Spartan-3E FPGA Family: Functional Description

DS312-2 (v1.0) March 1, 2005

Advance Product Specification

Introduction

As described in **Architectural Overview**, the Spartan™-3E FPGA architecture consists of five fundamental functional elements:

- **Input/Output Blocks (IOBs)**
- **Configurable Logic Block (CLB) and Slice Resources**
- **Block RAM**
- **Dedicated Multipliers**
- **Digital Clock Managers (DCMs)**

The following sections provide detailed information on each of these functions. In addition, this section also describes the following functions:

- **Clocking Infrastructure**
- **Interconnect**
- **Configuration**
- **Powering Spartan-3E FPGAs**

Input/Output Blocks (IOBs)

IOB Overview

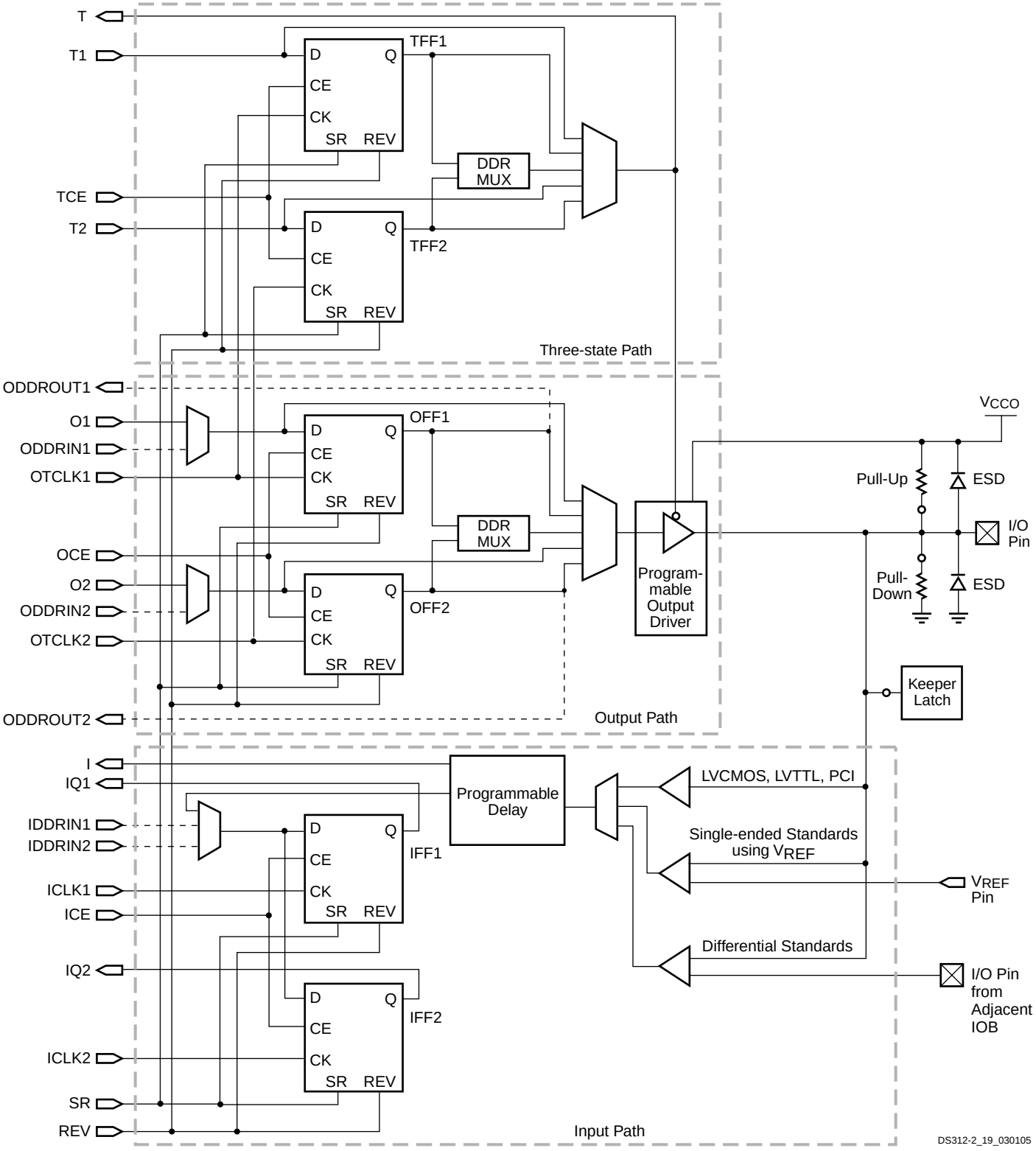
The Input/Output Block (IOB) provides a programmable, unidirectional or bidirectional interface between a package pin and the FPGA's internal logic. The IOB is similar to that of the Spartan-3 family with the following differences:

- Input-only blocks are added
- Programmable input delays are added to all blocks
- DDR flip-flops can be shared between adjacent IOBs

The unidirectional input-only block has a subset of the full IOB capabilities. Thus there are no connections or logic for an output path. The following paragraphs assume that any reference to output functionality does not apply to the input-only blocks. The number of input-only blocks varies with device size, but is never more than 25% of the total IOB count.

Figure 1, page 2 is a simplified diagram of the IOB's internal structure. There are three main signal paths within the IOB: the output path, input path, and 3-state path. Each path has its own pair of storage elements that can act as either registers or latches. For more information, see **Storage Element Functions**. The three main signal paths are as follows:

- The input path carries data from the pad, which is bonded to a package pin, through an optional programmable delay element directly to the I line. After the delay element, there are alternate routes through a pair of storage elements to the IQ1 and IQ2 lines. The IOB outputs I, IQ1, and IQ2 lead to the FPGA's internal logic. The delay element can be set to ensure a hold time of zero (see **Input Delay Functions**).
- The output path, starting with the O1 and O2 lines, carries data from the FPGA's internal logic through a multiplexer and then a three-state driver to the IOB pad. In addition to this direct path, the multiplexer provides the option to insert a pair of storage elements.
- The 3-state path determines when the output driver is high impedance. The T1 and T2 lines carry data from the FPGA's internal logic through a multiplexer to the output driver. In addition to this direct path, the multiplexer provides the option to insert a pair of storage elements.
- All signal paths entering the IOB, including those associated with the storage elements, have an inverter option. Any inverter placed on these paths is automatically absorbed into the IOB.



Notes:

1. All IOB signals communicating with the FPGA's internal logic have the option of inverting polarity inside the IOB.
2. Signals shown with dashed lines connect to the adjacent IOB in a differential pair only, not to the FPGA fabric.

Figure 1: **Simplified IOB Diagram**

Input Delay Functions

Each IOB has a programmable delay block that can delay the input signal from 0 to nominally 4000 ps. In Figure 2, the signal is first delayed by either 0 or 2000 ps (nominal) and is then applied to an 8 tap delay line. This delay line has a nominal value of 250 ps per tap. All 8 taps are available via a multiplexer for use as an asynchronous input directly into the FPGA fabric. In this way, the delay is programmable from 0 to 4000 ps in 250 ps steps. Four of the 8 taps are also available via a multiplexer to the D inputs of the synchronous storage elements. The delay inserted in the path to the storage element can be varied from 0 to 4000 ps in 500 ps steps. The first, coarse delay element is common to both asynchronous and synchronous paths, and must be either used or not used for both paths.

The delay values are set up in the silicon once at configuration time—they are non-modifiable in device operation.

The primary use for the input delay element is as an adequate delay to ensure that there is no hold time requirement when using the input flip-flop(s) with a global clock. The necessary value for this function is chosen by the Xilinx software tools and depends on device size. If the design is using a DCM in the clock path, then the delay element can be safely set to zero in the user's design, and there is still no hold time requirement.

Both asynchronous and synchronous values can be modified by the user, which is useful where extra delay is required on clock or data inputs, for example, in interfaces to various types of RAM.

See Module 3 of the Spartan-3E data sheet for exact values for the delay elements.

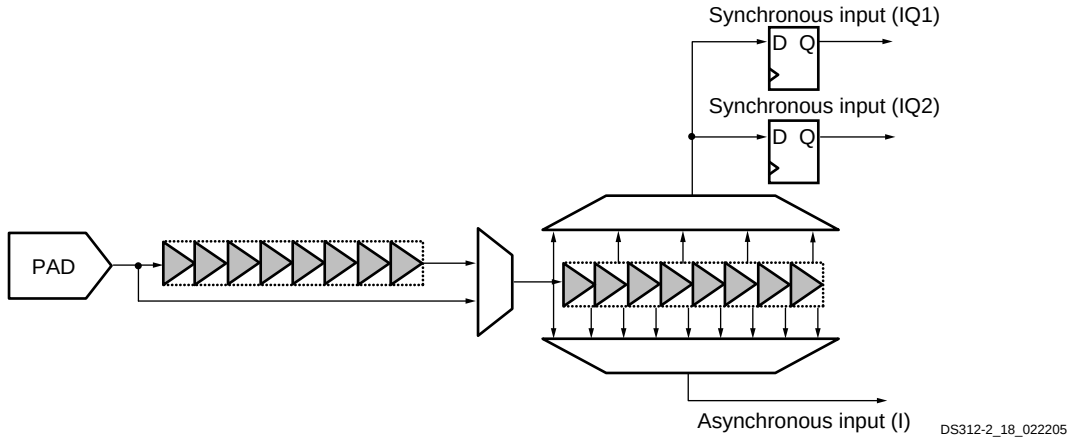


Figure 2: **Input Delay Elements**

Storage Element Functions

There are three pairs of storage elements in each IOB, one pair for each of the three paths. It is possible to configure each of these storage elements as an edge-triggered D-type flip-flop (FD) or a level-sensitive latch (LD).

The storage-element pair on either the Output path or the Three-State path can be used together with a special multiplexer to produce Double-Data-Rate (DDR) transmission. This is accomplished by taking data synchronized to the clock signal's rising edge and converting it to bits synchronized on both the rising and the falling edge. The combination of two registers and a multiplexer is referred to as a Double-Data-Rate D-type flip-flop (ODDR2).

Table 1 describes the signal paths associated with the storage element.

Table 1: **Storage Element Signal Description**

Storage Element Signal	Description	Function
D	Data input	Data at this input is stored on the active edge of CK and enabled by CE. For latch operation when the input is enabled, data passes directly to the output Q.
Q	Data output	The data on this output reflects the state of the storage element. For operation as a latch in transparent mode, Q mirrors the data at D.
CK	Clock input	Data is loaded into the storage element on this input's active edge with CE asserted.
CE	Clock Enable input	When asserted, this input enables CK. If not connected, CE defaults to the asserted state.
SR	Set/Reset input	This input forces the storage element into the state specified by the SRHIGH/SRLOW attributes. The SYNC/ASYNC attribute setting determines if the SR input is synchronized to the clock or not. If both SR and REV are active at the same time, the storage element gets a value of 0.
REV	Reverse input	This input is used together with SR. It forces the storage element into the state opposite from what SR does. The SYNC/ASYNC attribute setting determines whether the REV input is synchronized to the clock or not. If both SR and REV are active at the same time, the storage element gets a value of 0.

As shown in Figure 1, the upper registers in both the output and three-state paths share a common clock. The OTCLK1 clock signal drives the CK clock inputs of the upper registers on the output and three-state paths. Similarly, OTCLK2 drives the CK inputs for the lower registers on the output and three-state paths. The upper and lower registers on the input path have independent clock lines: ICLK1 and ICLK2.

The OCE enable line controls the CE inputs of the upper and lower registers on the output path. Similarly, TCE controls the CE inputs for the register pair on the three-state path and ICE does the same for the register pair on the input path.

The Set/Reset (SR) line entering the IOB controls all six registers, as is the Reverse (REV) line.

In addition to the signal polarity controls described in **IOB Overview**, each storage element additionally supports the controls described in Table 2.

Table 2: **Storage Element Options**

Option Switch	Function	Specificity
FF/Latch	Chooses between an edge-triggered flip-flop or a level-sensitive latch	Independent for each storage element
SYNC/ASYNC	Determines whether the SR set/reset control is synchronous or asynchronous	Independent for each storage element

Table 2: **Storage Element Options**

Option Switch	Function	Specificity
SRHIGH/SRLOW	Determines whether SR acts as a Set, which forces the storage element to a logic "1" (SRHIGH) or a Reset, which forces a logic "0" (SRLOW)	Independent for each storage element, except when using ODDR2. In the latter case, the selection for the upper element will apply to both elements.
INIT1/INIT0	When Global Set/Reset (GSR) is asserted or after configuration this option specifies the initial state of the storage element, either set (INIT1) or reset (INIT0). By default, choosing SRLOW also selects INIT0; choosing SRHIGH also selects INIT1.	Independent for each storage element, except when using ODDR2, which uses two IOBs. In the ODDR2 case, selecting INIT0 for one IOBs applies to both elements within the IOB, although INIT1 could be selected for the elements in the other IOB.

Double-Data-Rate Transmission

Double-Data-Rate (DDR) transmission describes the technique of synchronizing signals to both the rising and falling edges of the clock signal. Spartan-3E devices use register pairs in all three IOB paths to perform DDR operations.

The pair of storage elements on the IOB's Output path (OFF1 and OFF2), used as registers, combine with a special multiplexer to form a DDR D-type flip-flop (ODDR2). This primitive permits DDR transmission where output data bits are synchronized to both the rising and falling edges of a clock. DDR operation requires two clock signals (usually 50% duty cycle), one the inverted form of the other. These signals trigger the two registers in alternating fashion, as shown in Figure 3. The Digital Clock Manager (DCM) generates the two clock signals by mirroring an incoming signal, and then shifting it 180 degrees. This approach ensures minimal skew between the two signals. Alternatively, the inverter inside the IOB can be used to invert the clock signal, thus only using one clock line and both rising and falling edges of that clock line as the two clocks for the DDR flip-flops.

The storage-element pair on the Three-State path (TFF1 and TFF2) also can be combined with a local multiplexer to form a DDR primitive. This permits synchronizing the output enable to both the rising and falling edges of a clock. This DDR operation is realized in the same way as for the output path.

The storage-element pair on the input path (IFF1 and IFF2) allows an I/O to receive a DDR signal. An incoming DDR clock signal triggers one register, and the inverted clock signal triggers the other register. The registers take turns capturing bits of the incoming DDR data signal. The primitive to allow this functionality is called IDDR2.

Aside from high bandwidth data transfers, DDR outputs also can be used to reproduce, or *mirror*, a clock signal on the output. This approach is used to transmit clock and data signals together (source synchronously). A similar approach is used to reproduce a clock signal at multiple outputs. The advantage for both approaches is that skew across the outputs is minimal.

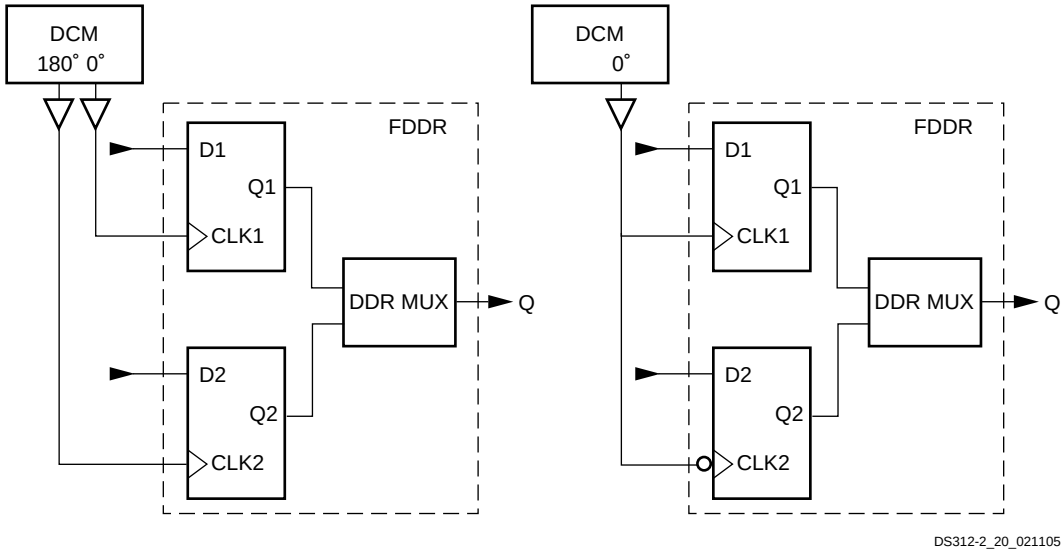


Figure 3: **Two Methods for Clocking the DDR Register**

Register Cascade Feature

In the Spartan-3E family, one of the IOBs in a differential pair can cascade either its input or output storage elements with those in the other IOB of the differential pair. This is intended to make DDR operation at high speed much simpler to implement. The new DDR connections that are available are shown in Figure 1 (dashed lines), and are only available for routing between IOBs and are not accessible to the FPGA fabric. Note that this feature is only available when using differential I/O.

IDDR2

As a DDR input pair, the master IOB registers incoming data on the rising edge of ICLK1 (= D1) and the rising edge of ICLK2 (= D2), which is typically the same as the falling edge of ICLK1. This data is then transferred into the FPGA fabric. At some point, both signals must be brought into the same clock domain, typically ICLK1. This can be difficult at high frequencies because the available time is only one half of a clock cycle assuming a 50% duty cycle. See Figure 4 for a graphical illustration of this function.

In the Spartan-3E device, the signal D2 can be cascaded into the storage element of the adjacent slave IOB. There it is re-registered to ICLK1, and only then fed to the FPGA fabric where it is now already in the same time domain as D1. Here, the FPGA fabric uses only the clock ICLK1 to process the received data. See Figure 5 for a graphical illustration of this function.

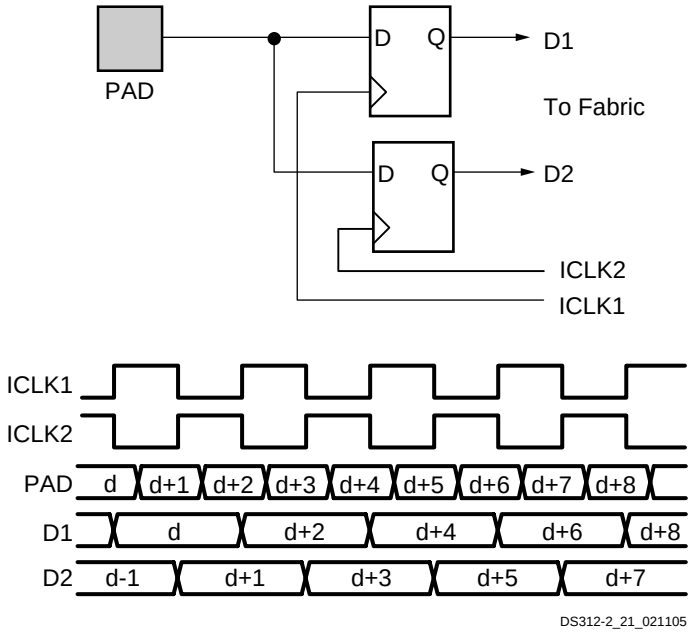


Figure 4: **Input DDR (without Cascade Feature)**

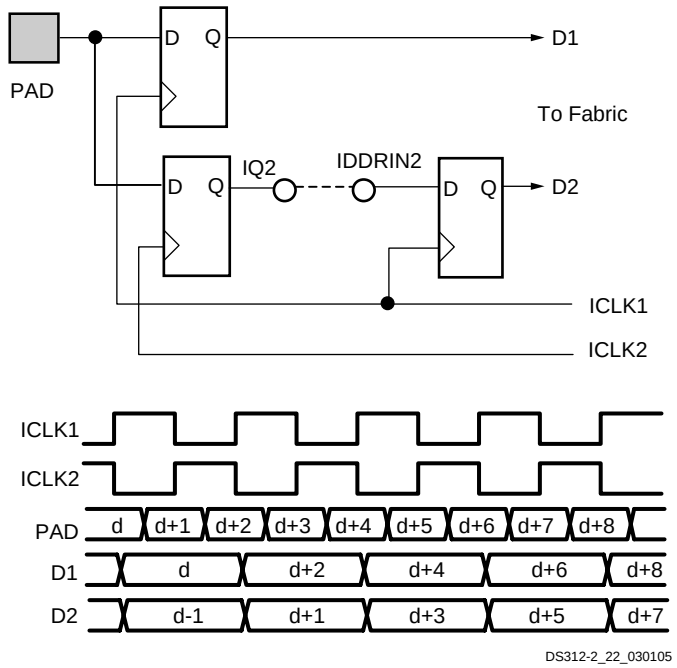


Figure 5: **Input DDR Using Spartan-3E Cascade Feature**

ODDR2

As a DDR output pair, the master IOB registers data coming from the FPGA fabric on the rising edge of OCLK1 (= D1) and the rising edge of OCLK2 (= D2), which is typically the same as the falling edge of OCLK1. These two bits of data are multiplexed by the DDR mux and forwarded to the output pin. At some point in the FPGA fabric, the signal D2 must be brought into the clock domain OCLK2 from the domain OCLK1. This can be difficult at high frequencies, because the time available is only one half a clock cycle. See Figure 6 for a graphical illustration of this function.

In the Spartan-3E device, the signal D2 can be cascaded via the storage element of the adjacent slave IOB. Here, it is registered by OCLK1 and then forwarded to the master IOB where it is re-registered to OCLK2, selected as usual by the DDR multiplexer, and then forwarded to the output pin. This way the data for transmission can be processed using just the clock OCLK1 in the FPGA fabric. See Figure 7 for a graphical illustration of this function.

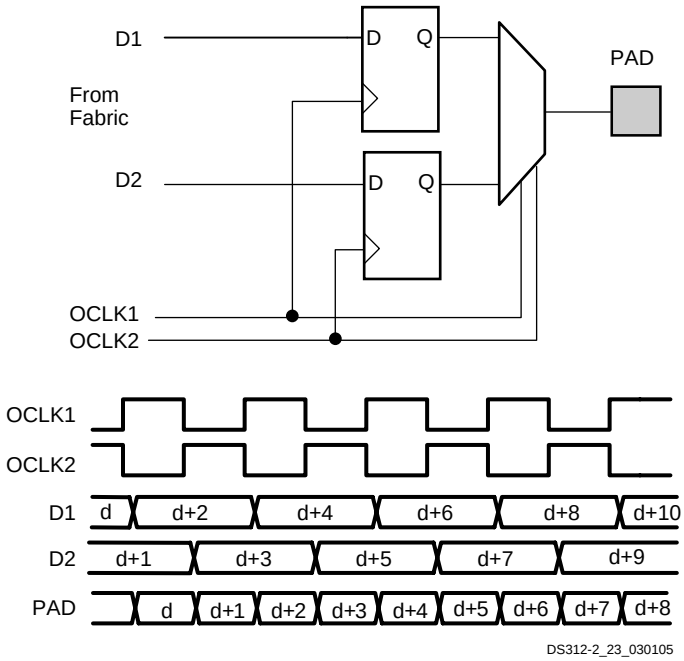


Figure 6: **Output DDR (without Cascade Feature)**

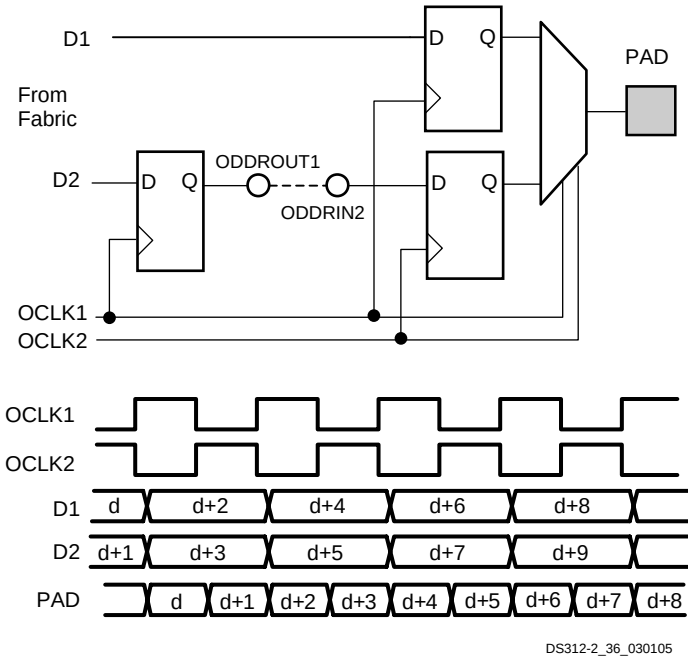


Figure 7: **Output DDR Using Spartan-3E Cascade Feature**

SelectIO Signal Standards

The Spartan-3E I/Os feature inputs and outputs that support a wide range of I/O signaling standards (Table 3 and Table 4). The majority of the I/Os also can be used to form differential pairs to support any of the differential signaling standards (Table 4).

To define the I/O signaling standard in a design, set the IOSTANDARD attribute to the appropriate setting. Xilinx provides a variety of different methods for applying the IOSTANDARD for maximum flexibility. For a full description of different methods of applying attributes to control IOSTANDARD, refer to “Entry Strategies for Xilinx Constraints” in the Xilinx Software Manuals and Help.

Spartan-3E FPGAs provide additional input flexibility by allowing I/O standards to be mixed in different banks. Special care must be taken to ensure the input voltages do not exceed V_{CCO} (see Module 3 for the specifications). For a particular V_{CCO} voltage, Table 3 and Table 4 list all of the IOSTANDARDs that can be combined and if the IOSTANDARD is supported as an input only or can be used for both inputs and outputs.

Table 3: **Single-Ended IOSTANDARD Bank Compatibility**

Single-Ended IOSTANDARD	V_{CCO} Supply/Compatibility						Input Requirements	
	1.2 V	1.5 V	1.8 V	2.5 V	3.0 V	3.3 V	V_{REF} for Inputs	Board Termination Voltage (V_{TT})
LVTTL	-	-	-	-	-	Input/ Output	N/R	N/R
LVCMOS33	-	-	-	-	-	Input/ Output	N/R	N/R
LVCMOS25	-	-	-	Input/ Output	Input	Input	N/R	N/R
LVCMOS18	-	-	Input/ Output	Input	Input	Input	N/R	N/R
LVCMOS15	-	Input/ Output	Input	Input	Input	Input	N/R	N/R
LVCMOS12	Input/ Output	Input	Input	Input	Input	Input	N/R(1)	N/R
PCI33_3	-	-	-	-	Input/ Output	Input	N/R	N/R
PCI66_3	-	-	-	-	Input/ Output	Input	N/R	N/R
PCIX					Input/ Output	Input	N/R	N/R
HSTL_I_18	-	-	Input/ Output	Input	Input	Input	0.9	0.9
HSTL_III_18	-	-	Input/ Output	Input	Input	Input	1.1	1.8
SSTL18_I	-	-	Input/ Output	Input	Input	Input	0.9	0.9
SSTL2_I	-	-	-	Input/ Output	Input	Input	1.25	1.25

Notes:

1. N/R - Not required for input operation.

Table 4: **Differential IOSTANDARD Bank Compatibility**

Differential IOSTANDARD	V_{CCO} Supply		Input Requirements: V_{REF}
	2.5V	3.3V	
LVDS_25	Input, On-chip Differential Termination, Output(1)	Input	N/R (Not Required)
RSDS_25	Input, On-chip Differential Termination, Output(1)	Input	
MINI_LVDS_25	Input, On-chip Differential Termination, Output(1)	Input	
LVPECL_25	Input, On-chip Differential Termination	Input	
BLVDS_25	Input, On-chip Differential Termination, Output	Input	

Notes:

1. Each bank can support any two of the following: LVDS_25 outputs, MINI_LVDS_25 outputs, RSDS_25 outputs.

HSTL and SSTL inputs use the Reference Voltage (V_{REF}) to bias the input-switching threshold. Once a configuration data file is loaded into the FPGA that calls for the I/Os of a given bank to use HSTL/SSTL, a few specifically reserved I/O pins on the same bank automatically convert to V_{REF} inputs. For banks that do not contain HSTL or SSTL, V_{REF} pins remain available for user I/Os or input pins.

Differential standards employ a pair of signals, one the opposite polarity of the other. The noise canceling properties (for example, Common-Mode Rejection) of these standards permit exceptionally high data transfer rates. This subsection introduces the differential signaling capabilities of Spartan-3E devices.

Each device-package combination designates specific I/O pairs specially optimized to support differential standards. Differential pairs can be shown in the Pin and Area Constraints Editor (PACE) with the “Show Differential Pairs” option. A unique *L-number*, part of the pin name, identifies the line-pairs associated with each bank (see Module 4). For each pair, the letters *P* and *N* designate the true and inverted lines, respectively. For example, the pin names IO_L43P_3 and IO_L43N_3 indicate the true and inverted lines comprising the line pair L43 on Bank 3.

V_{CCO} provides current to the outputs and additionally powers the On-Chip Differential Termination. V_{CCO} must be 2.5V when using the On-Chip Differential Termination. The V_{REF} lines are not required for differential operation.

To further understand how to combine multiple IOSTANDARDs within a bank, refer to **IOBs Organized into Banks**, page 10.

On-Chip Differential Termination

Spartan-3E devices provide an on-chip 100Ω differential termination across the input differential receiver terminals (See Module 3 for the specific range). The on-chip input differential termination in Spartan-3E devices eliminates the external 100Ω termination resistor commonly found in differential receiver circuits. Use differential termination for LVDS, mini-LVDS, and BLVDS as applications permit.

On-chip Differential Termination is available in banks with V_{CCO} = 2.5V and is not supported on dedicated input pins. Set the DIFF_TERM attribute to TRUE to enable Differential Termination on a differential I/O pin pair.

The DIFF_TERM attribute uses the following syntax in the UCF file:

```
INST <I/O_BUFFER_INSTANTIATION_NAME>
DIFF_TERM = "<TRUE/FALSE>";
```

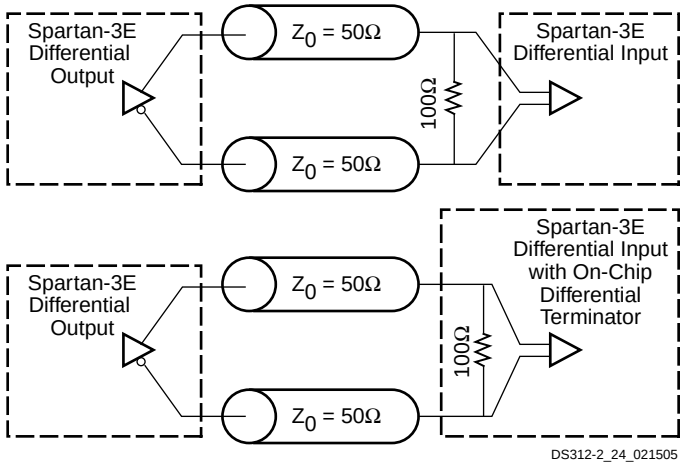


Figure 8: **Differential Inputs and Outputs**

Pull-Up and Pull-Down Resistors

Pull-up and pull-down resistors inside each IOB optionally force a floating I/O pin to a determined state. Pull-up and

pull-down resistors are commonly applied to unused I/Os, inputs, and three-state outputs, but can be used on any I/O. The pull-up resistor connects an I/O to V_{CCO} through a resistor. The resistance value depends on the V_{CCO} voltage (see Module 3 for the specifications). The pull-down resistor similarly connects an I/O to ground with a resistor. The PULLUP and PULLDOWN attributes and library primitives turn on these optional resistors.

By default, PULLDOWN resistors terminate all unused I/Os. Unused I/Os can alternatively be set to PULLUP or FLOAT. To change the unused I/O Pad setting, set the Bitstream Generator (BitGen) option ***UnusedPin*** to PULLUP, PULLDOWN, or FLOAT. The ***UnusedPin*** option is accessed through the Properties for Generate Programming File in ISE.

During configuration a Low logic level on HSWAP activates the pull-up resistors for all I/Os not used directly in the selected configuration mode.

Keeper Circuit

Each I/O has an optional keeper circuit (see Figure 9) that keeps bus lines from floating when not being actively driven. The KEEPER circuit retains the last logic level on a line after all drivers have been turned off. Apply the KEEPER attribute or use the KEEPER library primitive to use the KEEPER circuitry. Pull-up and pull-down resistors override the KEEPER settings.

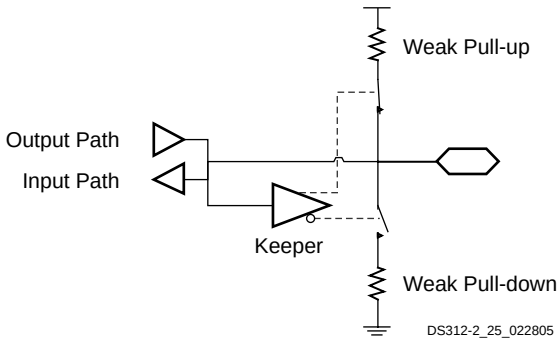


Figure 9: **Keeper Circuit**

Slew Rate Control and Drive Strength

Each IOB has a slew-rate control that sets the output switching edge-rate for LVCMOS and LVTTL outputs. The SLEW attribute controls the slew rate and can either be set to SLOW (default) or FAST.

Each LVCMOS and LVTTL output additionally supports up to six different drive current strengths as shown in Table 5.

To adjust the drive strength for each output set the DRIVE attribute to the desired drive strength: 2, 4, 6, 8, 12, and 16.

Table 5: **Programmable Output Drive Current**

Signal Standard	Output Drive Current (mA)					
	2	4	6	8	12	16
LVTTL	✓	✓	✓	✓	✓	✓
LVCMOS33	✓	✓	✓	✓	✓	✓
LVCMOS25	✓	✓	✓	✓	✓	-
LVCMOS18	✓	✓	✓	✓	-	-
LVCMOS15	✓	✓	✓	-	-	-
LVCMOS12	✓	-	-	-	-	-

High output current drive strength and FAST output slew rates generally result in fastest I/O performance. However, these same settings generally also result in transmission line effects on the printed circuit board (PCB) for all but the shortest board traces. Each IOB has independent slew rate and drive strength controls. Use the slowest slew rate and lowest output drive current that meets the performance requirements for the end application.

Likewise, due to lead inductance, a given package supports a limited number of simultaneous switching outputs (SSOs) when using fast, high-drive outputs. Only use fast, high-drive outputs when required by the application.

IOBs Organized into Banks

The Spartan-3E architecture organizes IOBs into four I/O banks as shown in Figure 10. Each bank maintains separate V_{CCO} and V_{REF} supplies. The separate supplies allow each bank to independently set V_{CCO}. Similarly, the V_{REF} supplies may be set for each bank. Refer to Table 3 and Table 4 for V_{CCO} and V_{REF} requirements.

When working with Spartan-3E devices, most of the differential I/O standards are compatible and can be combined within any given bank. Each bank can support any two of the following differential standards: LVDS_25 outputs, MINI_LVDS_25 outputs, and RSDS_25 outputs. As an example, LVDS_25 outputs, RSDS_25 outputs, and any other differential inputs while using on-chip differential termination are a valid combination. A combination not allowed is a single bank with LVDS_25 outputs, RSDS_25 outputs, and MINI_LVDS_25 outputs.

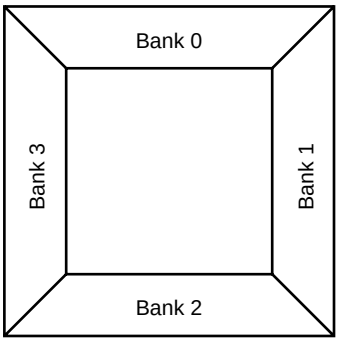


Figure 10: **Spartan-3E I/O Banks (top view)**

I/O Banking Rules

When assigning I/Os to banks, these V_{CCO} rules must be followed:

1. All V_{CCO} pins on the FPGA must be connected even if a bank is unused.
2. All V_{CCO} lines associated within a bank must be set to the same voltage level.
3. The V_{CCO} levels used by all standards assigned to the I/Os of any given bank must agree. The Xilinx development software checks for this. Table 3 and Table 4 describe how different standards use the V_{CCO} supply.
4. If a bank does not have any V_{CCO} requirements, connect V_{CCO} to an available voltage, such as 2.5V or 3.3V. Some configuration modes might place additional V_{CCO} requirements. Refer to **Configuration**, page 56 for more information.

If any of the standards assigned to the Inputs of the bank use V_{REF}, then the following additional rules must be observed:

1. All V_{REF} pins must be connected within a bank.
2. All V_{REF} lines associated with the bank must be set to the same voltage level.
3. The V_{REF} levels used by all standards assigned to the Inputs of the bank must agree. The Xilinx development software checks for this. Table 3 describes how different standards use the V_{REF} supply.

If V_{REF} is not required to bias the input switching thresholds, all associated V_{REF} pins within the bank can be used as user I/Os or input pins.

Package Footprint Compatibility

Sometimes, applications outgrow the logic capacity of a specific Spartan-3E FPGA. Fortunately, the Spartan-3E family is designed so that multiple part types are available in pin-compatible package footprints, as described in Module 4. In some cases, there are subtle differences between devices available in the same footprint. These differences are outlined for each package, such as pins that are unconnected on one device but connected on another in the same package or pins that are dedicated inputs on one package but full I/O on another. When designing the printed circuit board (PCB), plan for potential future upgrades and package migration.

The Spartan-3E family is not pin-compatible with any previous Xilinx FPGA family.

Dedicated Inputs

Dedicated Inputs are IOBs used only as inputs. Pin names designate a Dedicated Input if the name starts with *IP*, for example, IP or IP_Lxxx_x. Dedicated inputs retain the full functionality of the IOB for input functions with a single exception for differential inputs (IP_Lxxx_x). For the differential Dedicated Inputs, the on-chip differential termination is not available. To replace the on-chip differential termination, choose a differential pair that supports outputs (IO_Lxxx_x) or use an external 100Ω termination resistor on the board.

ESD Protection

Clamp diodes protect all device pads against damage from Electro-Static Discharge (ESD) as well as excessive voltage transients. Each I/O has two clamp diodes: one diode extends P-to-N from the pad to V_{CCO} and a second diode extends N-to-P from the pad to GND. During operation, these diodes are normally biased in the off state. These clamp diodes are always connected to the pad, regardless of the signal standard selected. The presence of diodes limits the ability of Spartan-3E I/Os to tolerate high signal voltages. The V_{IN} absolute maximum rating in Table 1 of Module 3 specifies the voltage range that I/Os can tolerate.

Supply Voltages for the IOBs

The IOBs are powered by three supplies:

1. The V_{CCO} supplies, one for each of the FPGA's I/O banks, power the output drivers. The voltage on the V_{CCO} pins determines the voltage swing of the output signal.
2. V_{CCINT} is the main power supply for the FPGA's internal logic.
3. V_{CCAUX} is an auxiliary source of power, primarily to optimize the performance of various FPGA functions such as I/O switching.

The I/Os During Power-On, Configuration, and User Mode

All I/Os have ESD clamp diodes to their respective V_{CCO} supply and from GND, as shown in Figure 1. The V_{CCINT} (1.2V), V_{CCAUX} (2.5V), and V_{CCO} supplies can be applied in any order. Before the FPGA can start its configuration process, V_{CCINT}, V_{CCO} Bank 2, and V_{CCAUX} must have reached their respective minimum recommended operating

levels (see Table 2 of Module 3). At this time, all I/O drivers are in a high-impedance state. V_{CCO} Bank 2, V_{CCINT}, and V_{CCAUX} serve as inputs to the internal Power-On Reset circuit (POR).

A Low level applied to the HSWAP input enables pull-up resistors on User I/Os from power-on throughout configuration. A High level on HSWAP disables the pull-up resistors, allowing the I/Os to float. HSWAP contains a weak pull-up and defaults to High if left floating. As soon as power is applied, the FPGA begins initializing its configuration memory. At the same time, the FPGA internally asserts the Global Set-Reset (GSR), which asynchronously resets all IOB storage elements to a default Low state.

Upon the completion of initialization and the beginning of configuration, INIT_B goes High, sampling the M0, M1, and M2 inputs to determine the configuration mode. At this point in time, the configuration data is loaded into the FPGA. The I/O drivers remain in a high-impedance state (with or without pull-up resistors, as determined by the HSWAP input) throughout configuration.

At the end of configuration, the GSR net is released, placing the IOB registers in a Low state by default, unless the loaded design reverses the polarity of their respective SR inputs.

The Global Three State (GTS) net is released during Start-Up, marking the end of configuration and the beginning of design operation in the User mode. After the GTS net is released, all user I/Os go active while all unused I/Os are weakly pulled down (PULLDOWN). The designer can control how the unused I/Os are terminated after GTS is released by setting the Bitstream Generator (BitGen) option UnusedPin to PULLUP, PULLDOWN, or FLOAT.

One clock cycle later (default), the Global Write Enable (GWE) net is released allowing the RAM and registers to change states. Once in User mode, any pull-up resistors enabled by HSWAP revert to the user settings and HSWAP is available as a general-purpose I/O. For more information on PULLUP and PULLDOWN, see **Pull-Up and Pull-Down Resistors**.

JTAG Boundary-Scan Capability

All Spartan-3E IOBs support boundary-scan testing compatible with IEEE 1149.1/1532 standards. See **JTAG Mode**, page 86 for more information on programming via JTAG.

Configurable Logic Block (CLB) and Slice Resources

CLB Overview

The Configurable Logic Blocks (CLBs) constitute the main logic resource for implementing synchronous as well as combinatorial circuits. Each CLB contains four slices, and each slice contains two Look-Up Tables (LUTs) to implement logic and two dedicated storage elements that can be used as flip-flops or latches. The LUTs can be used as a 16x1 memory (RAM16) or as a 16-bit shift register (SRL16), and additional multiplexers and carry logic simplify wide logic and arithmetic functions. Most general-purpose logic in a design is automatically mapped to the slice resources in the CLBs. Each CLB is identical, and the Spartan-3E family CLB structure is identical to that for the Spartan-3 family.

CLB Array

The CLBs are arranged in a regular array of rows and columns as shown in Figure 11.

Each density varies by the number of rows and columns of CLBs (see Table 6).

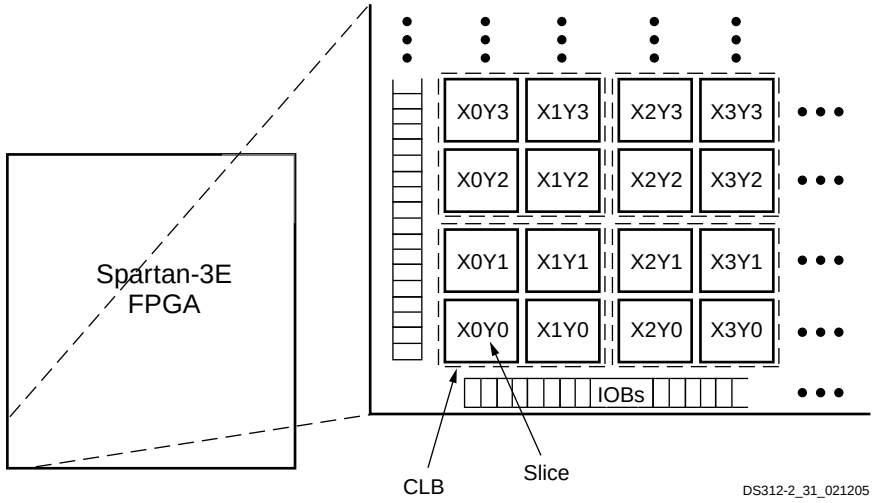


Figure 11: **CLB Locations**

Table 6: **Spartan-3E CLB Resources**

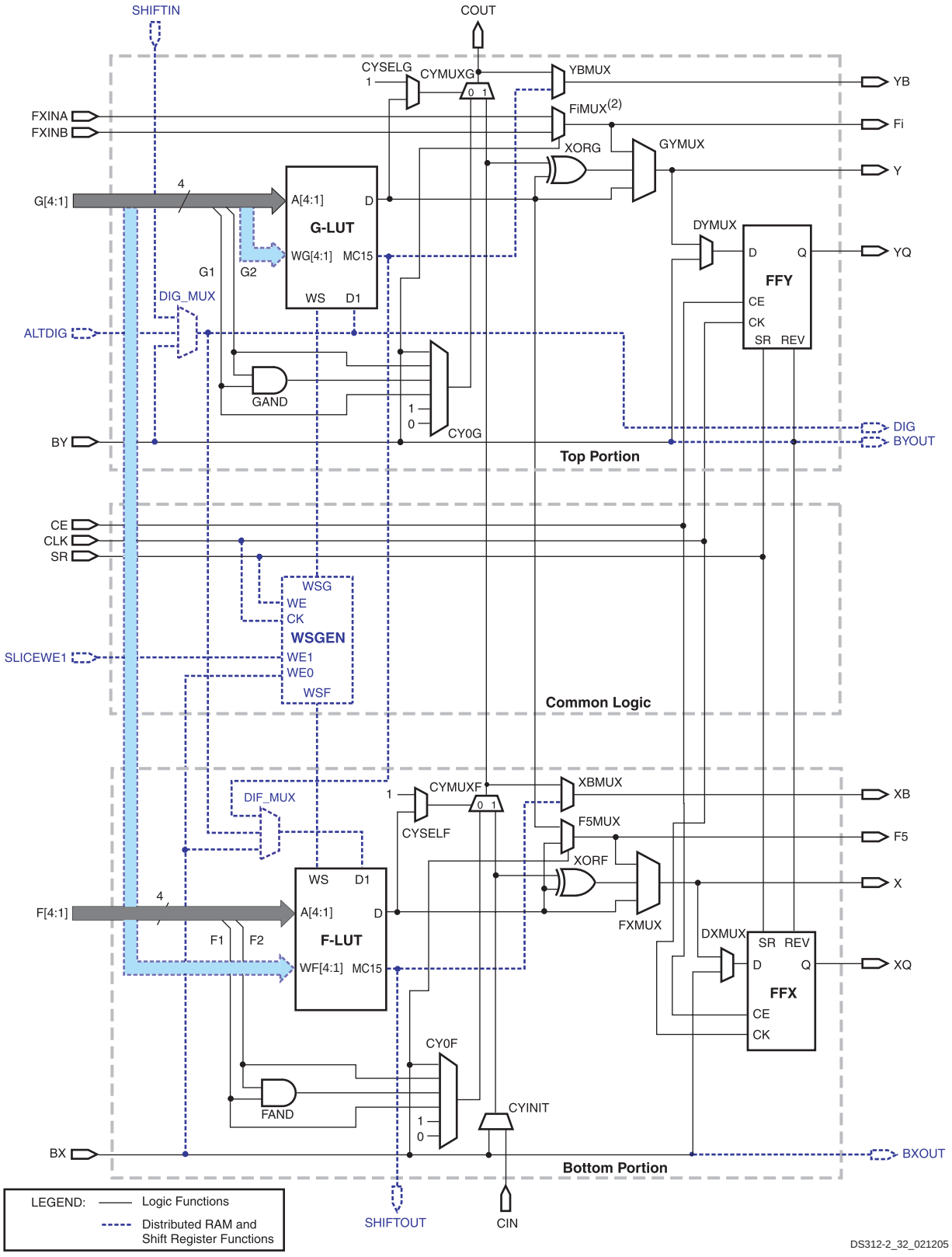
Device	CLB Rows	CLB Columns	CLB Total(1)	Slices	LUTs / Flip-Flops	Equivalent Logic Cells	RAM16 / SRL16	Distributed RAM Bits
XC3S100E	22	16	240	960	1920	2160	960	15360
XC3S250E	34	26	612	2448	4896	5508	2448	39168
XC3S500E	46	34	1164	4656	9312	10476	4656	74496
XC3S1200E	60	46	2168	8672	17344	19512	8672	138752
XC3S1600E	76	58	3688	14752	29504	33192	14752	236032

Notes:

1. The number of CLBs is less than the multiple of the rows and columns because the block RAM/multiplier blocks and the DCMs are embedded in the array (see Module 1, Figure 1).

Slices

Each CLB comprises four interconnected slices, as shown in Figure 13. These slices are grouped in pairs. Each pair is organized as a column with an independent carry chain. The left pair supports both logic and memory functions and its slices are called SLICEM. The right pair supports logic only and its slices are called SLICEL. Therefore half the LUTs support both logic and memory (including both RAM16 and SRL16 shift registers) while half support logic only, and the two types alternate throughout the array columns. The SLICEL reduces the size of the CLB and lowers the cost of the device, and can also provide a performance advantage over the SLICEM.



Notes:

1. Options to invert signal polarity as well as other options that enable lines for various functions are not shown.
2. The index i can be 6, 7, or 8, depending on the slice. The upper SLICEL has an F8MUX, and the upper SLICEM has an F7MUX. The lower SLICEL and SLICEM both have an F6MUX.

Figure 12: **Simplified Diagram of the Left-Hand SLICEM**

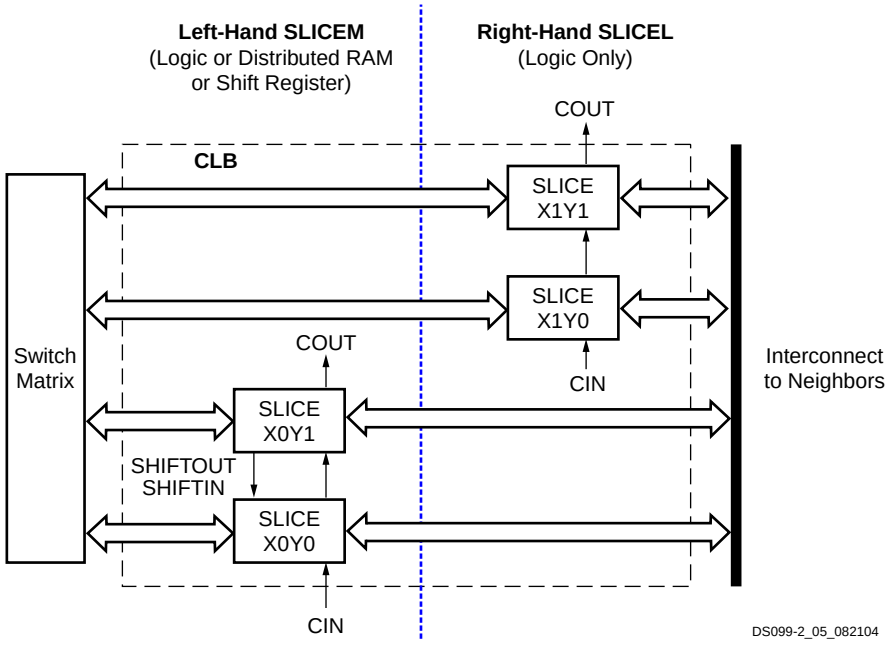


Figure 13: **Arrangement of Slices within the CLB**

Slice Location Designations

The Xilinx development software designates the location of a slice according to its X and Y coordinates, starting in the bottom left corner, as shown in Figure 11. The letter 'X' followed by a number identifies columns of slices, incrementing from the left side of the die to the right. The letter 'Y' followed by a number identifies the position of each slice in a pair as well as indicating the CLB row, incrementing from the bottom of the die. Figure 13 shows the CLB located in the lower left-hand corner of the die. The SLICEM always has an even 'X' number, and the SLICEL always has an odd 'X' number.

Slice Overview

A slice includes two LUT function generators and two storage elements, along with additional logic, as shown in Figure 14.

Both SLICEM and SLICEL have the following elements in common to provide logic, arithmetic, and ROM functions:

- Two 4-input LUT function generators, F and G
- Two storage elements
- Two wide-function multiplexers, F5MUX and FiMUX
- Carry and arithmetic logic

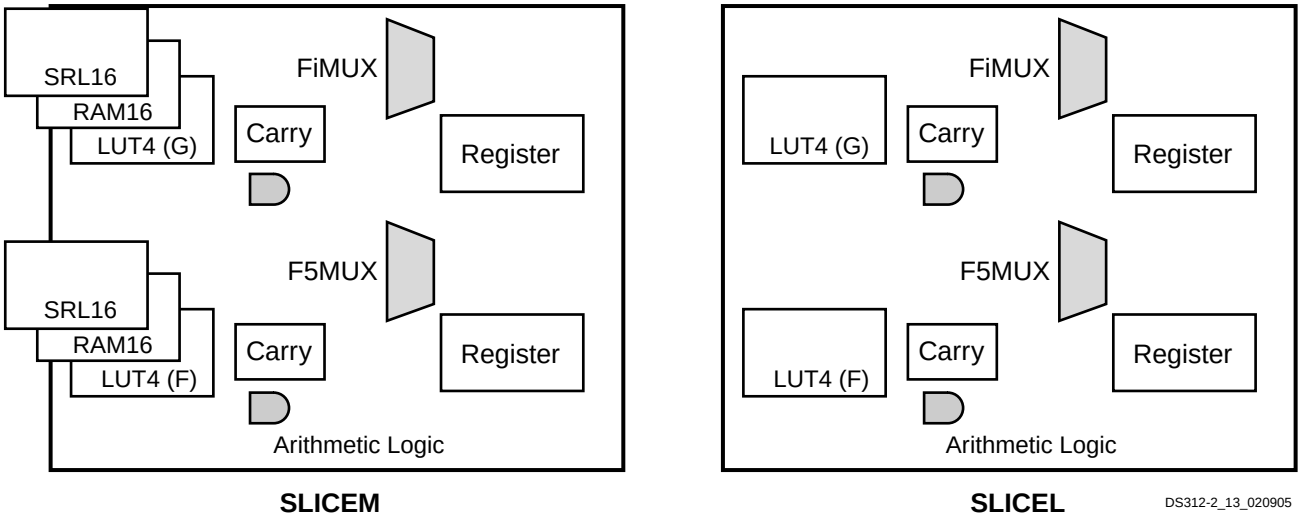


Figure 14: **Resources in a Slice**

The SLICEM pair supports two additional functions:

- Two 16x1 distributed RAM blocks, RAM16
- Two 16-bit shift registers, SRL16

Each of these elements is described in more detail in the following sections.

Logic Cells

The combination of a LUT and a storage element is known as a "Logic Cell". The additional features in a slice, such as the wide multiplexers, carry logic, and arithmetic gates, add to the capacity of a slice, implementing logic that would otherwise require additional LUTs. Benchmarks have shown that the overall slice is equivalent to 2.25 simple logic cells. This calculation provides the equivalent logic cell count shown in Table 6.

Slice Details

Figure 16 is a detailed diagram of the SLICEM. It represents a superset of the elements and connections to be found in all slices. The dashed and gray lines (blue when viewed in color) indicate the resources found only in the SLICEM and not in the SLICEL.

Each slice has two halves, which are differentiated as top and bottom to keep them distinct from the upper and lower slices in a CLB. The control inputs for the clock (CLK), Clock Enable (CE), Slice Write Enable (SLICEWE1), and Reset/Set (RS) are shared in common between the two halves.

The LUTs located in the top and bottom portions of the slice are referred to as "G" and "F", respectively, or the "G-LUT" and the "F-LUT". The storage elements in the top and bottom portions of the slice are called FFY and FFX, respectively.

Each slice has two multiplexers with F5MUX in the bottom portion of the slice and FiMUX in the top portion. Depending on the slice, the FiMUX takes on the name F6MUX, F7MUX, or F8MUX, according to its position in the multiplexer chain. The lower SLICEL and SLICEM both have an F6MUX. The upper SLICEM has an F7MUX, and the upper SLICEL has an F8MUX.

The carry chain enters the bottom of the slice as CIN and exits at the top as COUT. Five multiplexers control the chain: CYINIT, CY0F, and CYMUXF in the bottom portion and CY0G and CYMUXG in the top portion. The dedicated arithmetic logic includes the exclusive-OR gates XORF and XORG (bottom and top portions of the slice, respectively) as well as the AND gates FAND and GAND (bottom and top portions, respectively).

See Table 7 for a description of all the slice input and output signals.

Table 7: **Slice Inputs and Outputs**

Name	Location	Direction	Description
F[4:1]	SLICEL/M Bottom	Input	F-LUT and FAND inputs
G[4:1]	SLICEL/M Top	Input	G-LUT and GAND inputs or Write Address (SLICEM)
BX	SLICEL/M Bottom	Input	Bypass to or output (SLICEM) or storage element, or control input to F5MUX, input to carry logic, or data input to RAM (SLICEM)
BY	SLICEL/M Top	Input	Bypass to or output (SLICEM) or storage element, or control input to FiMUX, input to carry logic, or data input to RAM (SLICEM)
BXOUT	SLICEM Bottom	Output	BX bypass output
BYOUT	SLICEM Top	Output	BY bypass output
ALTDIG	SLICEM Top	Input	Alternate data input to RAM
DIG	SLICEM Top	Output	ALTDIG or SHIFTIN bypass output
SLICEWE1	SLICEM Common	Input	RAM Write Enable
F5	SLICEL/M Bottom	Output	Output from F5MUX; direct feedback to FiMUX
FXINA	SLICEL/M Top	Input	Input to FiMUX; direct feedback from F5MUX or another FiMUX
FXINB	SLICEL/M Top	Input	Input to FiMUX; direct feedback from F5MUX or another FiMUX
Fi	SLICEL/M Top	Output	Output from FiMUX; direct feedback to another FiMUX
CE	SLICEL/M Common	Input	FFX/Y Clock Enable
SR	SLICEL/M Common	Input	FFX/Y Set or Reset or RAM Write Enable (SLICEM)

Table 7: **Slice Inputs and Outputs *(Continued)***

Name	Location	Direction	Description
CLK	SLICEL/M Common	Input	FFX/Y Clock or RAM Clock (SLICEM)
SHIFTIN	SLICEM Top	Input	Data input to G-LUT RAM
SHIFTOUT	SLICEM Bottom	Output	Shift data output from F-LUT RAM
CIN	SLICEL/M Bottom	Input	Carry chain input
COUT	SLICEL/M Top	Output	Carry chain output
X	SLICEL/M Bottom	Output	Combinatorial output
Y	SLICEL/M Top	Output	Combinatorial output
XB	SLICEL/M Bottom	Output	Combinatorial output from carry or F-LUT SRL16 (SLICEM)
YB	SLICEL/M Top	Output	Combinatorial output from carry or G-LUT SRL16 (SLICEM)
XQ	SLICEL/M Bottom	Output	FFX output
YQ	SLICEL/M Top	Output	FFY output

Main Logic Paths

Central to the operation of each slice are two nearly identical data paths at the top and bottom of the slice. The description that follows uses names associated with the bottom path. (The top path names appear in parentheses.) The basic path originates at an interconnect switch matrix outside the CLB. See **Interconnect** for more information on the switch matrix and the routing connections.

Four lines, F1 through F4 (or G1 through G4 on the upper path), enter the slice and connect directly to the LUT. Once inside the slice, the lower 4-bit path passes through a LUT "F" (or "G") that performs logic operations. The LUT Data output, "D", offers five possible paths:

1. Exit the slice via line "X" (or "Y") and return to interconnect.
2. Inside the slice, "X" (or "Y") serves as an input to the DXMUX (or DYMUX) which feeds the data input, "D", of the FFY (or FFX) storage element. The "Q" output of the storage element drives the line XQ (or YQ) which exits the slice.
3. Control the CYMUXF (or CYMUXG) multiplexer on the carry chain.
4. With the carry chain, serve as an input to the XORF (or XORG) exclusive-OR gate that performs arithmetic operations, producing a result on "X" (or "Y").
5. Drive the multiplexer F5MUX to implement logic functions wider than four bits. The "D" outputs of both the F-LUT and G-LUT serve as data inputs to this multiplexer.

In addition to the main logic paths described above, there are two bypass paths that enter the slice as BX and BY. Once inside the FPGA, BX in the bottom half of the slice (or BY in the top half) can take any of several possible branches:

1. Bypass both the LUT and the storage element, and then exit the slice as BXOUT (or BYOUT) and return to interconnect.
2. Bypass the LUT, and then pass through a storage element via the D input before exiting as XQ (or YQ).
3. Control the wide function multiplexer F5MUX (or FiMUX).
4. Via multiplexers, serve as an input to the carry chain.
5. Drive the DI input of the LUT.
6. BY can control the REV inputs of both the FFY and FFX storage elements. See **Storage Element Functions**.
7. Finally, the DIG_MUX multiplexer can switch BY onto the DIG line, which exits the slice.

The control inputs CLK, CE, SR, BX and BY have programmable polarity. The LUT inputs do not need programmable polarity because their function can be inverted inside the LUT.

The sections that follow provide more detail on individual functions of the slice.

Look-Up Tables

The Look-Up Table or LUT is a RAM-based function generator and is the main resource for implementing logic functions. Furthermore, the LUTs in each SLICEM pair can be configured as Distributed RAM or a 16-bit shift register, as described later.

Each of the two LUTs (F and G) in a slice have four logic inputs (A1-A4) and a single output (D). Any four-variable Boolean logic operation can be implemented in one LUT. Functions with more inputs can be implemented by cascad-

ing LUTs or by using the wide function multiplexers that are described later.

The output of the LUT can connect to the wide multiplexer logic, the carry and arithmetic logic, or directly to a CLB output or to the CLB storage element. See Figure 15.

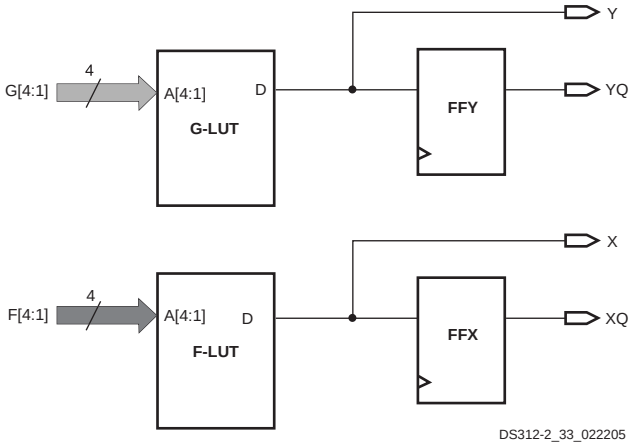


Figure 15: **LUT Resources in a Slice**

Wide Multiplexers

Wide-function multiplexers effectively combine LUTs in order to permit more complex logic operations. Each slice has two of these multiplexers with F5MUX in the bottom portion of the slice and FiMUX in the top portion. The F5MUX multiplexes the two LUTs in a slice. The FiMUX multiplexes two CLB inputs which connect directly to the F5MUX and FiMUX results from the same slice or from other slices. See Figure 16.

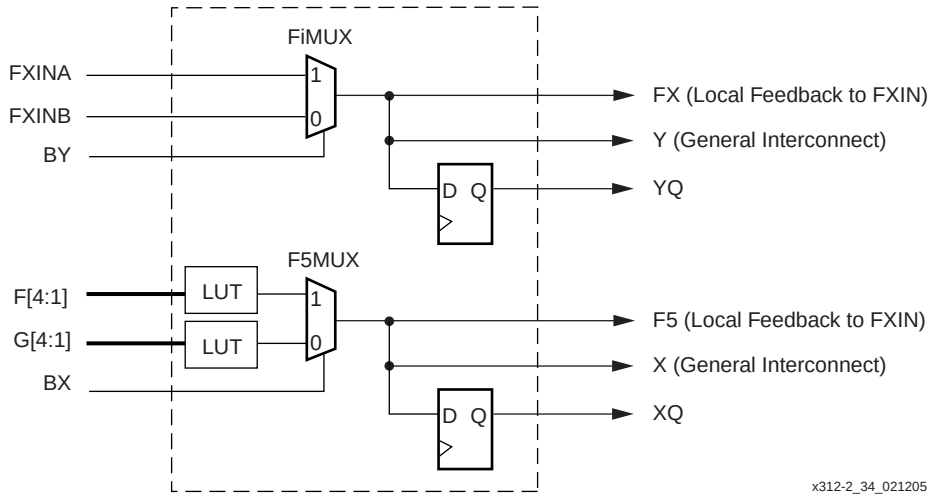


Figure 16: **Dedicated Multiplexers in Spartan-3E CLB**

Depending on the slice, FiMUX takes on the name F6MUX, F7MUX, or F8MUX. The designation indicates the number of inputs possible without restriction on the function. For example, an F7MUX can generate any function of seven inputs. Figure 17 shows the names of the multiplexers in each position in the Spartan-3E CLB. The figure also includes the direct connections within the CLB, along with the F7MUX connection to the CLB below.

Each mux can create logic functions of more inputs than indicated by its name. The F5MUX, for example, can generate any function of five inputs, with four inputs duplicated to two LUTs and the fifth input controlling the mux. Because each LUT can implement independent 2:1 muxes, the F5MUX can combine them to create a 4:1 mux, which is a six-input function. If the two LUTs have completely independent sets of inputs, some functions of all nine inputs can be implemented. Table 8 shows the connections for each multiplexer and the number of inputs possible for different types of functions.

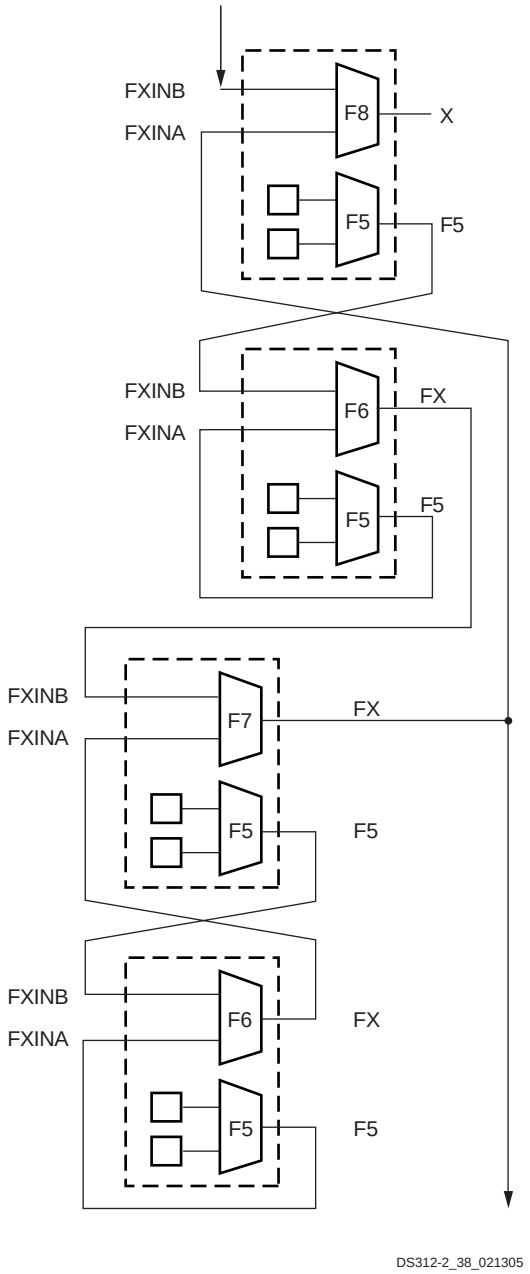


Figure 17: **Muxes and Dedicated Feedback in Spartan-3E CLB**

Table 8: **Mux Capabilities**

Mux	Usage	Input Source	Total Number of Inputs per Function		
			For Any Function	For Mux	For Limited Functions
F5MUX	F5MUX	LUTs	5	6 (4:1 mux)	9
FiMUX	F6MUX	F5MUX	6	11 (8:1 mux)	19
	F7MUX	F6MUX	7	20 (16:1 mux)	39
	F8MUX	F7MUX	8	37 (32:1 mux)	79

The wide multiplexers can be used by the automatic tools or instantiated in a design using a component such as the F5MUX. The symbol, signals, and function are described below. The description is similar for the F6MUX, F7MUX, and F8MUX. Each has versions with a general output, local output, or both.

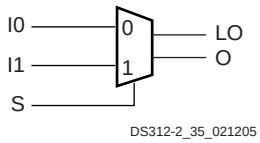


Figure 18: **F5MUX with Local and General Outputs**

Table 9: **F5MUX Inputs and Outputs**

Signal	Function
I0	Input selected when S is Low
I1	Input selected when S is High
S	Select input
LO	Local Output that connects to the F5 or FX CLB pins, which use local feedback to the FXIN inputs to the FiMUX for cascading
O	General Output that connects to the general-purpose combinatorial or registered outputs of the CLB

Table 10: **F5MUX Function**

Inputs			Outputs	
S	**I0**	**I1**	**O**	**LO**
0	1	X	1	1
0	0	X	0	0
1	X	1	1	1
1	X	0	0	0

For more details on using the multiplexers, see XAPP466: "Using Dedicated Multiplexers in Spartan-3 FPGAs".

Carry and Arithmetic Logic

The carry chain, together with various dedicated arithmetic logic gates, support fast and efficient implementations of math operations. The carry logic is automatically used for most arithmetic functions in a design. The gates and multiplexers of the carry and arithmetic logic can also be used for general-purpose logic, including simple wide Boolean functions.

The carry chain enters the slice as CIN and exits as COUT, controlled by several multiplexers. The carry chain connects directly from one CLB to the CLB above. The carry chain can be initialized at any point from the BX (or BY) inputs.

The dedicated arithmetic logic includes the exclusive-OR gates XORF and XORG (upper and lower portions of the slice, respectively) as well as the AND gates GAND and FAND (upper and lower portions, respectively). These gates work in conjunction with the LUTs to implement efficient arithmetic functions, including counters and multipliers, typically at two bits per slice. See Figure 19 and Table 11.

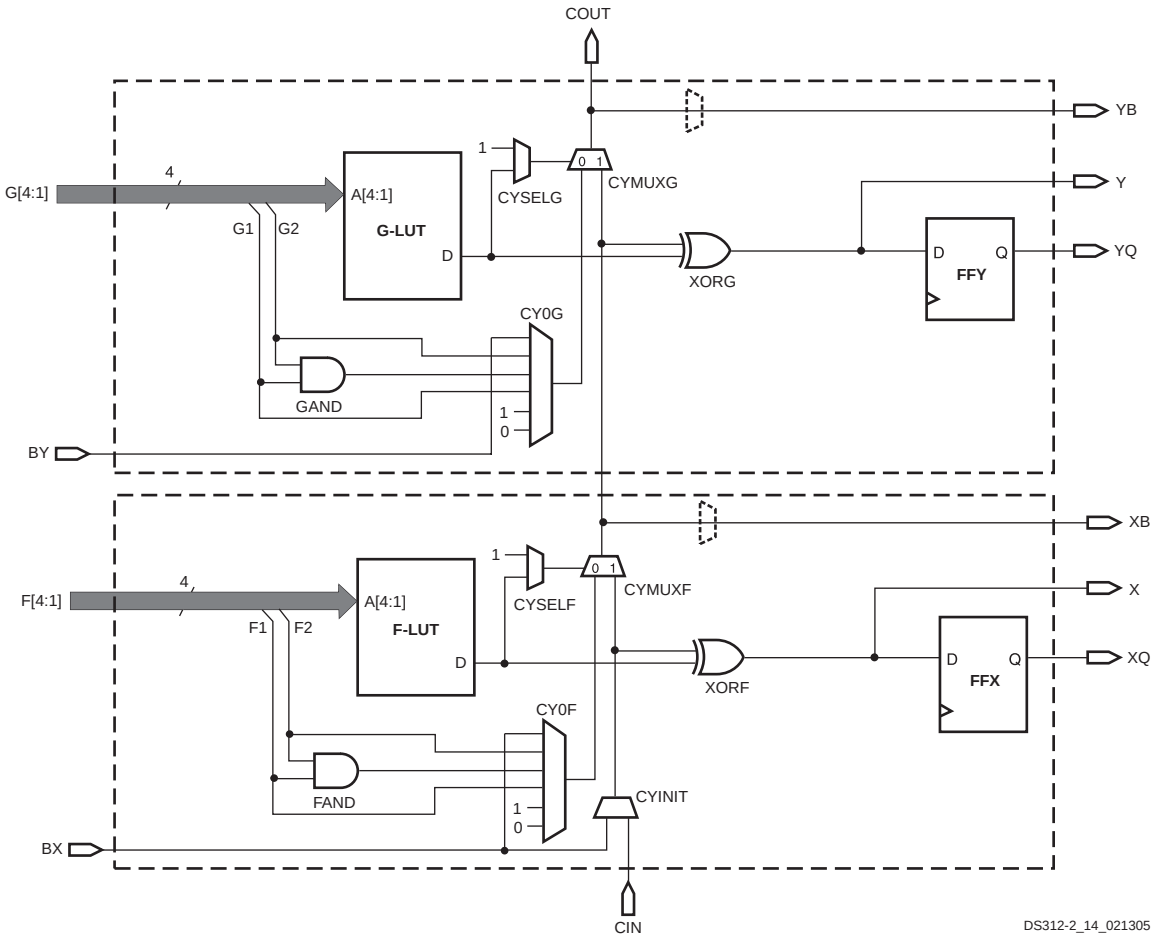


Figure 19: **Carry Logic**

Table 11: **Carry Logic Functions**

Function	Description
CYINIT	Initializes carry chain for a slice. Fixed selection of: • CIN carry input from the slice below • BX input
CY0F	Carry generation for bottom half of slice. Fixed selection of: • F1 or F2 inputs to the LUT (both equal 1 when a carry is to be generated) • FAND gate for multiplication • BX input for carry initialization • Fixed "1" or "0" input for use as a simple Boolean function
CY0G	Carry generation for top half of slice. Fixed selection of: • G1 or G2 inputs to the LUT (both equal 1 when a carry is to be generated) • GAND gate for multiplication • BY input for carry initialization • Fixed "1" or "0" input for use as a simple Boolean function
CYMUXF	Carry generation or propagation mux for bottom half of slice. Dynamic selection via CYSELF of: • CYINIT carry propagation (CYSELF = 1) • CY0F carry generation (CYSELF = 0)

Table 11: **Carry Logic Functions *(Continued)***

Function	Description
CYMUXG	Carry generation or propagation mux for top half of slice. Dynamic selection via CYSELF of: • CYMUXF carry propagation (CYSELG = 1) • CY0G carry generation (CYSELG = 0)
CYSELF	Carry generation or propagation select for bottom half of slice. Fixed selection of: • F-LUT output (typically XOR result) • Fixed "1" to always propagate
CYSELG	Carry generation or propagation select for top half of slice. Fixed selection of: • G-LUT output (typically XOR result) • Fixed "1" to always propagate
XORF	Sum generation for bottom half of slice. Inputs from: • F-LUT • CYINIT carry signal from previous stage Result is sent to either the combinatorial or registered output for the top of the slice.
XORG	Sum generation for top half of slice. Inputs from: • G-LUT • CYMUXF carry signal from previous stage Result is sent to either the combinatorial or registered output for the top of the slice.
FAND	Multiplier partial product for bottom half of slice. Inputs: • F-LUT F1 input • F-LUT F2 input Result is sent through CY0F to become the carry generate signal into CYMUXF
GAND	Multiplier partial product for top half of slice. Inputs: • G-LUT G1 input • G-LUT G2 input Result is sent through CY0G to become the carry generate signal into CYMUXG

The basic usage of the carry logic is to generate a half-sum in the LUT via an XOR function, which generates or propagates a carry out COUT via the carry mux CYMUXF (or CYMUXG), and then complete the sum with the dedicated XORF (or XORG) gate and the carry input CIN. This structure allows two bits of an arithmetic function in each slice. The CYMUXF (or CYMUXG) can be instantiated using the MUXCY element, and the XORF (or XORG) can be instantiated using the XORCY element.

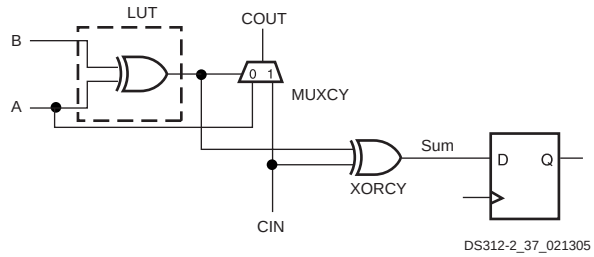


Figure 20: **Using the MUXCY and XORCY in the Carry Logic**

The FAND (or GAND) gate is used for partial product multiplication and can be instantiated using the MULT_AND component. Partial products are generated by two-input AND gates and then added. The carry logic is efficient for the adder, but one of the inputs must be outside the LUT as shown in Figure 20. The FAND (or GAND) gate is used to duplicate one of the partial products, while the LUT generates both partial products and the XOR function, as shown in Figure 21.

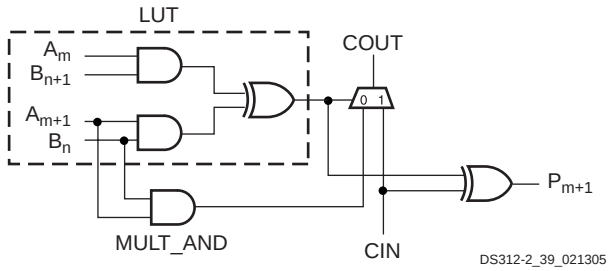


Figure 21: **Using the MULT_AND for Multiplication in Carry Logic**

The MULT_AND is useful for small multipliers. Larger multipliers can be built using the dedicated 18x18 multiplier blocks (see **Dedicated Multipliers**).

Storage Elements

The storage element, which is programmable as either a D-type flip-flop or a level-sensitive transparent latch, provides a means for synchronizing data to a clock signal, among other uses. The storage elements in the top and bottom portions of the slice are called FFY and FFX, respectively. FFY has a fixed multiplexer on the D input selecting either the combinatorial output Y or the bypass signal BY. FFX selects between the combinatorial output X or the bypass signal BX.

The functionality of a slice storage element is identical to that described earlier for the I/O storage elements. All signals have programmable polarity; the default active-High function is described.

Table 12: **Storage Element Signals**

Signal	Description
D	Input. For a flip-flop data on the D input is loaded when R and S (or CLR and PRE) are Low and CE is High during the Low-to-High clock transition. For a latch, Q reflects the D input while the gate (G) input and gate enable (GE) are High and R and S (or CLR and PRE) are Low. The data on the D input during the High-to-Low gate transition is stored in the latch. The data on the Q output of the latch remains unchanged as long as G or GE remains Low.
Q	Output. Toggles after the Low-to-High clock transition for a flip-flop and immediately for a latch.
C	Clock for edge-triggered flip-flops.
G	Gate for level-sensitive latches.
CE	Clock Enable for flip-flops.
GE	Gate Enable for latches.
S	Synchronous Set (Q = High). When the S input is High and R is Low, the flip-flop is set, output High, during the Low-to-High clock (C) transition. A latch output is immediately set, output High.
R	Synchronous Reset (Q = Low); has precedence over Set.
PRE	Asynchronous Preset (Q = High). When the PRE input is High and CLR is Low, the flip-flop is set, output High, during the Low-to-High clock (C) transition. A latch output is immediately set, output High.
CLR	Asynchronous Clear (Q = Low); has precedence over Preset to reset Q output Low
SR	CLB input for R, S, CLR, or PRE
REV	CLB input for opposite of SR. Must be asynchronous or synchronous to match SR.

The control inputs R, S, CE, and C are all shared between the two flip-flops in a slice.

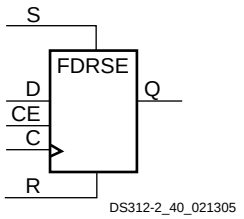


Figure 22: **FD Flip-Flop Component with Synchronous Reset, Set, and Clock Enable**

Table 13: **FD Flip-Flop Functionality with Synchronous Reset, Set, and Clock Enable**

Inputs					Outputs
R	S	CE	D	C	Q
1	X	X	X	↑	0
0	1	X	X	↑	1
0	0	0	X	X	No Change
0	0	1	1	↑	1
0	0	1	0	↑	0

Initialization

The CLB storage elements are initialized at power-up, during configuration, by the global GSR signal, and by the individual SR or REV inputs to the CLB.

Table 14: **Slice Storage Element Initialization**

Signal	Description
SR	Set/Reset input. Forces the storage element into the state specified by the attribute SRHIGH or SRLOW. SRHIGH forces a logic “1” when SR is asserted. SRLOW forces a logic “0”. For each slice, set and reset can be set to be synchronous or asynchronous.
REV	Reverse of Set/Reset input. A second input (BY) forces the storage element into the opposite state. The reset condition is predominant over the set condition if both are active. Same synchronous/asynchronous setting as for SR.
GSR	Global Set/Reset. GSR defaults to active High but can be inverted by adding an inverter in front of the GSR input of the STARTUP_SPARTAN3E element. The initial state after configuration or GSR is defined by a separate INIT0 and INIT1 attribute. By default, setting the SRLOW attribute sets INIT0, and setting the SRHIGH attribute sets INIT1.

Distributed RAM

The LUTs in the SLICEM can be programmed as distributed RAM. This type of memory affords moderate amounts of data buffering anywhere along a data path. One SLICEM LUT stores 16 bits (RAM16). The four LUT inputs F[4:1] or G[4:1] become the address lines labeled A[4:1] in the device model and A[3:0] in the design components, providing a 16x1 configuration in one LUT. Multiple SLICEM LUTs can be combined in various ways to store larger amounts of data, including 16x4, 32x2, or 64x1 configurations in one CLB. The fifth and sixth address lines required for the 32-deep and 64-deep configurations, respectively, are implemented using the BX and BY inputs, which connect to the write enable logic for writing and the F5MUX and F6MUX for reading.

Writing to distributed RAM is always synchronous to the SLICEM clock (WCLK for distributed RAM) and enabled by the SLICEM SR input which functions as the active-High Write Enable (WE). The read operation is asynchronous, and, therefore, during a write, the output initially reflects the old data at the address being written.

The distributed RAM outputs can be captured using the flip-flops within the SLICEM element. The WE write-enable control for the RAM and the CE clock-enable control for the flip-flop are independent, but the WCLK and CLK clock inputs are shared. Because the RAM read operation is asynchronous, the output data always reflects the currently addressed RAM location.

A dual-port option combines two LUTs so that memory access is possible from two independent data lines. The same data is written to both 16x1 memories but they have independent read address lines and outputs. The dual-port function is implemented by cascading the G-LUT address lines, which are used for both read and write, to the F-LUT write address lines (WF[4:1] in Figure 12), and by cascading the G-LUT data input D1 through the DIF_MUX in Figure 12 and to the D1 input on the F-LUT. One CLB provides a 16x1 dual-port memory as shown in Figure 23.

Any write operation on the D input and any read operation on the SPO output can occur simultaneously with and independently from a read operation on the second read-only port, DPO.

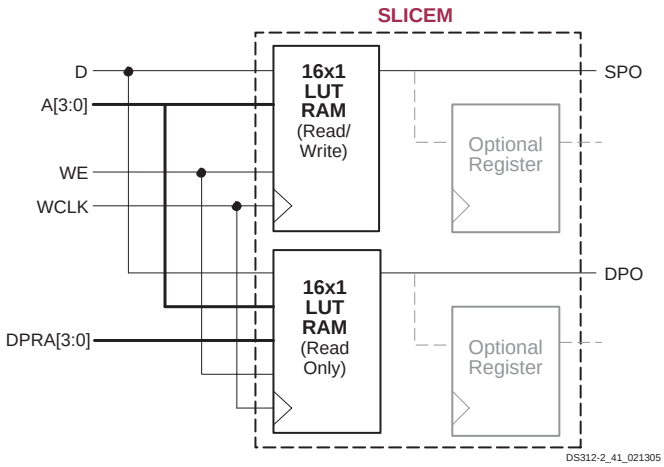


Figure 23: **RAM16X1D Dual-Port Usage**

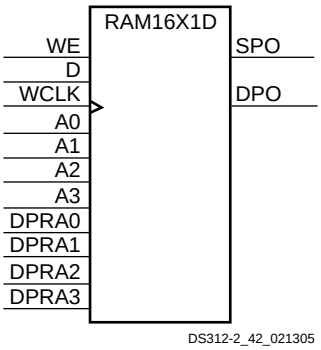


Figure 24: **Dual-Port RAM Component**

Table 15: **Dual-Port RAM Function**

Inputs			Outputs	
WE (mode)	**WCLK**	**D**	**SPO**	**DPO**
0 (read)	X	X	data_a	data_d
1 (read)	0	X	data_a	data_d
1 (read)	1	X	data_a	data_d
1 (write)	↑	D	D	data_d
1 (read)	↓	X	data_a	data_d

Notes:

1. data_a = word addressed by bits A3-A0.
2. data_d = word addressed by bits DPRA3-DPRA0.

Table 16: **Distributed RAM Signals**

Signal	Description
WCLK	The clock is used for synchronous writes. The data and the address input pins have setup times referenced to the WCLK pin. Active on the positive edge by default with built-in programmable polarity.
WE	The enable pin affects the write functionality of the port. An inactive Write Enable prevents any writing to memory cells. An active Write Enable causes the clock edge to write the data input signal to the memory location pointed to by the address inputs. Active High by default with built-in programmable polarity.

Table 16: **Distributed RAM Signals *(Continued)***

Signal	Description
A0, A1, A2, A3 (A4, A5)	The address inputs select the memory cells for read or write. The width of the port determines the required address inputs.
D	The data input provides the new data value to be written into the RAM.
O, SPO, and DPO	The data output O on single-port RAM or the SPO and DPO outputs on dual-port RAM reflects the contents of the memory cells referenced by the address inputs. Following an active write clock edge, the data out (O or SPO) reflects the newly written data.

The INIT attribute can be used to preload the memory with data during FPGA configuration. The default initial contents for RAM is all zeros. If the WE is held Low, the element can be considered a ROM. The ROM function is possible even in the SLICEL.

The global write enable signal, GWE, is asserted automatically at the end of device configuration to enable all writable elements. The GWE signal guarantees that the initialized distributed RAM contents are not disturbed during the configuration process.

The distributed RAM is useful for smaller amounts of memory. Larger memory requirements can use the dedicated 18Kbit RAM blocks (see **Block RAM**).

For more information on distributed RAM, see XAPP464: "Using Look-Up Tables as Distributed RAM in Spartan-3 FPGAs".

Shift Registers

It is possible to program each SLICEM LUT as a 16-bit shift register (see Figure 25). Used in this way, each LUT can delay serial data anywhere from 1 to 16 clock cycles without using any of the dedicated flip-flops. The resulting programmable delays can be used to balance the timing of data pipelines.

The SLICEM LUTs cascade from the G-LUT to the F-LUT through the DIFMUX (see Figure 12). SHIFTIN and SHIFTOUT lines cascade a SLICEM to the SLICEM below to form larger shift registers. The four SLICEM LUTs of a single CLB can be combined to produce delays up to 64 clock cycles. It is also possible to combine shift registers across more than one CLB.

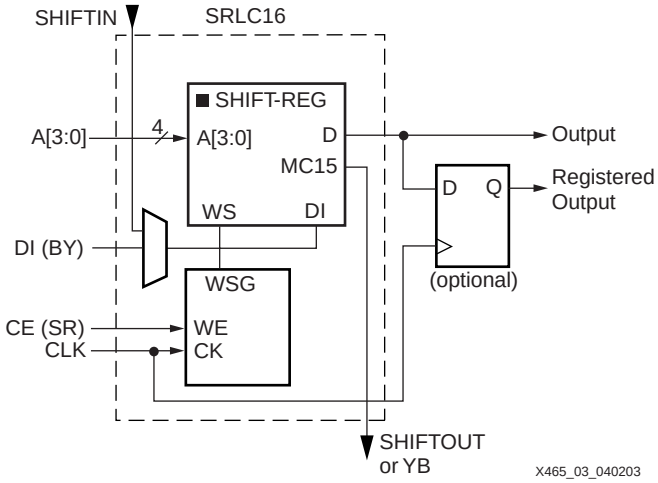


Figure 25: **Logic Cell SRL16 Structure**

Each shift register provides a shift output MC15 for the last bit in each LUT, in addition to providing addressable access to any bit in the shift register through the normal D output. The address inputs A[3:0] are the same as the distributed RAM address lines, which come from the LUT inputs F[4:1] or G[4:1]. At the end of the shift register, the CLB flip-flop can be used to provide one more shift delay for the addressable bit.

The shift register element is known as the SRL16 (Shift Register LUT 16-bit), with a 'C' added to signify a cascade ability (Q15 output) and 'E' to indicate a Clock Enable. See Figure 26 for an example of the SRLC16E component.

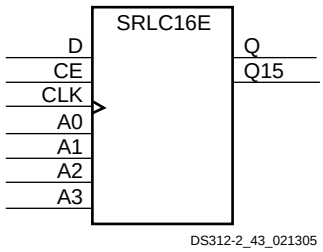


Figure 26: **SRL16 Shift Register Component with Cascade and Clock Enable**

The functionality of the shift register is shown in Table 17. The SRL16 shifts on the rising edge of the clock input when the Clock Enable control is High. This shift register cannot be initialized either during configuration or during operation except by shifting data into it. The clock enable and clock inputs are shared between the two LUTs in a SLICEM. The clock enable input is automatically kept active if unused.

Table 17: **SRL16 Shift Register Function**

Inputs				Outputs	
Am	**CLK**	**CE**	**D**	**Q**	**Q15**
Am	X	0	X	Q[Am]	Q[15]
Am	↑	1	D	Q[Am-1]	Q[15]

Notes:

1. m = 0, 1, 2, 3.

For more information on the SRL16, refer to XAPP465: "Using Look-Up Tables as Shift Registers (SRL16) in Spartan-3 FPGAs".

Block RAM

Spartan-3E devices incorporate 4 to 36 dedicated block RAMs, which are organized as dual-port configurable 18 Kbit blocks. Functionally, the block RAM is identical to the Spartan-3 architecture block RAM. Block RAM synchronously stores large amounts of data while distributed RAM, previously described, is better suited for buffering small amounts of data anywhere along signal paths. This section describes basic block RAM functions. For detailed implementation information, refer to XAPP463: "Using Block RAM in Spartan-3 Series FPGAs".

Each block RAM is configurable by setting the content's initial values, default signal value of the output registers, port aspect ratios, and write modes. Block RAM can be used in single-port or dual-port modes.

Arrangement of RAM Blocks on Die

The block RAMs are located together with the multipliers on the die in one or two columns depending on the size of the device. The XC3S100E has one column of block RAM. The Spartan-3E devices ranging from the XC3S250E to XC3S1600E have two columns of block RAM. Table 18 shows the number of RAM blocks, the data storage capacity, and the number of columns for each device. Row(s) of CLBs are located above and below each block RAM column.

Table 18: **Number of RAM Blocks by Device**

Device	Total Number of RAM Blocks	Total Addressable Locations (bits)	Number of Columns
XC3S100E	4	73,728	1
XC3S250E	12	221,184	2
XC3S500E	20	368,640	2
XC3S1200E	28	516,096	2
XC3S1600E	36	663,552	2

Immediately adjacent to each block RAM is an embedded 18x18 hardware multiplier. The upper 16 bits of the block RAM's Port A Data input bus are shared with the upper 16 bits of the A multiplicand input bus of the multiplier. Similarly, the upper 16 bits of Port B's data input bus are shared with the B multiplicand input bus of the multiplier. Details on the block RAM's shared connectivity with the multipliers are located in XAPP463.

The Internal Structure of the Block RAM

The block RAM has a dual port structure. The two identical data ports called A and B permit independent access to the common block RAM, which has a maximum capacity of 18,432 bits, or 16,384 bits with no parity bits (see parity bits description in Table 19). Each port has its own dedicated set of data, control, and clock lines for synchronous read and write operations. There are four basic data paths, as shown in Figure 27:

1. Write to and read from Port A
2. Write to and read from Port B
3. Data transfer from Port A to Port B
4. Data transfer from Port B to Port A

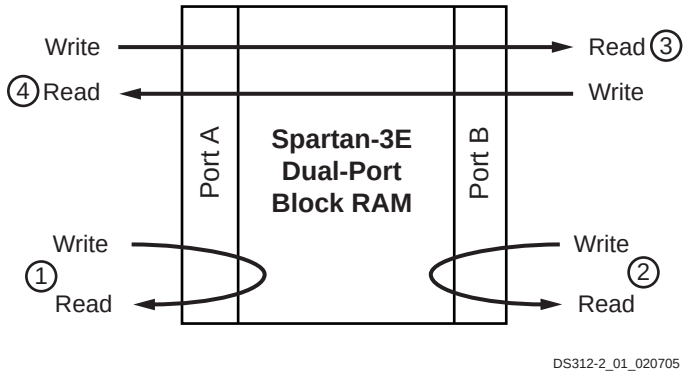


Figure 27: **Block RAM Data Paths**

Number of Ports

A choice among primitives determines whether the block RAM functions as dual- or single-port memory. A name of the form RAMB16_S[w_A]_S[w_B] calls out the dual-port primitive, where the integers w_A and w_B specify the total data path width at ports A and B, respectively. Thus, a RAMB16_S9_S18 is a dual-port RAM with a 9-bit Port A and an 18-bit Port B. A name of the form RAMB16_S[w] identifies the single-port primitive, where the integer w specifies the total data path width of the lone port A. A RAMB16_S18 is a single-port RAM with an 18-bit port.

Port Aspect Ratios

Each port of the block RAM can be configured independently to select a number of different possible widths for the data input (DI) and data output (DO) signals as shown in Table 19.

Table 19: **Port Aspect Ratios**

Total Data Path Width (w bits)	DI/DO Data Bus Width (w-p bits)[1]	DIP/DOP Parity Bus Width (p bits)	ADDR Bus Width (r bits)[2]	DI/DO [w-p-1:0]	DIP/DOP [p-1:0]	ADDR [r-1:0]	No. of Addressable Locations (n)[3]	Block RAM Capacity (w*n bits)[4]
1	1	0	14	[0:0]	-	[13:0]	16,384	16,384
2	2	0	13	[1:0]	-	[12:0]	8,192	16,384
4	4	0	12	[3:0]	-	[11:0]	4,096	16,384
9	8	1	11	[7:0]	[0:0]	[10:0]	2,048	18,432
18	16	2	10	[15:0]	[1:0]	[9:0]	1,024	18,432
36	32	4	9	[31:0]	[3:0]	[8:0]	512	18,432

Notes:

1. The width of the total data path (w) is the sum of the DI/DO bus width (w-p) and any parity bits (p).
2. The width selection made for the DI/DO bus determines the number of address lines (r) according to the relationship expressed as: $r = 14 – [\log(w–p)/\log(2)]$.
3. The number of address lines delimits the total number (n) of addressable locations or depth according to the following equation: $n = 2^r$.
4. The product of w and n yields the total block RAM capacity.

If the data bus width of Port A differs from that of Port B, the block RAM automatically performs a bus-matching function as described in Figure 28. When data is written to a port with a narrow bus and then read from a port with a wide bus, the latter port effectively combines “narrow” words to form “wide” words. Similarly, when data is written into a port with a wide bus and then read from a port with a narrow bus, the latter port divides “wide” words to form “narrow” words. Parity bits are not available if the data port width is configured as x4, x2, or x1. For example, if a x36 data word (32 data, 4 parity) is addressed as two x18 halfwords (16 data, 2 parity), the parity bits associated with each data byte are mapped within the block RAM to the appropriate parity bits. The same effect happens when the x36 data word is mapped as four x9 words.

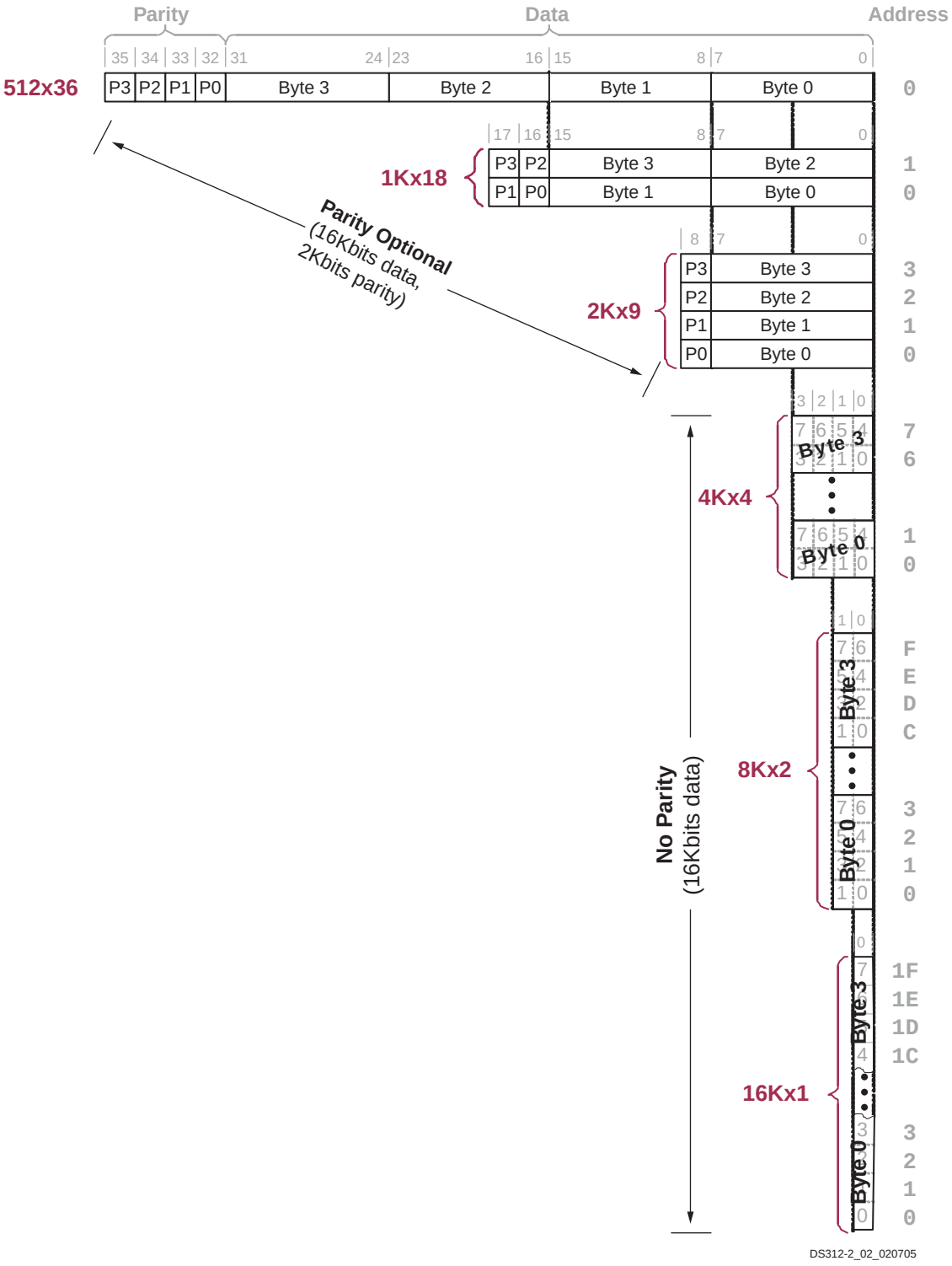
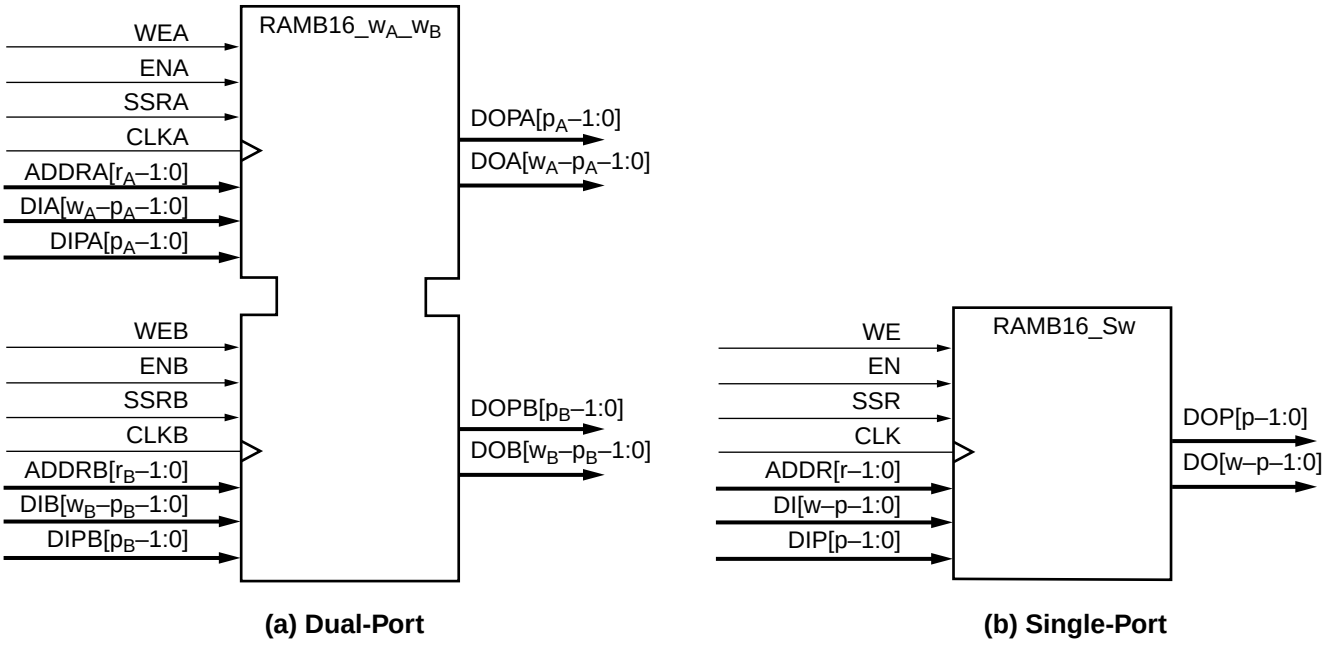


Figure 28: **Data Organization and Bus-matching Operation with Different Port Widths on Port A and Port B**

Block RAM Port Signal Definitions

Representations of the dual-port primitive RAMB16_S[w_A]_S[w_B] and the single-port primitive RAMB16_S[w] with their associated signals are shown in Figure 29a and Figure 29b, respectively. These signals are defined in Table 20. The control signals (WE, EN, CLK, and SSR) on the block RAM are active High. However, optional inverters on the control signals change the polarity of the active edge to active Low.



Notes:

1. w_A and w_B are integers representing the total data path width (i.e., data bits plus parity bits) at Ports A and B, respectively.
2. p_A and p_B are integers that indicate the number of data path lines serving as parity bits.
3. r_A and r_B are integers representing the address bus width at ports A and B, respectively.
4. The control signals CLK, WE, EN, and SSR on both ports have the option of inverted polarity.

Figure 29: **Block RAM Primitives**

Table 20: **Block RAM Port Signals**

Signal Description	Port A Signal Name	Port B Signal Name	Direction	Function
Address Bus	ADDRA	ADDRB	Input	The Address Bus selects a memory location for read or write operations. The width (w) of the port's associated data path determines the number of available address lines (r), as per Table 18.
Data Input Bus	DIA	DIB	Input	Data at the DI input bus is written to the RAM location specified by the address input bus (ADDR) during the active edge of the CLK input, when the clock enable (EN) and write enable (WE) inputs are active. It is possible to configure a port's DI input bus width (w-p) based on Table 18. This selection applies to both the DI and DO paths of a given port.
Parity Data Input(s)	DIPA	DIPB	Input	Parity inputs represent additional bits included in the data input path. Although referred to herein as "parity" bits, the parity inputs and outputs have no special functionality for generating or checking parity and can be used as additional data bits. The number of parity bits 'p' included in the DI (same as for the DO bus) depends on a port's total data path width (w). See Table 18.

Table 20: **Block RAM Port Signals *(Continued)***

Signal Description	Port A Signal Name	Port B Signal Name	Direction	Function
Data Output Bus	DOA	DOB	Output	Data is written to the DO output bus from the RAM location specified by the address input bus, ADDR. See the DI signal description for DO port width configurations. Basic data access occurs on the active edge of the CLK when WE is inactive and EN is active. The DO outputs mirror the data stored in the address ADDR memory location. Data access with WE active if the WRITE_MODE attribute is set to the value: WRITE_FIRST, which accesses data after the write takes place. READ_FIRST accesses data before the write occurs. A third attribute, NO_CHANGE, latches the DO outputs upon the assertion of WE. See **Block RAM Data Operations** for details on the WRITE_MODE attribute.
Parity Data Output(s)	DOPA	DOPB	Output	Parity outputs represent additional bits included in the data input path. The number of parity bits 'p' included in the DI bus (same as for the DO bus) depends on a port's total data path width (w). See the DIP signal description for configuration details.
Write Enable	WEA	WEB	Input	When asserted together with EN, this input enables the writing of data to the RAM. When WE is inactive with EN asserted, read operations are still possible. In this case, a latch passes data from the addressed memory location to the DO outputs.
Clock Enable	ENA	ENB	Input	When asserted, this input enables the CLK signal to perform read and write operations to the block RAM. When inactive, the block RAM does not perform any read or write operations.
Set/Reset	SSRA	SSRB	Input	When asserted, this pin forces the DO output latch to the value of the SRVAL attribute. It is synchronized to the CLK signal.
Clock	CLKA	CLKB	Input	This input accepts the clock signal to which read and write operations are synchronized. All associated port inputs are required to meet setup times with respect to the clock signal's active edge. The data output bus responds after a clock-to-out delay referenced to the clock signal's active edge.

Block RAM Attribute Definitions

A block RAM has a number of attributes that control its behavior as shown in Table 21.

Table 21: **Block RAM Attributes**

Function	Attribute	Possible Values
Initial Content for Data Memory, Loaded during Configuration	`INITxx` (INIT_00 through INIT3F)	Each initialization string defines 32 hex values of the 16384-bit data memory of the block RAM.
Initial Content for Parity Memory, Loaded during Configuration	`INITPxx` (INITP_00 through INITP0F)	Each initialization string defines 32 hex values of the 2048-bit parity data memory of the block RAM.
Data Output Latch Initialization	`INIT` (single-port) `INITA`, `INITB` (dual-port)	Hex value the width of the chosen port.

Table 21: **Block RAM Attributes *(Continued)***

Function	Attribute	Possible Values
Data Output Latch Synchronous Set/Reset Value	SRVAL (single-port) SRVAL_A, SRVAL_B (dual-port)	Hex value the width of the chosen port.
Data Output Latch Behavior during Write (see **Block RAM Data Operations**)	WRITE_MODE	WRITE_FIRST, READ_FIRST, NO_CHANGE

Block RAM Data Operations

Writing data to and accessing data from the block RAM are synchronous operations that take place independently on each of the two ports. Table 22 describes the data operations of each port as a result of the block RAM control signals in their default active-High edges.

The waveforms for the write operation are shown in the top half of Figure 30, Figure 31, and Figure 32. When the WE and EN signals enable the active edge of CLK, data at the DI input bus is written to the block RAM location addressed by the ADDR lines.

Table 22: **Block RAM Function Table**

Input Signals								Output Signals		RAM Data	
GSR	**EN**	**SSR**	**WE**	**CLK**	**ADDR**	**DIP**	**DI**	**DOP**	**DO**	**Parity**	**Data**
Immediately After Configuration											
Loaded During Configuration								X	X	INITP_xx	INIT_xx
Global Set/Reset Immediately After Configuration											
1	X	X	X	X	X	X	X	INIT	INIT	No Chg	No Chg
RAM Disabled											
0	0	X	X	X	X	X	X	No Chg	No Chg	No Chg	No Chg
Synchronous Set/Reset											
0	1	1	0	↑	X	X	X	SRVAL	SRVAL	No Chg	No Chg
Synchronous Set/Reset During Write RAM											
0	1	1	1	↑	addr	pdata	Data	SRVAL	SRVAL	RAM(addr) ← pdata	RAM(addr) ← data
Read RAM, no Write Operation											
0	1	0	0	↑	addr	X	X	RAM(pdata)	RAM(data)	No Chg	No Chg
Write RAM, Simultaneous Read Operation											
0	1	0	1	↑	addr	pdata	Data	WRITE_MODE = WRITE_FIRST			
								pdata	data	RAM(addr) ← pdata	RAM(addr) ← data
								WRITE_MODE = READ_FIRST			
								RAM(data)	RAM(data)	RAM(addr) ← pdata	RAM(addr) ← pdata
								WRITE_MODE = NO_CHANGE			
								No Chg	No Chg	RAM(addr) ← pdata	RAM(addr) ← pdata

There are a number of different conditions under which data can be accessed at the DO outputs. Basic data access always occurs when the WE input is inactive. Under this condition, data stored in the memory location addressed by the ADDR lines passes through a output latch to the DO outputs. The timing for basic data access is shown in the portions of Figure 30, Figure 31, and Figure 32 during which WE is Low.

Data also can be accessed on the DO outputs when asserting the WE input based on the value of the WRITE_MODE attribute as described in Table 23.

Table 23: **WRITE_MODE Effect on Data Output Latches During Write Operations**

Write Mode	Effect on Same Port	Effect on Opposite Port (dual-port only with same address)
WRITE_FIRST Read After Write	Data on DI and DIP inputs is written into specified RAM location and simultaneously appears on DO and DOP outputs.	Invalidates data on DO and DOP outputs.
READ_FIRST Read Before Write	Data from specified RAM location appears on DO and DOP outputs. Data on DI and DIP inputs is written into specified location.	Data from specified RAM location appears on DO and DOP outputs.
NO_CHANGE No Read on Write	Data on DO and DOP outputs remains unchanged. Data on DI and DIP inputs is written into specified location.	Invalidates data on DO and DOP outputs.

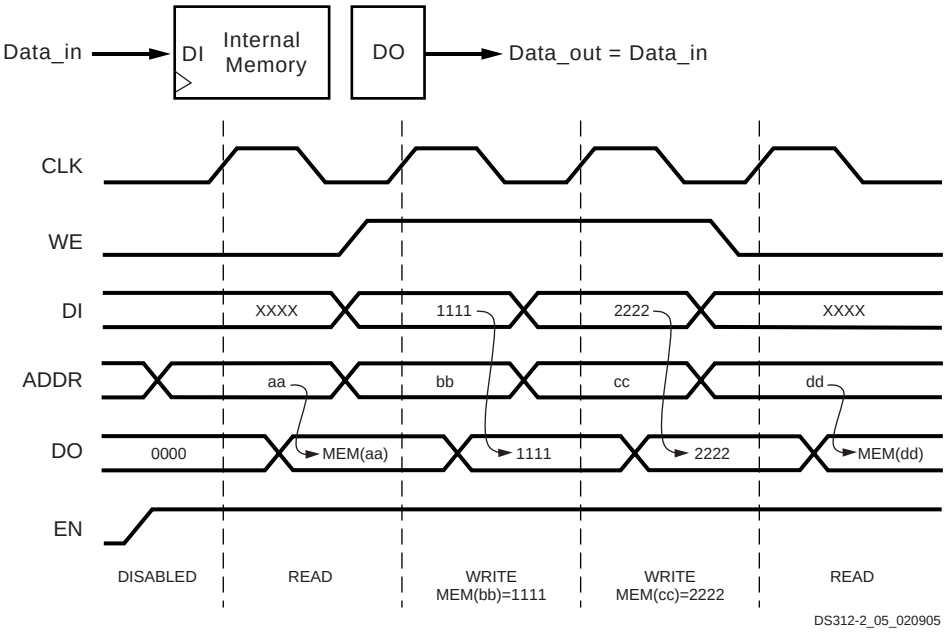


Figure 30: **Waveforms of Block RAM Data Operations with WRITE_FIRST Selected**

Setting the WRITE_MODE attribute to a value of WRITE_FIRST, data is written to the addressed memory location on an enabled active CLK edge and is also passed to the DO outputs. WRITE_FIRST timing is shown in the portion of Figure 30 during which WE is High.

Setting the WRITE_MODE attribute to a value of READ_FIRST, data already stored in the addressed location passes to the DO outputs before that location is overwritten with new data from the DI inputs on an enabled active CLK edge. READ_FIRST timing is shown in the portion of Figure 31 during which WE is High.

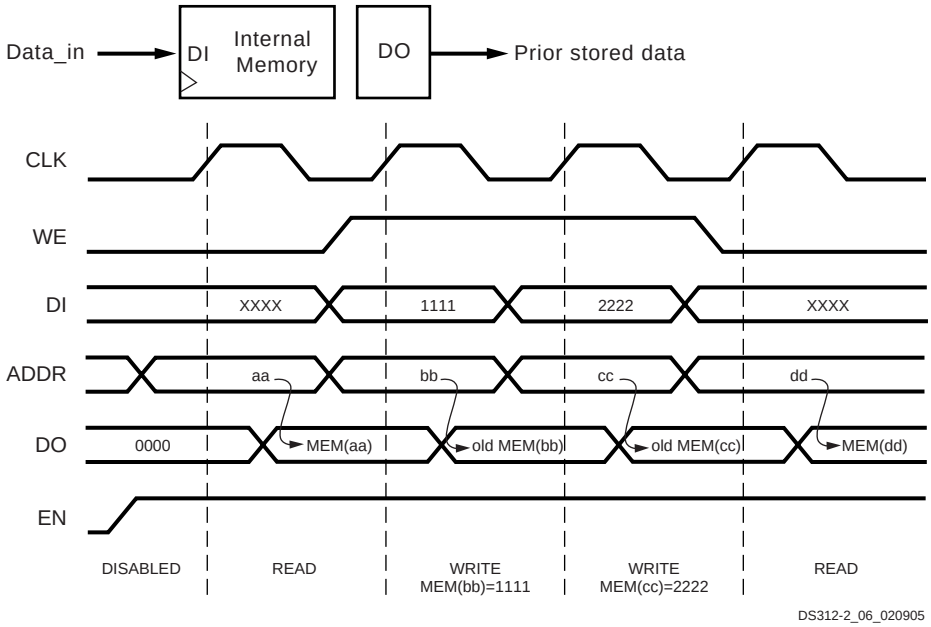


Figure 31: **Waveforms of Block RAM Data Operations with READ_FIRST Selected**

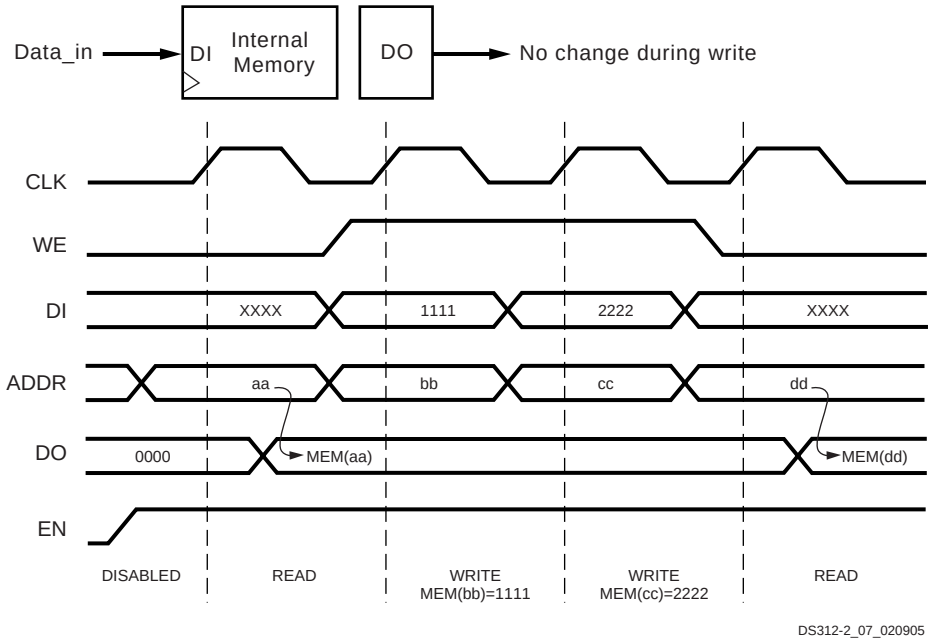


Figure 32: **Waveforms of Block RAM Data Operations with NO_CHANGE Selected**

Setting the WRITE_MODE attribute to a value of NO_CHANGE, puts the DO outputs in a latched state when asserting WE. Under this condition, the DO outputs retain the data driven just before WE is asserted. NO_CHANGE timing is shown in the portion of Figure 32 during which WE is High.

Dedicated Multipliers

The Spartan-3E devices provide 4 to 36 dedicated multiplier blocks per device. The multipliers are located together with the block RAM in one or two columns depending on device density. See **Arrangement of RAM Blocks on Die** for details on the location of these blocks and their connectivity.

The multiplier blocks primarily perform two's complement numerical multiplication but can also perform some less obvious applications such as simple data storage and barrel shifting. Logic slices also implement efficient small multipliers and thereby supplement the dedicated multipliers. The Spartan-3E dedicated multiplier blocks have additional features beyond those provided in Spartan-3 FPGAs.

Each multiplier performs the principle operation P = A × B, where 'A' and 'B' are 18-bit words in two's complement form, and 'P' is the full-precision 36-bit product, also in two's complement form. The 18-bit inputs represent values ranging from $-131{,}072_{10}$ to $+131{,}071_{10}$ with a resulting product ranging from $-17{,}179{,}738{,}112_{10}$ to $+17{,}179{,}869{,}184_{10}$.

Implement multipliers with inputs less than 18 bits by sign-extending the inputs (i.e., replicating the most-significant bit). Wider multiplication operations are performed by combining the dedicated multipliers and slice-based logic in any viable combination or by time-sharing a single multiplier. Perform unsigned multiplication by restricting the inputs to the positive range. Tie the most-significant bit Low and represent the unsigned value in the remaining 17 lesser-significant bits.

As shown in Figure 33, each multiplier block has optional registers on each of the multiplier inputs and the output. The registers are named AREG, BREG, and PREG and can be used in any combination. The clock input is common to all the registers within a block, but each register has an independent clock enable and synchronous reset controls making them ideal for storing data samples and coefficients. When used for pipelining, the registers boost the multiplier clock rate, beneficial for higher performance applications.

Figure 33 illustrates the principle features of the multiplier block.

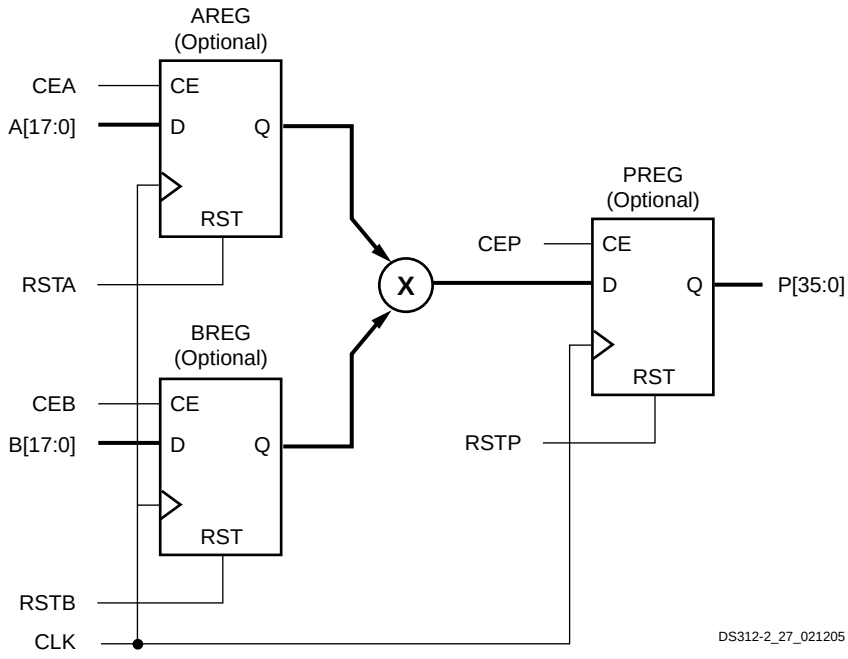


Figure 33: **Principle Ports and Functions of Dedicated Multiplier Blocks**

Use the MULT18X18SIO primitive shown in Figure 34 to instantiate a multiplier within a design. Although high-level logic synthesis software usually automatically infers a multiplier, adding the pipeline registers usually requires the MULT18X18SIO primitive. Connect the appropriate signals to the MULT18X18SIO multiplier ports and set the individual AREG, BREG, and PREG attributes to '1' to insert the associated register, or to 0 to remove it and make the signal path combinatorial.

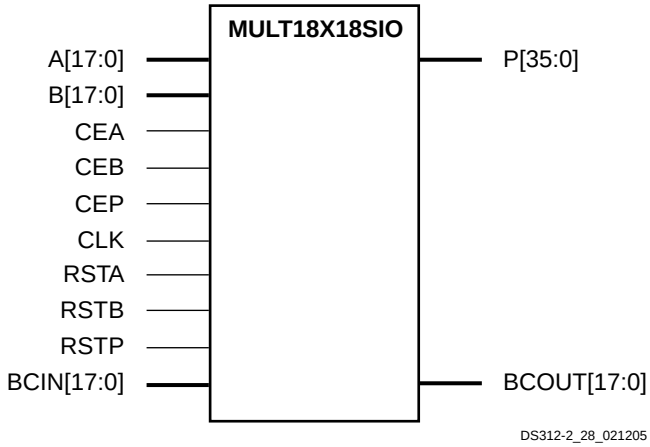


Figure 34: **MULT18X18SIO Primitive**

The MULT18X18SIO primitive has two additional ports called BCIN and BCOUT to cascade or share the multiplier's 'B' input among several multiplier bocks. The 18-bit BCIN "cascade" input port offers an alternate input source from the more typical 'B' input. The B_INPUT attribute specifies whether the specific implementation uses the BCIN or 'B' input path. Setting B_INPUT to DIRECT chooses the 'B' input. Setting B_INPUT to CASCADE selects the alternate BCIN input. The BREG register then optionally holds the selected input value, if required.

BCOUT is an 18-bit output port that always reflects the value that is applied to the multiplier's second input, which is either the 'B' input, the cascaded value from the BCIN input, or the output of the BREG if it is inserted.

Figure 35 illustrates the four possible configurations using different settings for the B_INPUT attribute and the BREG attribute.

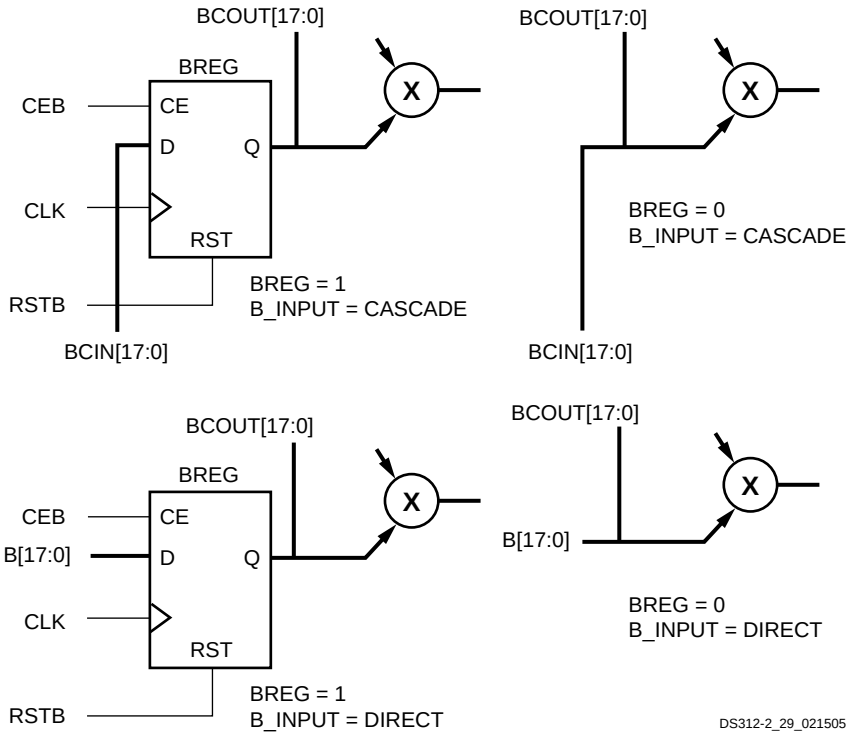


Figure 35: **Four Configurations of the B Input**

The BCIN and BCOUT ports have associated dedicated routing that connects adjacent multipliers within the same column. Via the cascade connection, the BCOUT port of one multiplier block drives the BCIN port of the multiplier block directly above it. There is no connection to the BCIN port of the bottom-most multiplier block in a column or a connection from the BCOUT port of the top-most block in a column. As an example, Figure 36 shows the multiplier cascade capability within the XC3S100E FPGA, which has a single column of multiplier, four blocks tall. For clarity, the figure omits the register control inputs.

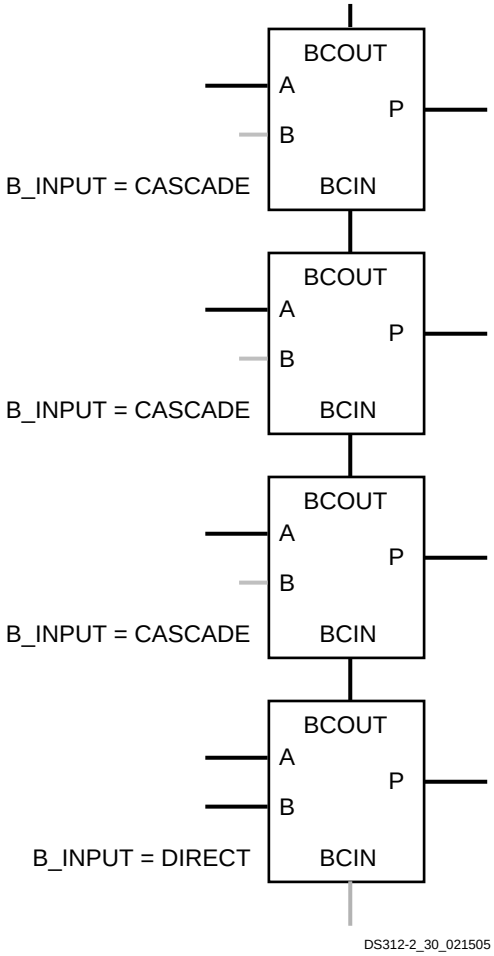


Figure 36: **Multiplier Cascade Connection**

When using the BREG register, the cascade connection forms a shift register structure typically used in DSP algorithms such as direct-form FIR filters. When the BREG register is omitted, the cascade structure essentially feeds the same input value to more than one multiplier. This parallel connection serves to create wide-input multipliers, implement transpose FIR filters, and is used in any application that requires that several multipliers have the same input value.

Table 24 defines each port of the MULT18X18SIO primitive.

Table 24: **MULT18X18SIO Embedded Multiplier Primitives Description**

Signal Name	Direction	Function
A[17:0]	Input	The primary 18-bit two's complement value for multiplication. The block multiplies by this value asynchronously if the optional AREG and PREG registers are omitted. When AREG and/or PREG are used, the value provided on this port is qualified by the rising edge of CLK, subject to the appropriate register controls.
B[17:0]	Input	The second 18-bit two's complement value for multiplication if the B_INPUT attribute is set to DIRECT. The block multiplies by this value asynchronously if the optional BREG and PREG registers are omitted. When BREG and/or PREG are used, the value provided on this port is qualified by the rising edge of CLK, subject to the appropriate register controls.
BCIN[17:0]	Input	The second 18-bit two's complement value for multiplication if the B_INPUT attribute is set to CASCADE. The block multiplies by this value asynchronously if the optional BREG and PREG registers are omitted. When BREG and/or PREG are used, the value provided on this port is qualified by the rising edge of CLK, subject to the appropriate register controls.
P[35:0]	Output	The 36-bit two's complement product resulting from the multiplication of the two input values applied to the multiplier. If the optional AREG, BREG and PREG registers are omitted, the output operates asynchronously. Use of PREG causes this output to respond to the rising edge of CLK with the value qualified by CEP and RSTP. If PREG is omitted, but AREG and BREG are used, this output responds to the rising edge of CLK with the value qualified by CEA, RSTA, CEB, and RSTB. If PREG is omitted and only one of AREG or BREG is used, this output responds to both asynchronous and synchronous events.
BCOUT[17:0]	Output	The value being applied to the second input of the multiplier. When the optional BREG register is omitted, this output responds asynchronously in response to changes at the B[17:0] or BCIN[17:0] ports according to the setting of the B_INPUT attribute. If BREG is used, this output responds to the rising edge of CLK with the value qualified by CEB and RSTB.
CEA	Input	Clock enable qualifier for the optional AREG register. The value provided on the A[17:0] port is captured by AREG in response to a rising edge of CLK when this signal is High, provided that RSTA is Low.
RSTA	Input	Synchronous reset for the optional AREG register. AREG content is forced to the value zero in response to a rising edge of CLK when this signal is High.
CEB	Input	Clock enable qualifier for the optional BREG register. The value provided on the B[17:0] or BCIN[17:0] port is captured by BREG in response to a rising edge of CLK when this signal is High, provided that RSTB is Low.
RSTB	Input	Synchronous reset for the optional BREG register. BREG content is forced to the value zero in response to a rising edge of CLK when this signal is High.
CEP	Input	Clock enable qualifier for the optional PREG register. The value provided on the output of the multiplier port is captured by PREG in response to a rising edge of CLK when this signal is High, provided that RSTP is Low.
RSTP	Input	Synchronous reset for the optional PREG register. PREG content is forced to the value zero in response to a rising edge of CLK when this signal is High.

Notes:

1. The control signals CLK, CEA, RSTA, CEB, RSTB, CEP, and RSTP have the option of inverted polarity.

Digital Clock Managers (DCMs)

Differences from the Spartan-3 Architecture

- Spartan-3E FPGAs have two, four, or eight DCMs, depending on device size.
- The Spartan-3E DCM has a maximum phase shift range of ±180°. The Spartan-3 DCM range is ±360°.
- The Spartan-3E DLL supports lower input frequencies, down to 5 MHz. Spartan-3 DLLs supports down to 24 MHz.

Overview

Spartan-3E Digital Clock Managers (DCMs) provide flexible, complete control over clock frequency, phase shift and skew. To accomplish this, the DCM employs a Delay-Locked Loop (DLL), a fully digital control system that uses feedback to maintain clock signal characteristics with a high degree of precision despite normal variations in operating temperature and voltage. This section provides a fundamental description of the DCM. See XAPP462: "Using Digital Clock Managers (DCMs) in Spartan-3 Series FPGAs" for further information.

The XC3S100E FPGA has two DCMs, one at the top and one at the bottom of the device. The XC3S250E and XC3S500E FPGAs each include four DCMs, two at the top and two at the bottom. The XC3S1200E and XC3S1600E FPGAs contain eight DCMs with two on each edge (see also Figure 42). The DCM in Spartan-3E FPGAs is surrounded by CLBs within the logic array and is no longer located at the top and bottom of a column of block RAM as in the Spartan-3 architecture,. The Digital Clock Manager is instantiated into a design as the “DCM” primitive.

The DCM supports three major functions:

- **Clock-skew Elimination:** Clock skew describes the extent to which clock signals may, under normal circumstances, deviate from zero-phase alignment. It occurs when slight differences in path delays cause the clock signal to arrive at different points on the die at different times. This clock skew can increase set-up and hold time requirements as well as clock-to-out time, which may be undesirable in applications operating at a high frequency, when timing is critical. The DCM eliminates clock skew by aligning the output clock signal it generates with another version of the clock signal that is fed back. As a result, the two clock signals establish a zero-phase relationship. This effectively cancels out clock distribution delays that might lie in the signal path leading from the clock output of the DCM to its feedback input.
- **Frequency Synthesis:** Provided with an input signal, the DCM can generate a wide range of different output clock frequencies. This is accomplished by either multiplying and/or dividing the frequency of the input clock signal by any of several different factors.
- **Phase Shifting:** The DCM provides the ability to shift the phase of all its output clock signals with respect to its input clock signal.

The DCM has four functional components: the Delay-Locked Loop (DLL), the Digital Frequency Synthesizer (DFS), the Phase Shifter (PS), and the Status Logic. Each component has its associated signals, as shown in Figure 37.

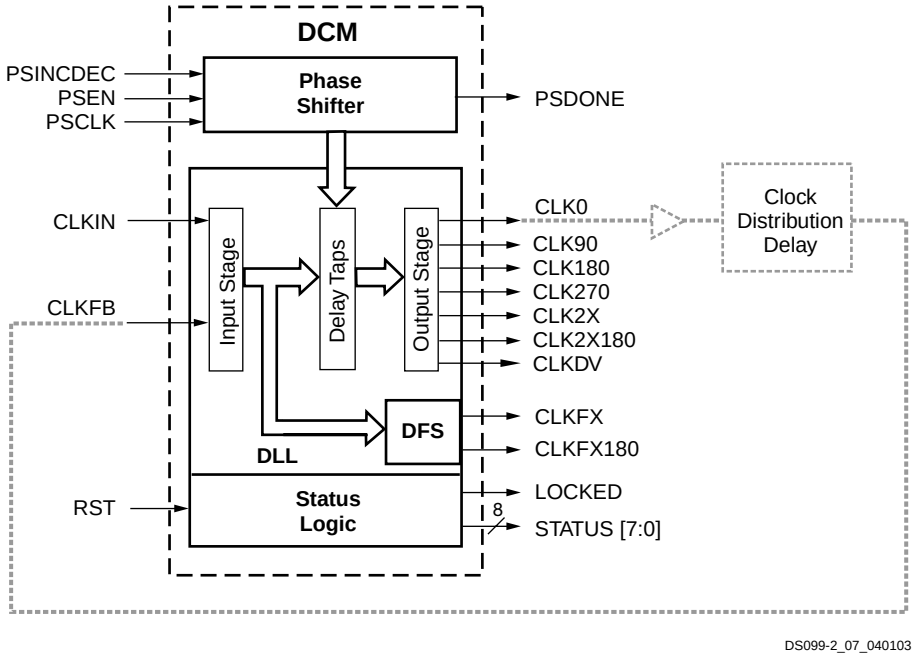


Figure 37: **DCM Functional Blocks and Associated Signals**

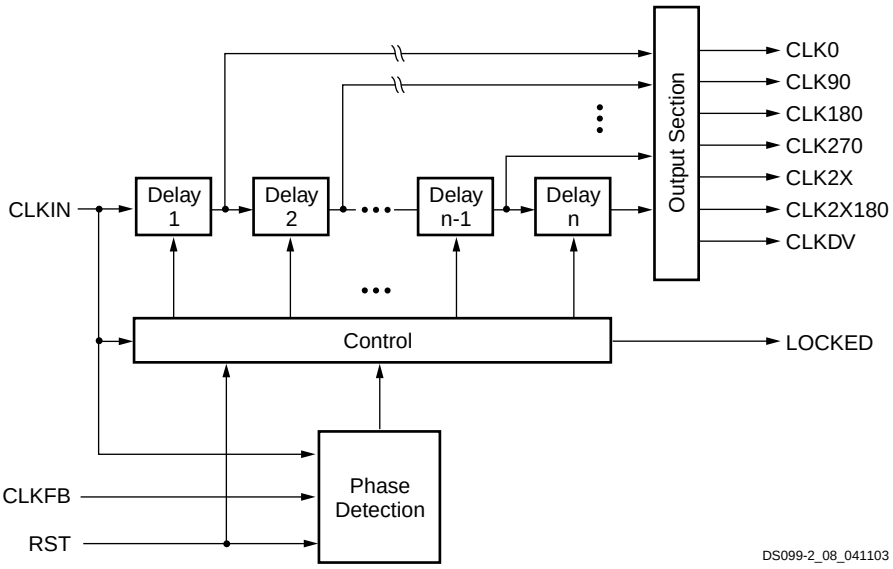


Figure 38: **Simplified Functional Diagram of DLL**

Table 25: **DLL Signals**

Signal	Direction	Description
CLKIN	Input	Accepts original clock signal.
CLKFB	Input	Accepts either CLK0 or CLK2X as the feedback signal. (Set CLK_FEEDBACK attribute accordingly).
CLK0	Output	Generates a clock signal with same frequency and phase as CLKIN.
CLK90	Output	Generates a clock signal with same frequency as CLKIN, only phase-shifted 90°.
CLK180	Output	Generates a clock signal with same frequency as CLKIN, only phase-shifted 180°.
CLK270	Output	Generates a clock signal with same frequency as CLKIN, only phase-shifted 270°.
CLK2X	Output	Generates a clock signal with same phase as CLKIN, only twice the frequency.
CLK2X180	Output	Generates a clock signal with twice the frequency of CLKIN, phase-shifted 180° with respect to CLKIN.
CLKDV	Output	Divides the CLKIN frequency by CLKDV_DIVIDE value to generate lower frequency clock signal that is phase-aligned to CLKIN.

Delay-Locked Loop (DLL)

The most basic function of the DLL component is to eliminate clock skew. The main signal path of the DLL consists of an input stage, followed by a series of discrete delay elements or *taps,* which in turn leads to an output stage. This path together with logic for phase detection and control forms a system complete with feedback as shown in Figure 38. In Spartan-3E FPGAs, the DLL is implemented using a counter-based delay line.

The DLL component has two clock inputs, CLKIN and CLKFB, as well as seven clock outputs, CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, and CLKDV as described in Table 25. The clock outputs drive simultaneously. Signals that initialize and report the state of the DLL are discussed in the Status Logic Component section.

The clock signal supplied to the CLKIN input serves as a reference waveform. The DLL seeks to align the rising-edge of feedback signal at the CLKFB input with the rising-edge of CLKIN input. When eliminating clock skew, the common approach to using the DLL is as follows: The CLK0 signal is passed through the clock distribution network to all the registers it synchronizes. These registers are either internal or external to the FPGA. After passing through the clock distribution network, the clock signal returns to the DLL via a feedback line called CLKFB. The control block inside the DLL measures the phase error between CLKFB and CLKIN.

This phase error is a measure of the clock skew that the clock distribution network introduces. The control block activates the appropriate number of delay elements to cancel out the clock skew. Once the DLL has brought the CLK0 signal in phase with the CLKIN signal, it asserts the LOCKED output, indicating a lock on to the CLKIN signal.

DLL Attributes and Related Functions

A number of different functional options can be set for the DLL component through the use of the attributes described in Table 26. Each attribute is described in detail in the sections that follow:

Table 26: **DLL Attributes**

Attribute	Description	Values
CLK_FEEDBACK	Chooses either the CLK0 or CLK2X output to drive the CLKFB input	NONE, 1X, 2X
CLKIN_DIVIDE_BY_2	Halves the frequency of the CLKIN signal just as it enters the DCM	TRUE, FALSE
CLKDV_DIVIDE	Selects the constant used to divide the CLKIN input frequency to generate the CLKDV output frequency	1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6.0, 6.5, 7.0, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, and 16
DUTY_CYCLE_CORRECTION	Enables 50% duty cycle correction for the CLK0, CLK90, CLK180, and CLK270 outputs	TRUE, FALSE

DLL Clock Input Connections

An external clock source enters the FPGA using a Global Clock Input Buffer (IBUFG), which directly accesses the global clock network or via an Input Buffer (IBUF). Clock signals within the FPGA drive a global clock net using a Global Clock Multiplexer Buffer (BUFGMUX). The global clock net connects directly to the CLKIN input. The internal and external connections are shown in Figure 39a and Figure 39c, respectively. A differential clock (e.g., LVDS) can serve as an input to CLKIN.

DLL Clock Output and Feedback Connections

As many as four of the nine DCM clock outputs can simultaneously drive four of the BUFGMUX buffers on the same die edge. All DCM clock outputs can simultaneously drive general routing resources, including interconnect leading to OBUF buffers.

The feedback loop is essential for DLL operation and is established by driving the CLKFB input with either the CLK0 or the CLK2X signal so that any undesirable clock distribution delay is included in the loop. It is possible to use either of these two signals for synchronizing any of the seven DLL outputs: CLK0, CLK90, CLK180, CLK270, CLKDV, CLK2X, or CLK2X180. The value assigned to the CLK_FEEDBACK attribute must agree with the physical feedback connection: a value of 1X for the CLK0 case, 2X for the CLK2X case. If the DCM is used in an application that does not require the DLL — that is, only the DFS is used — then there is no required feedback loop so CLK_FEEDBACK is set to NONE.

There are two basic cases that determine how to connect the DLL clock outputs and feedback connections: on-chip synchronization and off-chip synchronization, which are illustrated in Figure 39a through Figure 39d.

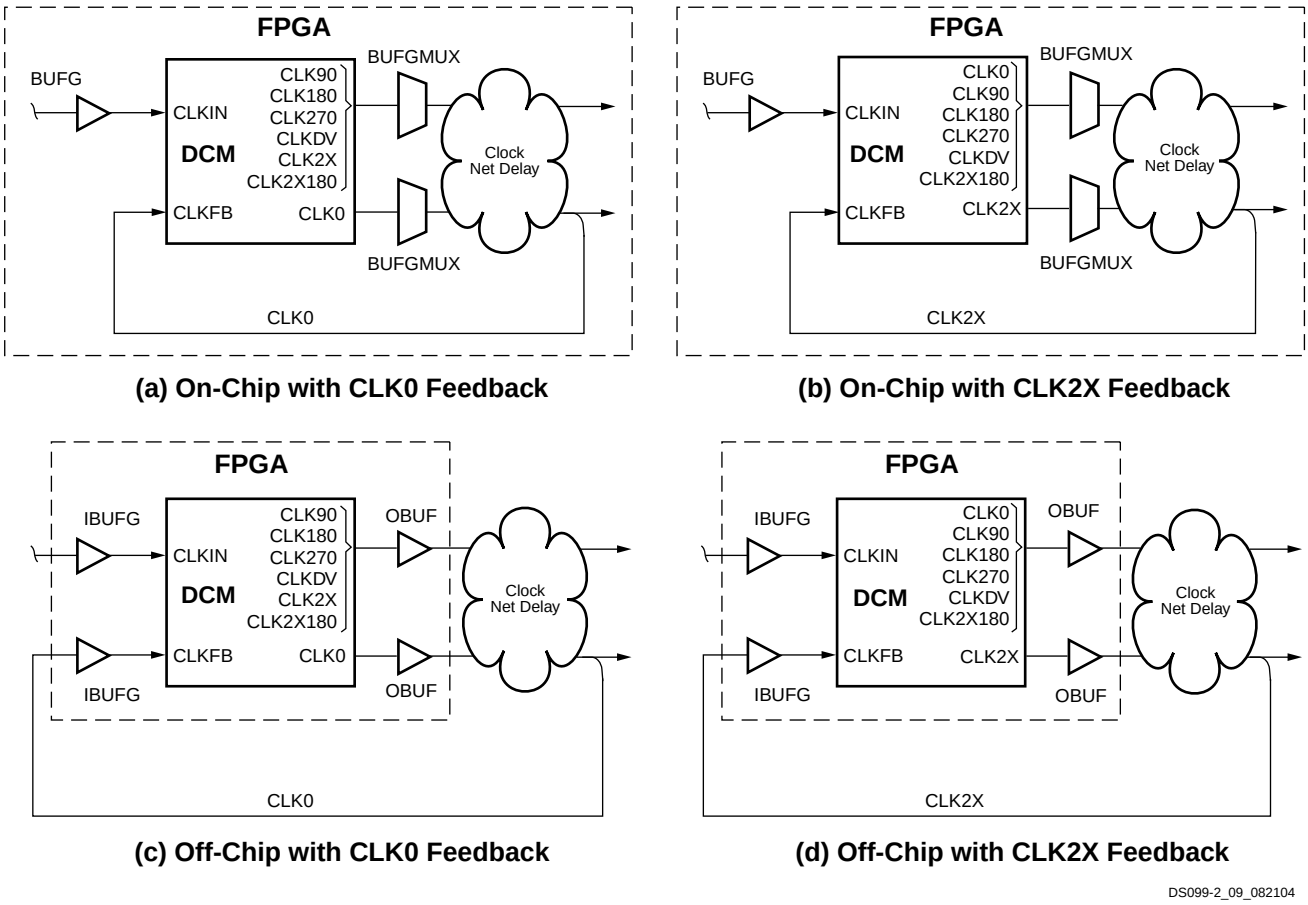


Figure 39: **Input Clock, Output Clock, and Feedback Connections for the DLL**

In the on-chip synchronization case in Figure 39a and Figure 39b, it is possible to connect any of the DLL's seven output clock signals through general routing resources to the FPGA's internal registers. Either a Global Clock Buffer (BUFG) or a BUFGMUX affords access to the global clock network. As shown in Figure 39a, the feedback loop is created by routing CLK0 (or CLK2X, in Figure 39b to a global clock net, which in turn drives the CLKFB input.

In the off-chip synchronization case in Figure 39c and Figure 39d, CLK0 (or CLK2X) plus any of the DLL's other output clock signals exit the FPGA using output buffers (OBUF) to drive an external clock network plus registers on the board. As shown in Figure 39c, the feedback loop is formed by feeding CLK0 (or CLK2X, in Figure 39d) back into the FPGA using an IBUFG, which directly accesses the global clock network, or an IBUF. Then, the global clock net is connected directly to the CLKFB input.

Accommodating High Input Frequencies

If the frequency of the CLKIN signal is high such that it exceeds the maximum permitted, divide it down to an acceptable value using the CLKIN_DIVIDE_BY_2 attribute. When this attribute is set to TRUE, the CLKIN frequency is divided by a factor of two just as it enters the DCM.

Coarse Phase Shift Outputs of the DLL Component

In addition to CLK0 for zero-phase alignment to the CLKIN signal, the DLL also provides the CLK90, CLK180, and CLK270 outputs for 90°, 180°, and 270° phase-shifted signals, respectively. These signals are described in Table 25. Their relative timing is shown in Figure 40. For control in finer increments than 90°, see **Phase Shifter (PS)**.

Basic Frequency Synthesis Outputs of the DLL Component

The DLL component provides basic options for frequency multiplication and division in addition to the more flexible synthesis capability of the DFS component, described in a later section. These operations result in output clock signals with frequencies that are either a fraction (for division) or a multiple (for multiplication) of the incoming clock frequency. The CLK2X output produces an in-phase signal that is twice the frequency of CLKIN. The CLK2X180 output also doubles the frequency, but is 180° out-of-phase with respect to CLKIN. The CLKDIV output generates a clock frequency that is a predetermined fraction of the CLKIN frequency. The CLKDV_DIVIDE attribute determines the factor used to divide the CLKIN frequency. The attribute can be set to var-

ious values as described in Table 26. The basic frequency synthesis outputs are described in Table 25.

Duty Cycle Correction of DLL Clock Outputs

The CLK2X[1], CLK2X180, and CLKDV[2] output signals ordinarily exhibit a 50% duty cycle – even if the incoming CLKIN signal has a different duty cycle. Fifty-percent duty cycle means that the High and Low times of each clock cycle are equal. The DUTY_CYCLE_CORRECTION attribute determines whether or not duty cycle correction is applied to the CLK0, CLK90, CLK180, and CLK270 outputs. If DUTY_CYCLE_CORRECTION is set to TRUE, then the duty cycle of these four outputs is corrected to 50%. If DUTY_CYCLE_CORRECTION is set to FALSE, then these outputs exhibit the same duty cycle as the CLKIN signal. Figure 40 compares the characteristics of the DLL's output signals to those of the CLKIN signal.

The CLK2X output generates a 25% duty cycle clock at the same frequency as the CLKIN signal until the DLL has achieved lock.

The duty cycle of the CLKDV outputs may differ somewhat from 50% (i.e., the signal is High for less than 50% of the period) when the CLKDV_DIVIDE attribute is set to a non-integer value *and* the DLL is operating in the High Frequency mode.

Digital Frequency Synthesizer (DFS)

The DFS component generates clock signals the frequency of which is a product of the clock frequency at the CLKIN input and a ratio of two user-determined integers. Because of the wide range of possible output frequencies such a ratio permits, the DFS feature provides still further flexibility than the DLL's basic synthesis options as described in the preceding section. The DFS component's two dedicated outputs, CLKFX and CLKFX180, are defined in Table 28.

The signal at the CLKFX180 output is essentially an inversion of the CLKFX signal. These two outputs always exhibit a 50% duty cycle. This is true even when the CLKIN signal does not. These DFS clock outputs are driven at the same time as the DLL's seven clock outputs.

The numerator of the ratio is the integer value assigned to the attribute CLKFX_MULTIPLY and the denominator is the integer value assigned to the attribute CLKFX_DIVIDE. These attributes are described in Table 27.

The output frequency (f_{CLKFX}) can be expressed as a function of the incoming clock frequency (f_{CLKIN}) as follows:

$$f_{CLKFX} = f_{CLKIN} * (CLKFX\_MULTIPLY / CLKFX\_DIVIDE) \quad \textbf{(3)}$$

Regarding the two attributes, it is possible to assign any combination of integer values, provided that two conditions are met:

1. The two values fall within their corresponding ranges, as specified in Table 27.
2. The f_{CLKFX} frequency calculated from the above expression accords with the DCM's operating frequency specifications.

For example, if CLKFX_MULTIPLY = 5 and CLKFX_DIVIDE = 3, then the frequency of the output clock signal is 5/3 that of the input clock signal.

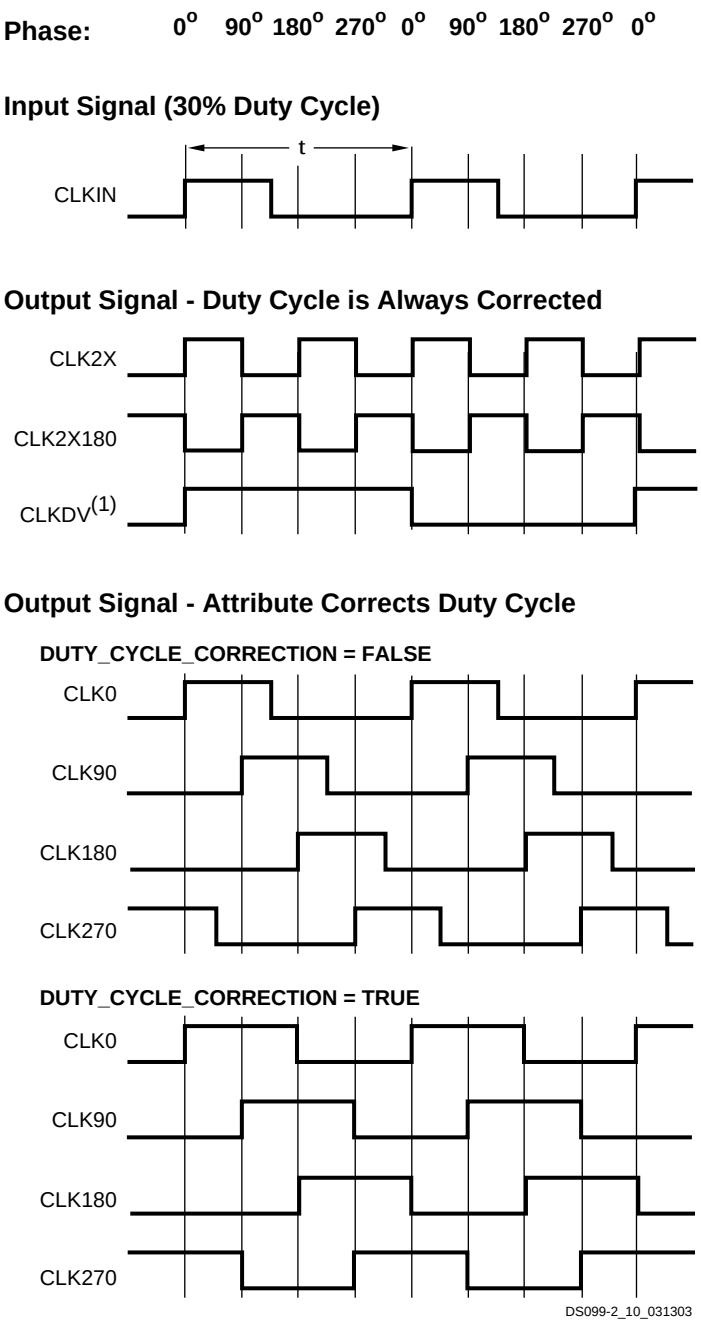


Figure 40: **Characteristics of the DLL Clock Outputs**

DFS With or Without the DLL

The DFS component can be used with or without the DLL component: Without the DLL, the DFS component multiplies or divides the CLKIN signal frequency according to the respective CLKFX_MULTIPLY and CLKFX_DIVIDE values,

generating a clock with the new target frequency on the CLKFX and CLKFX180 outputs. Though classified as belonging to the DLL component, the CLKIN input is shared with the DFS component. This case does not employ feedback loop. Therefore, it cannot correct for clock distribution delay.

With the DLL, the DFS operates as described in the preceding case, only with the additional benefit of eliminating the clock distribution delay. In this case, a feedback loop from the CLK0 output to the CLKFB input must be present.

The DLL and DFS components work together to achieve this phase correction as follows: Given values for the CLKFX_MULTIPLY and CLKFX_DIVIDE attributes, the DLL selects the delay element for which the output clock edge coincides with the input clock edge whenever mathematically possible. For example, when CLKFX_MULTIPLY = 5 and CLKFX_DIVIDE = 3, the input and output clock edges coincide every three input periods, which is equivalent in time to five output periods.

Smaller CLKFX_MULTIPLY and CLKFX_DIVIDE values achieve faster lock times. With no factors common to the two attributes, alignment occurs once with every number of cycles equal to the CLKFX_DIVIDE value. Therefore, it is recommended that the user reduce these values by factoring wherever possible. For example, given CLKFX_MULTIPLY = 9 and CLKFX_DIVIDE = 6, removing a factor of three yields CLKFX_MULTIPLY = 3 and CLKFX_DIVIDE = 2. While both value-pairs result in the multiplication of clock frequency by 3/2, the latter value-pair enables the DLL to lock more quickly.

Table 27: **DFS Attributes**

Attribute	Description	Values
CLKFX_MULTIPLY	Frequency multiplier constant	Integer from 2 to 32, inclusive
CLKFX_DIVIDE	Frequency divisor constant	Integer from 1 to 32, inclusive

Table 28: **DFS Signals**

Signal	Direction	Description
CLKFX	Output	Multiplies the CLKIN frequency by the attribute-value ratio (CLKFX_MULTIPLY/ CLKFX_DIVIDE) to generate a clock signal with a new target frequency.
CLKFX180	Output	Generates a clock signal with same frequency as CLKFX, only shifted 180° out-of-phase.

DFS Clock Output Connections

There are two basic cases that determine how to connect the DFS clock outputs: on-chip and off-chip, which are illustrated in Figure 39a and Figure 39c, respectively. This is similar to what has already been described for the DLL component. See **DLL Clock Output and Feedback Connections**.

In the on-chip case, it is possible to connect either of the DFS's two output clock signals through general routing resources to the FPGA's internal registers. Either a Global Clock Buffer (BUFG) or a BUFGMUX affords access to the global clock network. The optional feedback loop is formed in this way, routing CLK0 to a global clock net, which in turn drives the CLKFB input.

In the off-chip case, the DFS's two output clock signals, plus CLK0 for an optional feedback loop, can exit the FPGA using output buffers (OBUF) to drive a clock network plus registers on the board. The feedback loop is formed by feeding the CLK0 signal back into the FPGA using an IBUFG, which directly accesses the global clock network, or an IBUF. Then the global clock net is connected directly to the CLKFB input.

Phase Shifter (PS)

The DCM provides two approaches to controlling the phase of a DCM clock output signal relative to the CLKIN signal: First, there are nine clock outputs that employ the DLL to achieve a desired phase relationship: CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, CLKDV CLKFX, and CLKFX180. These outputs afford "coarse" phase control.

The second approach uses the PS component described in this section to provide a still finer degree of control. The PS component accomplishes this by introducing a "fine phase shift" (T_{PS}) between the CLKFB and CLKIN signals inside the DLL component. The user can control this fine phase shift down to a resolution of 1/512 of a CLKIN cycle or one tap delay (DCM_TAP), whichever is greater. When in use, the PS component shifts the phase of all nine DCM clock output signals together. If the PS component is used together with a DCM clock output such as the CLK90, CLK180, CLK270, CLK2X180, and CLKFX180, then the fine phase shift of the former gets added to the coarse phase shift of the latter.

PS Component Enabling and Mode Selection

The CLKOUT_PHASE_SHIFT attribute enables the PS component for use in addition to selecting between two operating modes. As described in Table 29, this attribute has three possible values: NONE, FIXED, and VARIABLE. When CLKOUT_PHASE_SHIFT is set to NONE, the PS component is disabled and its inputs, PSEN, PSCLK, and PSINCDEC, must be tied to GND. The set of waveforms in Figure 41a shows the disabled case, where the DLL maintains a zero-phase alignment of signals CLKFB and CLKIN upon which the PS component has no effect. The PS com-

ponent is enabled by setting the attribute to either the FIXED or VARIABLE values, which select the Fixed Phase mode and the Variable Phase mode, respectively. These two modes are described in the sections that follow.

Table 29: **PS Attributes**

Attribute	Description	Values
CLKOUT_PHASE_SHIFT	Disables the PS component or chooses between Fixed Phase and Variable Phase modes.	NONE, FIXED, VARIABLE
PHASE_SHIFT	Determines size and direction of initial fine phase shift.	Integers from –255 to +255

Determining the Fine Phase Shift

The user controls the phase shift of CLKFB relative to CLKIN by setting and/or adjusting the value of the PHASE_SHIFT attribute. This value must be an integer ranging from –255 to +255. This corresponds to a phase shift range of –180 to +180 degrees, which is different from the original Spartan-3 DCM. The PS component uses this value to calculate the desired fine phase shift (T_{PS}) as a fraction of the CLKIN period (T_{CLKIN}). Given values for PHASE_SHIFT and T_{CLKIN}, it is possible to calculate T_{PS} as follows:

$$T_{PS} = (\text{PHASE\_SHIFT}/512) * T_{CLKIN} \quad \textbf{(4)}$$

Both the Fixed Phase and Variable Phase operating modes employ this calculation. If the PHASE_SHIFT value is zero, then CLKFB and CLKIN are in phase, the same as when the PS component is disabled. When the PHASE_SHIFT value is positive, the CLKFB signal is shifted later in time with respect to CLKIN. If the attribute value is negative, the CLKFB signal is shifted earlier in time with respect to CLKIN.

The Fixed Phase Mode

This mode fixes the desired fine phase shift to a fraction of the T_{CLKIN}, as determined by Equation (4) and its user-selected PHASE_SHIFT value P. The set of waveforms in Figure 41b illustrates the relationship between CLKFB and CLKIN in the Fixed Phase mode. In the Fixed Phase mode, the PSEN, PSCLK, and PSINCDEC inputs are not used and must be tied to GND.

In Figure 41:

- P represents the integer value ranging from –255 to +255 to which the PHASE_SHIFT attribute is assigned. (P = approximately -90 as shown)
- N is an integer value ranging from (P – 255) to (+255 – P) that represents the net phase shift effect from a series of increment and/or decrement operations.
- N = {Total number of increments} – {Total number of decrements} provided the user does not try to increment past + 255 or decrement past -255. A positive value for N indicates a net increment; a negative value indicates a net decrement.

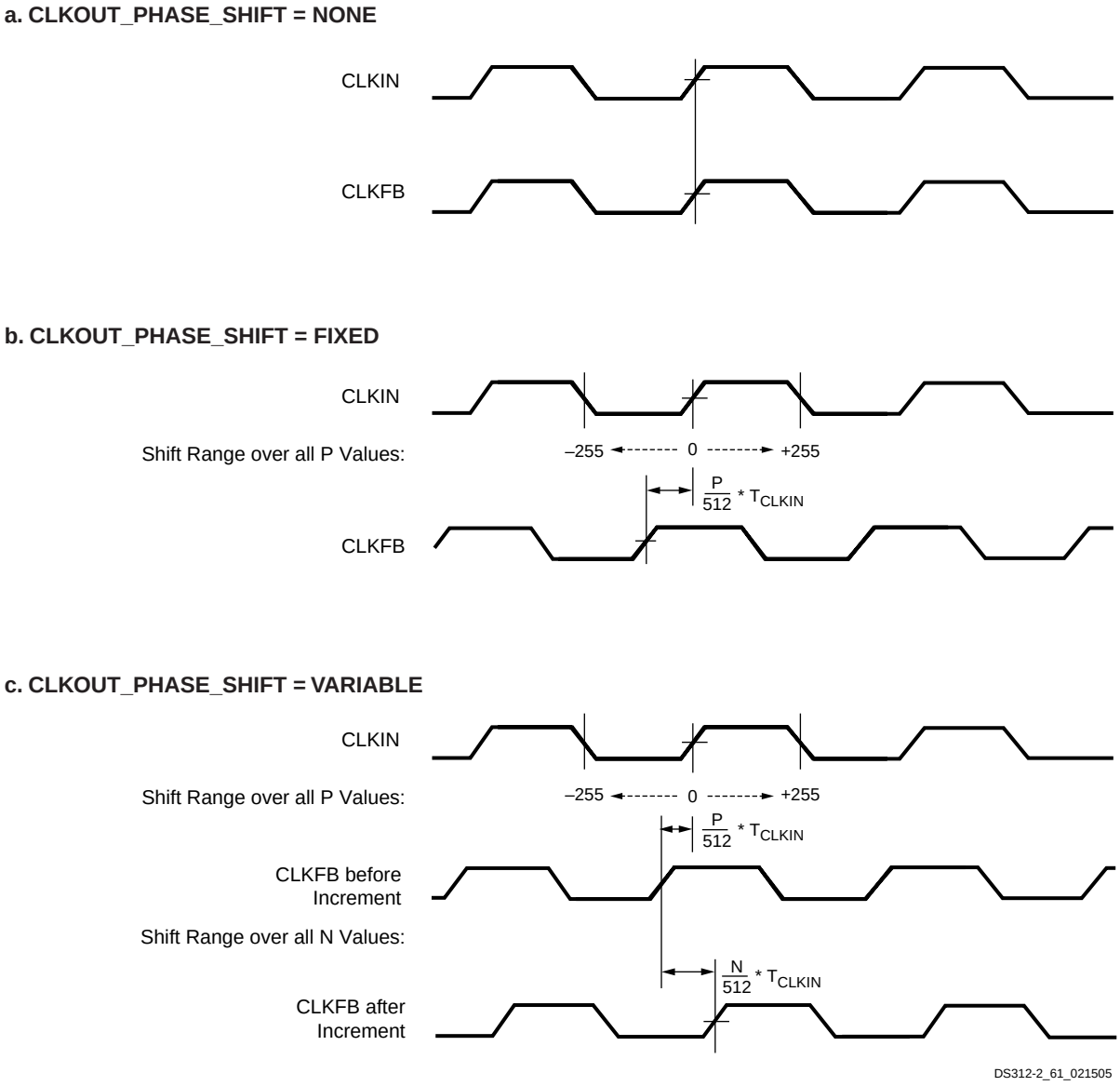


Figure 41: **Phase Shifter Waveforms**

The Variable Phase Mode

The Variable Phase mode dynamically adjusts the fine phase shift over time using three inputs to the PS component (PSEN, PSCLK, and PSINCDEC), as defined in Table 30.

Table 30: **Signals for Variable Phase Mode**

Signal	Direction	Description
PSEN(1)	Input	Enables PSCLK for variable phase adjustment.
PSCLK(1)	Input	Clock to synchronize phase shift adjustment.
PSINCDEC(1)	Input	Chooses between increment and decrement for phase adjustment. It is synchronized to the PSCLK signal.
PSDONE	Output	Goes High to indicate that present phase adjustment is complete and PS component is ready for next phase adjustment request. It is synchronized to the PSCLK signal.

Notes:

1. It is possible to program this input for either a true or inverted polarity.

Just following device configuration, the PS component initially determines T_{PS} by evaluating Equation (4) for the value assigned to the PHASE_SHIFT attribute. Then to dynamically adjust that phase shift, use the three PS inputs to increase or decrease the fine phase shift.

PSINCDEC is synchronized to the PSCLK clock signal, which is enabled by asserting PSEN. It is possible to drive the PSCLK input with the CLKIN signal or any other clock signal. A request for phase adjustment is entered as follows: For each PSCLK cycle that PSINCDEC is High, the PS component adds 1/512 of a CLKIN cycle to T_{PS}. Similarly, for each enabled PSCLK cycle that PSINCDEC is Low, the PS component subtracts 1/512 of a CLKIN cycle from T_{PS}. The phase adjustment may require as many as 100 CLKIN cycles plus three PSCLK cycles to take effect, at which point the output PSDONE goes High for one PSCLK cycle. This pulse indicates that the PS component has finished the present adjustment and is now ready for the next request.

Asserting the Reset (RST) input, returns T_{PS} to its original shift time, as determined by the PHASE_SHIFT attribute value. The set of waveforms in Figure 41c illustrates the relationship between CLKFB and CLKIN in the Variable Phase mode.

The Status Logic Component

The Status Logic component not only reports on the state of the DCM but also provides a means of resetting the DCM to an initial known state. The signals associated with the Status Logic component are described in Table 31.

As a rule, the Reset (RST) input is asserted only upon configuring the device or changing the CLKIN frequency. A DCM reset does not affect attribute values (e.g., CLKFX_MULTIPLY and CLKFX_DIVIDE). If not used, tie RST to GND.

The eight bits of the STATUS bus are defined in Table 32.

Table 31: **Status Logic Signals**

Signal	Direction	Description
RST	Input	A High resets the entire DCM to its initial power-on state. Initializes the DLL taps for a delay of zero. Sets the LOCKED output Low. This input is asynchronous.
STATUS[7:0]	Output	The bit values on the STATUS bus provide information regarding the state of DLL and PS operation
LOCKED	Output	Indicates that the CLKIN and CLKFB signals are in phase by going High. The two signals are out-of-phase when Low.

Table 32: **DCM Status Bus**

Bit	Name	Description
0	Reserved	-
1	CLKIN Stopped	A value of 1 indicates that the CLKIN input signal is not toggling. A value of 0 indicates toggling. This bit functions only when the CLKFB input is connected.(1)
2	CLKFX Stopped	A value of 1 indicates that the CLKFX output is not toggling. A value of 0 indicates toggling. This bit functions only when the CLKFX or CLKFX180 output are connected.
3-6	Reserved	-

Notes:

1. If only the DFS clock outputs are used, but none of the DLL clock outputs, this bit does not go High when the CLKIN signal stops.

Stabilizing DCM Clocks Before User Mode

The STARTUP_WAIT attribute shown in Table 33 optionally delays the end of the FPGA's configuration process until after the DCM locks to the incoming clock frequency. This option ensures that the FPGA remains in the Startup phase of configuration until all clock outputs generated by the DCM are stable. When all the DCMs with their STARTUP_WAIT attribute set to TRUE assert the LOCKED signal, then the FPGA completes its configuration process and proceeds to user mode. The associated bitstream generator (BitGen) option ***LCK_cycle*** specifies one of the six cycles in the Startup phase. The selected cycle defines the point at which configuration halts until the all the LOCKED outputs go High. Also see **Start-Up**, page 91.

Table 33: **Status Attributes**

Attribute	Description	Values
STARTUP_WAIT	Delays transition from configuration to user mode until DCM locks to input clock.	TRUE, FALSE

Clocking Infrastructure

The Spartan-3E clocking infrastructure, shown in Figure 42, provides a series of low-capacitance, low-skew interconnect lines well-suited to carrying high-frequency signals throughout the FPGA. The infrastructure also includes the clock inputs and BUFGMUX clock buffers/multiplexers. The Xilinx Place-and-Route (PAR) software automatically routes high-fanout clock signals using these resources.

Clock Inputs

Clock pins accept external clock signals and connect directly to DCMs and BUFGMUX elements. Each Spartan-3E FPGA has:

- 16 Global Clock inputs (GCLK0 through GCLK15) located along the top and bottom edges of the FPGA
- 8 Right-Half Clock inputs (RHCLK0 through RHCLK7) located along the right edge
- 8 Left-Half Clock inputs (LHCLK0 through LHCLK7) located along the left edge

Clock inputs optionally connect directly to DCMs using dedicated connections. Table 35 shows the clock inputs that feed a specific DCM within a given Spartan-3E part number. Different Spartan-3E FPGA densities have different numbers of DCMs.

Each clock input is also optionally a user-I/O pin and connects to internal interconnect. Some clock pad pins are input-only pins as indicated in Module 4 of the Spartan-3E Data Sheet.

Clock Buffers/Multiplexers

Clock Buffers/Multiplexers either drive clock input signals directly onto a clock line (BUFG) or optionally provide a multiplexer to switch between two unrelated, possibly asynchronous clock signals (BUFGMUX).

Each BUFGMUX element, shown in Figure 43, is a 2-to-1 multiplexer. The select line, S, chooses which of the two inputs, I0 or I1, drives the BUFGMUX's output signal, O, as described in Table 34. The switching from one clock to the other is glitch-less, and done in such a way that the output High and Low times are never shorter than the shortest High or Low time of either input clock.

Table 34: **BUFGMUX Select Mechanism**

S Input	O Output
0	I0 Input
1	I1 Input

The BUFG clock buffer primitive drives a single clock signal onto the clock network and is essentially the same element as a BUFGMUX, just without the clock select mechanism. Similarly, the BUFGCE primitive creates an enabled clock buffer using the BUFGMUX select mechanism.

The I0 and I1 inputs to an BUFGMUX element originate from clock input pins, DCMs, or Double-Line interconnect, as shown in Figure 43. As shown in Figure 42, there are 24 BUFGMUX elements distributed around the four edges of the device. Clock signals from the four BUFGMUX elements at the top edge and the four at the bottom edge are truly global and connect to all clocking quadrants. The eight left-edge BUFGMUX elements only connect to the two clock quadrants in the left half of the device. Similarly, the eight right-edge BUFGMUX elements only connect to the right half of the device.

BUFGMUX elements are organized in pairs and share I0 and I1 connections with adjacent BUFGMUX elements from a common clock switch matrix as shown in Figure 43. For example, the input on I0 of one BUFGMUX also a shared input to I1 of the adjacent BUFGMUX.

The clock switch matrix for the left- and right-edge BUFGMUX elements receive signals from any of the three following sources: an LHCLK or RHCLK pin as appropriate, a Double-Line interconnect, or a DCM in the XC3S1200E and XC3S1600E devices.

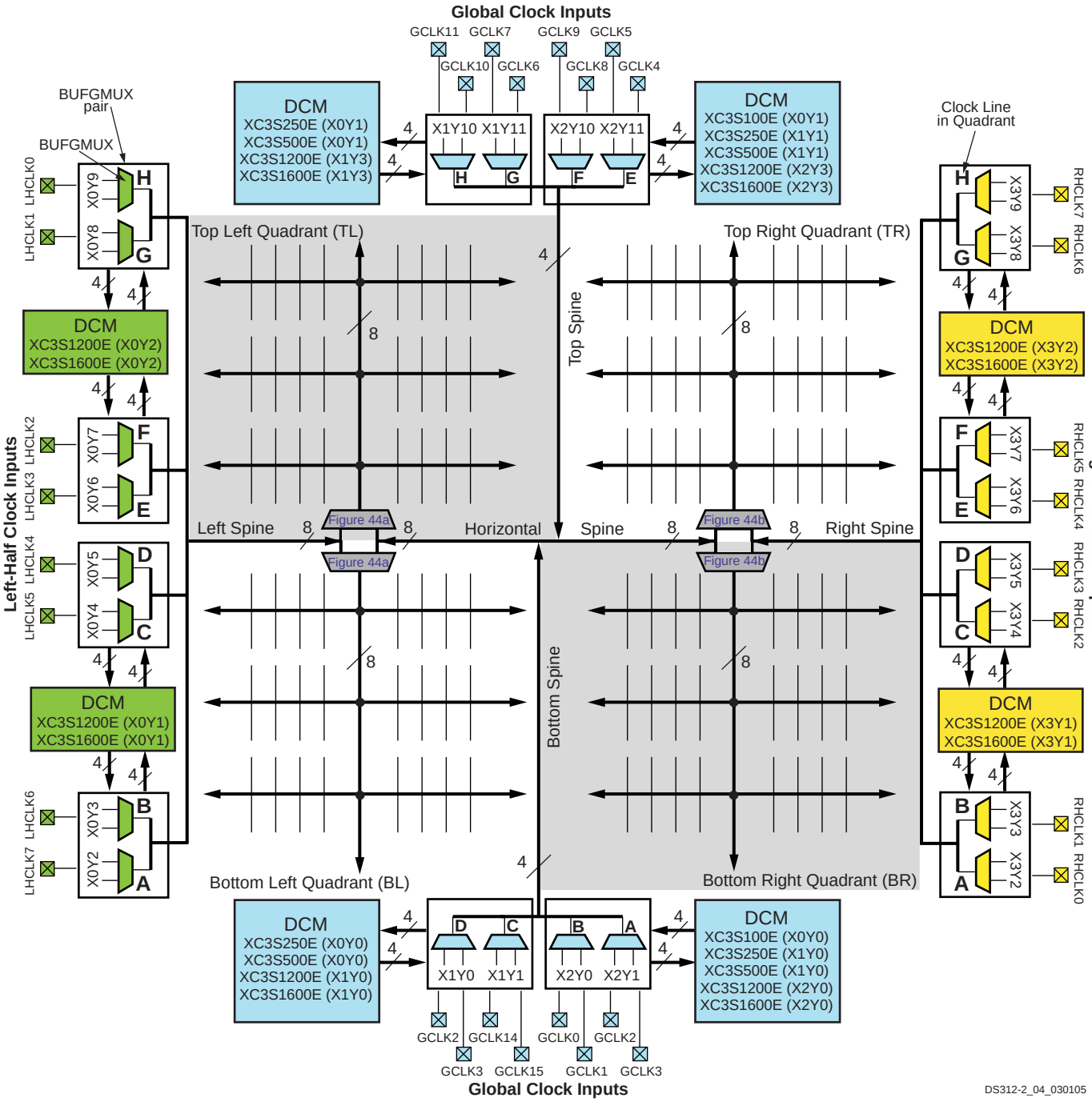
By contrast, the clock switch matrixes on the top and bottom edges receive signals from any of the five following sources: two GCLK pins, two DCM outputs, or one Double-Line interconnect.

Table 36 indicates permissible connections between clock inputs and BUFGMUX elements. The four BUFGMUX elements on the top edge are paired together and share inputs from the eight global clock inputs along the top edge. Each

BUFGMUX pair connects to four of the eight global clock inputs, as shown in Figure 42. This optionally allows differential inputs to the global clock inputs without wasting a BUFGMUX element.

The connections for the bottom-edge BUFGMUX elements is similar to the top-edge connections.

On the left and right edges, only two clock inputs feed each pair of BUFGMUX elements.



Notes:

1. Number of DCMs and locations of these DCM varies for different device densities.
2. The left and right DCMs are only in the XC3S1200E and XC3S1600E. The XC3S100E has only two DCMs, one on the top right and one on the bottom right of the die.

Figure 42: **Spartan-3E Internal Quadrant-Based Clock Network (Top View)**

Table 35: **Direct Connections from Clock Inputs to DCMs and Associated DCM Location String**

Clock Input	XC3S100E	XC3S250E/XC3S500E	XC3S1200E/XC3S1600E
GCLK[3:0]	DCM_X0Y0	DCM_X1Y0	DCM_X2Y0
RHCLK[3:0]	N/A	N/A	DCM_X3Y1
RHCLK[7:4]	N/A	N/A	DCM_X3Y2
GCLK[7:4]	DCM_X0Y1	DCM_X1Y1	DCM_X2Y2
GCLK[11:8]	N/A	DCM_X0Y1	DCM_X1Y3
LHCLK[3:0]	N/A	N/A	DCM_X0Y2
LHCLK[7:4]	N/A	N/A	DCM_X0Y1
GCLK[15:12]	N/A	DCM_X0Y0	DCM_X1Y0

Table 36: **Connections from Clock Inputs to BUFGMUX Elements and Associated Quadrant Clock**

Quadrant Clock Line(1)	Left-Half BUFGMUX			Top or Bottom BUFGMUX			Right-Half BUFGMUX		
	Location(2)	I0 Input	I1 Input	Location(2)	I0 Input	I1 Input	Location(2)	I0 Input	I1 Input
A	*X0Y2*	LHCLK7	LHCLK6	*X2Y1*	GCLK0 or GCLK12	GCLK1 or GCLK13	*X3Y2*	RHCLK0	RHCLK1
B	*X0Y3*	LHCLK6	LHCLK7	*X2Y0*	GCLK1 or GCLK13	GCLK0 or GCLK12	*X3Y3*	RHCLK1	RHCLK0
C	*X0Y4*	LHCLK5	LHCLK4	*X1Y1*	GCLK2 or GCLK14	GCLK3 or GCLK15	*X3Y4*	RHCLK2	RHCLK3
D	*X0Y5*	LHCLK4	LHCLK5	*X1Y0*	GCLK3 or GCLK15	GCLK2 or GCLK14	*X3Y5*	RHCLK3	RHCLK2
E	*X0Y6*	LHCLK3	LHCLK2	*X2Y11*	GCLK4 or GCLK8	GCLK5 or GCLK9	*X3Y6*	RHCLK4	RHCLK5
F	*X0Y7*	LHCLK2	LHCLK3	*X2Y10*	GCLK5 or GCLK9	GCLK4 or GCLK8	*X3Y7*	RHCLK5	RHCLK4
G	*X0Y8*	LHCLK1	LHCLK0	*X1Y11*	GCLK6 or GCLK10	GCLK7 or GCLK11	*X3Y8*	RHCLK6	RHCLK7
H	*X0Y9*	LHCLK0	LHCLK1	*X1Y10*	GCLK7 or GCLK11	GCLK6 or GCLK10	*X3Y9*	RHCLK7	RHCLK6

Notes:

1. See **Quadrant Clock Routing** for connectivity details for the eight quadrant clocks.
2. See Figure 42 for specific BUFGMUX locations and Figure 44 for information on how BUFGMUX elements drive onto a specific clock line within a quadrant.

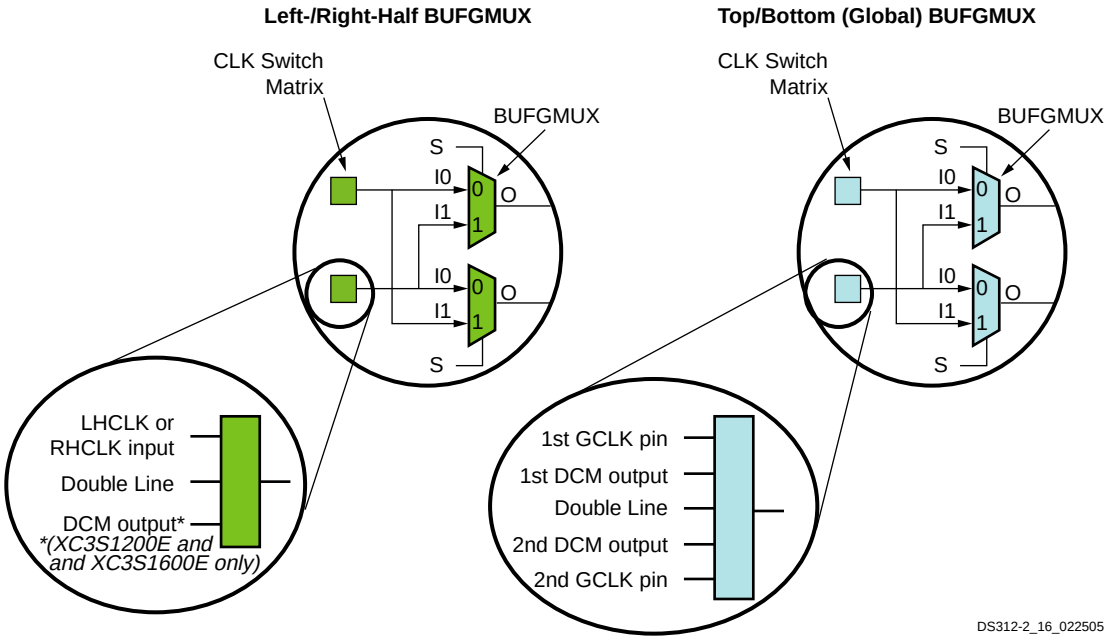


Figure 43: **Clock Switch Matrix to BUFGMUX Pair Connectivity**

Quadrant Clock Routing

The clock routing within the FPGA is quadrant-based, as shown in Figure 42. Each clock quadrant supports eight total clock signals, labeled ‘A’ through ‘H’ in Table 36 and Figure 44. The clock source for an individual clock line originates either from a global BUFGMUX element along the top and bottom edges or from a BUFGMUX element along the associated edge, as shown in Figure 44. The clock lines feed the synchronous resource elements (CLBs, IOBs, block RAM, multipliers, and DCMs) within the quadrant.

The four quadrants of the device are:

- Top Right (TR)
- Bottom Right (BR)
- Bottom Left (BL)
- Top Left (TL)

Note that the quadrant clock notation (TR, BR, BL, TL) is separate from that used for similar IOB placement constraints.

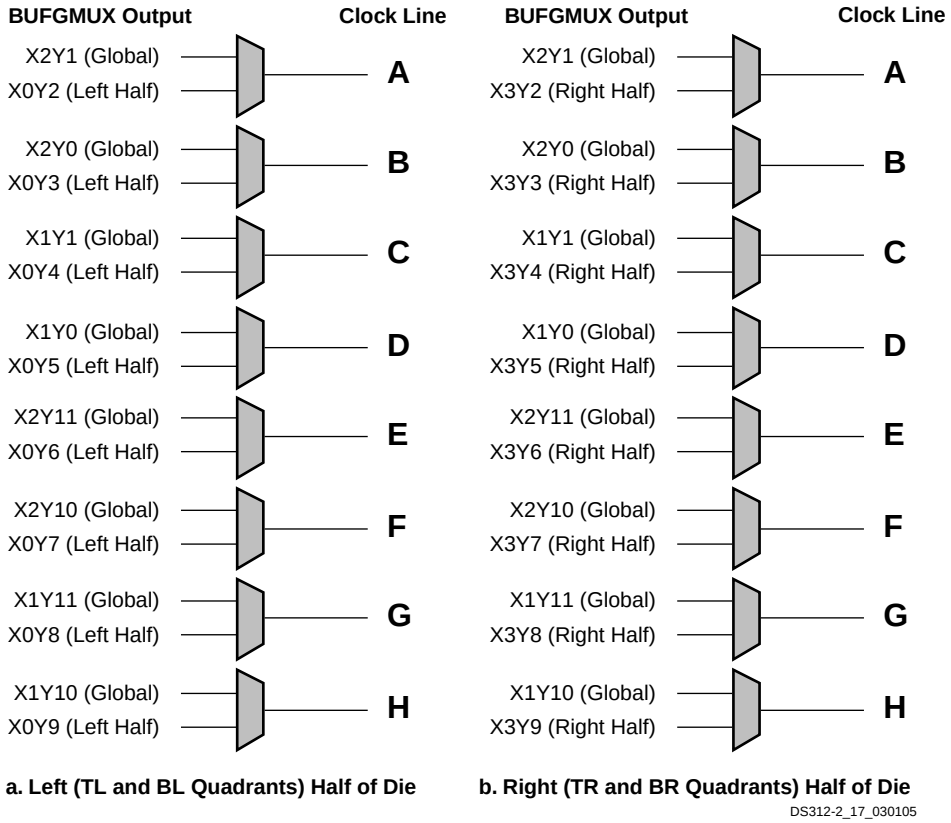


Figure 44: **Clock Sources for the Eight Clock Lines within a Clock Quadrant**

The outputs of the top or bottom BUFGMUX elements connect to two vertical spines, each comprising four vertical clock lines as shown in Figure 42. At the center of the die, these clock signals connect to the eight-line horizontal clock spine.

Outputs of the left and right BUFGMUX elements are routed onto the left or right horizontal spines, each comprising eight horizontal clock lines.

Each of the eight clock signals in a clock quadrant derives either from a global clock signal or a half clock signal. In other words, there are up to 24 total potential clock inputs to the FPGA, eight of which can connect to clocked elements in a single clock quadrant. Figure 44 shows how the clock lines in each quadrant are selected from associated BUFGMUX sources. For example, if quadrant clock ‘A’ in the bottom left (BL) quadrant originates from BUFGMUX_X2Y1, then the clock signal from BUFGMUX_X0Y2 is unavailable in the bottom left quadrant. However, the top left (TL) quadrant clock ‘A’ can still solely use the output from either BUFGMUX_X2Y1 or BUFGMUX_X0Y2 as the source.

To minimize the dynamic power dissipation of the clock network, the Xilinx development software automatically disables all clock segments not in use.

Interconnect

Interconnect is the programmable network of signal pathways between the inputs and outputs of functional elements within the FPGA, such as IOBs, CLBs, DCMs, block RAM, etc.

Interconnect, also called routing, is segmented for optimal connectivity. Functionally, interconnect resources are identical to that of the Spartan-3 architecture. There are four kinds of interconnects: long lines, hex lines, double lines, and direct lines. The Xilinx Place and Route (PAR) software exploits the rich interconnect array to deliver optimal system performance and the fastest compile times.

The switch matrix connects to the different kinds of interconnects across the device. An interconnect tile, shown in Figure 45, is defined as a single switch matrix connected to a functional element, such as a CLB, IOB, or DCM. If a functional element spans across multiple switch matrices such as the block RAM or multipliers, then an interconnect tile is defined by the number of switch matrices connected to that functional element. A Spartan-3E device can be represented as an array of interconnect tiles where interconnect resources are for the channel between any two adjacent interconnect tile rows or columns as shown in Figure 46.

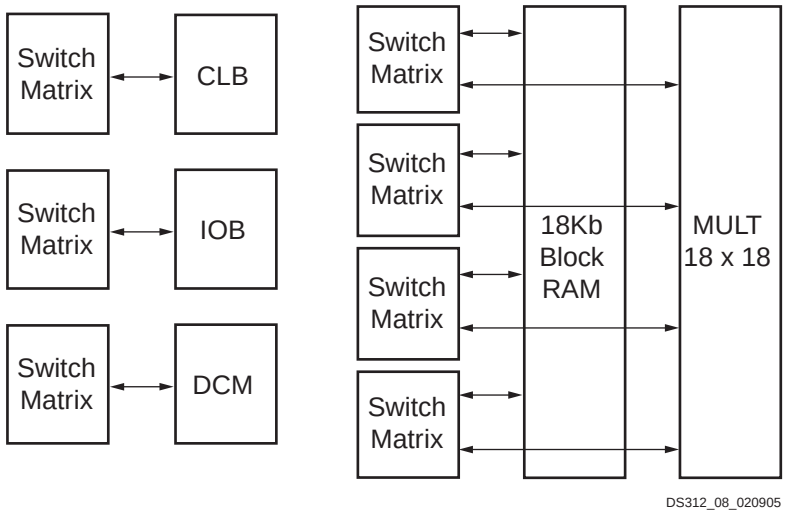


Figure 45: **Four Types of Interconnect Tiles (CLBs, IOBs, DCMs, and Block RAM/Multiplier)**

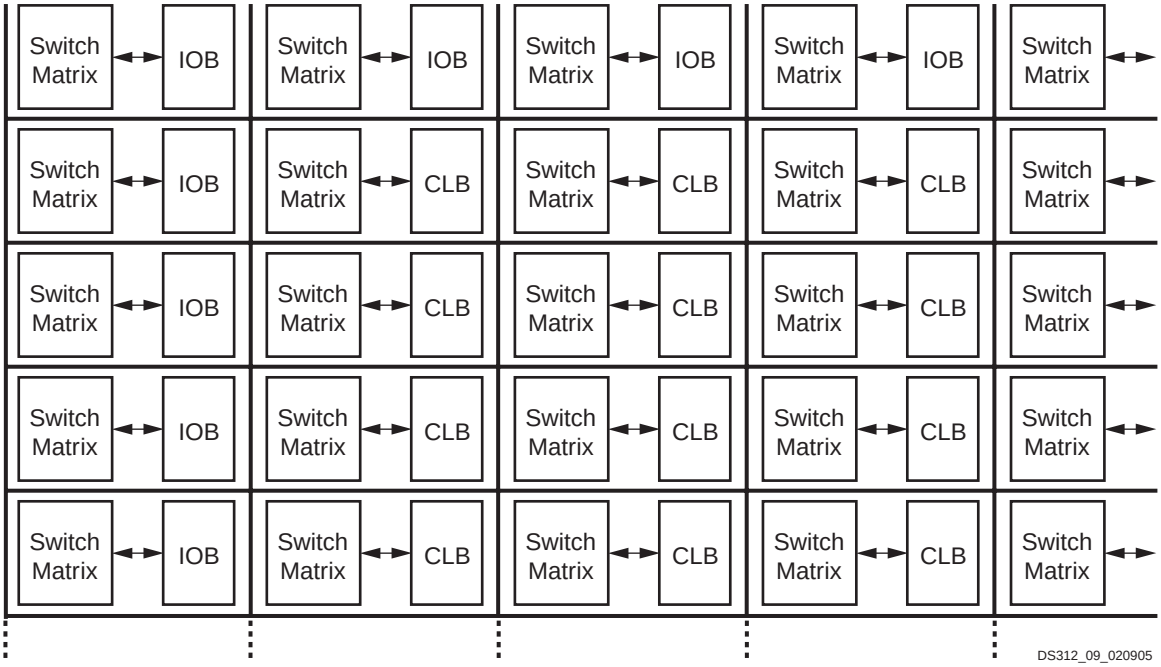


Figure 46: **Array of Interconnect Tiles in Spartan-3E FPGA**

There are four type of general-purpose interconnect available in each channel, as shown in Figure 47 and described below.

Long Lines

Each set of 24 long line signals spans the die both horizontally and vertically and connects to one out of every six interconnect tiles. At any tile, four of the long lines drive or receive signals from a switch matrix. Because of their low capacitance, these lines are well-suited for carrying high-frequency signals with minimal loading effects (e.g. skew). If all global clock lines are already committed and additional clock signals remain to be assigned, long lines serve as a good alternative.

Hex Lines

Each set of eight hex lines are connected to one out of every three tiles, both horizontally and vertically. Thirty-two hex lines are available between any given interconnect tile. Hex lines are only driven from one end of the route.

Double Lines

Each set of eight double lines are connected to every other tile, both horizontally and vertically. in all four directions. Thirty-two double lines available between any given interconnect tile. Double lines are more connections and more flexibility, compared to long line and hex lines.

Direct Connections

Direct connect lines route signals to neighboring tiles: vertically, horizontally, and diagonally. These lines most often drive a signal from a "source" tile to a double, hex, or long line and conversely from the longer interconnect back to a direct line accessing a "destination" tile.

Global Controls (STARTUP_SPARTAN3E)

In addition to the general-purpose interconnect, Spartan-3E FPGAs have two global logic control signals, as described in Table 37. These signals are available to the FPGA application via the STARTUP_SPARTAN3E primitive.

Table 37: **Spartan-3E Global Logic Control Signals**

Global Control Input	Description
GSR	When driven High, asynchronously places all registers and flip-flops in their initial state (see **Initialization**, page 24). Asserted automatically during the FPGA configuration process (see **Start-Up**, page 91).
GTS	When driven High, asynchronously forces all I/O pins to a high-impedance state (Hi-Z, three-state).

The Global Set/Reset (GSR) signal replaces the global reset signal included in many ASIC-style designs. Use the GSR control instead of a separate global reset signal in the design to free up CLB inputs, resulting in a smaller, more efficient design. Similarly, the GSR signal is asserted automatically during the FPGA configuration process, guaranteeing that the FPGA starts-up in a known state.

The STARTUP_SPARTAN3E primitive also includes two other signals used specifically during configuration. The MBT signals are for **Dynamically Loading Multiple Configuration Images Using MultiBoot Option**, page 78. The CLK input is an alternate clock for configuration **Start-Up**, page 91.

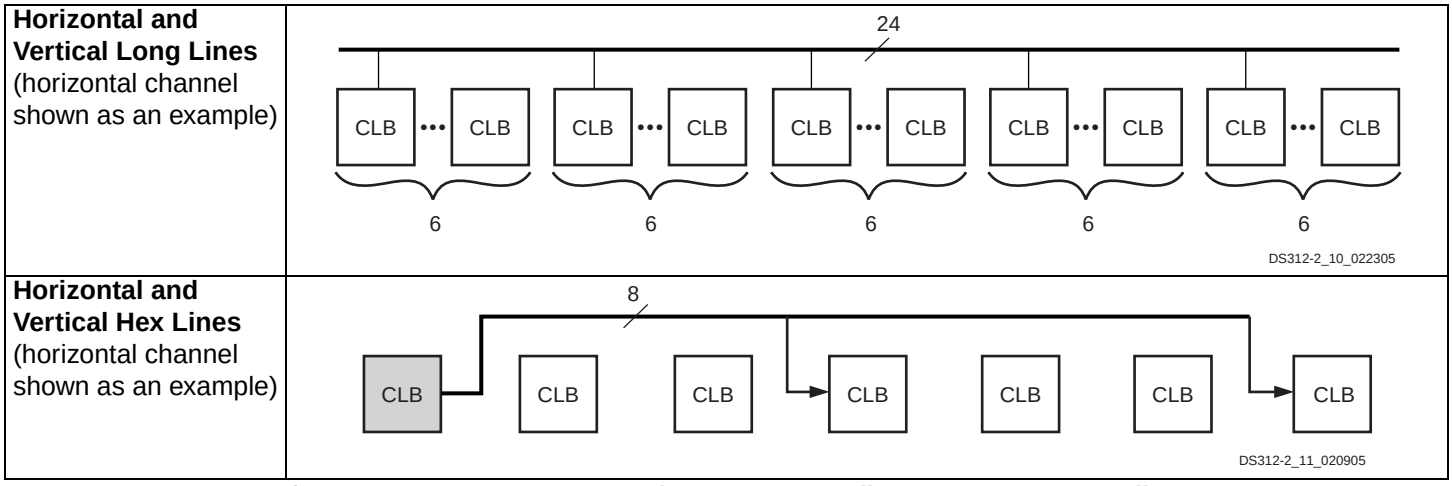


Figure 47: **Interconnect Types between Two Adjacent Interconnect Tiles**

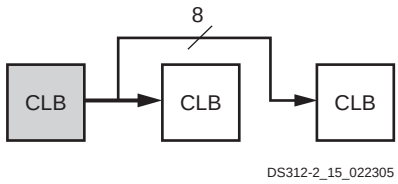
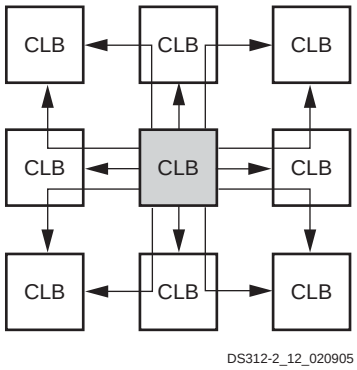
Horizontal and Vertical Double Lines (horizontal channel shown as an example)	
Direct Connections	

Figure 47: **Interconnect Types between Two Adjacent Interconnect Tiles**

Configuration

Differences from Spartan-3 FPGAs

In general, Spartan-3E FPGA configuration modes are a superset to those available in Spartan-3 FPGAs. Two new modes added in Spartan-3E FPGAs provide a glue-less configuration interface to industry-standard parallel NOR Flash and SPI serial Flash memories. Unlike Spartan-3 FPGAs, nearly all of the Spartan-3E configuration pins become available as user I/Os after configuration.

Configuration Process

The function of a Spartan-3E FPGA is defined by loading application-specific configuration data into the FPGA's internal, reprogrammable CMOS configuration latches (CCLs), similar to the way a microprocessor's function is defined by its application program. For FPGAs, this configuration process uses a subset of the device pins, some of which are dedicated to configuration; other pins are merely borrowed and returned to the application as general-purpose user I/Os after configuration completes.

Spartan-3E FPGAs offer several configuration options to minimize the impact of configuration on the overall system design. In some configuration modes, the FPGA generates a clock and loads itself from an external memory source, either serially or via a byte-wide data path. Alternatively, an external host such as a microprocessor downloads the FPGA's configuration data using a simple synchronous serial interface or via a byte-wide peripheral-style interface. Furthermore, multiple-FPGA designs share a single configuration memory source, creating a structure called a daisy chain.

Three FPGA pins—M2, M1, and M0—select the desired configuration mode. The mode pin settings appear in Table 38. The mode pin values are sampled during the start of configuration when the FPGA's INIT_B output goes High. After the FPGA completes configuration, the mode pins are available as user I/Os.

Table 38: **Spartan-3E Configuration Mode Pin Settings**

	Master Serial	SPI	BPI	Slave Parallel	Slave Serial	JTAG
M[2:0] mode pin settings	<0:0:0>	<0:0:1>	<0:1:0>=Up <0:1:1>=Down	<1:1:0>	<1:1:1>	<1:0:1>
Data width	Serial	Serial	Byte-wide	Byte-wide	Serial	Serial
Configuration memory source	Xilinx Platform Flash	Industry-standard SPI Serial Flash	Industry-standard parallel NOR Flash	Any source via microcontroller, CPU, Xilinx parallel Platform Flash, etc.	Any source via microcontroller, CPU, Xilinx Platform Flash, etc.	Any source via microcontroller, CPU, etc. and System Ace CF
Clock source	Internal oscillator	Internal oscillator	Internal oscillator	External clock on CCLK pin	External clock on CCLK pin	External clock on TCK pin
Total I/O pins borrowed during configuration	8	13	46	21	8	0
Configuration mode for downstream daisy-chained FPGAs	Slave Serial	Slave Serial	Slave Parallel	Slave Parallel or Memory Mapped	Slave Serial	JTAG
Self-configuring applications (no external download host)	✓	✓	✓	Possible using XCFxxP Platform Flash, which optionally generates CCLK	Possible using XCFxxP Platform Flash, which optionally generates CCLK	
Uses low-cost, industry-standard Flash		✓	✓			

A specific Spartan-3E part type always requires a constant number of configuration bits, regardless of design complexity, as shown in Table 39. The configuration file size for a multiple-FPGA daisy-chain design equals the sum of the individual file sizes.

Table 39: **Number of Bits to Program a Spartan-E FPGA (Uncompressed Bitstreams)**

Device	Number of Configuration Bits
XC3S100E	581,344
XC3S250E	1,352,192
XC3S500E	2,267,136
XC3S1200E	3,832,320
XC3S1600E	5,957,760

Pin Behavior During Configuration

Table 40 shows how various pins behave during the FPGA configuration process. The actual behavior depends on the values applied to the M2, M1, and M0 mode select pins and the HSWAP pin. The mode select pins determine which of the I/O pins are borrowed during configuration and how they function. In JTAG configuration mode, no user-I/O pins are borrowed for configuration.

All I/O pins are high impedance (floating, three-stated, Hi-Z) during the configuration process. These pins are indicated in Table 40 as shaded table entries or cells. If the HSWAP input is Low, these pins have a pull-up resistor to their associated V_{CCO} supply that is active throughout configuration. After configuration, pull-up and pull-down resistors are available in the FPGA application as described in **Pull-Up and Pull-Down Resistors**, page 9.

Spartan-3E FPGAs have only six dedicated configuration pins, including the DONE and PROG_B pins, and the four JTAG boundary-scan pins: TDI, TDO, TMS, and TCK.

Table 40: **Pin Behavior during Configuration**

Pin Name	Master Serial	SPI (Serial Flash)	BPI (Parallel NOR Flash)	JTAG	Slave Parallel	Slave Serial	Supply/ I/O Bank
TDI	TDI	TDI	TDI	TDI	TDI	TDI	VCCAUX
TMS	TMS	TMS	TMS	TMS	TMS	TMS	VCCAUX
TCK	TCK	TCK	TCK	TCK	TCK	TCK	VCCAUX
TDO	TDO	TDO	TDO	TDO	TDO	TDO	VCCAUX
PROG_B	PROG_B	PROG_B	PROG_B	PROG_B	PROG_B	PROG_B	VCCAUX
DONE	DONE	DONE	DONE	DONE	DONE	DONE	VCCAUX
HSWAP	HSWAP	HSWAP	HSWAP	HSWAP	HSWAP	HSWAP	0
M2	0	0	0	1	1	1	2
M1	0	0	1	0	1	1	2
M0	0	1	0 = Up 1 = Down	1	0	1	2
CCLK	CCLK (O)	CCLK (O)	CCLK (O)		CCLK (I)	CCLK (I)	2
INIT_B	INIT_B	INIT_B	INIT_B		INIT_B	INIT_B	2
CSO_B		CSO_B	CSO_B		CSO_B		2
DOUT/BUSY	DOUT	DOUT	BUSY		BUSY	DOUT	2
MOSI/CSI_B		MOSI	CSI_B		CSI_B		2
D7			D7		D7		2
D6			D6		D6		2
D5			D5		D5		2
D4			D4		D4		2

Table 40: **Pin Behavior during Configuration *(Continued)***

Pin Name	Master Serial	SPI (Serial Flash)	BPI (Parallel NOR Flash)	JTAG	Slave Parallel	Slave Serial	Supply/ I/O Bank
D3			D3		D3		2
D2			D2		D2		2
D1			D1		D1		2
D0/DIN	DIN	DIN	D0		D0	DIN	2
RDWR_B			RDWR_B		RDWR_B		2
A23			A23				2
A22			A22				2
A21			A21				2
A20			A20				2
A19/VS2		VS2	A19				2
A18/VS1		VS1	A18				2
A17/VS0		VS0	A17				2
A16			A16				1
A15			A15				1
A14			A14				1
A13			A13				1
A12			A12				1
A11			A11				1
A10			A10				1
A9			A9				1
A8			A8				1
A7			A7				1
A6			A6				1
A5			A5				1
A4			A4				1
A3			A3				1
A2			A2				1
A1			A1				1
A0			A0				1
LDC0			LDC0		LDC0		1
LDC1			LDC1		LDC1		1
LDC2			LDC2		LDC2		1
HDC			HDC		HDC		1

Table 41 shows the default I/O standard setting for the various configuration pins during the configuration process. The configuration interface is designed primarily for 2.5V operation when the VCCO_2 (and VCCO_1 in BPI mode) connects to 2.5V.

The configuration pins also operate at other voltages by setting VCCO_2 (and VCCO_1 in BPI mode) to either 3.3V or 1.8V. The change on the VCCO supply also changes the I/O drive characteristics. For example, with VCCO = 3.3V, the output current when driving High, I_{OH}, increases to approximately 12 to 16 mA, while the current when driving Low, I_{OL}, remains 8 mA. At VCCO = 1.8V, the output current when driving High, I_{OH}, decreases slightly to approximately 6 to 8 mA. Again, the current when driving Low, I_{OL}, remains 8 mA.

Table 41: **Default I/O Standard Setting During Configuration (VCCO_2 = 2.5V)**

Pin(s)	I/O Standard	Output Drive	Slew Rate
All, including CCLK	LVCMOS25	8 mA	Slow

Master Serial Mode

In Master Serial mode (M[2:0] = <0:0:0>), the Spartan-3E FPGA configures itself from an attached Xilinx Platform Flash PROM, as illustrated in Figure 48. The FPGA supplies the CCLK output clock from its internal oscillator to the attached Platform Flash PROM. In response, the Platform Flash PROM supplies bit-serial data to the FPGA's DIN input and the FPGA accepts this data on each rising CCLK edge.

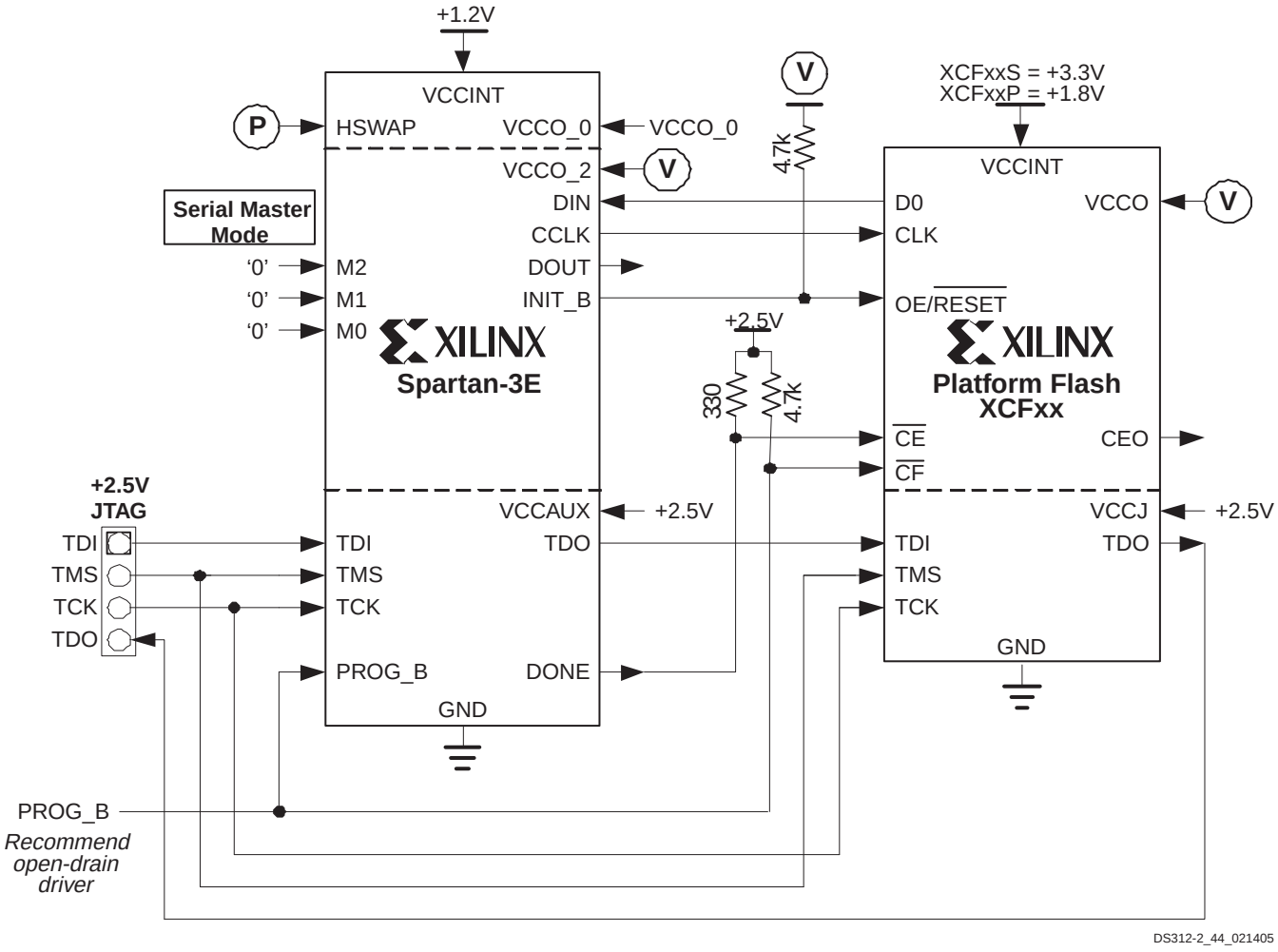


Figure 48: **Master Serial Mode using Platform Flash PROM**

The mode select pins, M[2:0], must all be Low when sampled, when the FPGA's INIT_B output goes High. After configuration, when the FPGA's DONE output goes High, the mode select pins are available as full-featured user-I/O pins.

(P) Similarly, the FPGA's HSWAP pin must be Low to enable pull-up resistors on all user-I/O pins during configuration or High to disable the pull-up resistors. The HSWAP control must remain at a constant logic level throughout FPGA configuration. After configuration, when the FPGA's DONE output goes High, the HSWAP pin is available as full-featured user-I/O pin and is powered by the VCCO_0 supply.

The FPGA's DOUT pin is used in daisy-chain applications, described later. In a single-FPGA application, the FPGA's DOUT pin is not used but is actively driving during the configuration process.

Table 42: **Serial Master Mode Connections**

Pin Name	FPGA Direction	Description	During Configuration	After Configuration
HSWAP (P)	Input	**User I/O Pull-Up Control**. When Low during configuration, enables pull-up resistors in all I/O pins to respective I/O bank V_{CCO} input. 0: Pull-ups during configuration 1: No pull-ups	Drive at valid logic level throughout configuration.	User I/O
M[2:0]	Input	**Mode Select**. Selects the FPGA configuration mode.	M2 = 0, M1 = 0, M0 = 0. Sampled when INIT_B goes High.	User I/O
DIN	Input	**Serial Data Input**.	Receives serial data from PROM's D0 output.	User I/O
CCLK	Output	**Configuration Clock**. Generated by FPGA internal oscillator. Frequency controlled by ***ConfigRate*** bitstream generator option. If CCLK PCB trace is long or has multiple connections, terminate this output to maintain signal integrity.	Drives PROM's CLK clock input.	User I/O
DOUT	Output	**Serial Data Output**.	Actively drives. Not used in single-FPGA designs. In a daisy-chain configuration, this pin connects to DIN input of the next FPGA in the chain.	User I/O
INIT_B	Open-drain bidirectional I/O	**Initialization Indicator**. Active Low. Goes Low at start of configuration during Initialization memory clearing process. Released at end of memory clearing, when mode select pins are sampled. Requires external 4.7 kΩ pull-up resistor to VCCO_2.	Connects to PROM's OE/RESET input. FPGA clears PROM's address counter at start of configuration, enables outputs during configuration. PROM also holds FPGA in Initialization state until PROM reaches Power-On Reset (POR) state. If CRC error detected during configuration, FPGA drives INIT_B Low.	User I/O

Table 42: **Serial Master Mode Connections *(Continued)***

Pin Name	FPGA Direction	Description	During Configuration	After Configuration
DONE	Open-drain bidirectional I/O	**FPGA Configuration Done**. Low during configuration. Goes High when FPGA successfully completes configuration. Requires external 330 Ω pull-up resistor to 2.5V.	Connects to PROM's chip-enable (CE) input. Enables PROM during configuration. Disables PROM after configuration.	Pulled High via external pull-up. When High, indicates that the FPGA successfully configured.
PROG_B	Input	**Program FPGA**. Active Low. When asserted Low for 300 ns or longer, forces the FPGA to restart its configuration process by clearing configuration memory and resetting the DONE and INIT_B pins once PROG_B returns High. Requires external 4.7 kΩ pull-up resistor to 2.5V. If driving externally, use an open-drain or open-collector driver.	Must be High during configuration to allow configuration to start. Connects to PROM's CF pin, allowing JTAG PROM programming algorithm to reprogram the FPGA.	Drive PROG_B Low and release to reprogram FPGA.

Voltage Compatibility

The PROM's V_{CCINT} supply must be either 3.3V for the serial XCFxxS Platform Flash PROMs or 1.8V for the serial/parallel XCFxxP PROMs.

Ⓥ The FPGA's VCCO_2 supply input and the Platform Flash PROM's V_{CCO} supply input must be the same voltage, ideally +2.5V. Both devices also support 1.8V and 3.3V interfaces but the FPGA's PROG_B and DONE pins require special attention as they are powered by the FPGA's VCCAUX supply, nominally 2.5V. See application note XAPP453: "The 3.3V Configuration of Spartan-3 FPGAs" for additional information.

Supported Platform Flash PROMs

Table 43 shows the smallest available Platform Flash PROM to program a single Spartan-3E FPGA. A multiple-FPGA daisy-chain application requires a Platform Flash PROM large enough to contain the sum of the various FPGA file sizes.

Table 43: **Number of Bits to Program a Spartan-3E FPGA and Smallest Platform Flash PROM**

Device	Number of Configuration Bits	Smallest Available Platform Flash
XC3S100E	581,344	XCF01S
XC3S250E	1,352,192	XCF02S
XC3S500E	2,267,136	XCF04S
XC3S1200E	3,832,320	XCF04S
XC3S1600E	5,957,760	XCF08P or 2 x XCF04S

The XC3S1600E requires an 8 Mbit PROM. There are two possible solutions. Either use a single 8 Mbit XCF08P parallel/serial PROM or cascade two 4 Mbit XCF04S serial PROMs. The two XCF04S PROMs use a 3.3V V_{CCINT} supply while the XCF08P requires a 1.8V V_{CCINT} supply. If the board does not already have a 1.8V supply available, the two cascaded XCF04S PROM solution is recommended.

CCLK Frequency

In Master Serial mode, the FPGA's internal oscillator generates the configuration clock frequency. The FPGA provides this clock on its CCLK output pin, driving the PROM's CLK input pin. The FPGA starts configuration at its lowest frequency and increases its frequency for the remainder of the configuration process if so specified in the configuration bitstream. The maximum frequency is specified using the ***ConfigRate*** bitstream generator option. Table 44 shows the maximum ***ConfigRate*** settings, approximately equal to MHz, for various Platform Flash devices and I/O voltages. For the serial XCFxxS PROMs, the maximum frequency also depends on the interface voltage.

Table 44: **Maximum ConfigRate Settings for Platform Flash**

Platform Flash Part Number	I/O Voltage (VCCO_2, VCCO)	Maximum *ConfigRate* Setting
XCF01S XCF02S XCF04S	3.3V or 2.5V	25
	1.8V	12
XCF08P XCF16P XCF32P	3.3V, 2.5V, or 1.8V	25

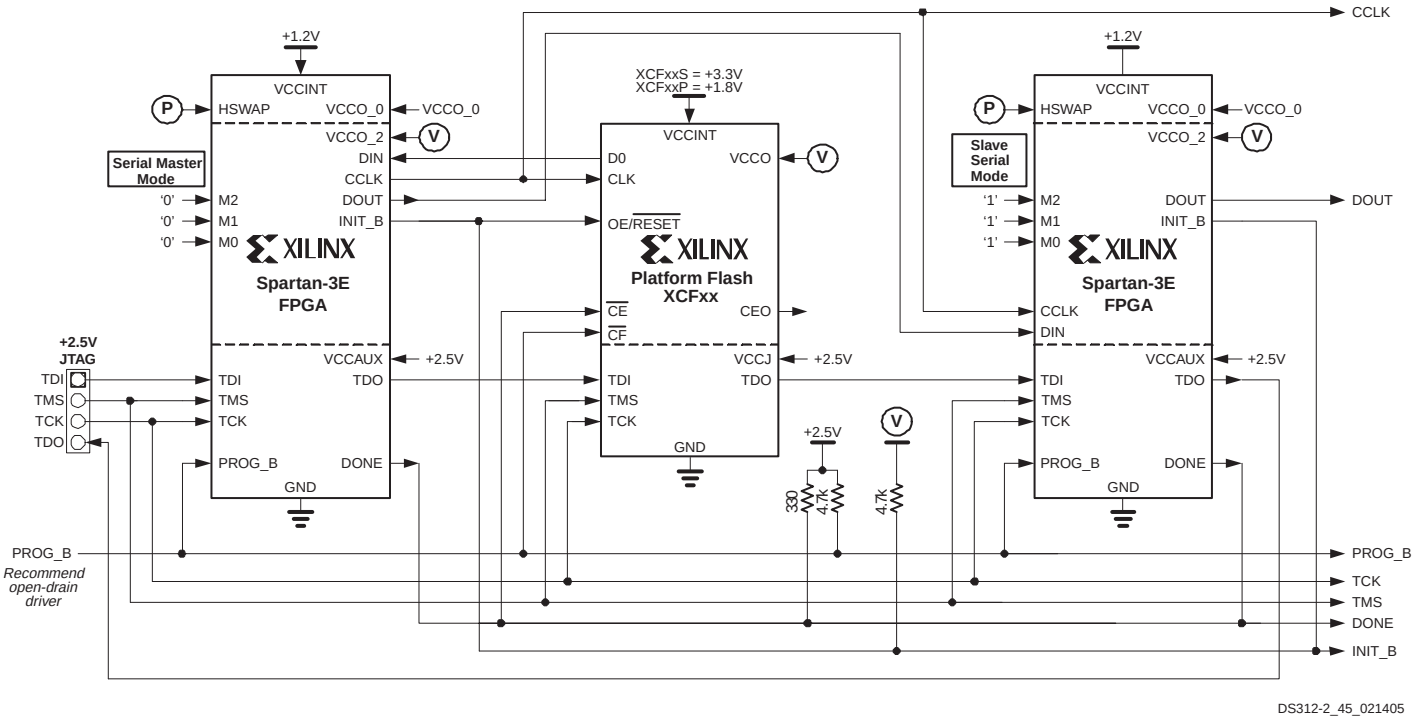


Figure 49: **Daisy-Chaining from Master Serial Mode**

Daisy-Chaining

If the application requires multiple FPGAs with different configurations, then configure the FPGAs using a daisy chain, as shown in Figure 49. Use Master Serial mode (M[2:0] = <0:0:0>) for the FPGA connected to the Platform Flash PROM and Slave Serial mode (M[2:0] = <1:1:1>) for all other FPGAs in the daisy-chain. After the master FPGA—the FPGA on the left in the diagram—finishes loading its configuration data from the Platform Flash, the master device supplies data using its DOUT output pin to the next device in the daisy-chain, on the falling CCLK edge.

JTAG Interface

Both the Spartan-3E FPGA and the Platform Flash PROM have a four-wire IEEE 1149.1/1532 JTAG port. Both devices share the TCK clock input and the TMS mode select input. The devices may connect in either order on the JTAG chain with the TDO output of one device feeding the TDI input of the following device in the chain. The TDO output of the last device in the JTAG chain drives the JTAG connector.

The JTAG interface on Spartan-3E FPGAs is powered by the 2.5V VCCAUX supply. Consequently, the PROM's V_{CCJ} supply input must also be 2.5V. To create a 3.3V JTAG interface, please refer to application note XAPP453: "The 3.3V Configuration of Spartan-3 FPGAs" for additional information.

In-System Programming Support

Both the FPGA and the Platform Flash PROM are in-system programmable via the JTAG chain. Download support is provided by the Xilinx iMPACT programming software and the associated Xilinx Parallel Cable IV, MultiPRO, or Platform Cable USB programming cables.

Storing Additional User Data in Platform Flash

After configuration, the FPGA application can continue to use the Master Serial interface pins to communicate with the Platform Flash PROM. If desired, use a larger Platform Flash PROM to hold additional non-volatile application data, such as MicroBlaze processor code, or other user data such as serial numbers and Ethernet MAC IDs. The FPGA first configures from Platform Flash PROM. Then using FPGA logic after configuration, the FPGA copies MicroBlaze code from Platform Flash into external DDR SDRAM for code execution.

See XAPP694: "Reading User Data from Configuration PROMs" and XAPP482: "MicroBlaze Platform Flash/PROM Boot Loader and User Data Storage" for specific details on how to implement such an interface.

SPI Serial Flash Mode

In SPI Serial Flash mode (M[2:0] = <0:0:0>), the Spartan-3E FPGA configures itself from an attached industry-standard SPI serial Flash PROM, as illustrated in Figure 50 and Figure 52. The FPGA supplies the CCLK output clock from its internal oscillator to the clock input of the attached SPI Flash PROM.

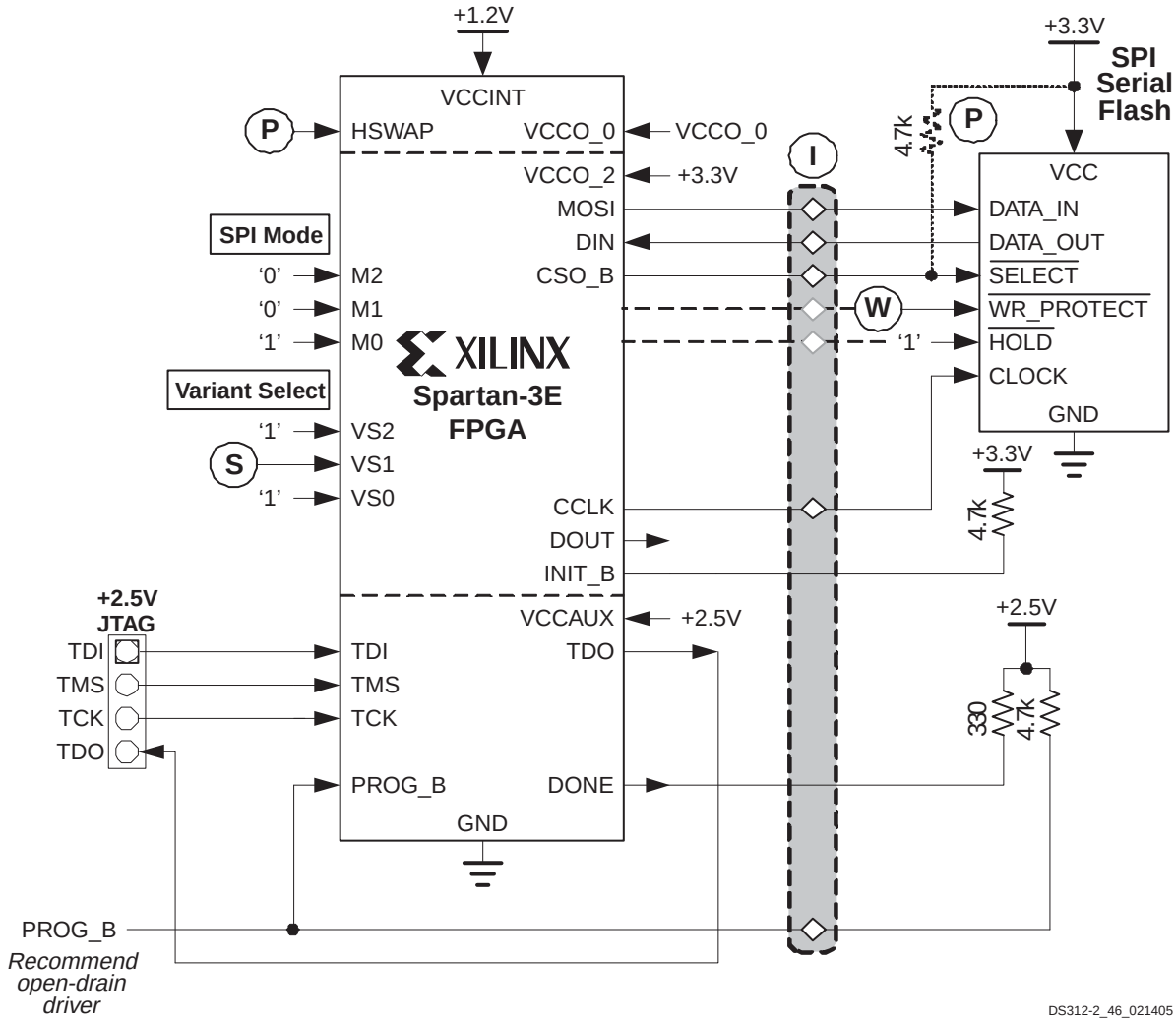


Figure 50: **SPI Flash PROM Interface for PROMs Supporting READ (0x03) and FAST_READ (0x0B)**

(S) Although SPI is a standard four-wire interface, various available SPI Flash PROMs use different command protocols. The FPGA's variant select pins, VS[2:0], define how the FPGA communicates with the SPI Flash, including which SPI Flash command the FPGA issues to start the read operation and the number of dummy bytes inserted before the FPGA expects to receive valid data from the SPI Flash. Table 45 shows the available SPI Flash PROMs expected to operate with Spartan-3E FPGAs. Other compatible devices might work but have not been tested for suitability with Spartan-3E FPGAs. All other VS[2:0] values are reserved for future use.

Figure 50 shows the general connection diagram for those SPI Flash PROMs that support the 0x03 READ command or the 0x0B FAST READ commands.

Figure 51 shows the connection diagram for Atmel DataFlash serial PROMs, which also use an SPI-based protocol.

Figure 54 demonstrates how to configure multiple FPGAs with different configurations, all stored in a single SPI Flash. The diagram uses standard SPI Flash memories but the same general technique applies for Atmel DataFlash.

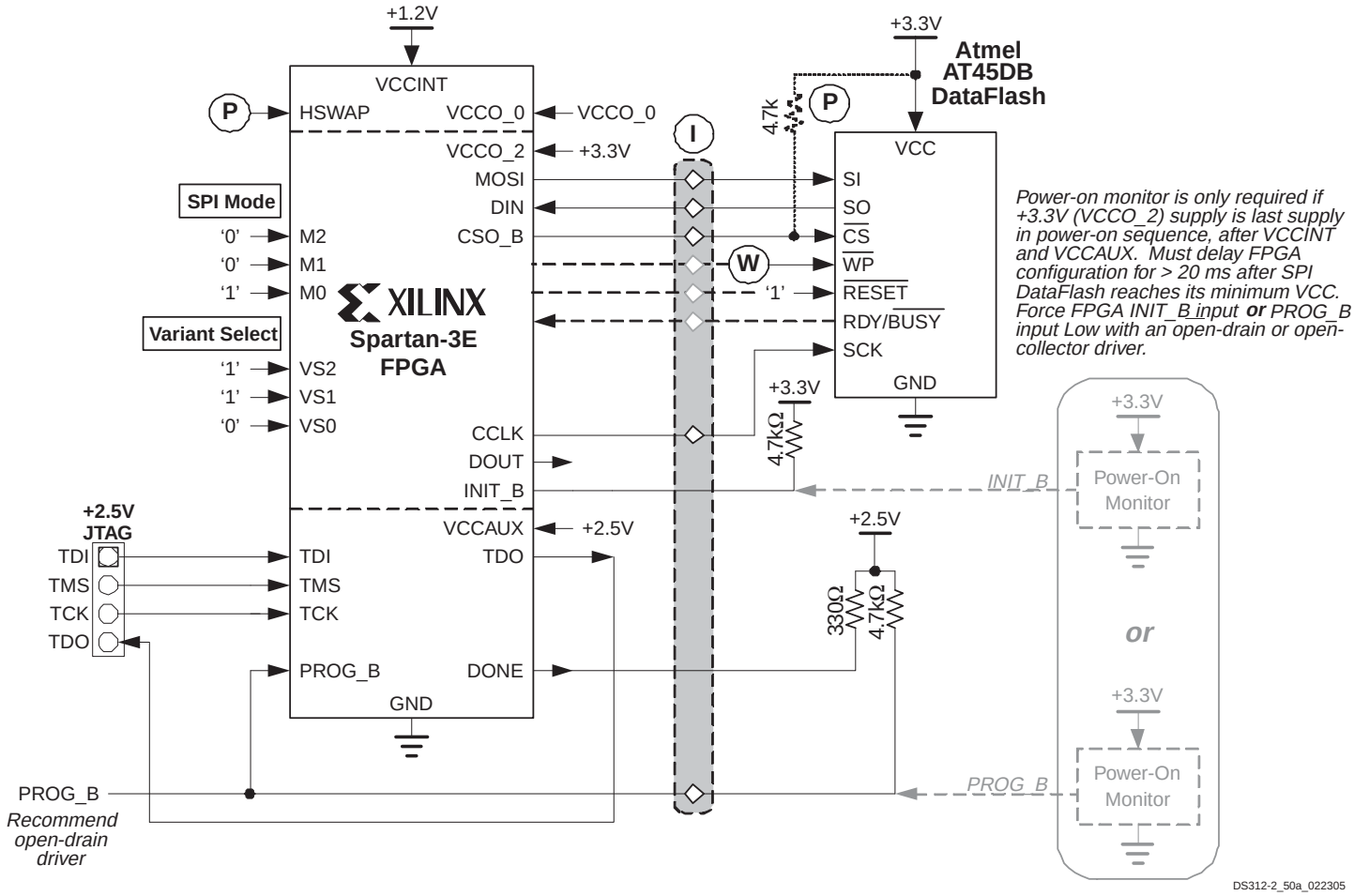


Figure 51: **Atmel SPI-based DataFlash Configuration Interface**

Table 45: **Variant Select Codes for SPI Serial Flash PROMs**

VS2	VS1	VS0	SPI Read Command	Dummy Bytes	SPI Serial Flash Vendor	SPI Flash Family
1	1	1	FAST READ (0x0B) (see Figure 50)	1	STMicroelectronics (ST)	M25Pxx
					NexFlash	NX25Pxx
					Silicon Storage Technology (SST)	SST25LFxxxA SST25VFxxxA
					Programmable Microelectronics Corp. (PMC)	Pm25LVxxx
1	0	1	READ (0x03) (see Figure 50)	0	STMicroelectronics (ST)	M25Pxx
					NexFlash	NX25Pxx
					Silicon Storage Technology (SST)	SST25LFxxxA SST25VFxxxA SST25VFxxx
					Programmable Microelectronics Corp. (PMC)	Pm25LVxxx
1	1	0	READ ARRAY (0xE8) (see Figure 51)	3	Atmel Corporation	AT45DB DataFlash
Others			Reserved			

(W) Table 46 shows the connections between the SPI Flash PROM and the FPGA's SPI configuration interface. Each SPI Flash PROM vendor uses slightly different signal naming. The SPI Flash PROM's write protect and hold controls are not used by the FPGA during configuration. However, the $\overline{\text{HOLD}}$ pin must be High during the configuration process. The PROM's write protect input must be High in order to write or program the Flash memory.

Table 46: **SPI Flash PROM Connections and Pin Naming**

SPI Flash Pin	FPGA Connection	STMicro	NexFlash	Silicon Storage Technology	Atmel DataFlash
DATA_IN	MOSI	D	DI	SI	SI
DATA_OUT	DIN	Q	DO	SO	SO
$\overline{\text{SELECT}}$	CSO_B	$\overline{\text{S}}$	$\overline{\text{CS}}$	CE#	$\overline{\text{CS}}$
CLOCK	CCLK	C	CLK	SCK	SCK
$\overline{\text{WR\_PROTECT}}$ (W)	Not required for FPGA configuration. Must be High to program SPI Flash. Optional connection to FPGA user I/O after configuration.	$\overline{\text{W}}$	$\overline{\text{WP}}$	WP#	$\overline{\text{WP}}$
$\overline{\text{HOLD}}$ (see Figure 50)	Not required for FPGA configuration but must be High during configuration. Optional connection to FPGA user I/O after configuration. Not applicable to Atmel DataFlash.	$\overline{\text{HOLD}}$	$\overline{\text{HOLD}}$	HOLD#	N/A

Table 46: **SPI Flash PROM Connections and Pin Naming *(Continued)***

SPI Flash Pin	FPGA Connection	STMicro	NexFlash	Silicon Storage Technology	Atmel DataFlash
$\overline{\text{RESET}}$ (see Figure 51)	Only applicable to Atmel DataFlash. Not required for FPGA configuration but must be High during configuration. Optional connection to FPGA user I/O after configuration. Do not connect to FPGA's PROG_B as this will prevent direct programming of the DataFlash.	N/A	N/A	N/A	$\overline{\text{RESET}}$
RDY/$\overline{\text{BUSY}}$ (see Figure 51)	Only applicable to Atmel DataFlash and only available on certain packages. Not required for FPGA configuration. Output from DataFlash PROM. Optional connection to FPGA user I/O after configuration.	N/A	N/A	N/A	RDY/$\overline{\text{BUSY}}$

The mode select pins, M[2:0], and the variant select pins, VS[2:0] are sampled when the FPGA's INIT_B output goes High and must be at defined logic levels during this time. After configuration, when the FPGA's DONE output goes High, these pins are all available as full-featured user-I/O pins.

(P) Similarly, the FPGA's HSWAP pin must be Low to enable pull-up resistors on all user-I/O pins or High to disable the pull-up resistors. The HSWAP control must remain at a constant logic level throughout FPGA configuration. After configuration, when the FPGA's DONE output goes High, the HSWAP pin is available as full-featured user-I/O pin and is powered by the VCCO_0 supply.

In a single-FPGA application, the FPGA's DOUT pin is not used but is actively driving during the configuration process.

Table 47: **Serial Peripheral Interface (SPI) Connections**

Pin Name	FPGA Direction	Description	During Configuration	After Configuration
HSWAP (P)	Input	**User I/O Pull-Up Control**. When Low during configuration, enables pull-up resistors in all I/O pins to respective I/O bank V_{CCO} input. 0: Pull-ups during configuration 1: No pull-ups	Drive at valid logic level throughout configuration.	User I/O
M[2:0]	Input	**Mode Select**. Selects the FPGA configuration mode.	M2 = 0, M1 = 0, M0 = 1. Sampled when INIT_B goes High.	User I/O
VS[2:0] (S)	Input	**Variant Select**. Instructs the FPGA how to communicate with the attached SPI Flash PROM.	Must be at the logic levels shown in Table 45. Sampled when INIT_B goes High.	User I/O
MOSI	Output	**Serial Data Output**.	FPGA sends SPI Flash memory read commands and starting address to the PROM's serial data input.	User I/O
DIN	Input	**Serial Data Input**.	FPGA receives serial data from PROM's serial data output.	User I/O

Table 47: **Serial Peripheral Interface (SPI) Connections *(Continued)***

Pin Name	FPGA Direction	Description	During Configuration	After Configuration
CSO_B	Output	**Chip Select Output**. Active Low.	Connects to the SPI Flash PROM's chip-select input. If HSWAP = 1, connect this signal to a 4.7 kΩ pull-up resistor to 3.3V.	Drive CSO_B High after configuration to disable the SPI Flash and reclaim the MOSI, DIN, and CCLK pins. Optionally, re-use this pin and MOSI, DIN, and CCLK to continue communicating with SPI Flash.
CCLK	Output	**Configuration Clock**. Generated by FPGA internal oscillator. Frequency controlled by ***ConfigRate*** bitstream generator option. If CCLK PCB trace is long or has multiple connections, terminate this output to maintain signal integrity.	Drives PROM's clock input.	User I/O
DOUT	Output	**Serial Data Output**.	Actively drives. Not used in single-FPGA designs. In a daisy-chain configuration, this pin connects to DIN input of the next FPGA in the chain.	User I/O
INIT_B	Open-drain bidirectional I/O	**Initialization Indicator**. Active Low. Goes Low at start of configuration during Initialization memory clearing process. Released at end of memory clearing, when mode select pins are sampled. In daisy-chain applications, this signal requires an external 4.7 kΩ pull-up resistor to VCCO_2.	Active during configuration. If SPI Flash PROM requires > 2 ms to awake after powering on, hold INIT_B Low until PROM is ready. If CRC error detected during configuration, FPGA drives INIT_B Low.	User I/O
DONE	Open-drain bidirectional I/O	**FPGA Configuration Done**. Low during configuration. Goes High when FPGA successfully completes configuration. Requires external 330 Ω pull-up resistor to 2.5V.	Low indicates that the FPGA is not yet configured.	Pulled High via external pull-up. When High, indicates that the FPGA successfully configured.
PROG_B	Input	**Program FPGA**. Active Low. When asserted Low for 300 ns or longer, forces the FPGA to restart its configuration process by clearing configuration memory and resetting the DONE and INIT_B pins once PROG_B returns High. Requires external 4.7 kΩ pull-up resistor to 2.5V. If driving externally, use an open-drain or open-collector driver.	Must be High to allow configuration to start.	Drive PROG_B Low and release to reprogram FPGA. Hold PROG_B to force FPGA I/O pins into Hi-Z, allowing direct programming access to SPI Flash PROM pins.

Voltage Compatibility

Available SPI Flash PROMs use a single 3.3V supply voltage. All of the FPGA's SPI Flash interface signals are within I/O Bank 2. Consequently, the FPGA's VCCO_2 supply voltage must also be 3.3V to match the SPI Flash PROM.

Power-On Precautions if 3.3V Supply is Last in Sequence

Spartan-3E FPGAs have a built-in power-on reset (POR) circuit, as shown in Figure 63. The FPGA waits for its three power supplies — VCCINT, VCCAUX, and VCCO to I/O Bank 2 (VCCO_2) — to reach their respective power-on thresholds before beginning the configuration process.

The SPI Flash PROM is powered by the same voltage supply feeding the FPGA's VCCO_2 voltage input, typically 3.3V. SPI Flash PROMs specify that they cannot be accessed until their VCC supply reaches its minimum data sheet voltage, followed by an additional delay. For some devices, this additional delay is as little as 10 µs as shown in Table 48. For other vendors, it is as much as 20 ms.

Table 48: **Example Minimum Power-On to Select Times for Various SPI Flash PROMs**

Vendor	SPI Flash PROM Part Number	Data Sheet Minimum Time from VCC, min. to Select = Low		
		Symbol	Value	Units
STMicroelectronics	M25Pxx	T_{VSL}	10	µs
NexFlash	NX25xx	T_{VSL}	10	µs
Silicon Storage Technology	SST25LFxx	$T_{PU-READ}$	10	µs
Programmable Microelectronics Corporation	Pm25LVxxx	T_{VCS}	50	µs
Atmel Corporation	AT45DBxx		20	ms

In many systems, the 3.3V supply feeding the FPGA's VCCO_2 input is valid before the FPGA's other VCCINT and VCCAUX supplies, and consequently, there is no issue. However, if the 3.3V supply feeding the FPGA's VCCO_2 supply is last in the sequence, a potential race occurs between the FPGA and the SPI Flash PROM, as shown in Figure 52.

If the FPGA's VCCINT and VCCAUX supplies are already valid, then the FPGA waits for VCCO_2 to reach its minimum threshold voltage before starting configuration. This threshold voltage is labeled as V_{CCO2T} in Module 3 and ranges from approximately 0.4V to 1.0V, substantially lower than the SPI Flash PROM's minimum voltage. Once all three FPGA supplies reach their respective Power On Reset (POR) thresholds, the FPGA starts the configuration process and begins initializing its internal configuration memory. Initialization requires approximately 1 ms (T_{POR}, minimum in Module 3), after which the FPGA deasserts INIT_B, selects the SPI Flash PROM, and starts sending the appropriate read command. The SPI Flash PROM must be ready for read operations at this time.

If the 3.3V supply is last in the sequence and does not ramp fast enough, or if the SPI Flash PROM cannot be ready when required by the FPGA, delay the FPGA configuration process by holding either the FPGA's PROG_B input or INIT_B input Low, as highlighted in Figure 51. Release the FPGA when the SPI Flash PROM is ready. For example, a simple R-C delay circuit attached to the INIT_B pin forces the FPGA to wait for a preselected amount of time. Alternately, a Power Good signal from the 3.3V supply or a system reset signal accomplishes the same purpose. Use an open-drain or open-collector output when driving PROG_B or INIT_B.

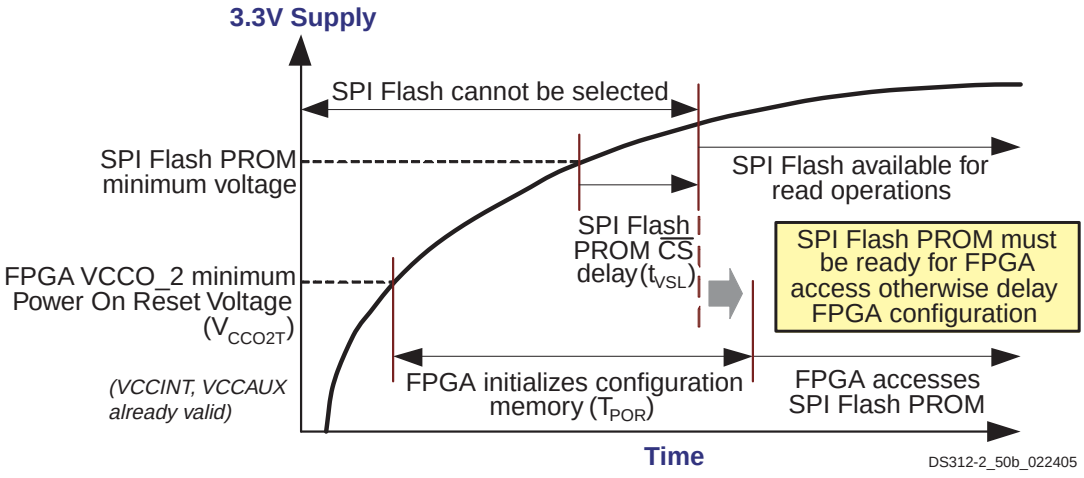


Figure 52: **SPI Flash PROM/FPGA Power-On Timing if 3.3V Supply is Last in Power-On Sequence**

SPI Flash PROM Density Requirements

Table 49 shows the smallest usable SPI Flash PROM to program a single Spartan-3E FPGA. Commercially available SPI Flash PROMs range in density from 1 Mbit to 128 Mbits. A multiple-FPGA daisy-chained application requires a SPI Flash PROM large enough to contain the sum of the FPGA file sizes. An application can also use a larger-density SPI Flash PROM to hold additional data beyond just FPGA configuration data. For example, the SPI Flash PROM can also store application code for a MicroBlaze™ RISC processor core integrated in the Spartan-3E FPGA. See **Using the SPI Flash Interface after Configuration**.

Table 49: **Number of Bits to Program a Spartan-3E FPGA and Smallest SPI Flash PROM**

Device	Number of Configuration Bits	Smallest Usable SPI Flash PROM
XC3S100E	581,344	1 Mbit
XC3S250E	1,352,192	2 Mbit
XC3S500E	2,267,136	4 Mbit
XC3S1200E	3,832,320	4 Mbit
XC3S1600E	5,957,760	8 Mbit

CCLK Frequency

In SPI Flash mode, the FPGA's internal oscillator generates the configuration clock frequency. The FPGA provides this clock on its CCLK output pin, driving the PROM's clock input pin. The FPGA starts configuration at its lowest frequency and increases its frequency for the remainder of the configuration process if so specified in the configuration bitstream. The maximum frequency is specified using the ***ConfigRate*** bitstream generator option. The maximum frequency supported by the FPGA configuration logic depends on the timing for the SPI Flash device. Without examining the timing for a specific SPI Flash PROM, use ***ConfigRate*** = 12, which is approximately 12 MHz. SPI Flash PROMs that support the FAST READ command support higher data rates. Some such PROMs support up to ***ConfigRate*** = 25 and beyond but require careful data sheet analysis.

Using the SPI Flash Interface after Configuration

After the FPGA successfully completes configuration, all of the pins connected to the SPI Flash PROM are available as user-I/O pins.

If not using the SPI Flash PROM after configuration, drive CSO_B High to disable the PROM. The MOSI, DIN, and CCLK pins are then available to the FPGA application.

Because all the interface pins are user I/O after configuration, the FPGA application can continue to use the SPI Flash interface pins to communicate with the SPI Flash PROM, as shown in Figure 53. SPI Flash PROMs offer random-accessible, byte-addressable, read/write, non-volatile storage to the FPGA application.

SPI Flash PROMs are available in densities ranging from 1 Mbit up to 128 Mbits. However, a single Spartan-3E FPGA requires less than 6 Mbits. If desired, use a larger SPI Flash PROM to contain additional non-volatile application data, such as MicroBlaze processor code, or other user data such as serial numbers and Ethernet MAC IDs. In the example shown in Figure 53, the FPGA configures from SPI Flash PROM. Then using FPGA logic after configuration, the FPGA copies MicroBlaze code from SPI Flash into external DDR SDRAM for code execution. Similarly, the FPGA application can store non-volatile application data within the SPI Flash PROM.

The FPGA configuration data is stored starting at location 0. Store any additional data beginning in the next available SPI Flash PROM sector or page. Do not mix configuration data and user data in the same sector or page.

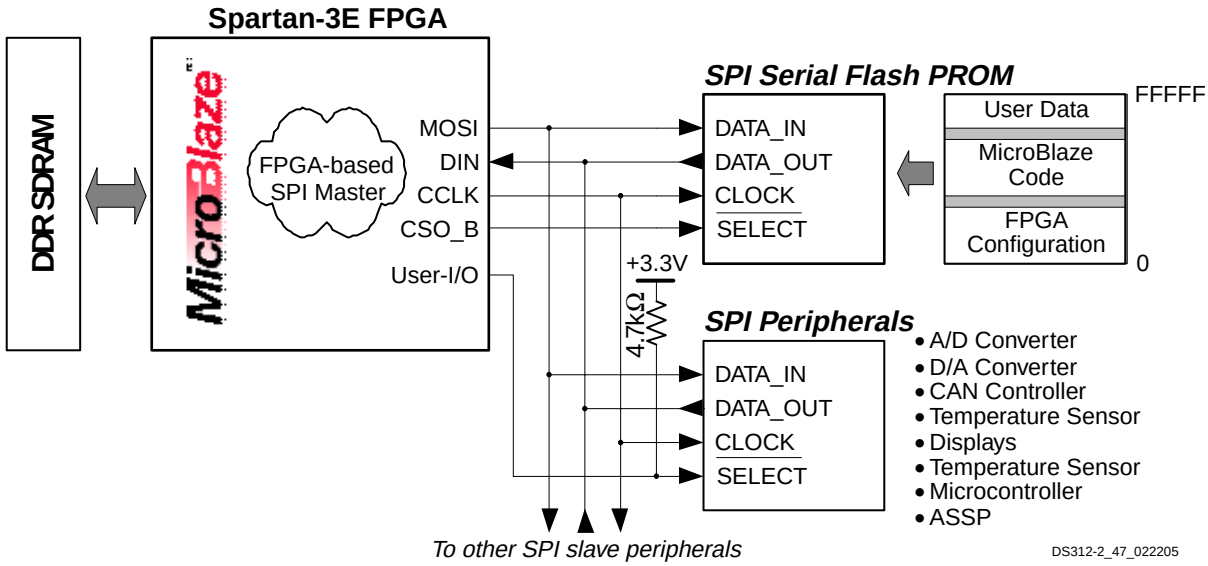


Figure 53: **Using the SPI Flash Interface After Configuration**

Similarly, the SPI bus can be expanded to additional SPI peripherals. Because SPI is a common industry-standard interface, there are a variety of SPI-based peripherals available, including analog-to-digital (A/D) converters, digital-to-analog (D/A) converters, CAN controllers, and temperature sensors.

The MOSI, DIN, and CCLK pins are common to all SPI peripherals. Connect the select input on each additional SPI peripheral to one of the FPGA user I/O pins. If HSWAP = 0 during configuration, the FPGA holds the select line High. If HSWAP = 1, connect the select line to +3.3V via an external 4.7 kΩ pull-up resistor to avoid spurious read or write operations. After configuration, drive the select line Low to select the desired SPI peripheral. Refer to the individual SPI peripheral data sheet for specific interface and communication protocol requirements.

Daisy-Chaining

If the application requires multiple FPGAs with different configurations, then configure the FPGAs using a daisy chain, as shown in Figure 54. Use SPI Flash mode (M[2:0] = <0:0:1>) for the FPGA connected to the Platform Flash PROM and Slave Serial mode (M[2:0] = <1:1:1>) for all other FPGAs in the daisy-chain. After the master FPGA—the FPGA on the left in the diagram—finishes loading its configuration data from the SPI Flash PROM, the master device uses its DOUT output pin to supply data to the next device in the daisy-chain, on the falling CCLK edge.

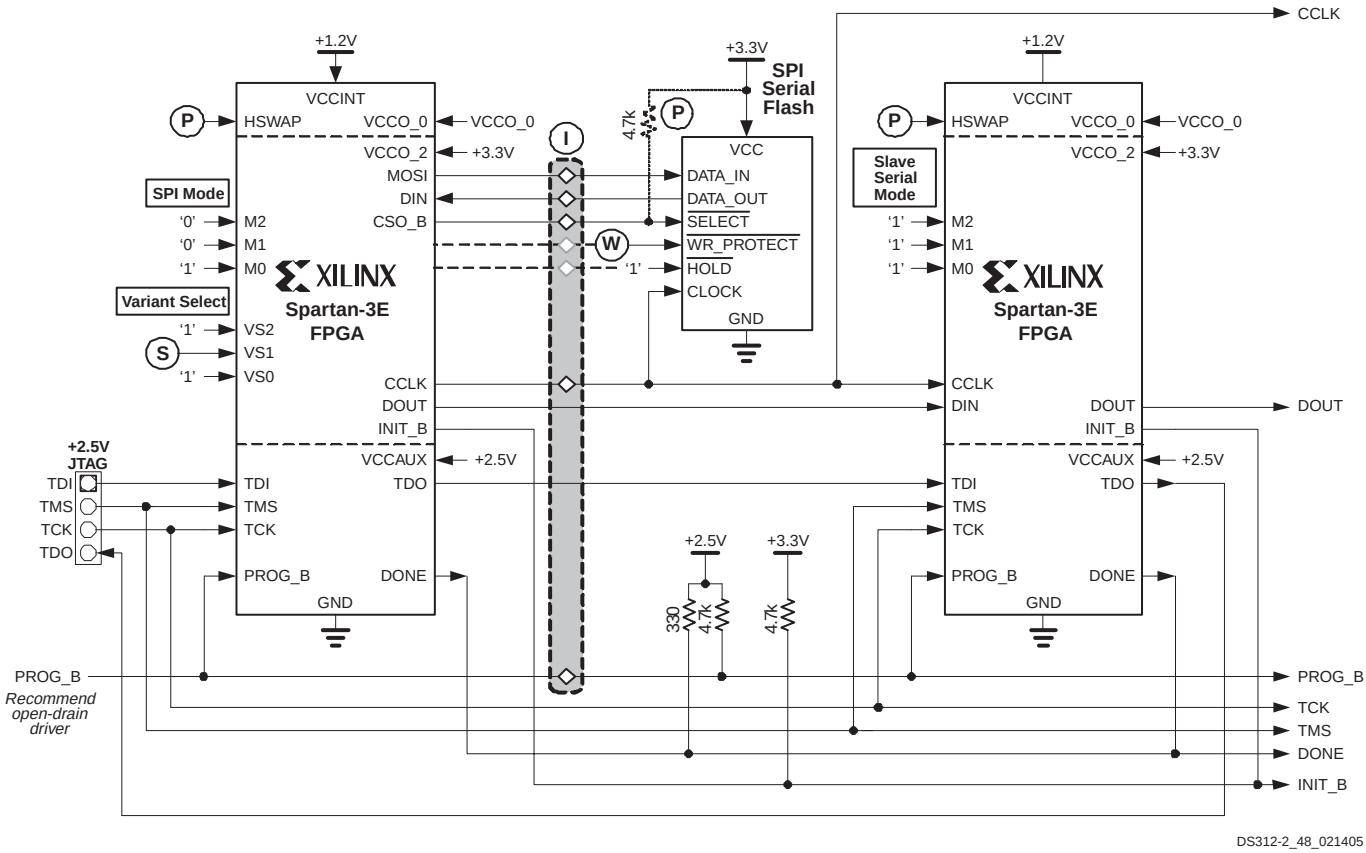


Figure 54: **Daisy-Chaining from SPI Flash Mode**

In-System Programming Support

(I) In a production application, the SPI Flash PROM is usually pre-programmed before it is mounted on the printed circuit board. In-system programming support is available from some third-party PROM programmers using a socket adapter with attached wires. To gain access to the SPI Flash signals, drive the FPGA's PROG_B input Low with an open-drain driver. This action places all FPGA I/O pins, including those attached to the SPI Flash, in high-impedance (Hi-Z). If the HSWAP input is High, the I/Os have pull-up resistors to the V_{CCO} input on their respective I/O bank. The external programming hardware then has direct access to the SPI Flash pins. The programming access points are highlighted in the gray box in Figure 50, Figure 51, and Figure 54.

Byte-Wide Peripheral Interface (BPI) Parallel Flash Mode

In Byte-wide Peripheral Interface (BPI) mode (M[2:0] = <0:1:0> or <0:1:1>), a Spartan-3E FPGA configures itself from an industry-standard parallel NOR Flash PROM, as illustrated in Figure 55. The FPGA generates up to a 24-bit address lines to access an attached parallel Flash. Only 20 address lines are generated for Spartan-3E FPGAs in the TQ144 package. The BPI mode is not available for Spartan-3E FPGAs in the VQ100 package.

The interface is designed for standard parallel NOR Flash PROMs and supports both byte-wide (x8) and byte-wide/halfword (x8/x16) PROMs. The interface does not support halfword-only (x16) PROMs. The interface works equally wells with other memories that use a similar interface such as SRAM, EPROM, or masked ROM but is primarily designed for Flash memory.

There is another type of Flash memory called NAND Flash, which is commonly used in memory cards for digital cameras, etc. Spartan-3E FPGAs do not configure directly from NAND Flash memories.

The FPGA's internal oscillator controls the interface timing and the FPGA supplies the clock on the CCLK output pin. However, the CCLK signal is not used in single FPGA applications. Similarly, the FPGA drives three pins Low during configuration (LDC[2:0]) and one pin High during configuration (HDC) to the PROM's control inputs.

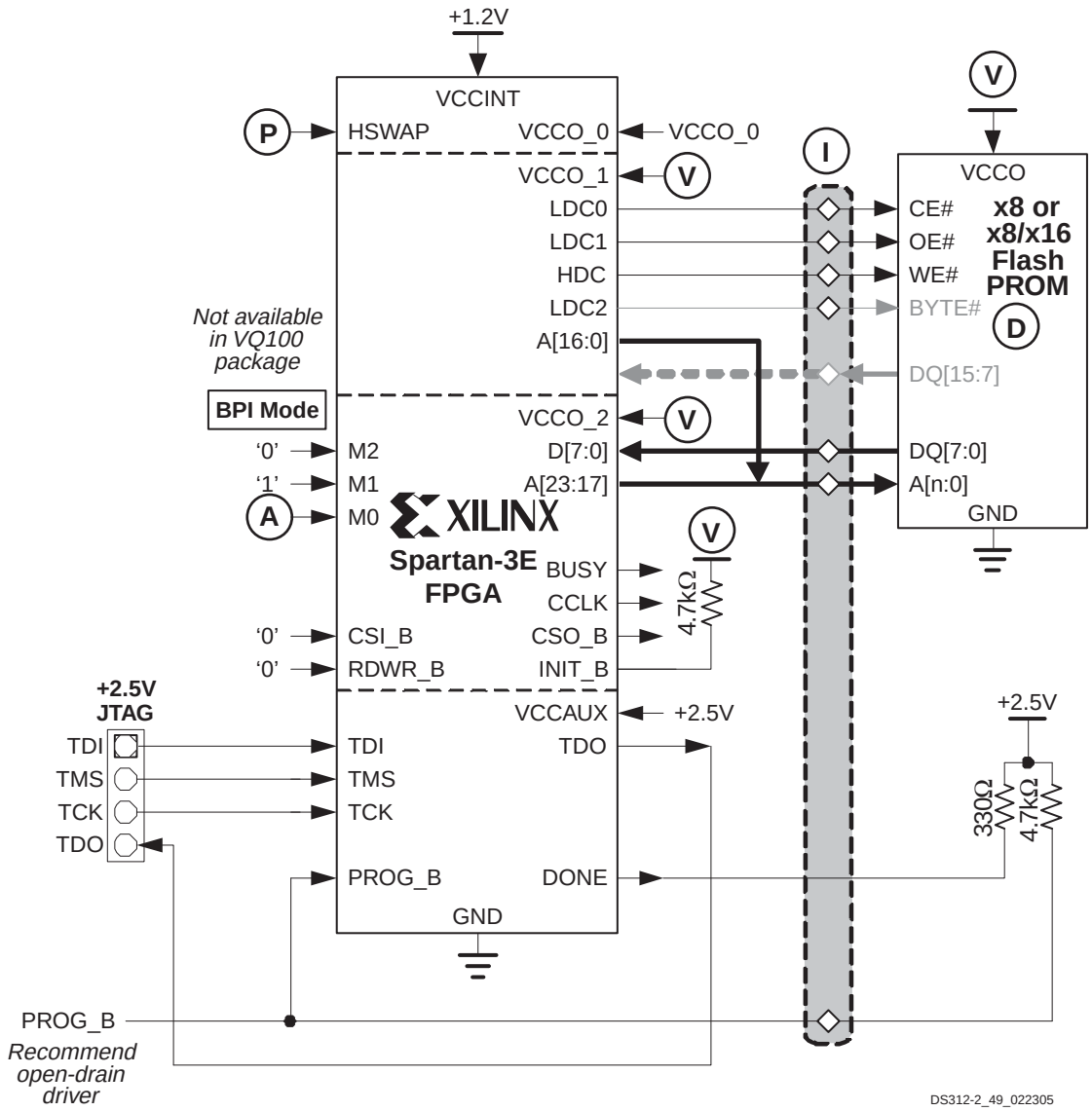


Figure 55: **Byte-wide Peripheral Interface (BPI) Mode Configured from Parallel NOR Flash PROMs**

(A) During configuration, the value of the M0 mode pin determines how the FPGA generates addresses, as shown Table 50. When M0 = 0, the FPGA generates addresses starting at 0 and increments the address on every falling CCLK edge. Conversely, when M0 = 1, the FPGA generates addresses starting at 0xFF_FFFF (all ones) and decrements the address on every falling CCLK edge.

Table 50: **BPI Addressing Control**

M2	M1	M0	Start Address	Addressing
0	1	0	0	Incrementing
		1	0xFF_FFFF	Decrementing

This addressing flexibility allows the FPGA to share the parallel Flash PROM with an external or embedded processor. Depending on the specific processor architecture, the processor boots either from the top or bottom of memory. The FPGA is flexible and boots from the opposite end of memory from the processor. Only the processor or the FPGA can boot at any given time. The FPGA can configure first, holding the processor in reset or the processor can boot first, asserting the FPGA's PROG_B pin.

The mode select pins, M[2:0], are sampled when the FPGA's INIT_B output goes High and must be at defined logic levels during this time. After configuration, when the FPGA's DONE output goes High, the mode pins are available as full-featured user-I/O pins.

(P) Similarly, the FPGA's HSWAP pin must be Low to enable pull-up resistors on all user-I/O pins or High to disable the pull-up resistors. The HSWAP control must remain at a constant logic level throughout FPGA configuration. After configuration, when the FPGA's DONE output goes

High, the HSWAP pin is available as full-featured user-I/O pin and is powered by the VCCO_0 supply.

The RDWR_B and CSI_B must be Low throughout the configuration process. After configuration, these pins also become user I/O.

In a single-FPGA application, the FPGA's CSO_B and CCLK pins are not used but are actively driving during the configuration process. The BUSY pin is not used but also actively drives during configuration and is available as a user I/O after configuration.

After configuration, all of the interface pins except DONE and PROG_B are available as user I/Os. Furthermore, the bidirectional SelectMAP configuration peripheral interface (see **Slave Parallel Mode**) is available after configuration. To continue using SelectMAP mode, set the ***Persist*** bitstream generator option to ***Yes***. An external host can then read and verify configuration data.

Table 51: **Byte-Wide Peripheral Interface (BPI) Connections**

Pin Name	FPGA Direction	Description	During Configuration	After Configuration
HSWAP (P)	Input	**User I/O Pull-Up Control**. When Low during configuration, enables pull-up resistors in all I/O pins to respective I/O bank V_{CCO} input. 0: Pull-ups during configuration 1: No pull-ups	Drive at valid logic level throughout configuration.	User I/O
M[2:0] (A)	Input	**Mode Select**. Selects the FPGA configuration mode.	M2 = 0, M1 = 1. Set M0 = 0 to start at address 0, increment addresses. Set M0 = 1 to start at address 0xFFFFFF and decrement addresses. Sampled when INIT_B goes High.	User I/O
CSI_B	Input	**Chip Select Input**. Active Low.	Must be Low throughout configuration.	User I/O. If bitstream option ***Persist=Yes***, becomes part of SelectMap parallel peripheral interface.
RDWR_B	Input	**Read/Write Control**. Active Low write enable. Read functionality typically only used after configuration, if bitstream option ***Persist=Yes***.	Must be Low throughout configuration.	User I/O. If bitstream option ***Persist=Yes***, becomes part of SelectMap parallel peripheral interface.
LDC0	Output	PROM Chip Enable	Connect to PROM chip-select input (CE#). FPGA drives this signal Low throughout configuration.	User I/O
LDC1	Output	PROM Output Enable	Connect to PROM output-enable input (OE#). FPGA drives this signal Low throughout configuration.	User I/O
HDC	Output	PROM Write Enable	Connect to PROM write-enable input (WE#). FPGA drives this signal High throughout configuration.	User I/O
LDC2 (D)	Output	PROM Byte Mode	This signal is not used for x8 PROMs. For PROMs with a x8/x16 data width control, connect to PROM byte-mode input (BYTE#). See **Precautions Using x8/x16 Flash PROMs**. FPGA drives this signal Low throughout configuration.	User I/O. Drive this pin High after configuration to use a x8/x16 PROM in x16 mode.

Table 51: **Byte-Wide Peripheral Interface (BPI) Connections *(Continued)***

Pin Name	FPGA Direction	Description	During Configuration	After Configuration
A[23:0]	Output	Address	Connect to PROM address inputs. High order address lines may not be available in all packages and not all may be required. Number of address lines required depends on the size of the attached Flash PROM. FPGA address generation controlled by M0 mode pin. Addresses presented on falling CCLK edge. Only 20 address lines are available in TQ144 package.	User I/O
D[7:0]	Input	Data Input	FPGA receives byte-wide data on these pins in response the address presented on A[23:0]. Data captured by FPGA	User I/O If bitstream option ***Persist=Yes***, becomes part of SelectMap parallel peripheral interface.
CSO_B	Output	**Chip Select Output**. Active Low.	Not used in single FPGA applications. In a daisy-chain configuration, this pin connects to the CSI_B pin of the next FPGA in the chain. Actively drives.	User I/O
BUSY	Output	**Busy Indicator**. Typically only used after configuration, if bitstream option ***Persist=Yes***.	Not used during configuration but actively drives.	User I/O. If bitstream option ***Persist=Yes***, becomes part of SelectMap parallel peripheral interface.
CCLK	Output	**Configuration Clock**. Generated by FPGA internal oscillator. Frequency controlled by ***ConfigRate*** bitstream generator option. If CCLK PCB trace is long or has multiple connections, terminate this output to maintain signal integrity.	Not used in single FPGA applications but actively drives. In a daisy-chain configuration, drives the CCLK inputs of all other FPGAs in the daisy-chain.	User I/O If bitstream option ***Persist=Yes***, becomes part of SelectMap parallel peripheral interface.
INIT_B	Open-drain bidirectional I/O	**Initialization Indicator**. Active Low. Goes Low at start of configuration during Initialization memory clearing process. Released at end of memory clearing, when mode select pins are sampled. In daisy-chain applications, this signal requires an external 4.7 kΩ pull-up resistor to VCCO_2.	Active during configuration. If CRC error detected during configuration, FPGA drives INIT_B Low.	User I/O

Table 51: **Byte-Wide Peripheral Interface (BPI) Connections *(Continued)***

Pin Name	FPGA Direction	Description	During Configuration	After Configuration
DONE	Open-drain bidirectional I/O	**FPGA Configuration Done**. Low during configuration. Goes High when FPGA successfully completes configuration. Requires external 330 Ω pull-up resistor to 2.5V.	Low indicates that the FPGA is not yet configured.	Pulled High via external pull-up. When High, indicates that the FPGA successfully configured.
PROG_B	Input	**Program FPGA**. Active Low. When asserted Low for 300 ns or longer, forces the FPGA to restart its configuration process by clearing configuration memory and resetting the DONE and INIT_B pins once PROG_B returns High. Requires external 4.7 kΩ pull-up resistor to 2.5V. If driving externally, use an open-drain or open-collector driver.	Must be High to allow configuration to start.	Drive PROG_B Low and release to reprogram FPGA. Hold PROG_B to force FPGA I/O pins into Hi-Z, allowing direct programming access to Flash PROM pins.

Voltage Compatibility

Ⓥ The FPGA's parallel Flash interface signals are within I/O Banks 1 and 2. The majority of parallel Flash PROMs use a single 3.3V supply voltage. Consequently, in most cases, the FPGA's VCCO_1 and VCCO_2 supply voltages must also be 3.3V to match the parallel Flash PROM. There are some 1.8V parallel Flash PROMs available and the FPGA interfaces with these devices if the VCCO_1 and VCCO_2 supplies are also 1.8V.

Supported Parallel NOR Flash PROM Densities

Table 52 indicates the smallest usable parallel Flash PROM to program a single Spartan-3E FPGA. Parallel Flash density is specified in bits but addressed as bytes. The FPGA presents up to 24 address lines during configuration but not all are required for single FPGA applications. Table 52 shows the minimum required number of address lines between the FPGA and parallel Flash PROM. The actual number of address line required depends on the density of the attached parallel Flash PROM.

A multiple-FPGA daisy-chained application requires a parallel Flash PROM large enough to contain the sum of the FPGA file sizes. An application may also use a larger-density parallel Flash PROM to hold additional data beyond just FPGA configuration data. For example, the parallel Flash PROM could also contain the application code for a MicroBlaze RISC processor core implemented within the Spartan-3E FPGA. After configuration, the MicroBlaze processor could execute directly from external Flash or could copy the code to other, faster system memory before executing the code.

Table 52: **Number of Bits to Program a Spartan-3E FPGA and Smallest Parallel Flash PROM**

Device	Uncompressed File Sizes (bits)	Smallest Usable Parallel Flash PROM	Minimum Required Address Lines
XC3S100E	581,344	1 Mbit	A[16:0]
XC3S250E	1,352,192	2 Mbit	A[17:0]
XC3S500E	2,267,136	4 Mbit	A[18:0]
XC3S1200E	3,832,320	4 Mbit	A[18:0]
XC3S1600E	5,957,760	8 Mbit	A[19:0]

CCLK Frequency

In BPI mode, the FPGA's internal oscillator generates the configuration clock frequency that controls all the interface timing. The FPGA starts configuration at its lowest frequency and increases its frequency for the remainder of the configuration process if so specified in the configuration bitstream. The maximum frequency is specified using the ***ConfigRate*** bitstream generator option. Table 53 shows the maximum ***ConfigRate*** settings, approximately equal to MHz, for various PROM read access times. Despite using slower ***ConfigRate*** settings, BPI mode is equally fast as the other configuration modes. In BPI mode, data is accessed

at the ***ConfigRate*** frequency and internally serialized with an 8X clock frequency.

Table 53: **Maximum ConfigRate Settings for Parallel Flash PROMs**

Flash Read Access Time	Maximum *ConfigRate* Setting
≤ 200 ns	3
≤ 90 ns	6

Using the BPI Interface after Configuration

After the FPGA successfully completes configuration, all of the pins connected to the parallel Flash PROM are available as user I/Os.

If not using the parallel Flash PROM after configuration, drive LDC0 High to disable the PROM's chip-select input. The remainder of the BPI pins then become available to the FPGA application, including all 24 address lines, the eight data lines, and the LDC2, LDC1, and HDC control pins.

Because all the interface pins are user I/Os after configuration, the FPGA application can continue to use the interface pins to communicate with the parallel Flash PROM. Parallel Flash PROMs are available in densities ranging from 1 Mbit up to 128 Mbits and beyond. However, a single Spartan-3E FPGA requires less than 6 Mbits for configuration. If desired, use a larger parallel Flash PROM to contain additional non-volatile application data, such as MicroBlaze processor code, or other user data such as serial numbers, Ethernet MAC IDs, etc. In such an example, the FPGA configures from parallel Flash PROM. Then using FPGA logic after configuration, a MicroBlaze processor embedded within the FPGA can either execute code directly from parallel Flash PROM or copy the code to external DDR SDRAM and execute from DDR SDRAM. Similarly, the FPGA application can store non-volatile application data within the parallel Flash PROM.

The FPGA configuration data is stored starting at either at location 0 or the top of memory (addresses all ones) or at both locations for MultiBoot mode. Store any additional data beginning in other available parallel Flash PROM sectors. Do not mix configuration data and user data in the same sector.

Similarly, the parallel Flash PROM interface can be expanded to additional parallel peripherals.

The address, data, and LDC1 (OE#) and HDC (WE#) control signals are common to all parallel peripherals. Connect the chip-select input on each additional peripheral to one of the FPGA user I/O pins. If HSWAP = 0 during configuration, the FPGA holds the chip-select line High via an internal pull-up resistor. If HSWAP = 1, connect the select line to +3.3V via an external 4.7 kΩ pull-up resistor to avoid spurious read or write operations. After configuration, drive the select line Low to select the desired peripheral. Refer to the individual peripheral data sheet for specific interface and communication protocol requirements.

The FPGA optionally supports a 16-bit peripheral interface by driving the LDC2 (BYTE#) control pin High after configuration. See **Precautions Using x8/x16 Flash PROMs** for additional information.

The FPGA provides up to 24 address lines during configuration, addressing up to 128 Mbits (16 Mbytes). If using a larger parallel PROM, connect the upper address lines to FPGA user I/O. During configuration, the upper address lines will be pulled High if HSWAP = 0. Otherwise, use external pull-up or pull-down resistors on these address lines to define their values during configuration.

Precautions Using x8/x16 Flash PROMs

(D) Most low- to mid-density PROMs are byte-wide (x8) only. Many higher-density Flash PROMs support both byte-wide (x8) and halfword-wide (x16) data paths and include a mode input called BYTE# that switches between x8 or x16. During configuration, Spartan-3E FPGAs only support byte-wide data. However, after configuration, the FPGA supports either x8 or x16 modes. In x16 mode, up to eight additional user I/O pins are required for the upper data bits, D[15:8].

Connecting a Spartan-3E FPGA to a x8/x16 Flash PROM is simple, but does require a precaution. Various Flash PROM vendors use slightly different interfaces to support both x8 and x16 modes. Some vendors (Intel, Micron, some STMicroelectronics devices) use a straightforward interface with pin naming that matches the FPGA connections. However, the PROM's A0 pin is wasted in x16 applications and a separate FPGA user-I/O pin is required for the D15 data line. Fortunately, the FPGA A0 pin is still available as a user I/O after configuration, even though it connects to the Flash PROM.

Other vendors (AMD, Atmel, Silicon Storage Technology, some STMicroelectronics devices) use a pin-efficient interface but change the function of one pin, called IO15/A-1, depending if the PROM is in x8 or x16 mode. In x8 mode, BYTE# = 0, this pin is the least-significant address line. The A0 address line selects the halfword location. The A-1 address line selects the byte location. When in x16 mode, BYTE# = 1, the IO15/A-1 pin becomes the most-significant data bit, D15 because byte addressing is not required in this mode. Check to see if the Flash PROM has a pin named "IO15/A-1" or "DQ15/A-1". If so, be careful to connect x8/x16 Flash PROMs correctly, as shown in Table 54. Also, remember that the D[14:8] data connections require FPGA user I/O pins but that the D15 data is already connected for the FPGA's A0 pin.

Table 54: **FPGA Connections to Flash PROM with "IO15/A-1" Pin**

FPGA Pin	Connection to Flash PROM with IO15/A-1 Pin	x8 Flash PROM Interface After FPGA Configuration	x16 Flash PROM Interface After FPGA Configuration
LDC2	BYTE#	Drive LDC2 Low or leave unconnected and tie PROM BYTE# input to GND	Drive LCD2 High
LDC1	OE#	Active-Low Flash PROM output-enable control	Active-Low Flash PROM output-enable control
LDC0	CS#	Active-Low Flash PROM chip-select control	Active-Low Flash PROM chip-select control
HDC	WE#	Flash PROM write-enable control	Flash PROM write-enable control
A[23:1]	A[n:0]	A[n:0]	A[n:0]
A0	IO15/A-1	IO15/A-1 is least-significant address input	IO15/A-1 is most-significant data line, IO15
D[7:0]	IO[7:0]	IO[7:0]	IO[7:0]
User I/O	Upper data lines IO[14:8] not required unless used as x16 Flash interface after configuration	Upper data lines IO[14:8] not required	IO[14:8]

Daisy-Chaining

If the application requires multiple FPGAs with different configurations, then configure the FPGAs using a daisy chain, as shown in Figure 56. Use BPI mode (M[2:0] = <0:1:0> or <0:1:1>) for the FPGA connected to the parallel NOR Flash PROM and Slave Parallel mode (M[2:0] = <1:1:0>) for all other FPGAs in the daisy-chain. After the master FPGA—the FPGA on the left in the diagram—finishes loading its configuration data from the parallel Flash PROM, the master device continues generating addresses to the Flash PROM and asserts its CSO_B output Low, enabling the next FPGA in the daisy-chain. The next FPGA then receives parallel configuration data from the Flash PROM. The master FPGA's CCLK output synchronizes data capture.

The downstream devices in Slave Parallel mode also actively drive their LDC[2:0] and HDC outputs during configuration, although these signal are not used for configuration. These pins are in I/O Bank 1, powered by VCCO_1. Because these pins do not connect elsewhere in the configuration circuit, the voltage on VCCO_1 can be whatever is required by the end application.

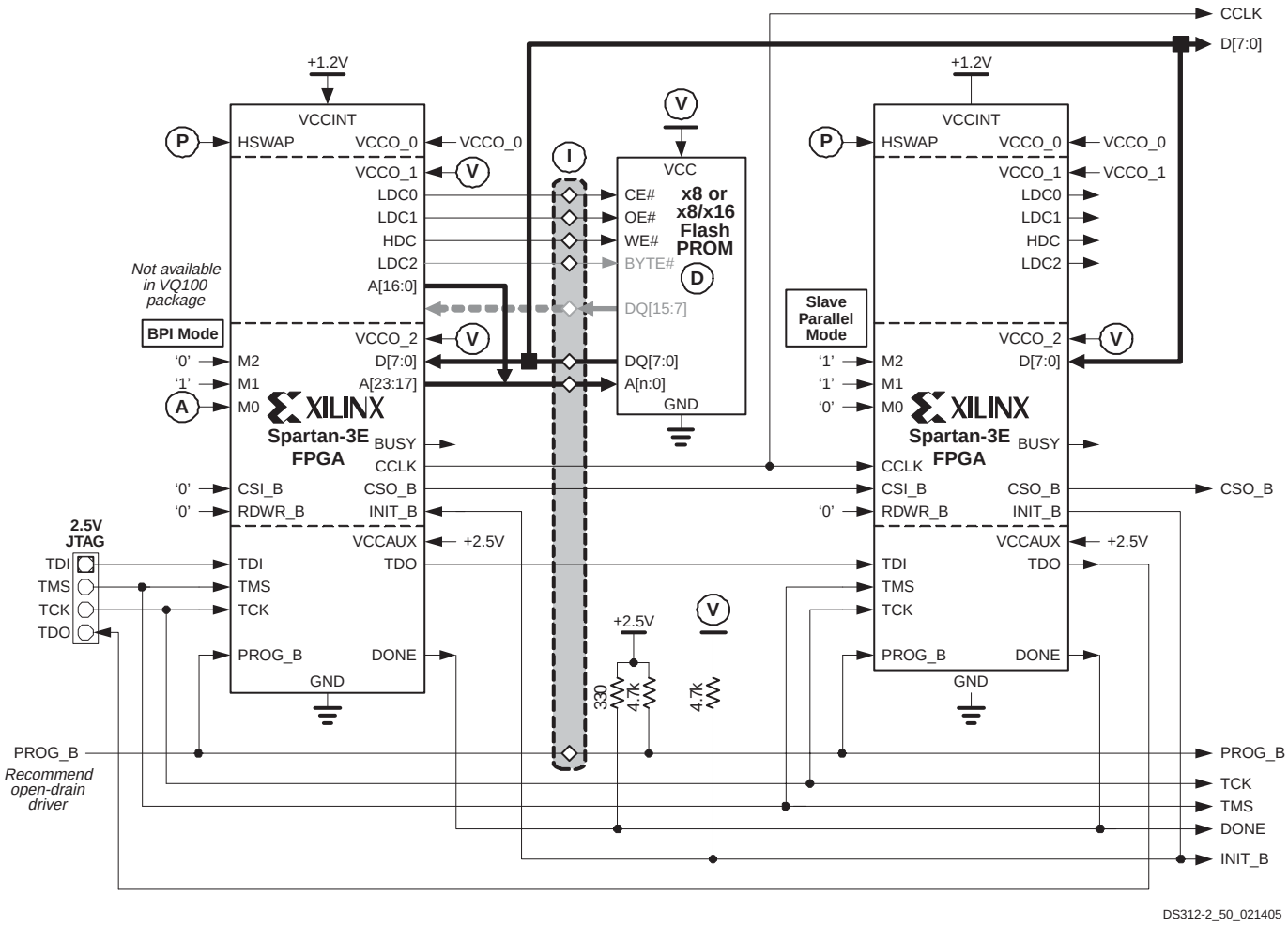


Figure 56: **Daisy-Chaining from BPI Flash Mode**

In-System Programming Support

(I) In a production application, the parallel Flash PROM is usually preprogrammed before it is mounted on the printed circuit board. In-system programming support is available from third-party boundary-scan tool vendors and from some third-party PROM programmers using a socket adapter with attached wires. To gain access to the parallel Flash signals, drive the FPGA's PROG_B input Low with an open-drain driver. This action places all FPGA I/O pins, including those attached to the parallel Flash, in high-impedance (Hi-Z). If the HSWAP input is High, the I/Os have pull-up resistors to the V_{CCO} input on their respective I/O bank. The external programming hardware then has direct access to the parallel Flash pins. The programming access points are highlighted in the gray boxes in Figure 55 and Figure 56.

The FPGA itself can also be used as a parallel Flash PROM programmer during development and test phases. Initially, an FPGA-based programmer is downloaded into the FPGA via JTAG. Then the FPGA performs the Flash PROM programming algorithms and receives programming data from the host via the FPGA's JTAG interface. See Chapter 11 in "Embedded System Tools Reference Manual".

Dynamically Loading Multiple Configuration Images Using MultiBoot Option

After the FPGA configures itself using BPI mode from one end of the parallel Flash PROM, then the FPGA can trigger a MultiBoot event and reconfigure itself from the opposite end of the parallel Flash PROM. MultiBoot is only available when using BPI mode and only for applications with a single Spartan-3E FPGA.

By default, MultiBoot mode is disabled. To trigger a MultiBoot event, assert a Low pulse lasting at least 300 ns on the MultiBoot Trigger (MBT) input to the STARTUP_SPARTAN3E library primitive. Figure 57 shows an example usage. At power up, the FPGA loads itself from the attached parallel Flash PROM. In this example, the M0 mode pin is Low so the FPGA starts at address 0 and increments through the Flash PROM memory locations. After the FPGA completes configuration, the application loaded into the FPGA performs a board-level or system test using FPGA logic. If the test is successful, the FPGA triggers a MultiBoot event, causing the FPGA to reconfigure from the opposite end of the Flash PROM memory. This second configuration contains the FPGA application for normal operation.

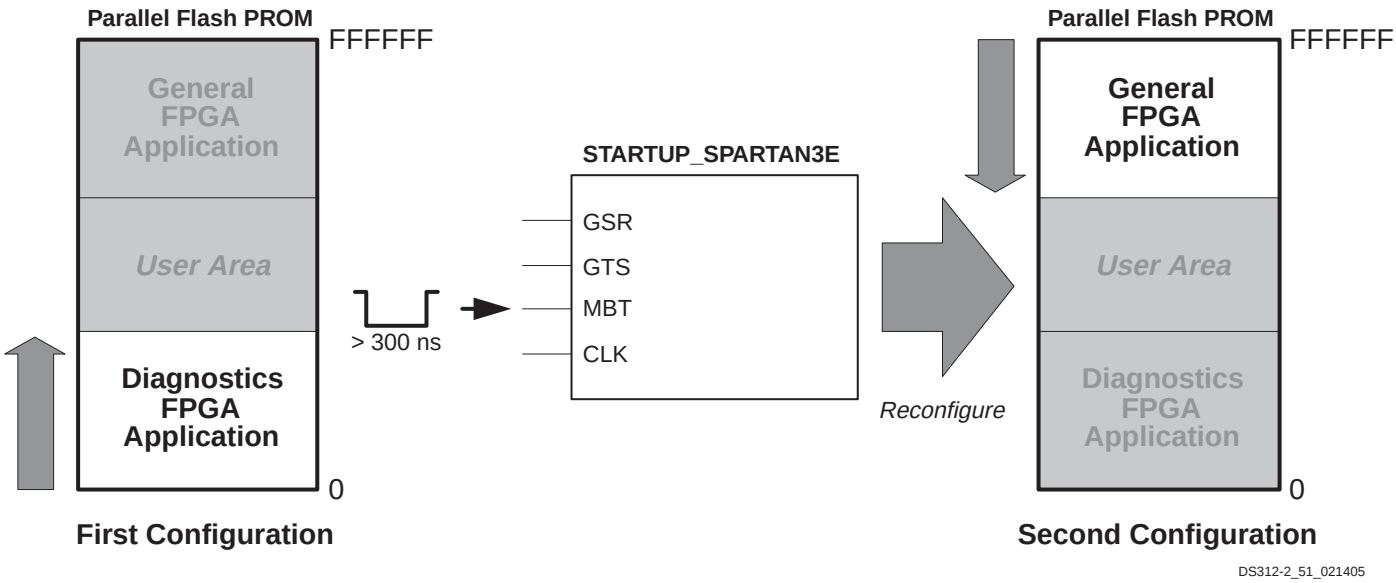


Figure 57: **Use MultiBoot to Load Alternate Configuration Images**

Similarly, the general FPGA application could trigger a MultiBoot event at any time to reload the diagnostics design.

In another potential application, the initial design loaded into the FPGA image contains a “golden” or “fail-safe” configuration image, which then communicates with the outside world and checks for a newer image. If there is a new configuration revision and the new image verifies as good, the “golden” configuration triggers a MultiBoot event to load the new image.

When a MultiBoot event is triggered, the FPGA then again drives its configuration pins as described in Table 51. However, the FPGA does not assert the PROG_B pin. The system design must ensure that no other device drives on these same pins during the reconfiguration process. The FPGA’s DONE, LDC[2:0], or HDC pins can temporarily disable any conflicting drivers during reconfiguration.

Slave Parallel Mode

In Slave Parallel mode (M[2:0] = <1:1:0>), an external host such as a microprocessor or microcontroller writes byte-wide configuration data into the FPGA, using a typical peripheral interface as shown in Figure 58.

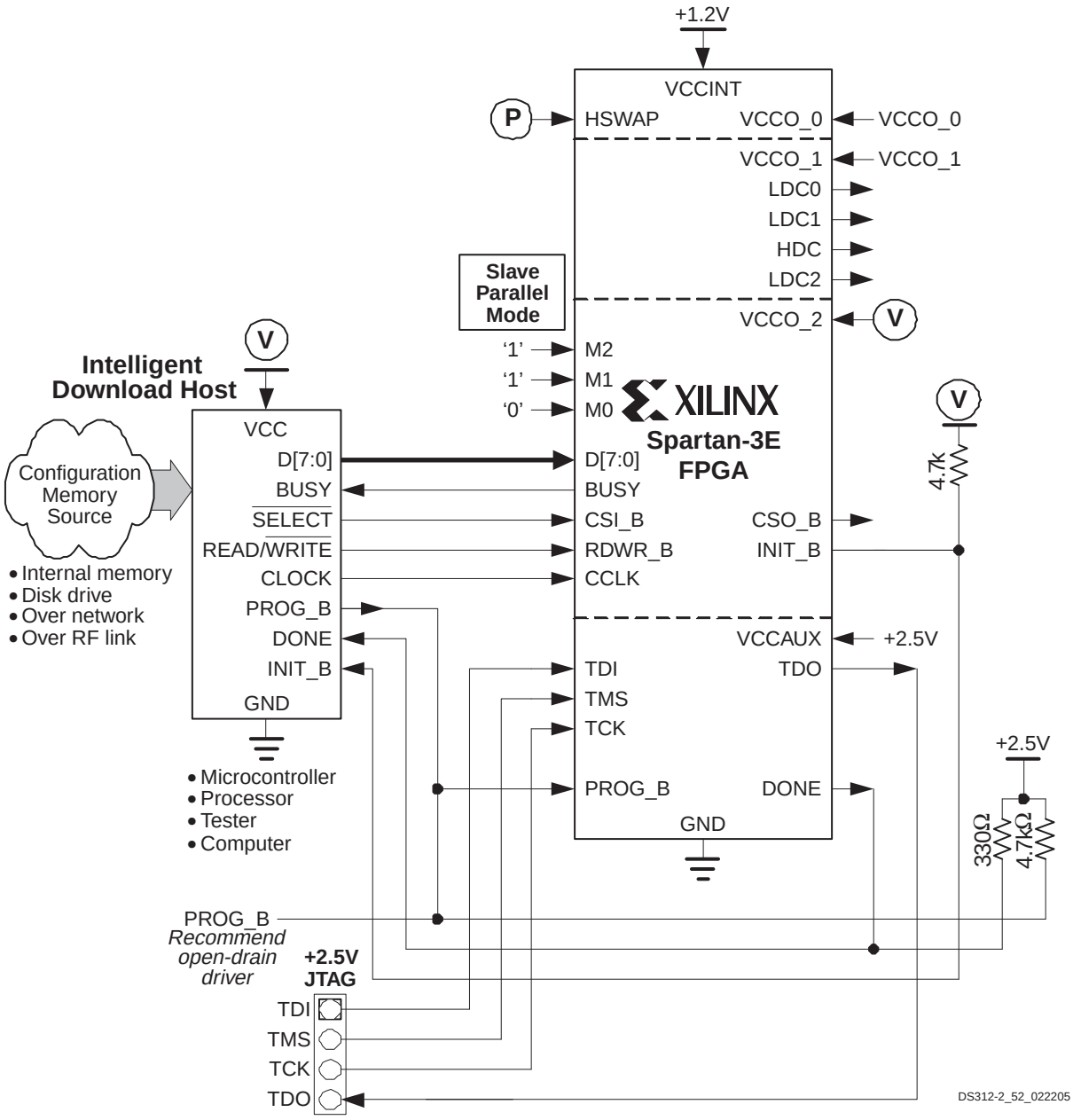


Figure 58: **Slave Parallel Configuration Mode**

The external download host starts the configuration process by pulsing PROG_B and monitoring that the INIT_B pin goes High, indicating that the FPGA is ready to receive its first data. The host asserts the active-Low chip-select signal (CSI_B) and the active-Low Write signal (RDWR_B). The host then continues supplying data and clock signals until either the FPGA's DONE pin goes High, indicating a successful configuration, or until the FPGA's INIT_B pin goes Low, indicating a configuration error.

The FPGA captures data on the rising CCLK edge. If the CCLK frequency exceeds 50 MHz, then the host must also monitor the FPGA's BUSY output. If the FPGA asserts BUSY High, the host must hold the data for an additional clock cycle, until BUSY returns Low. If the CCLK frequency is 50 MHz or below, the BUSY pin may be ignored but actively drives during configuration.

The configuration process requires more clock cycles than indicated from the configuration file size. Additional clocks are required during the FPGA's start-up sequence, especially if the FPGA is programmed to wait for selected Digital Clock Managers (DCMs) to lock to their respective clock inputs (see **Start-Up**, page 91).

If the Slave Parallel interface is only used to configure the FPGA, never to read data back, then the RDWR_B signal can also be eliminated from the interface. However, RDWR_B must remain Low during configuration.

After configuration, all of the interface pins except DONE and PROG_B are available as user I/Os. Alternatively, the bidirectional SelectMAP configuration interface is available after configuration. To continue using SelectMAP mode, set the ***Persist*** bitstream generator option to ***Yes***. The external host can then read and verify configuration data.

The Slave Parallel mode is also used with BPI mode to create multi-FPGA daisy-chains. The lead FPGA is set for BPI mode configuration; all the downstream daisy-chain FPGAs are set for Slave Parallel configuration, as highlighted in Figure 56.

Table 55: **Slave Parallel Mode Connections**

Pin Name	FPGA Direction	Description	During Configuration	After Configuration
HSWAP	Input	**User I/O Pull-Up Control**. When Low during configuration, enables pull-up resistors in all I/O pins to respective I/O bank V_{CCO} input. 0: Pull-ups during configuration 1: No pull-ups	Drive at valid logic level throughout configuration.	User I/O
M[2:0]	Input	**Mode Select**. Selects the FPGA configuration mode.	M2 = 1, M1 = 1, M0 = 0 Sampled when INIT_B goes High.	User I/O
D[7:0]	Input	**Data Input**.	Byte-wide data provided by host. FPGA captures data on rising CCLK edge.	User I/O. If bitstream option ***Persist=Yes***, becomes part of SelectMap parallel peripheral interface.
BUSY	Output	**Busy Indicator**.	If CCLK frequency is < 50 MHz, this pin may be ignored. When High, indicates that the FPGA is not ready to receive additional configuration data. Host must hold data an additional clock cycle.	User I/O. If bitstream option ***Persist=Yes***, becomes part of SelectMap parallel peripheral interface.
CSI_B	Input	**Chip Select Input**. Active Low.	Must be Low throughout configuration.	User I/O. If bitstream option ***Persist=Yes***, becomes part of SelectMap parallel peripheral interface.
RDWR_B	Input	**Read/Write Control**. Active Low write enable.	Must be Low throughout configuration.	User I/O. If bitstream option ***Persist=Yes***, becomes part of SelectMap parallel peripheral interface.
CCLK	Input	**Configuration Clock**. If CCLK PCB trace is long or has multiple connections, terminate this output to maintain signal integrity.	External clock.	User I/O If bitstream option ***Persist=Yes***, becomes part of SelectMap parallel peripheral interface.
LDC[2:0]	Output	**Low During Configuration**.	These pins are not used during configuration. Low throughout configuration.	User I/O
HDC	Output	**High During Configuration**.	This pin is not used during configuration. High throughout configuration.	User I/O

Table 55: **Slave Parallel Mode Connections *(Continued)***

Pin Name	FPGA Direction	Description	During Configuration	After Configuration
CSO_B	Output	**Chip Select Output**. Active Low.	Not used in single FPGA applications. In a daisy-chain configuration, this pin connects to the CSI_B pin of the next FPGA in the chain. Actively drives.	User I/O
INIT_B	Open-drain bidirectional I/O	**Initialization Indicator**. Active Low. Goes Low at start of configuration during Initialization memory clearing process. Released at end of memory clearing, when mode select pins are sampled. In daisy-chain applications, this signal requires an external 4.7 kΩ pull-up resistor to VCCO_2.	Active during configuration. If CRC error detected during configuration, FPGA drives INIT_B Low.	User I/O
DONE	Open-drain bidirectional I/O	**FPGA Configuration Done**. Low during configuration. Goes High when FPGA successfully completes configuration. Requires external 330 Ω pull-up resistor to 2.5V.	Low indicates that the FPGA is not yet configured.	Pulled High via external pull-up. When High, indicates that the FPGA successfully configured.
PROG_B	Input	**Program FPGA**. Active Low. When asserted Low for 300 ns or longer, forces the FPGA to restart its configuration process by clearing configuration memory and resetting the DONE and INIT_B pins once PROG_B returns High. Requires external 4.7 kΩ pull-up resistor to 2.5V. If driving externally, use an open-drain or open-collector driver.	Must be High to allow configuration to start.	Drive PROG_B Low and release to reprogram FPGA.

Voltage Compatibility

Ⓥ Most Slave Parallel interface signals are within the FPGA's I/O Bank 2, supplied by the VCCO_2 supply input. The VCCO_2 voltage can be 1.8V, 2.5V, or 3.3V to match the requirements of the external host, ideally 2.5V. Using 1.8V or 3.3V requires additional design considerations as the DONE and PROG_B pins are powered by the FPGA's 2.5V VCCAUX supply. See application note XAPP453: "The 3.3V Configuration of Spartan-3 FPGAs" for additional information.

The LDC[2:0] and HDC signal are active in I/O Bank 1 but are not used in the interface. Consequently, VCCO_1 can be set the appropriate voltage for the application.

Daisy-Chaining

If the application requires multiple FPGAs with different configurations, then configure the FPGAs using a daisy chain. Use Slave Parallel mode (M[2:0] = <1:1:0>) for all FPGAs in the daisy-chain. The schematic in Figure 59 is optimized for FPGA downloading and does not support the SelectMAP read interface. The FPGA's RDWR_B pin must be Low during configuration.

After the lead FPGA is filled with its configuration data, the lead FPGA enables the next FPGA in the daisy-chain by asserting is chip-select output, CSO_B.

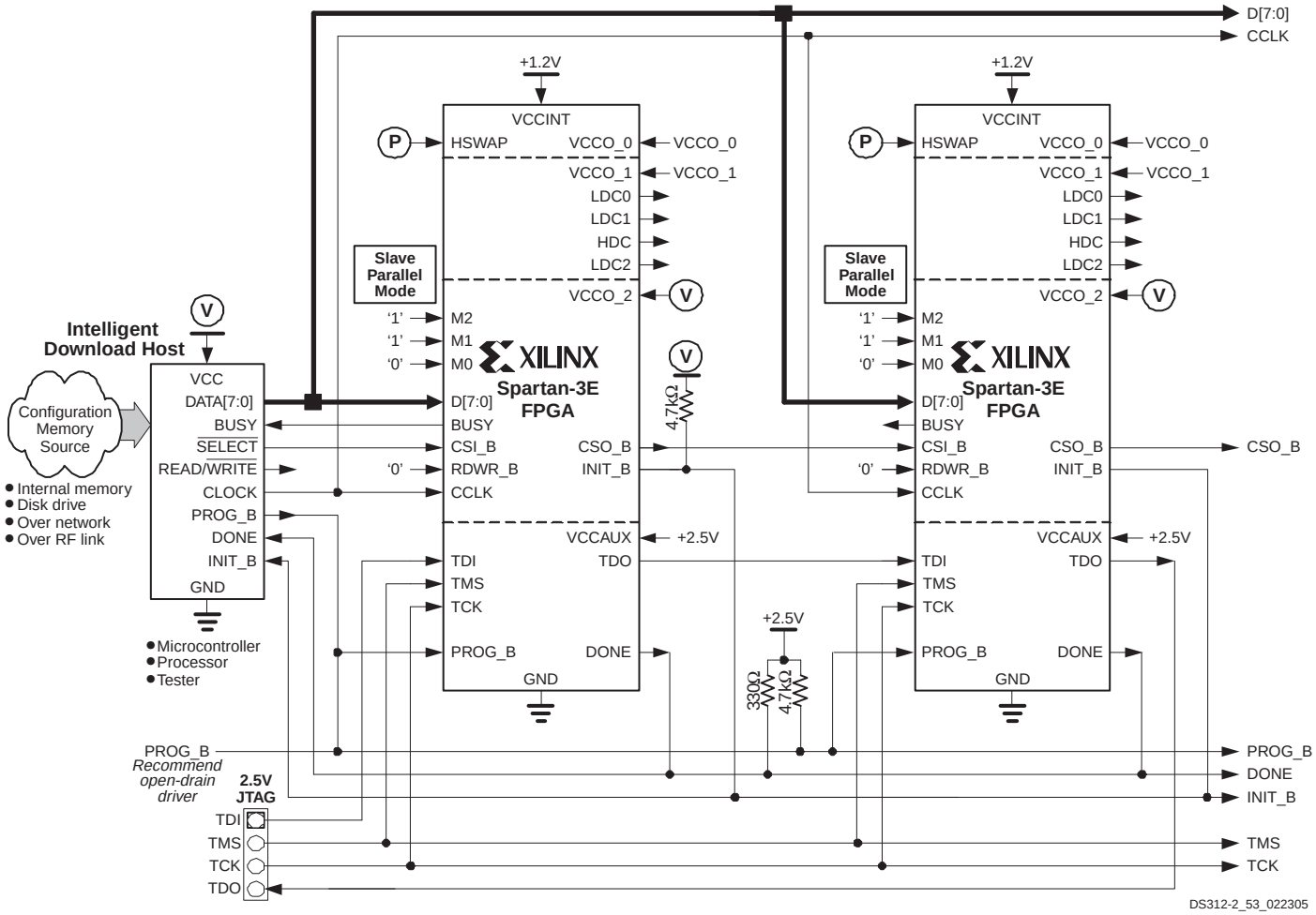


Figure 59: **Daisy-Chaining using Slave Parallel Mode**

Slave Serial Mode

In Slave Serial mode (M[2:0] = <1:1:1>), an external host such as a microprocessor or microcontroller writes serial configuration data into the FPGA, using the synchronous serial interface shown in Figure 60. The serial configuration data is presented on the FPGA's DIN input pin with sufficient setup time before each rising edge of the externally generated CCLK clock input.

The intelligent host starts the configuration process by pulsing PROG_B and monitoring that the INIT_B pin goes High, indicating that the FPGA is ready to receive its first data. The host then continues supplying data and clock signals until either the DONE pin goes High, indicating a successful configuration, or until the INIT_B pin goes Low, indicating a configuration error. The configuration process requires more clock cycles than indicated from the configuration file size. Additional clocks are required during the FPGA's start-up sequence, especially if the FPGA is programmed to wait for selected Digital Clock Managers (DCMs) to lock to their respective clock inputs (see **Start-Up**, page 91).

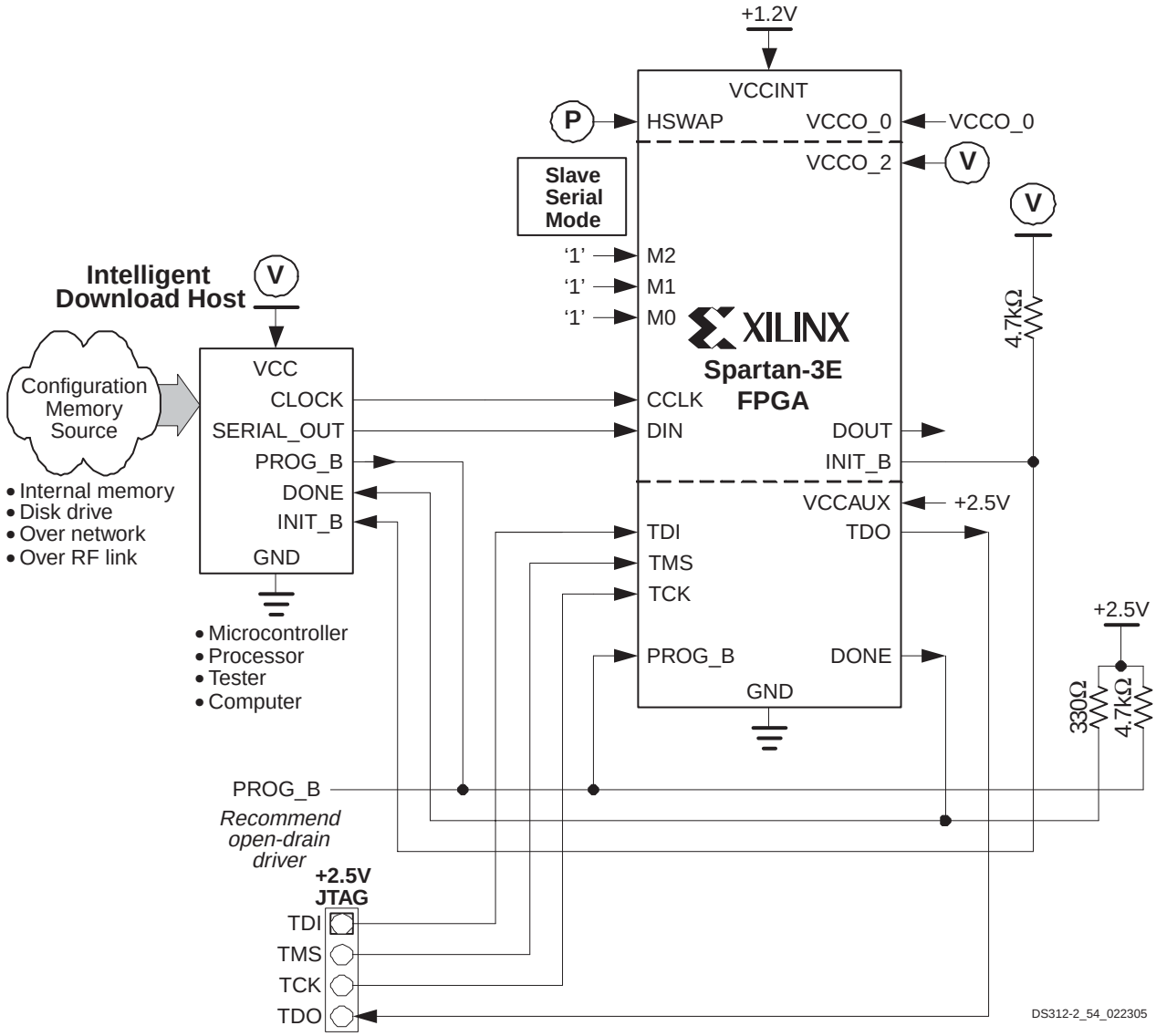


Figure 60: **Slave Serial Configuration**

The mode select pins, M[2:0], are sampled when the FPGA's INIT_B output goes High and must be at defined logic levels during this time. After configuration, when the FPGA's DONE output goes High, the mode pins are available as full-featured user-I/O pins.

(P) Similarly, the FPGA's HSWAP pin must be Low to enable pull-up resistors on all user-I/O pins or High to disable the pull-up resistors. The HSWAP control must remain at a constant logic level throughout FPGA configuration. After configuration, when the FPGA's DONE output goes High, the HSWAP pin is available as full-featured user-I/O pin and is powered by the VCCO_0 supply.

Table 56: **Slave Serial Mode Connections**

Pin Name	FPGA Direction	Description	During Configuration	After Configuration
HSWAP	Input	**User I/O Pull-Up Control**. When Low during configuration, enables pull-up resistors in all I/O pins to respective I/O bank V_{CCO} input. 0: Pull-up during configuration 1: No pull-ups	Drive at valid logic level throughout configuration.	User I/O
M[2:0]	Input	**Mode Select**. Selects the FPGA configuration mode.	M2 = 1, M1 = 1, M0 = 1 Sampled when INIT_B goes High.	User I/O
DIN	Input	**Data Input**.	Serial data provided by host. FPGA captures data on rising CCLK edge.	User I/O
CCLK	Input	**Configuration Clock**. If CCLK PCB trace is long or has multiple connections, terminate this output to maintain signal integrity.	External clock.	User I/O
INIT_B	Open-drain bidirectional I/O	**Initialization Indicator**. Active Low. Goes Low at start of configuration during Initialization memory clearing process. Released at end of memory clearing, when mode select pins are sampled. In daisy-chain applications, this signal requires an external 4.7 kΩ pull-up resistor to VCCO_2.	Active during configuration. If CRC error detected during configuration, FPGA drives INIT_B Low.	User I/O
DONE	Open-drain bidirectional I/O	**FPGA Configuration Done**. Low during configuration. Goes High when FPGA successfully completes configuration. Requires external 330 Ω pull-up resistor to 2.5V.	Low indicates that the FPGA is not yet configured.	Pulled High via external pull-up. When High, indicates that the FPGA successfully configured.
PROG_B	Input	**Program FPGA**. Active Low. When asserted Low for 300 ns or longer, forces the FPGA to restart its configuration process by clearing configuration memory and resetting the DONE and INIT_B pins once PROG_B returns High. Requires external 4.7 kΩ pull-up resistor to 2.5V. If driving externally, use an open-drain or open-collector driver.	Must be High to allow configuration to start.	Drive PROG_B Low and release to reprogram FPGA.

Voltage Compatibility

Ⓥ Most Slave Serial interface signals are within the FPGA's I/O Bank 2, supplied by the VCCO_2 supply input. The VCCO_2 voltage can be 3.3V, 2.5V, or 1.8V to match the requirements of the external host, ideally 2.5V. Using 3.3V or 1.8V requires additional design considerations as the DONE and PROG_B pins are powered by the FPGA's 2.5V VCCAUX supply. See application note XAPP453: "The 3.3V Configuration of Spartan-3 FPGAs" for additional information.

Daisy-Chaining

If the application requires multiple FPGAs with different configurations, then configure the FPGAs using a daisy chain, as shown in Figure 61. Use Slave Serial mode (M[2:0] = <1:1:1>) for all FPGAs in the daisy-chain. After the lead FPGA is filled with its configuration data, the lead FPGA passes configuration data via its DOUT output pin to the next FPGA on the falling CCLK edge.

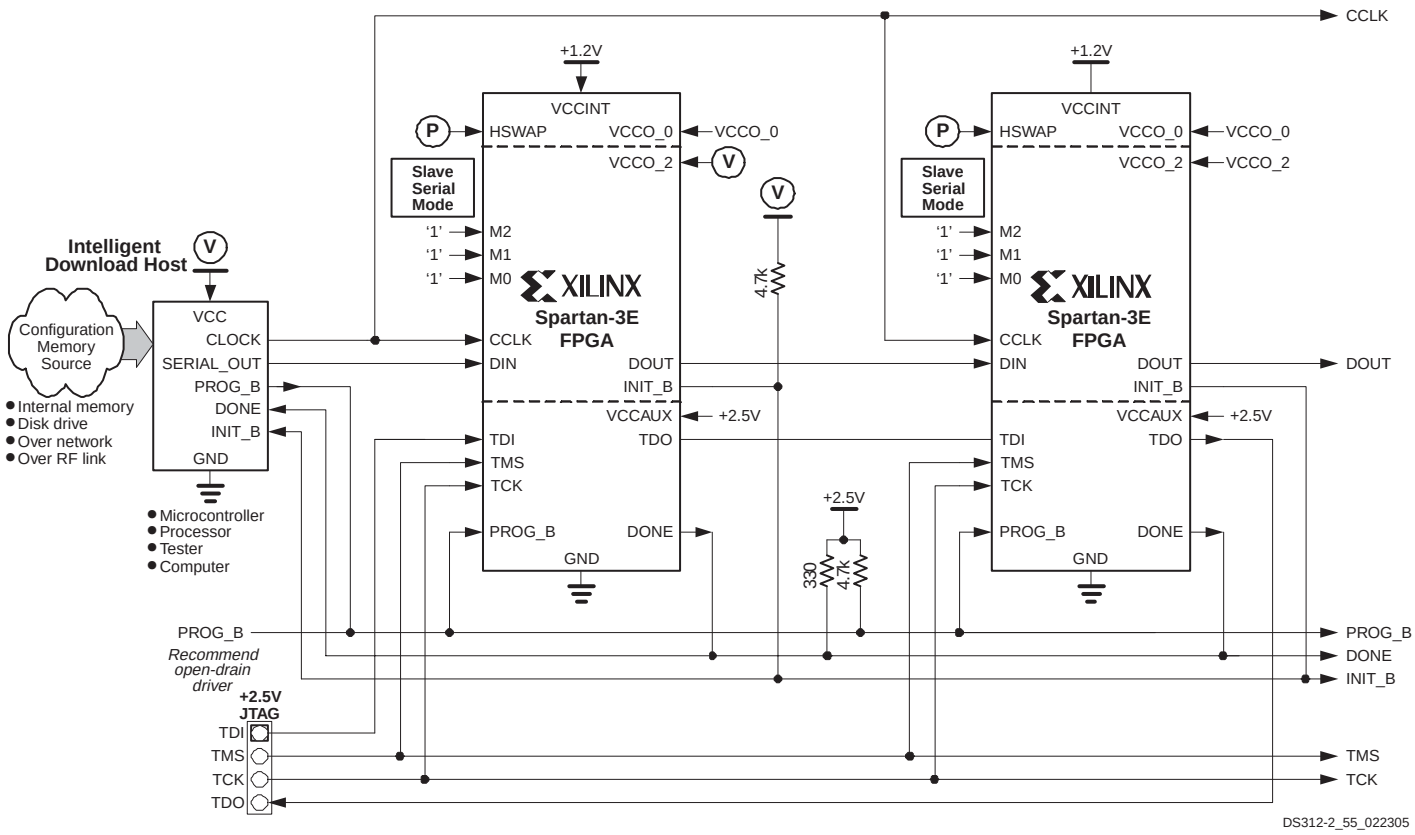


Figure 61: **Daisy-Chaining using Slave Serial Mode**

JTAG Mode

The Spartan-3E FPGA has a dedicated four-wire IEEE 1149.1/1532 JTAG port that is always available any time the FPGA is powered and regardless of the mode pin settings. However, when the FPGA mode pins are set for JTAG mode (M[2:0] = <1:0:1>), the FPGA waits to be configured via the JTAG port after a power-on event or when PROG_B is asserted. Selecting the JTAG mode simply disables the other configuration modes. No other pins are required as part of the configuration interface.

Figure 62 illustrates a JTAG-only configuration interface. The JTAG interface is easily cascaded to any number of FPGAs by connecting the TDO output of one device to the TDI input of the next device in the chain. The TDO output of the last device in the chain loops back to the port connector.

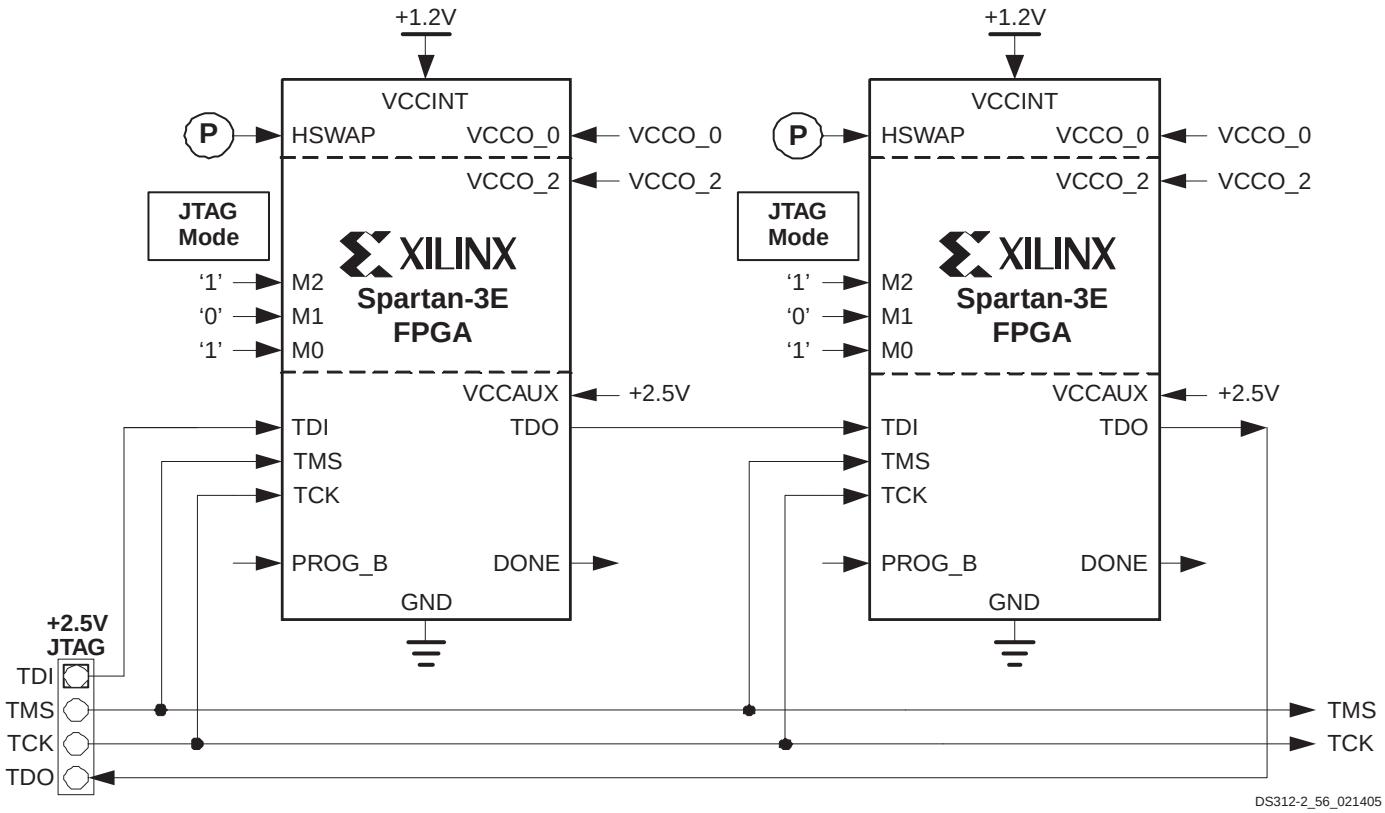


Figure 62: **JTAG Configuration Mode**

Voltage Compatibility

The 2.5V VCCAUX supply powers the JTAG interface. All of the user I/Os are separately powered by their respective VCCO_# supplies.

When connecting the Spartan-3E JTAG port to a 3.3V interface, the JTAG input pins must be current-limited to 10 mA or less using series resistors. Similarly, the TDO pin is a CMOS output powered from +2.5V. The TDO output can directly drive a 3.3V input but with reduced noise immunity. See application note XAPP453: "The 3.3V Configuration of Spartan-3 FPGAs" for additional information.

Maximum Bitstream Size for Daisy-Chains

The maximum bitstream length supported by Spartan-3E FPGAs in serial daisy-chains is 4,294,967,264 bits (4 Gbits), roughly equivalent to a daisy-chain with 720 XC3S1600E FPGAs. This is a limit only for serial daisy-chains where configuration data is passed via the FPGA's DOUT pin. There is no such limit for JTAG chains.

Configuration Sequence

The Spartan-3E configuration process is three-stage process that begins after the FPGA powers on (a POR event) or after the PROG_B input is asserted. Power-On Reset (POR) occurs after the V_{CCINT}, V_{CCAUX}, and the V_{CCO} Bank 2 supplies reach their respective input threshold levels. After either a POR or PROG_B event, the three-stage configuration process begins.

1. The FPGA clears (initializes) the internal configuration memory.
2. Configuration data is loaded into the internal memory.
3. The user-application is activated by a start-up process.

Figure 63 is a generalized block diagram of the Spartan-3E configuration logic, showing the interaction of different device inputs and Bitstream Generator (BitGen) options. A flow diagram for the configuration sequence of the Serial and Parallel modes appears in Figure 64. Figure 65 shows the Boundary-Scan or JTAG configuration sequence.

Initialization

Configuration automatically begins after power-on or after asserting the FPGA PROG_B pin, unless delayed using the FPGA's INIT_B pin. The FPGA holds the open-drain INIT_B signal Low while it clears its internal configuration memory. Externally holding the INIT_B pin Low forces the configuration sequencer to wait until INIT_B again goes High.

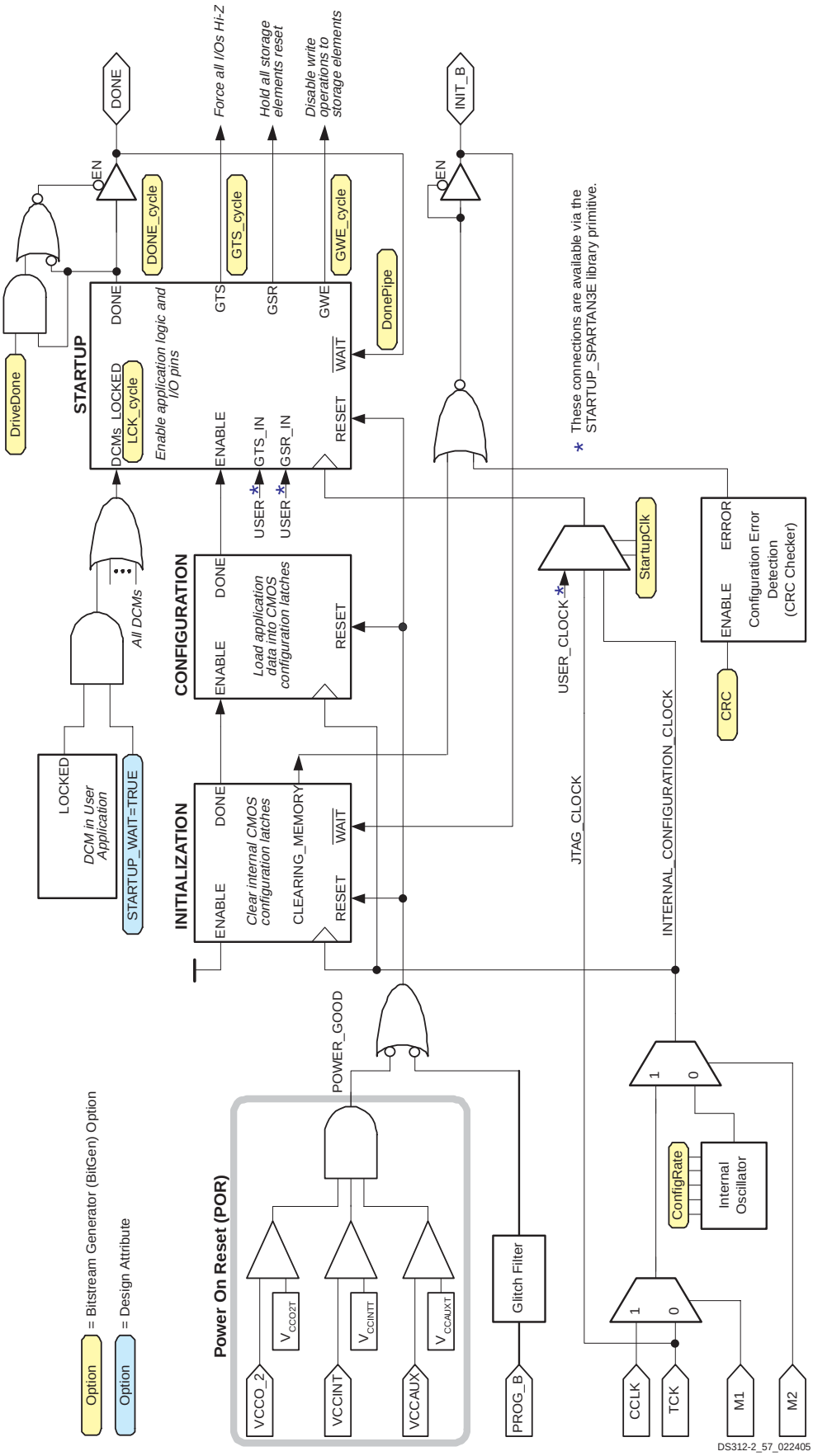


Figure 63: **Generalized Spartan-3E FPGA Configuration Logic Block Diagram**

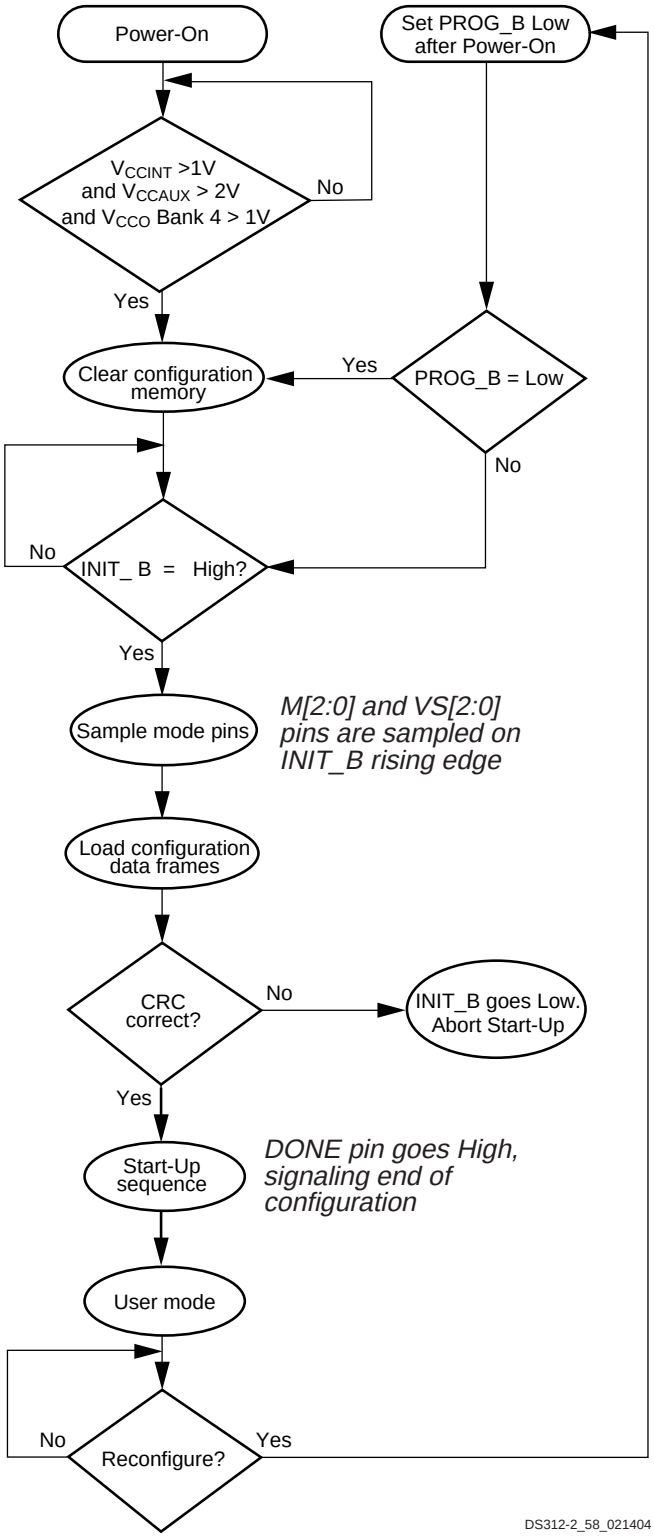


Figure 64: **General Configuration Process**

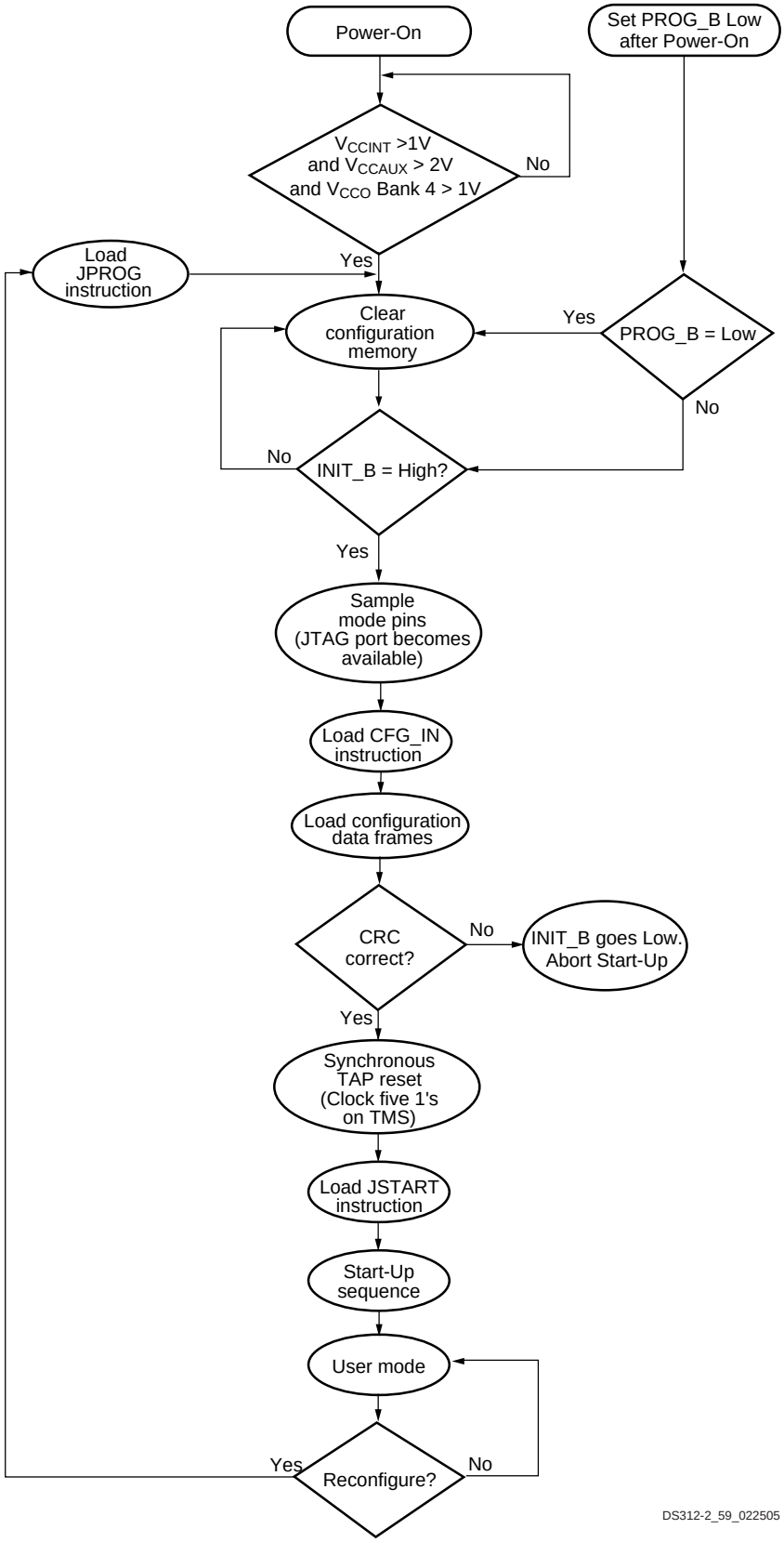


Figure 65: **Boundary-Scan Configuration Flow Diagram**

The FPGA signals when the memory-clearing phase is complete by releasing the open-drain INIT_B pin, allowing the pin to go High via the external pull-up resistor to VCCO_2.

Loading Configuration Data

Configuration data is then written to the FPGA's internal memory. The FPGA holds the Global Set/Reset (GSR) signal active throughout configuration, holding all FPGA flip-flops in a reset state. The FPGA signals when the entire configuration process completes be releasing the DONE pin, allowing it to go High.

The FPGA configuration sequence can also be initiated by asserting the PROG_B. Once release, the FPGA begins clearing its internal configuration memory, and progresses through the remainder of the configuration process.

Start-Up

At the end of configuration, the Global Set/Reset (GSR) signal is pulsed, placing all flip-flops in a known state. After configuration completes, the FPGA switches over to the user application loaded into the FPGA. The sequence and timing of how the FPGA switches over is programmable as is the clock source controlling the sequence.

The default start-up sequence appears in Figure 66, where the Global Three-State signal (GTS) is released one clock cycle after DONE goes High. This sequence allows the DONE signal to enable or disable any external logic used during configuration before the user application in the FPGA starts driving output signals. One clock cycle later, the Global Write Enable (GWE) signal is released. This allows signals to propagate within the FPGA before any clocked storage elements such as flip-flops and block ROM are enabled.

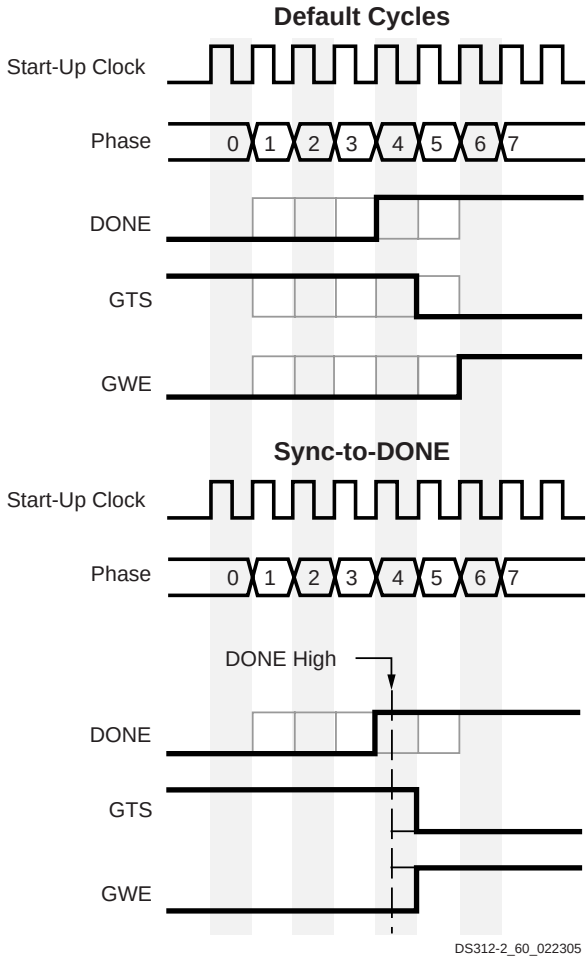


Figure 66: **Default Start-Up Sequence**

The relative timing of configuration events is programmed via the Bitstream Generator (BitGen) options in the Xilinx development software. For example, the GTS and GWE events can be programmed to wait for all the DONE pins to High on all the devices in a multiple-FPGA daisy-chain, forcing the FPGAs to start synchronously. Similarly, the start-up sequence can be paused at any stage, waiting for selected DCMs to lock to their respective input clock signals. See also **Stabilizing DCM Clocks Before User Mode**, page 48.

The start-up sequence can by synchronized to a clock within the FPGA application using the STARTUP_SPARTAN3E library primitive and by setting the ***StartupClk*** bitstream generator option. The FPGA application can optionally assert the Global Set/Reset (GSR) and Global Three-State signal (GTS) signals via the STARTUP_SPARTAN3E primitive.

Readback

Using Slave Parallel mode, configuration data from the FPGA can be read back. Readback is supported only in the Slave Parallel and JTAG modes.

Along with the configuration data, it is possible to read back the contents of all registers, distributed RAM, and block RAM resources. This capability is used for real-time debugging.

To synchronously control when registers values are captured for readback, using the CAPTURE_SPARTAN3 library primitive, which applies for both Spartan-3 and Spartan-3E FPGA families.

Bitstream Generator (BitGen) Options

Various Spartan-3E FPGA functions are controlled by specific bits in the configuration bitstream image. These values are specified when creating the bitstream image with the Bitstream Generator (BitGen) software.

Table 57 provides a list of all BitGen options for Spartan-3E FPGAs.

Table 57: **Spartan-3E FPGA Bitstream Generator (BitGen) Options**

Option Name	Pins/Function Affected	Values (***default***)	Description
ConfigRate	CCLK, Configuration	3, ***6***, 12, 25	Sets the approximate frequency, in MHz, of the internal oscillator using for Master Serial, SPI, and BPI configuration modes. The internal oscillator powers up at its lowest frequency and the new setting is loaded as part of the configuration bitstream. The software default value is 6 (~6 MHz).
StartupClk	Configuration, Startup	***Cclk***	***Default.*** The CCLK signal (internally or externally generated) controls the startup sequence when the FPGA transitions from configuration mode to the user mode. See **Start-Up**, page 91.
		UserClk	A clock signal from within the FPGA application controls the startup sequence when the FPGA transitions from configuration mode to the user mode. See **Start-Up**, page 91. The FPGA application supplies the user clock on the CLK pin on the STARTUP_SPARTAN3E primitive.
		Jtag	The JTAG TCK input controls the startup sequence when the FPGA transitions from configuration mode to the user mode. See **Start-Up**, page 91.
UnusedPin	Unused I/O Pins	***Pulldown***	***Default.*** All unused I/O pins have a pull-down resistor to GND.
		Pullup	All unused I/O pins have a pull-up resistor to the VCCO_# supply for its associated I/O bank.
		Pullnone	All unused I/O pins are left floating (Hi-Z, high-impedance, three-state). Use external pull-up or pull-down resistors or logic to apply a valid signal level.
DONE_cycle	DONE pin, Configuration Startup	1, 2, 3, ***4***, 5, 6	Selects the Configuration Startup phase that activates the FPGA's DONE pin. See **Start-Up**, page 91.

Table 57: **Spartan-3E FPGA Bitstream Generator (BitGen) Options *(Continued)***

Option Name	Pins/Function Affected	Values (***default***)	Description
GWE_cycle	All flip-flops, LUT RAMs, and SRL16 shift registers, Block RAM, Configuration Startup	1, 2, 3, 4, 5, ***6***	Selects the Configuration Startup phase that asserts the internal write-enable signal to all flip-flops, LUT RAMs and shift registers (SRL16). It also enables block RAM read and write operations. See **Start-Up**, page 91.
		Done	Waits for the DONE pin input to go High before asserting the internal write-enable signal to all flip-flops, LUT RAMs and shift registers (SRL16). Block RAM read and write operations are enabled at this time.
		Keep	Retains the current GWE_cycle setting for partial reconfiguration applications.
GTS_cycle	All I/O pins, Configuration	1, 2, 3, 4, ***5***, 6	Selects the Configuration Startup phase that releases the internal three-state control, holding all I/O buffers in high-impedance (Hi-Z). Output buffers actively drive, if so configured, after this point. See **Start-Up**, page 91.
		Done	Waits for the DONE pin input to go High before releasing the internal three-state control, holding all I/O buffers in high-impedance (Hi-Z). Output buffers actively drive, if so configured, after this point.
		Keep	Retains the current GTS_cycle setting for partial reconfiguration applications.
LCK_cycle	DCMs, Configuration Startup	***NoWait***	The FPGA does not wait for selected DCMs to lock before completing configuration.
		0, 1, 2, 3, 4, 5, 6	If one or more DCMs in the design have the STARTUP_WAIT attribute set to TRUE, the FPGA waits for such DCMs to acquire their respective input clock and assert their LOCKED output. This setting selects the Configuration Startup phase where the FPGA waits for the DCMs to lock.
DonePin	DONE pin	***Pullup***	Internally connects a pull-up resistor between DONE pin and VCCAUX. An external 330 Ω pull-up resistor to VCCAUX is still recommended.
		Pullnone	No internal pull-up resistor on DONE pin. An external 330 Ω pull-up resistor to VCCAUX is required.
DriveDone	DONE pin	***No***	When configuration completes, the DONE pin stops driving Low and relies on an external 330 Ω pull-up resistor to VCCAUX for a valid logic High.
		Yes	When configuration completes, the DONE pin actively drives High. When using this option, an external pull-up resistor is no longer required. Only one device in an FPGA daisy-chain should use this setting.
DonePipe	DONE pin	***No***	The input path from DONE pin input back to the Startup sequencer is not pipelined.
		Yes	This option adds a pipeline register stage between the DONE pin input and the Startup sequencer. Used for high-speed daisy-chain configurations when DONE cannot rise in a single CCLK cycle. Releases GWE and GTS signals on the first rising edge of StartupClk after the DONE pin input goes High.
ProgPin	PROG_B pin	***Pullup***	Internally connects a pull-up resistor or between PROG_B pin and VCCAUX. An external 4.7 kΩ pull-up resistor to VCCAUX is still recommended.
		Pullnone	No internal pull-up resistor on PROG_B pin. An external 4.7 kΩ pull-up resistor to VCCAUX is required.
TckPin	JTAG TCK pin	***Pullup***	Internally connects a pull-up resistor between JTAG TCK pin and VCCAUX.
		Pulldown	Internally connects a pull-down resistor between JTAG TCK pin and GND.
		Pullnone	No internal pull-up resistor on JTAG TCK pin.
TdiPin	JTAG TDI pin	***Pullup***	Internally connects a pull-up resistor between JTAG TDI pin and VCCAUX.
		Pulldown	Internally connects a pull-down resistor between JTAG TDI pin and GND.
		Pullnone	No internal pull-up resistor on JTAG TDI pin.

Table 57: **Spartan-3E FPGA Bitstream Generator (BitGen) Options *(Continued)***

Option Name	Pins/Function Affected	Values (***default***)	Description
TdoPin	JTAG TDO pin	***Pullup***	Internally connects a pull-up resistor between JTAG TDO pin and VCCAUX.
		Pulldown	Internally connects a pull-down resistor between JTAG TDO pin and GND.
		Pullnone	No internal pull-up resistor on JTAG TDO pin.
TmsPin	JTAG TMS pin	***Pullup***	Internally connects a pull-up resistor between JTAG TMS pin and VCCAUX.
		Pulldown	Internally connects a pull-down resistor between JTAG TMS pin and GND.
		Pullnone	No internal pull-up resistor on JTAG TMS pin.
UserID	JTAG User ID register	User string	The 32-bit JTAG User ID register value is loaded during configuration. The default value is all ones, 0xFFFF_FFFF hexadecimal. To specify another value, enter an 8-character hexadecimal value.
Security	JTAG, SelectMAP, Readback, Partial reconfiguration	***None***	Readback and partial reconfiguration are available via the JTAG port or via the SelectMAP interface, if the Persist option is set to Yes.
		Level1	Readback function is disabled. Partial reconfiguration is still available via the JTAG port or via the SelectMAP interface, if the Persist option is set to Yes.
		Level	Readback function is disabled. Partial reconfiguration is disabled.
CRC	Configuration	**Enable**	***Default.*** Enable CRC checking on the FPGA bitstream. If error detected, FPGA asserts INIT_B Low and DONE pin stays Low.
		Disable	Turn off CRC checking.
Persist	SelectMAP interface pins, BPI mode, Slave mode, Configuration	***No***	All BPI and Slave mode configuration pins are available as user-I/O after configuration.
		Yes	This option is required for Readback and partial reconfiguration using the SelectMAP interface. The SelectMAP interface pins (see **Slave Parallel Mode**, page 79) are reserved after configuration and are not available as user-I/O.

Powering Spartan-3E FPGAs

Voltage Supplies

Like Spartan-3 FPGAs, Spartan-3E FPGAs have multiple voltage supply inputs, as shown in Table 58. There are two supply inputs for internal logic functions, V_{CCINT} and V_{CCAUX}. Each of the four I/O banks has a separate V_{CCO} supply input that powers the output buffers within the associated I/O bank. All of the V_{CCO} connections to a specific I/O bank must be connected and must connect to the same voltage.

Table 58: **Spartan-3E Voltage Supplies**

Supply Input	Description	Nominal Supply Voltage
VCCINT	Internal core supply voltage. Supplies all internal logic functions such as CLBs, block RAM, multipliers, etc. Input to Power-On Reset (POR) circuit.	1.2V
VCCAUX	Auxiliary supply voltage. Supplies Digital Clock Managers (DCMs), differential drivers, dedicated configuration pins, JTAG interface. Input to Power-On Reset (POR) circuit.	2.5V
VCCO_0	Supplies the output buffers in I/O Bank 0, the bank along the top edge of the FPGA.	Selectable, 3.3V, 3.0V, 2.5V, 1.8, 1.5V, or 1.2V.
VCCO_1	Supplies the output buffers in I/O Bank 1, the bank along the right edge of the FPGA. In **Byte-Wide Peripheral Interface (BPI) Parallel Flash Mode**, connects to the save voltage as the Flash PROM.	Selectable, 3.3V, 3.0V, 2.5V, 1.8, 1.5V, or 1.2V.
VCCO_2	Supplies the output buffers in I/O Bank 2 the bank along the bottom edge of the FPGA. Connects to the same voltage as the FPGA configuration source. Input to Power-On Reset (POR) circuit.	Selectable, 3.3V, 3.0V, 2.5V, 1.8, 1.5V, or 1.2V.
VCCO_3	Supplies the output buffers in I/O Bank 0, the bank along the top edge of the FPGA.	Selectable, 3.3V, 3.0V, 2.5V, 1.8, 1.5V, or 1.2V.

In a 3.3V-only application, all four V_{CCO} supplies connect to 3.3V. However, Spartan-3E FPGAs provide the ability to bridge between different I/O voltages and standards by applying different voltages to the V_{CCO} inputs of different banks. Refer to **I/O Banking Rules** for which I/O standards can be intermixed within a single I/O bank.

Each I/O bank also has an separate, optional input voltage reference supply, called VREF. If the I/O bank includes an I/O standard that requires a voltage reference such as HSTL or SSTL, then all VREF pins within the I/O bank must be connected to the same voltage.

Voltage Regulators

Various power supply manufacturers offer complete power solutions for Xilinx FPGAs including some with integrated three-rail regulators specifically designed for Spartan-3 and Spartan-3E FPGAs. The Xilinx Power Corner web site provides links to vendor solution guides and Xilinx power estimation and analysis tools.

Power Distribution System (PDS) Design and Decoupling/Bypass Capacitors

Good power distribution system (PDS) design is important for all FPGA designs, but especially so for high performance applications, greater than 100 MHz. Proper design results in better overall performance, lower clock and DCM jitter, and a generally more robust system. Before designing the printed circuit board (PCB) for the FPGA design, please review XAPP623: "Power Distribution System (PDS) Design: Using Bypass/Decoupling Capacitors".

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/01/05	1.0	Initial Xilinx release.

The Spartan-3E Family Data Sheet

DS312-1, *Spartan-3E FPGA Family: Introduction and Ordering Information* (Module 1)

DS312-2, *Spartan-3E FPGA Family: Functional Description* (Module 2)

DS312-3, *Spartan-3E FPGA Family: DC and Switching Characteristics* (Module 3)

DS312-4, *Spartan-3E FPGA Family: Pinout Descriptions* (Module 4)