer inputs provide a direct connection from a PIO pin to an associated global buffer. This connection can be instantiated using an SB_GB_IO primitive. An SB_GB_IO primitive has all the ports for an SB_IO primitive,

shown in Table 9-5, plus the additional connection shown in Table 9-6. Global Buffer Input 7 (GBIN7) is the only one that supports differential clock inputs. See “Differential Clock Input” on page 5 for more information.

Table 9-6. Additional Port Names on SB_GB_IO Primitive

Port Name	Direction	Description
GLOBAL_BUFFER_OUTPUT	Output	Output from associated Global Buffer. This output is controlled by the iCE-gate latch if a control signal is connected to the iCEgate latch input (LATCH_INPUT_VALUE).

PIN_TYPE Parameter

The PIN_TYPE parameter defines the structure and the functionality of any instantiated SB_IO primitive. PIN_TYPE is a six-bit binary value. The upper four bits, PIN_TYPE[5:2], define the output structure while the lower two bits, PIN_TYPE[1:0] define the input structure. Both fields are required, but operate independently.

Global buffer inputs, defined using the SB_GB_IO primitive, are also full-featured PIO pins. However, if only the GLOBAL_BUFFER_OUTPUT is connected on the SB_GB_IO primitive, then the PIN_TYPE parameter has no real effect.

Output Field (PIN_TYPE[5:2])

Table 9-7 shows the various PIN_TYPE definitions for the output side an SB_IO or SB_GB_IO primitive.

Table 9-7. Output Structures and PIN_TYPE Values

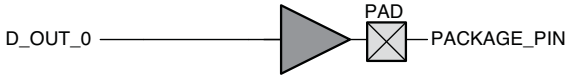
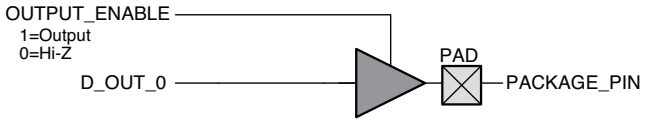
Style	Mnemonic (Diagram)	PIN_TYPE[5:2]			
None (output disabled)	PIN_NO_OUTPUT 	0	0	0	0
Non-registered Output	PIN_OUTPUT 	0	1	1	0
	PIN_OUTPUT_TRISTATE 	1	0	1	0

Table 9-7. Output Structures and PIN_TYPE Values (Continued)

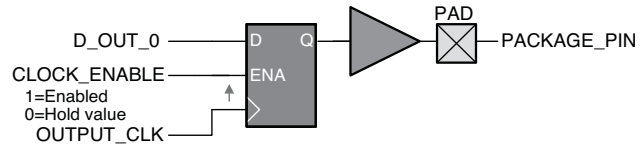
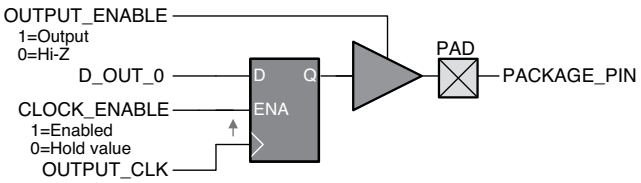
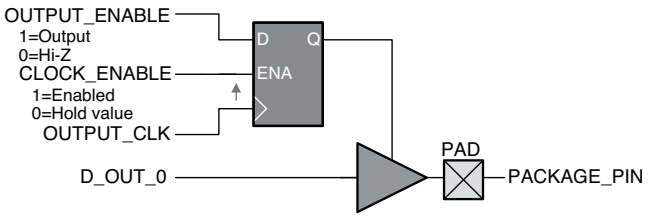
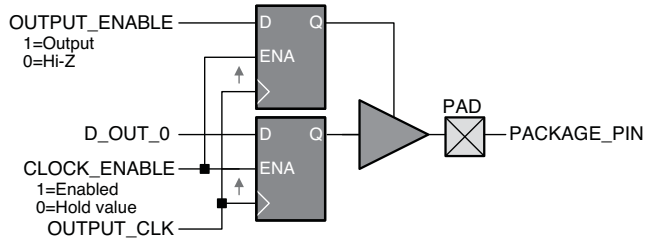
Style	Mnemonic (Diagram)	PIN_TYPE[5:2]			
Registered Outputs	PIN_OUTPUT_REGISTERED 	0	1	0	1
	PIN_OUTPUT_REGISTERED_ENABLE 	1	0	0	1
	PIN_OUTPUT_ENABLE_REGISTERED 	1	1	1	0
	PIN_OUTPUT_REGISTERED_ENABLE_REGISTERED 	1	1	0	1

Table 9-7. Output Structures and PIN_TYPE Values (Continued)

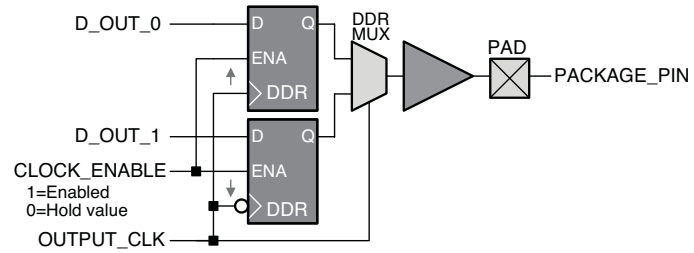
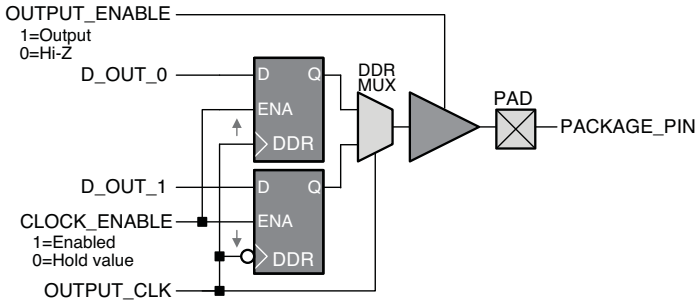
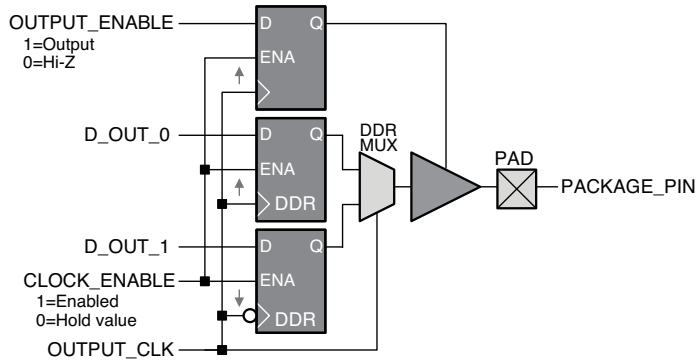
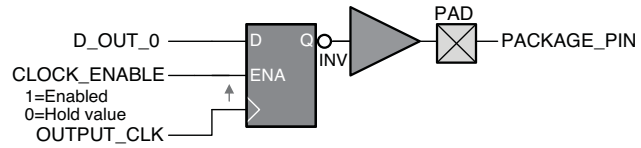
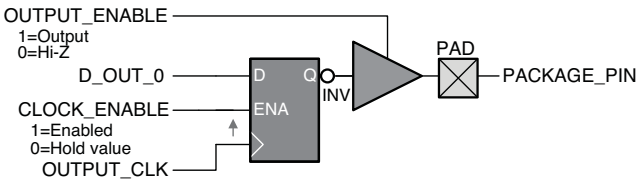
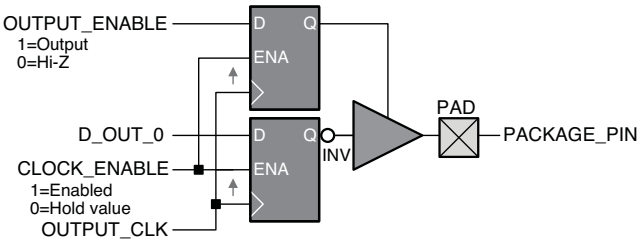
Style	Mnemonic (Diagram)	PIN_TYPE[5:2]			
Double Data Rage (DDR) Output	PIN_OUTPUT_DDR 	0	1	0	0
	PIN_OUTPUT_DDR_ENABLE 	1	0	0	0
	PIN_OUTPUT_DDR_ENABLE_REGISTERED 	1	1	0	0

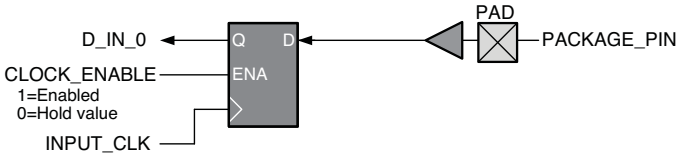
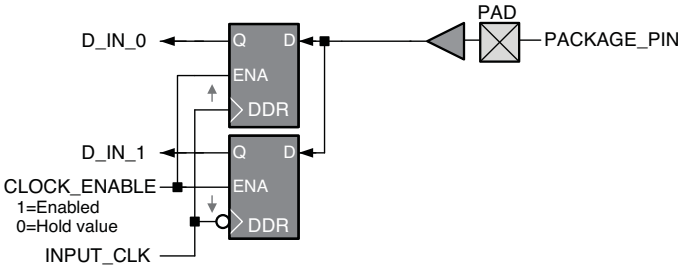
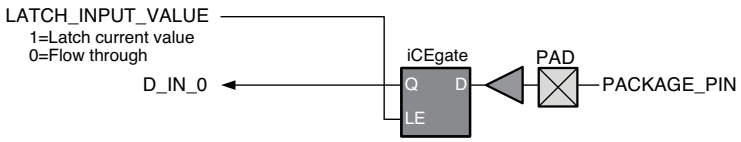
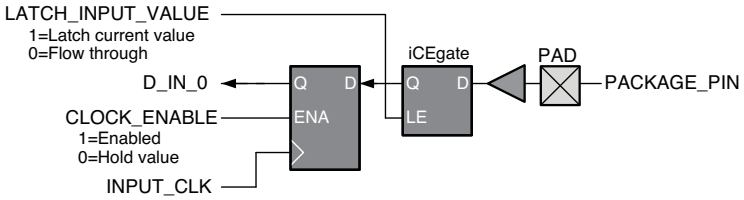
Table 9-7. Output Structures and PIN_TYPE Values (Continued)

Style	Mnemonic (Diagram)	PIN_TYPE[5:2]			
Registered Output, Inverted	PIN_OUTPUT_REGISTERED_INVERTED 	0	1	1	1
	PIN_OUTPUT_REGISTERED_ENABLE_INVERTED 	1	0	1	1
	PIN_OUTPUT_REGISTERED_ENABLE_REGISTERED_INVERTED 	1	1	1	1

Input Field (PIN_TYPE[1:0])

Table 9-8 shows the various PIN_TYPE definitions for the input side of an SB_IO or SB_GB_IO primitive.

Table 9-8. Input Structures and PIN_TYPE Values

Style	Mnemonic (Diagram)	PIN_TYPE[1:0]	
Direct	PIN_INPUT 	0	1
Registered	PIN_INPUT_REGISTERED 	0	0
Double Data Rage (DDR) Output	PIN_INPUT_DDR 	0	0
iCEgate Low-Power Latch	PIN_INPUT_LATCH 	1	1
	PIN_INPUT_REGISTERED_LATCH 	1	0

IO_STANDARD Parameter

Differential inputs or outputs require specific settings for the IO_STANDARD parameter, as summarized in Table 9-9. Regardless of actual electrical requirements (LVDS, Sub-LVDS), the IO_STANDARD parameter on differential inputs must be set to SB_LVDS_INPUT. Again, differential inputs are only available in I/O bank 3. For differential outputs, set IO_STANDARD to SB_LVCMOS. Differential outputs are available in I/O banks 0, 1, 2, or 3.

Table 9-9. IO_STANDARD Settings for Differential I/O

	Differential Inputs (I/O Bank 3 Only)	Differential Outputs (I/O Banks 0, 1, 2, 3)
IO_STANDARD Setting	SB_LVDS_INPUT	SB_LVCMOS

NEG_TRIGGER Parameter

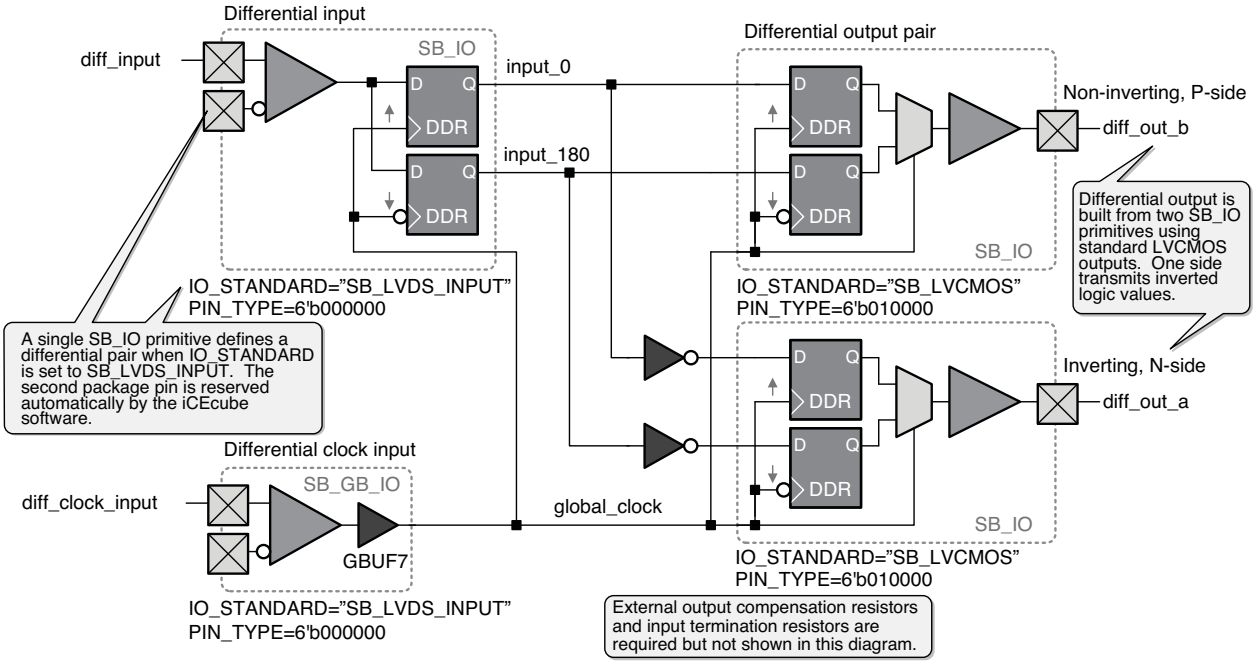
The optional NEG_TRIGGER parameter, when set to '1', inverts the clock polarity within the PIO pin.

HDL Implementation Example

Most applications that use differential I/O do so because they require very high data bandwidth. Consequently, most of these applications use also Double Data Rate (DDR) flip-flops to double the effective data rate at the PIO pin.

The design example shown in Figure 9-9 uses the DDR flip-flops embedded in every PIO pin.

Figure 9-9. Differential I/O Design Example



VHDL Component Declaration

For VHDL implementations, the SB_IO and SB_GB_IO primitives must be declared as components before they are first used. Verilog does not require this declaration.

Differential Clock Input

The following Verilog code snippets demonstrate how to instantiate a differential clock input using an SB_GB_IO primitive. The differential clock input is always connected to Global Buffer GBIN7 and always in I/O Bank. In this example, shown in Figure 9-9, the PIO pin only connects an external differential clock signal to the FPGA's GBUF7 global buffer. The only ports connected on the primitive are the PACKAGE_PIN and the GLOBAL_BUFFER_OUTPUT ports. Consequently, the PIN_TYPE setting for this SB_GB_IO primitive does not actually matter.

Set the IO_STANDARD parameter to "SB_LVDS_INPUT". Doing so also causes the iCEcube2 software to reserve a second pin for the other side of the differential input pair.

An external 100 Ω termination resistor must be connected between the two inputs. The resistor must be physically placed as close as possible to the two package balls.

For VHDL implementations, the SB_GB_IO component must be declared.

The PIN_TYPE for this primitive may be different, depending on whether only the global buffer input is used, or if the data input paths are used, or both. The example shown here is for a dedicated differential clock input.

Verilog

```
// Differential clock input example: Verilog
// IMPORTANT: The PIN_TYPE is different for a regular LVDS input used as a clock.
//       This example is specifically for the differential clock input.
defparam differential_clock_input.PIN_TYPE = 6'b000000 ; // {NO_OUTPUT, PIN_INPUT_REGISTERED}
defparam differential_clock_input.IO_STANDARD = "SB_LVDS_INPUT" ;
SB_GB_IO differential_clock_input (
  .PACKAGE_PIN(diff_clock_input),
  .LATCH_INPUT_VALUE ( ),
  .CLOCK_ENABLE ( ),
  .INPUT_CLK ( ),
  .OUTPUT_CLK ( ),
  .OUTPUT_ENABLE ( ),
  .D_OUT_0 ( ),
  .D_OUT_1 ( ),
  .D_IN_0 ( ),
  .D_IN_1 ( ),
  .GLOBAL_BUFFER_OUTPUT(global_clock) // Global buffer output
);
```

VHDL

Under development.

Differential Input

The following Verilog and VHDL code snippets demonstrate how to instantiate a differential input using an SB_ IO primitive. Differential inputs are always implemented in I/O Bank 3. In this example, shown in Figure 9-9, the differential input also connects to Double Data Rate (DDR) input flip-flops. There is no output connected. Consequently, set the PIN_TYPE parameter to the binary value “000000”, which defines no output (see Table 9-7) and DDR input (see Table 9-8).

Set the IO_STANDARD parameter to “SB_LVDS_INPUT”. Doing so also causes the iCEcube2 software to reserve a second pin for the other side of the differential input pair.

An external 100 Ω termination resistor must be connected between the two inputs. The resistor must be physically placed as close as possible to the two package balls.

For VHDL implementations, the SB_ IO component must be declared.

Verilog

```
// Differential input, DDR data
defparam differential_input.PIN_TYPE = 6'b000000 ; // {NO_OUTPUT, PIN_INPUT_DDR}
defparam differential_input.IO_STANDARD = "SB_LVDS_INPUT" ;
SB_IO differential_input (
  .PACKAGE_PIN(diff_input),
  .LATCH_INPUT_VALUE ( ),
  .CLOCK_ENABLE ( ),
  .INPUT_CLK (global_clock),
  .OUTPUT_CLK ( ),
  .OUTPUT_ENABLE ( ),
  .D_OUT_0 ( ),
  .D_OUT_1 ( ),
  .D_IN_0 (input_0),
  .D_IN_1 (input_180)
);
```

VHDL
Under development.

Differential Output Pair

The following Verilog and VHDL code snippets demonstrate how to instantiate a differential output pair using an SB_ IO primitive. Differential outputs are specified as two separate single-ended outputs. One output provides the non-inverted or P-side of the pair while the other output provides the inverted or N-side of the pair. In this example, shown in Figure 9-9, the differential output also connects to Double Data Rate (DDR) input flip-flops for higher output performance. There is no input connected. Consequently, set the PIN_TYPE parameter to the binary value “010000”, which defines DDR output (see Table) and a benign input (see Table 9-8).

Differential outputs can be placed in any I/O bank, although the pair must be part of an I/O tile, as shown in the iCE40 or iCE40 data sheet. Because of better slew rate control, place lower-speed differential outputs in I/O Banks 0, 1, or 2. Differential I/Os in I/O Bank 3 have the best performance, but also have faster, noisier switching edges. Set the IO_STANDARD parameter to “SB_LVCMOS”.

An external compensation resistor network must be connected between the two inputs. The resistors must be physically placed as close as possible to the two package balls.

For VHDL implementations, the SB_ IO component must be declared.

Verilog

```
// Differential output pair, DDR data
// Non-inverting, P-side of pair
defparam   differential_output_b.PIN_TYPE   =   6'b010000   ;   //   {PIN_OUTPUT_DDR,
PIN_INPUT_REGISTER }
defparam differential_output_b.IO_STANDARD = "SB_LVCMOS" ;
SB_IO differential_output_b (
  .PACKAGE_PIN(diff_output_b),
  .LATCH_INPUT_VALUE ( ),
  .CLOCK_ENABLE ( ),
  .INPUT_CLK ( ),
  .OUTPUT_CLK (global_clock),
  .OUTPUT_ENABLE ( ),
  .D_OUT_0 (input_0),   // Non-inverted
  .D_OUT_1 (input_180),  // Non-inverted
  .D_IN_0 ( ),
  .D_IN_1 ( )
);

// Inverting, N-side of pair
defparam   differential_output_a.PIN_TYPE   =   6'b010000   ;   //   {PIN_OUTPUT_DDR,
PIN_INPUT_REGISTER }
defparam differential_output_a.IO_STANDARD = "SB_LVCMOS" ;
SB_IO differential_output_a (
  .PACKAGE_PIN(diff_output_a),
  .LATCH_INPUT_VALUE ( ),
  .CLOCK_ENABLE ( ),
  .INPUT_CLK ( ),
  .OUTPUT_CLK (global_clock),
  .OUTPUT_ENABLE ( ),
  .D_OUT_0 (~input_0),  // Inverted
  .D_OUT_1 (~input_180), // Inverted
  .D_IN_0 ( ),
  .D_IN_1 ( )
);
```

VHDL
Under development.

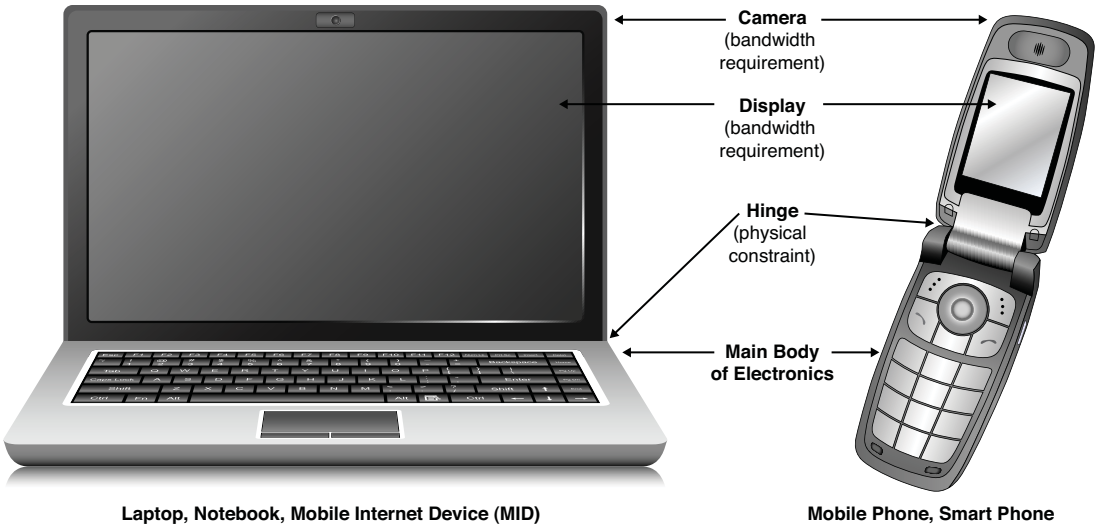
Applications

Applications that benefit most from differential I/O are those with high bandwidth communication requirements such as graphic displays, cameras and imagers, or chip-to-chip interfaces.

While driving such displays using single-ended LVCMOS I/O is possible, portable or hand-held applications place additional physical constraints on a design, as shown in Figure 9-10. Typically, the high bandwidth device—the graphic display or camera—is separate from the main body that holds the majority of the system electronics. The main body and the high-bandwidth device are often mechanically connected by some sort of hinge mechanism. In other applications, the display and camera or cameras are folded into a compact phone, camera, or tablet body. Sending a wide, LVCMOS signal cable bundle across the hinge to the display is simply impractical. Likewise, a custom, wide, flex-cable is prohibitively expensive.

The higher bandwidth possible with differential signaling allows the same data to be transported over fewer electrical connections. Few connections results in a smaller, lower-cost flexible cable. Likewise, the smaller voltage swing results in lower electromagnetic interference (EMI).

Figure 9-10. Portable Devices Benefit from Differential I/O



iCE40 FPGAs offer a broad range of possible solutions for handheld applications, primarily in bridging and format conversion applications.

- Covert RGB data to high-speed differential data and back.
- Connect a processor without an integrated differential interface to a differential display or camera.
- Convert from one high-speed differential interface to another.
- Scale, rotate, and rebroadcast one stream onto another display format or another differential I/O format.

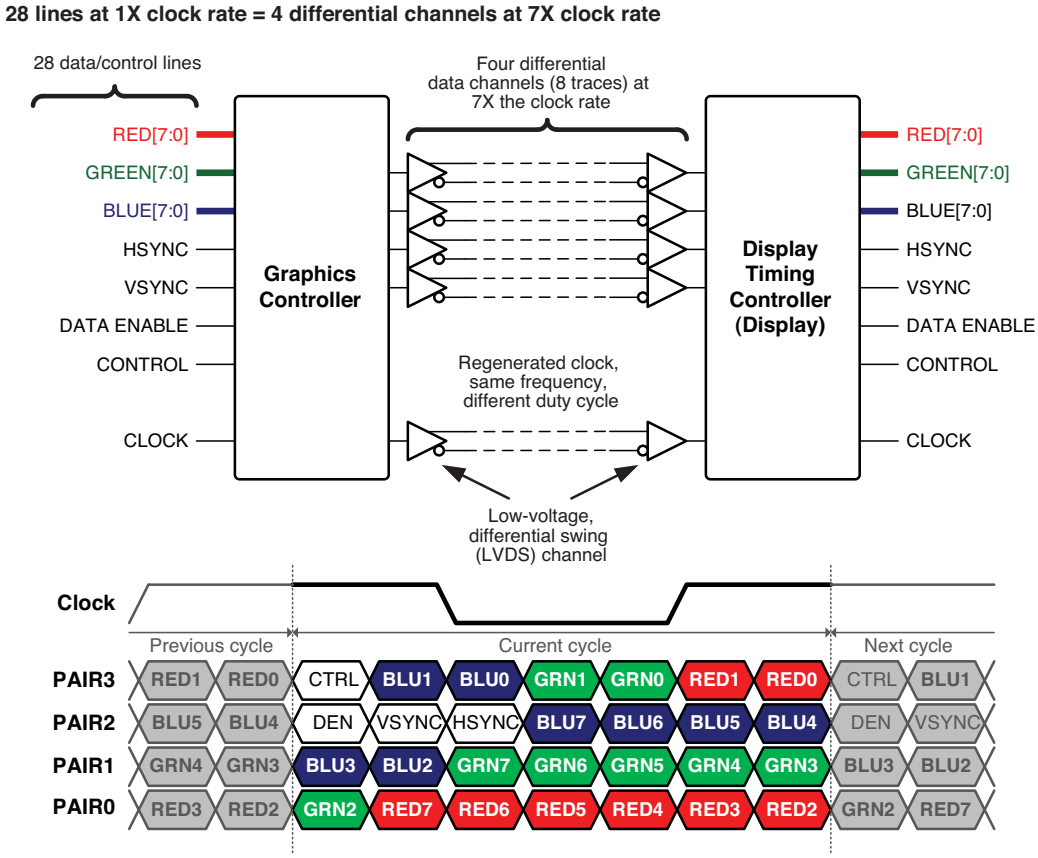
Graphic Displays

Graphic displays demand high data rates, especially high-resolution displays that support a broad color range. In portable or handheld applications, the challenge is to provide a high-bandwidth communications path between the graphics controller and the display that fits with the physical constraints of the hinge or the package body. There are a variety of standard interfaces that leverage differential switching. Perhaps the most widely-used example is Flat Panel Display Link (FDP-Link), shown in Figure 9-11. The example shows a 24-bit per pixel design, although there are other implementations that use fewer colors and differential pairs. A standard 24-bit RGB interface requires up

to 28 single-ended signals. Instead of sending a cable bundle with 24 wires across the hinge, FPD-Link serializes the 28 data/control lines onto four differential I/O pairs, plus a clock differential pair resulting in 64% fewer wires. FPD-Link uses the Low-Voltage Differential Swing (LVDS) I/O standard.

Dividing 28 lines by four differential pairs also means that the data rate across the interface is seven times higher than the output clock rate.

Figure 9-11. Example Differential I/O Solution: Flat Panel Display Interface



At the receiving end, the differential data is de-serialized and converted back into a wider bundle of single-ended signals.

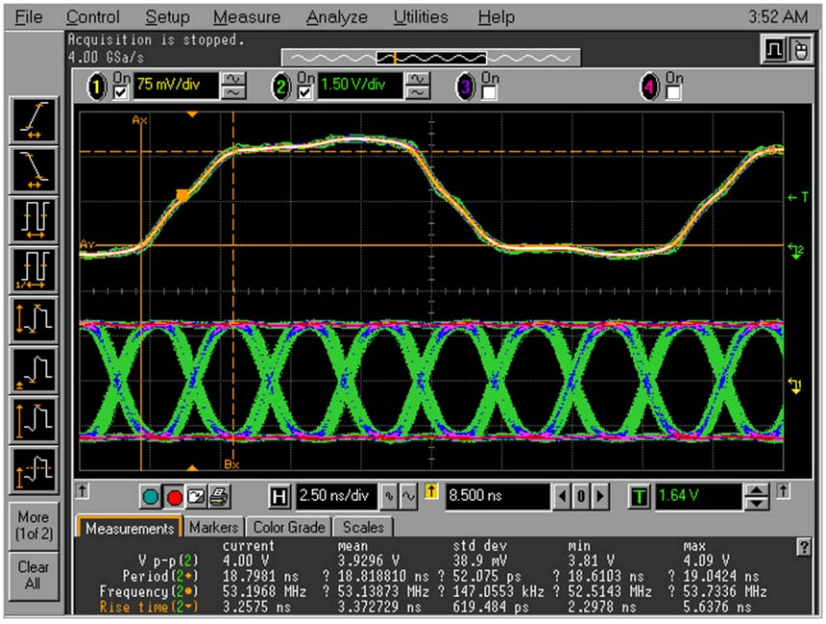
The integrated PLL in iCE40-family FPGAs simplifies this style of interface. An iCE40 FPGA can be at either end of the serial interface, either to allow a processor without an FPD-Link interface to communicate with an FPD-Link display, or to allow a processor with only an FPD-Link display interface to communicate to an RGB display or a display that uses a different format.

Figure 9-12 shows an example application running on an iCEman40 evaluation board. A 5-Megapixel camera captures live video images and provides the data to the iCE40 FPGA in RGGB format on a parallel LVTLL interface. The FPGA converts the RGGB data to 18-bit RGB data and then drives a 1,366x768 LCD panel on a three-channel FPD-Link interface, plus an LVDS output clock. The pixel clock operates at up to 75 MHz while data on the LVDS outputs operates up to 525 Megabits per second.

Figure 9-13 shows an oscilloscope output from an example FPD-Link transmitter interface build on an iCE40LP8K FPGA using the iCEman40 Evaluation Kit. The top-trace is the pixel clock, regenerated and phase aligned at the output. The iCE40 FPGA's PLL multiplies the pixel clock frequency by seven times to generate an internal 7X clock, which is divided internally to produce a 3.5X clock. Data is clocked onto the LVDS outputs using Double Data Rate (DDR) flip-flops, controlled by the internal 3.5X clock.

Figure 9-12. 5-Megapixel CMOS Camera to 1,366x768 LVDS Display on iCEman40 Evaluation Board

Figure 9-13. Example FPD-Link Transmitter on iCEman40 Evaluation Board



Cameras and Imagers

Differential interfaces are also popular for high-resolution and high-speed cameras and imagers.

Summary

This technical note provides an overview of iCE LVDS technology, focusing on its advantages, implementation, application, and its various electrical and timing characteristics. It also includes detailed recommendations for instantiating LVDS transmitter and receivers in your design and calculating the required external terminations to guarantee optimum performance.

References

- IEEE 1596.3 Standards
- Texas Instruments, “LVDS Owner’s Manual”, 2008
- Jimmy Ma, “A Closer Look at LVDS Technology”, Pericom, 2001
- Texas Instruments, “Application Note 1032: An Introduction to FPD-Link”
- Texas Instruments, “Application Note 1127: LVDS Display Interface (LDI) TFT Data Mapping for Interoperability with FPD-Link”

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
September 2012	01.0	Initial release.

Section III. iCE40 LP/HX Family Handbook Revision History

iCE40 LP/HX Family Handbook Revision History

Revision History

Date	Handbook Revision Number	Change Summary
October 2012	01.0	Initial release.

Note: For detailed revision changes, please refer to the revision history for each document.

© 2012 Lattice Semiconductor Corp. All Lattice trademarks, registered trademarks, patents, and disclaimers are as listed at www.latticesemi.com/legal. All other brand or product names are trademarks or registered trademarks of their respective holders. The specifications and information herein are subject to change without notice.