IOBs

IOB Overview

The Input/Output Block (IOB) provides a programmable, bidirectional interface between an I/O pin and the FPGA's internal logic.

A simplified diagram of the IOB's internal structure appears in Figure 1. 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 the Storage Element Functions section. 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 all lead to the FPGA's internal logic. The delay element can be set to ensure a hold time of zero.
- 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.

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).

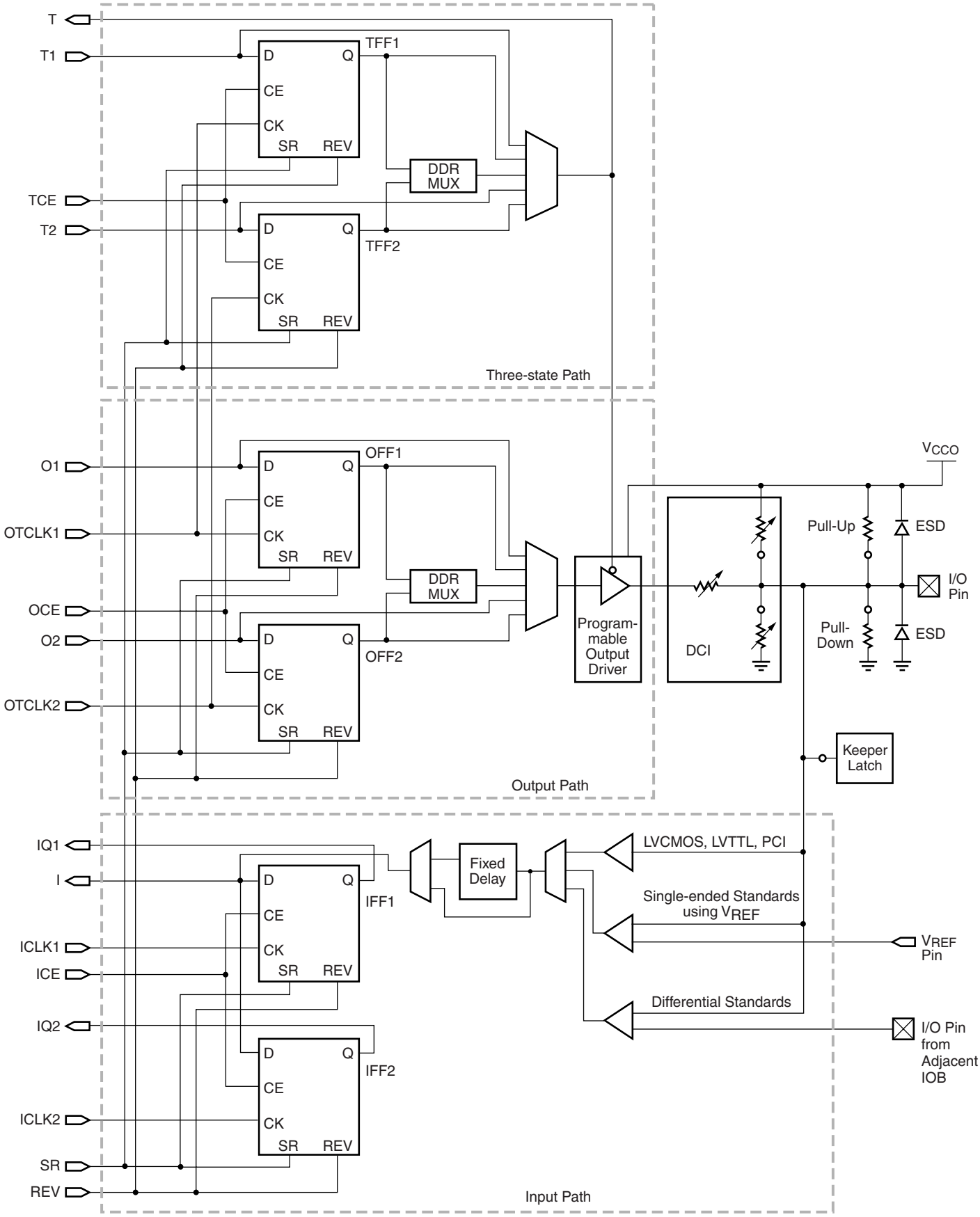
The storage-element-pair on either the Output path or the Three-State path can be used together with a special multiplexer to produce Double-Data-Rate (DDR) transmission. This is accomplished by taking data synchronized to the clock signal's rising edge and converting them to bits synchronized on both 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 (FDDR).

See **Double-Data-Rate Transmission**, page 3 for more information.

The signal paths associated with the storage element are described in Table 1.

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 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 will mirror the data at D.
CK	Clock input	A signal's active edge on this input with CE asserted, loads data into the storage element.
CE	Clock Enable input	When asserted, this input enables CK. If not connected, CE defaults to the asserted state.
SR	Set/Reset	Forces 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.
REV	Reverse	Used together with SR. Forces storage element into the state opposite from what SR does.



Note: All IOB signals communicating with the FPGA's internal logic have the option of inverting polarity.

DS099-2_01_082104

Figure 1: **Simplified IOB Diagram**

According to Figure 1, the clock line OTCLK1 connects the CK inputs of the upper registers on the output and three-state paths. Similarly, OTCLK2 connects 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 enable line OCE connects the CE inputs of the upper and lower registers on the output path. Similarly, TCE connects 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 is common to all six registers, as is the Reverse (REV) line.

Each storage element supports numerous options in addition to the control over signal polarity described in the IOB Overview section. These are described in Table 2.

Table 2: **Storage Element Options**

Option Switch	Function	Specificity
FF/Latch	Chooses between an edge-sensitive flip-flop or a level-sensitive latch	Independent for each storage element.
SYNC/ASYNC	Determines whether SR is synchronous or asynchronous	Independent for each storage element.
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 FDDR. In the latter case, the selection for the upper element (OFF1 or TFF2) will apply to both elements.
INIT1/INIT0	In the event of a Global Set/Reset, after configuration or upon activation of the GTS net, this switch decides whether to set or reset a storage element. By default, choosing SRLOW also selects INIT0; choosing SRHIGH also selects INIT1.	Independent for each storage element, except when using FDDR. In the latter case, selecting INIT0 for one element applies to both elements (even though INIT1 is selected for the other).

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-3 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 (FDDR). This primitive permits DDR transmission where output data bits are synchronized to both the rising and falling edges of a clock. It is possible to access this function by placing either an FDDRRSE or an FDDRCPE component or symbol into the design. DDR operation requires two clock signals (50% duty cycle), one the inverted form of the other. These signals trigger the two registers in alternating fashion, as shown in Figure 2. Commonly, the Digital Clock Manager (DCM) generates the two clock signals by mirroring an incoming signal, then shifting it 180 degrees. This approach ensures minimal skew between the two signals.

The storage-element-pair on the Three-State path (TFF1 and TFF2) can also be combined with a local multiplexer to form an FDDR 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. In this way, the registers take turns capturing bits of the incoming DDR data signal.

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

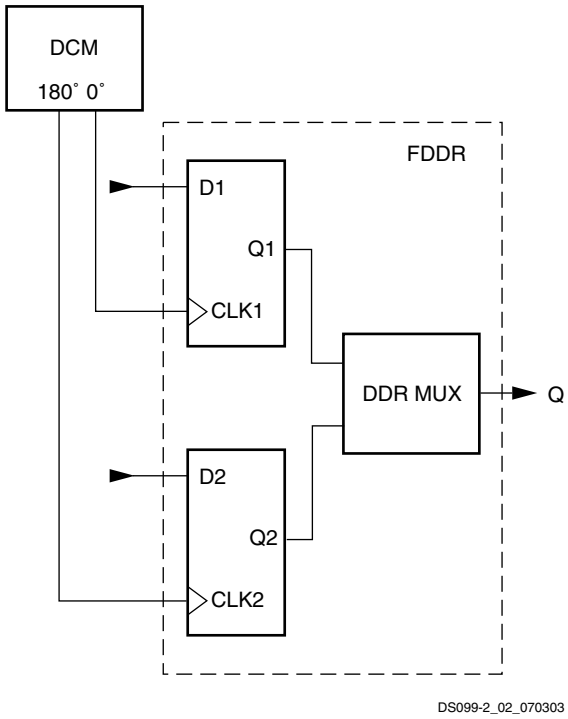


Figure 2: **Clocking the DDR Register**

Pull-Up and Pull-Down Resistors

The optional pull-up and pull-down resistors are intended to establish High and Low levels, respectively, at unused I/Os. The pull-up resistor optionally connects each IOB pad to V_{CCO}. A pull-down resistor optionally connects each pad to GND. These resistors are placed in a design using the PULLUP and PULLDOWN symbols in a schematic, respectively. They can also be instantiated as components, set as constraints or passed as attributes in HDL code. These resistors can also be selected for all unused I/O using the Bitstream Generator (BitGen) option UnusedPin. A Low logic level on HSWAP_EN activates the pull-up resistors on all I/Os during configuration.

Keeper Circuit

Each I/O has an optional keeper circuit that retains the last logic level on a line after all drivers have been turned off. This is useful to keep bus lines from floating when all connected drivers are in a high-impedance state. This function is placed in a design using the KEEPER symbol. Pull-up and pull-down resistors override the keeper circuit.

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-3 I/Os to tolerate high signal voltages. The V_{IN} absolute maximum rating in Table 1, Module 3 specifies the voltage range that I/Os can tolerate.

Slew Rate Control and Drive Strength

Two options, FAST and SLOW, control the output slew rate. The FAST option supports output switching at a high rate. The SLOW option reduces bus transients. These options are only available when using one of the LVCMOS or LVTTL standards, which also provide up to seven different levels of current drive strength: 2, 4, 6, 8, 12, 16, and 24 mA. Choosing the appropriate drive strength level is yet another means to minimize bus transients.

Table 3 shows the drive strengths that the LVCMOS and LVTTL standards support. The Fast option is indicated by appending an "F" attribute after the output buffer symbol OBUF or the bidirectional buffer symbol IOBUF. The Slow option appends an "S" attribute. The drive strength in milliamperes follows the slew rate attribute. For example, OBUF_LVCMOS18_S_6 or IOBUF_LVCMOS25_F_16.

Table 3: **Programmable Output Drive Current**

Signal Standard	Current Drive (mA)						
	2	4	6	8	12	16	24
LVCMOS12	✓	✓	✓	-	-	-	-
LVCMOS15	✓	✓	✓	✓	✓	-	-
LVCMOS18	✓	✓	✓	✓	✓	✓	-
LVCMOS25	✓	✓	✓	✓	✓	✓	✓
LVCMOS33	✓	✓	✓	✓	✓	✓	✓
LVTTL	✓	✓	✓	✓	✓	✓	✓

Boundary-Scan Capability

All Spartan-3 IOBs support boundary-scan testing compatible with IEEE 1149.1 standards. See **Boundary-Scan (JTAG) Mode**, page 36 for more information.

SelectIO Signal Standards

The IOBs support 17 different single-ended signal standards, as listed in Table 4. Furthermore, the majority of IOBs can be used in specific pairs supporting any of six differential signal standards, as shown in Table 5. The desired standard is selected by placing the appropriate I/O library symbol or component into the FPGA design. For example, the symbol named IOBUF_LVCMOS15_F_8 represents a bidirectional I/O to which the 1.5V LVCMOS signal standard has been assigned. The slew rate and current drive are set to Fast and 8 mA, respectively.

Together with placing the appropriate I/O symbol, two externally applied voltage levels, V_{CCO} and V_{REF} select the desired signal standard. The V_{CCO} lines provide current to the output driver. The voltage on these lines determines the output voltage swing for all standards except GTL and GTLP.

All single-ended standards except the LVCMOS, LVTTL, and PCI varieties require a 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 such a signal standard, a few specifically reserved I/O pins on the same bank automatically convert to V_{REF} inputs. When using one of the LVCMOS standards, these pins remain I/Os because the V_{CCO} voltage biases the input-switching threshold, so there is no need for V_{REF}. Select the V_{CCO} and V_{REF} levels to suit the desired single-ended standard according to Table 4.

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

Each device-package combination designates specific I/O pairs that are specially optimized to support differential standards. 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_7 and IO_L43N_7 indicate the true and inverted lines comprising the line pair L43 on Bank 7. The V_{CCO} lines provide current to the outputs. The V_{REF} lines are not used. Select the V_{CCO} level to suit the desired differential standard according to Table 5.

Table 4: **Single-Ended I/O Standards (Values in Volts)**

Signal Standard	V_{CCO}		V_{REF} for Inputs(1)	Board Termination Voltage (V_{TT})
	For Outputs	For Inputs		
GTL	Note 2	Note 2	0.8	1.2
GTLP	Note 2	Note 2	1	1.5
HSTL_I	1.5	-	0.75	0.75
HSTL_III	1.5	-	0.9	1.5
HSTL_I_18	1.8	-	0.9	0.9
HSTL_II_18	1.8	-	0.9	0.9
HSTL_III_18	1.8	-	1.1	1.8
LVCMOS12	1.2	1.2	-	-
LVCMOS15	1.5	1.5	-	-
LVCMOS18	1.8	1.8	-	-
LVCMOS25	2.5	2.5	-	-
LVCMOS33	3.3	3.3	-	-
LVTTL	3.3	3.3	-	-
PCI33_3	3.0	3.0	-	-
SSTL18_I	1.8	-	0.9	0.9
SSTL2_I	2.5	-	1.25	1.25
SSTL2_II	2.5	-	1.25	1.25

Notes:
1. Banks 4 and 5 of any Spartan-3 device in a VQ100 package do not support signal standards using V_{REF}.
2. The V_{CCO} level used for the GTL and GTLP standards must be no lower than the termination voltage (V_{TT}), nor can it be lower than the voltage at the I/O pad.
3. See Table 6 for a listing of the single-ended DCI standards.

Table 5: **Differential I/O Standards**

Signal Standard	V_{CCO} (Volts)		V_{REF} for Inputs (Volts)
	For Outputs	For Inputs	
LDT_25 (ULVDS_25)	2.5	-	-
LVDS_25	2.5	-	-
BLVDS_25	2.5	-	-
LVDSEXT_25	2.5	-	-
LVPECL_25	2.5	-	-
RSDS_25	2.5	-	-

Notes:
1. See Table 6 for a listing of the differential DCI standards.

The need to supply V_{REF} and V_{CCO} imposes constraints on which standards can be used in the same bank. See **The Organization of IOBs into Banks** section for additional guidelines concerning the use of the V_{CCO} and V_{REF} lines.

Digitally Controlled Impedance (DCI)

When the round-trip delay of an output signal — i.e., from output to input and back again — exceeds rise and fall times, it is common practice to add termination resistors to the line carrying the signal. These resistors effectively match the impedance of a device’s I/O to the characteristic impedance of the transmission line, thereby preventing reflections that adversely affect signal integrity. However, with the high I/O counts supported by modern devices, adding resistors requires significantly more components and board area. Furthermore, for some packages — e.g., ball grid arrays — it may not always be possible to place resistors close to pins.

DCI answers these concerns by providing two kinds of on-chip terminations: Parallel terminations make use of an integrated resistor network. Series terminations result from controlling the impedance of output drivers. DCI actively adjusts both parallel and series terminations to accurately match the characteristic impedance of the transmission line. This adjustment process compensates for differences in I/O impedance that can result from normal variation in the ambient temperature, the supply voltage and the manufac-

turing process. When the output driver turns off, the series termination, by definition, approaches a very high impedance; in contrast, parallel termination resistors remain at the targeted values.

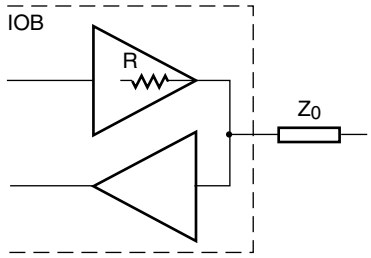
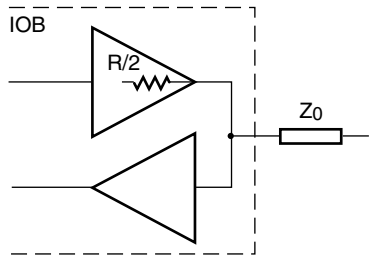
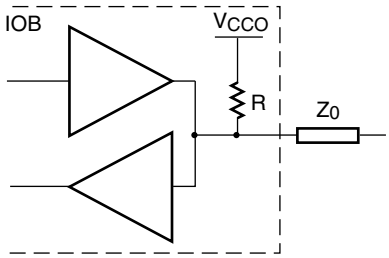
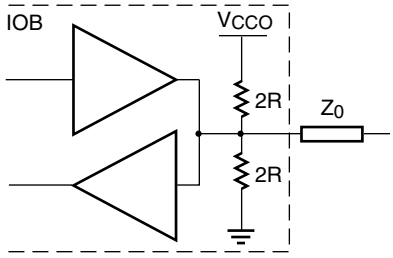
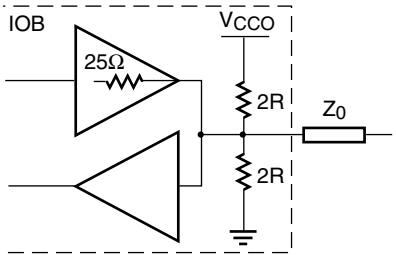
DCI is available only for certain I/O standards, as listed in Table 6. DCI is selected by applying the appropriate I/O standard extensions to symbols or components. There are five basic ways to configure terminations, as shown in Table 7. The DCI I/O standard determines which of these terminations is put into effect.

Table 6: **DCI I/O Standards**

Category of Signal Standard	Signal Standard	V_{CCO} (V)		V_{REF} for Inputs (V)	Termination Type	
		For Outputs	For Inputs		At Output	At Input
Single-Ended						
Gunning Transceiver Logic	GTL_DCI	1.2	1.2	0.8	Single	Single
	GTLP_DCI	1.5	1.5	1.0		
High-Speed Transceiver Logic	HSTL_I_DCI	1.5	1.5	0.75	None	Split
	HSTL_III_DCI	1.5	1.5	0.9	None	Single
	HSTL_I_DCI_18	1.8	1.8	0.9	None	Split
	HSTL_II_DCI_18	1.8	1.8	0.9	Split	
	HSTL_III_DCI_18	1.8	1.8	1.1	None	Single
Low-Voltage CMOS	LVDCI_15	1.5	1.5	-	Controlled impedance driver	None
	LVDCI_18	1.8	1.8	-		
	LVDCI_25	2.5	2.5	-		
	LVDCI_33	3.3	3.3	-		
	LVDCI_DV2_15	1.5	1.5	-	Controlled driver with half-impedance	
	LVDCI_DV2_18	1.8	1.8	-		
	LVDCI_DV2_25	2.5	2.5	-		
	LVDCI_DV2_33	3.3	3.3	-		
Stub Series Terminated Logic	SSTL18_I_DCI	1.8	1.8	0.9	25-Ohm driver	Split
	SSTL2_I_DCI	2.5	2.5	1.25	25-Ohm driver	
	SSTL2_II_DCI	2.5	2.5	1.25	Split with 25-Ohm driver	
Differential						
Low-Voltage Differential Signalling	LVDS_25_DCI	2.5	2.5	-	None	Split on each line of pair
	LVDSEXT_25_DCI	2.5	2.5	-		

Notes:
1. Bank 5 of any Spartan-3 device in a VQ100 or TQ144 package does not support DCI signal standards.

Table 7: **DCI Terminations**

Termination	Schematic[1]	I/O Standards
Controlled impedance output driver		LVDCI_15 LVDCI_18 LVDCI_25 LVDCI_33
Controlled output driver with half impedance		LVDCI_DV2_15 LVDCI_DV2_18 LVDCI_DV2_25 LVDCI_DV2_33
Single resistor		GTL_DCI GTLP_DCI HSTL_III_DCI[2] HSTL_III_DCI_18[2]
Split resistors		HSTL_I_DCI[2] HSTL_I_DCI_18[2] HSTL_II_DCI_18 LVDS_25_DCI LVDSEXT_25_DCI
Split resistors with output driver impedance fixed to 25Ω		SSTL18_I_DCI[3] SSTL2_I_DCI[3] SSTL2_II_DCI

Notes:

1. The value of R is equivalent to the characteristic impedance of the line connected to the I/O. It is also equal to half the value of R_{REF} for the DV2 standards and R_{REF} for all other DCI standards.
2. For DCI using HSTL Classes I and III, terminations only go into effect at inputs (not at outputs).
3. For DCI using SSTL Class I, the split termination only goes into effect at inputs (not at outputs).

The DCI feature operates independently for each of the device's eight banks. Each bank has an "N" reference pin (VRN) and a "P" reference pin, (VRP), to calibrate driver and termination resistance. Only when using a DCI standard on a given bank do these two pins function as VRN and VRP. When not using a DCI standard, the two pins function as user I/Os. As shown in Figure 3, add an external reference resistor to pull the VRN pin up to V_{CCO} and another reference resistor to pull the VRP pin down to GND. Both resistors have the same value — commonly 50 Ohms — with one-percent tolerance, which is either the characteristic impedance of the line or twice that, depending on the DCI standard in use. Standards having a symbol name that contains the letters "DV2" use a reference resistor value that is twice the line impedance. DCI adjusts the output driver impedance to match the reference resistors' value or half that, according to the standard. DCI always adjusts the on-chip termination resistors to directly match the reference resistors' value.

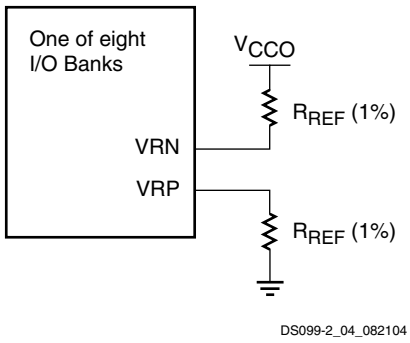


Figure 3: **Connection of Reference Resistors (R_{REF})**

The rules guiding the use of DCI standards on banks are as follows:

1. No more than one DCI I/O standard with a Single Termination is allowed per bank.
2. No more than one DCI I/O standard with a Split Termination is allowed per bank.
3. Single Termination, Split Termination, Controlled-Impedance Driver, and Controlled-Impedance Driver with Half Impedance can co-exist in the same bank.

See also **The Organization of IOBs into Banks**, page 8.

The Organization of IOBs into Banks

IOBs are allocated among eight banks, so that each side of the device has two banks, as shown in Figure 4. For all packages, each bank has independent V_{REF} lines. For example, V_{REF} Bank 3 lines are separate from the V_{REF} lines going to all other banks.

For the Very Thin Quad Flat Pack (VQ), Plastic Quad Flat Pack (PQ), Fine Pitch Thin Ball Grid Array (FT), and Fine Pitch Ball Grid Array (FG) packages, each bank has dedicated V_{CCO} lines. For example, the V_{CCO} Bank 7 lines are separate from the V_{CCO} lines going to all other banks. Thus, Spartan-3 devices in these packages support eight independent V_{CCO} supplies.

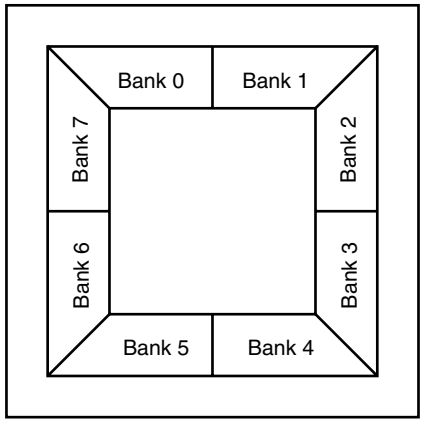


Figure 4: **Spartan-3 I/O Banks (top view)**

In contrast, the 144-pin Thin Quad Flat Pack (TQ144) package ties V_{CCO} together internally for the pair of banks on each side of the device. For example, the V_{CCO} Bank 0 and the V_{CCO} Bank 1 lines are tied together. The interconnected bank-pairs are 0/1, 2/3, 4/5, and 6/7. As a result, Spartan-3 devices in the TQ144 package support four independent V_{CCO} supplies.

Spartan-3 Compatibility

Within the Spartan-3 family, all devices are pin-compatible by package. When the need for future logic resources outgrows the capacity of the Spartan-3 device in current use, a larger device in the same package can serve as a direct replacement. Larger devices may add extra V_{REF} and V_{CCO} lines to support a greater number of I/Os. In the larger device, more pins can convert from user I/Os to V_{REF} lines. Also, additional V_{CCO} lines are bonded out to pins that were "not connected" in the smaller device. Thus, it is important to plan for future upgrades at the time of the board's initial design by laying out connections to the extra pins.

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

Rules Concerning Banks

When assigning I/Os to banks, it is important to follow the following V_{CCO} rules:

1. Leave no V_{CCO} pins unconnected on the FPGA.
2. Set all V_{CCO} lines associated with the (interconnected) bank to the same voltage level.
3. The V_{CCO} levels used by all standards assigned to the I/Os of the (interconnected) bank(s) must agree. The Xilinx development software checks for this. Tables 4, 5, and 6 describe how different standards use the V_{CCO} supply.

4. If none of the standards assigned to the I/Os of the (interconnected) bank(s) use V_{CCO}, tie all associated V_{CCO} lines to 2.5V.
5. In general, apply 2.5V to V_{CCO} Bank 4 from power-on to the end of configuration. Apply the same voltage to V_{CCO} Bank 5 during parallel configuration or a Readback operation. For information on how to program the FPGA using 3.3V signals and power, see the **3.3V-Tolerant Configuration Interface** section.

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

1. Leave no V_{REF} pins unconnected on any bank.
2. Set all V_{REF} lines associated with the bank 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. Tables 4 and 6 describe how different standards use the V_{REF} supply.

If none of the standards assigned to the Inputs of a bank use V_{REF} for biasing input switching thresholds, all associated V_{REF} pins function as User I/Os.

Exceptions to Banks Supporting I/O Standards

Bank 5 of any Spartan-3 device in a VQ100 or TQ144 package does not support DCI signal standards. In this case, bank 5 has neither VRN nor VRP pins.

Furthermore, banks 4 and 5 of any Spartan-3 device in a VQ100 package do not support signal standards using V_{REF} (see Table 4). In this case, the two banks do not have any V_{REF} pins.

Supply Voltages for the IOBs

Three different supplies power the IOBs:

1. The V_{CCO} supplies, one for each of the FPGA's I/O banks, power the output drivers, except when using the GTL and GTLP signal standards. 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. The 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

With no power applied to the FPGA, all I/Os are in a high-impedance state. The V_{CCINT} (1.2V), V_{CCAUX} (2.5V), and V_{CCO} supplies may be applied in any order. Before power-on can finish, V_{CCINT}, V_{CCO} Bank 4, and V_{CCAUX} must have reached their respective minimum recommended operating levels (see **Table 2 in Module 3**). At this time, all I/O drivers also will be in a high-impedance state. V_{CCO} Bank 4, V_{CCINT}, and V_{CCAUX} serve as inputs to the internal Power-On Reset circuit (POR).

A Low level applied to the HSWAP_EN input enables pull-up resistors on User I/Os from power-on throughout configuration. A High level on HSWAP_EN disables the pull-up resistors, allowing the I/Os to float. 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 Low state.

Upon the completion of initialization, INIT_B goes High, sampling the M0, M1, and M2 inputs to determine the configuration mode. At this point, 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_EN input) throughout configuration.

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. At this point, those I/Os to which signals have been assigned go active while all unused I/Os remain in a high-impedance state. The release of the GSR net, also part of Start-up, leaves the IOB registers in a Low state by default, unless the loaded design reverses the polarity of their respective RS inputs.

In User mode, all internal pull-up resistors on the I/Os are disabled and HSWAP_EN becomes a "don't care" input. If it is desirable to have pull-up or pull-down resistors on I/Os carrying signals, the appropriate symbol — e.g., PULLUP, PULLDOWN — must be placed at the appropriate pads in the design. The Bitstream Generator (Bitgen) option UnusedPin available in the Xilinx development software determines whether unused I/Os collectively have pull-up resistors, pull-down resistors, or no resistors in User mode.

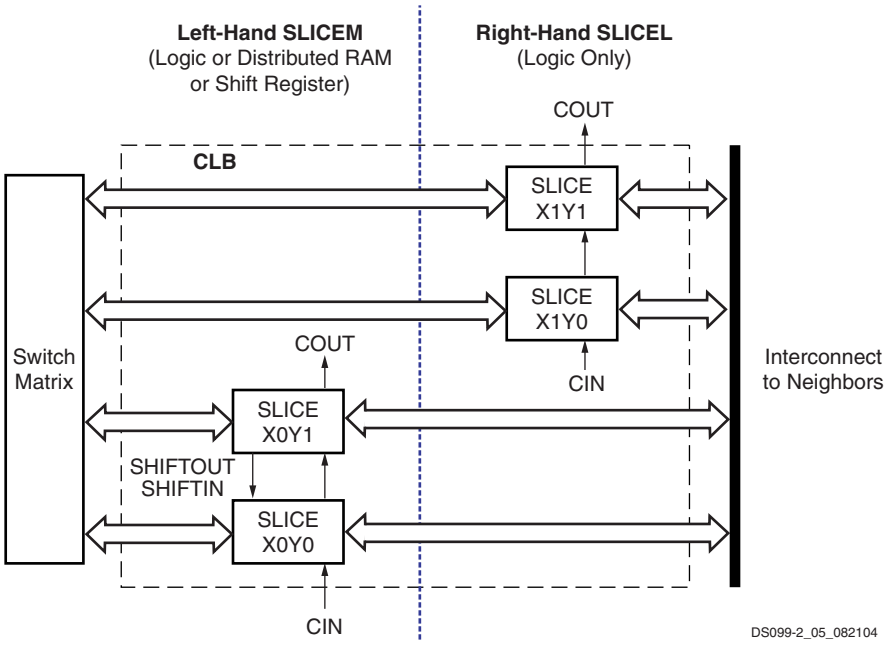


Figure 5: **Arrangement of Slices within the CLB**

CLB Overview

The Configurable Logic Blocks (CLBs) constitute the main logic resource for implementing synchronous as well as combinatorial circuits. Each CLB comprises four interconnected slices, as shown in Figure 5. These slices are grouped in pairs. Each pair is organized as a column with an independent carry chain.

The nomenclature that the FPGA Editor — part of the Xilinx development software — uses to designate slices is as follows: The letter "X" followed by a number identifies columns of slices. The "X" number counts up in sequence 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. The "Y" number counts slices starting from the bottom of the die according to the sequence: 0, 1, 0, 1 (the first CLB row); 2, 3, 2, 3 (the second CLB row); etc. Figure 5 shows the CLB located in the lower left-hand corner of the die. Slices X0Y0 and X0Y1 make up the column-pair on the left where as slices X1Y0 and X1Y1 make up the column-pair on the right. For each CLB, the term “left-hand” (or SLICEM) is used to indicated the pair of slices labeled with an even "X" number, such as X0, and the term “right-hand” (or SLICEL) designates the pair of slices with an odd "X" number, e.g., X1.

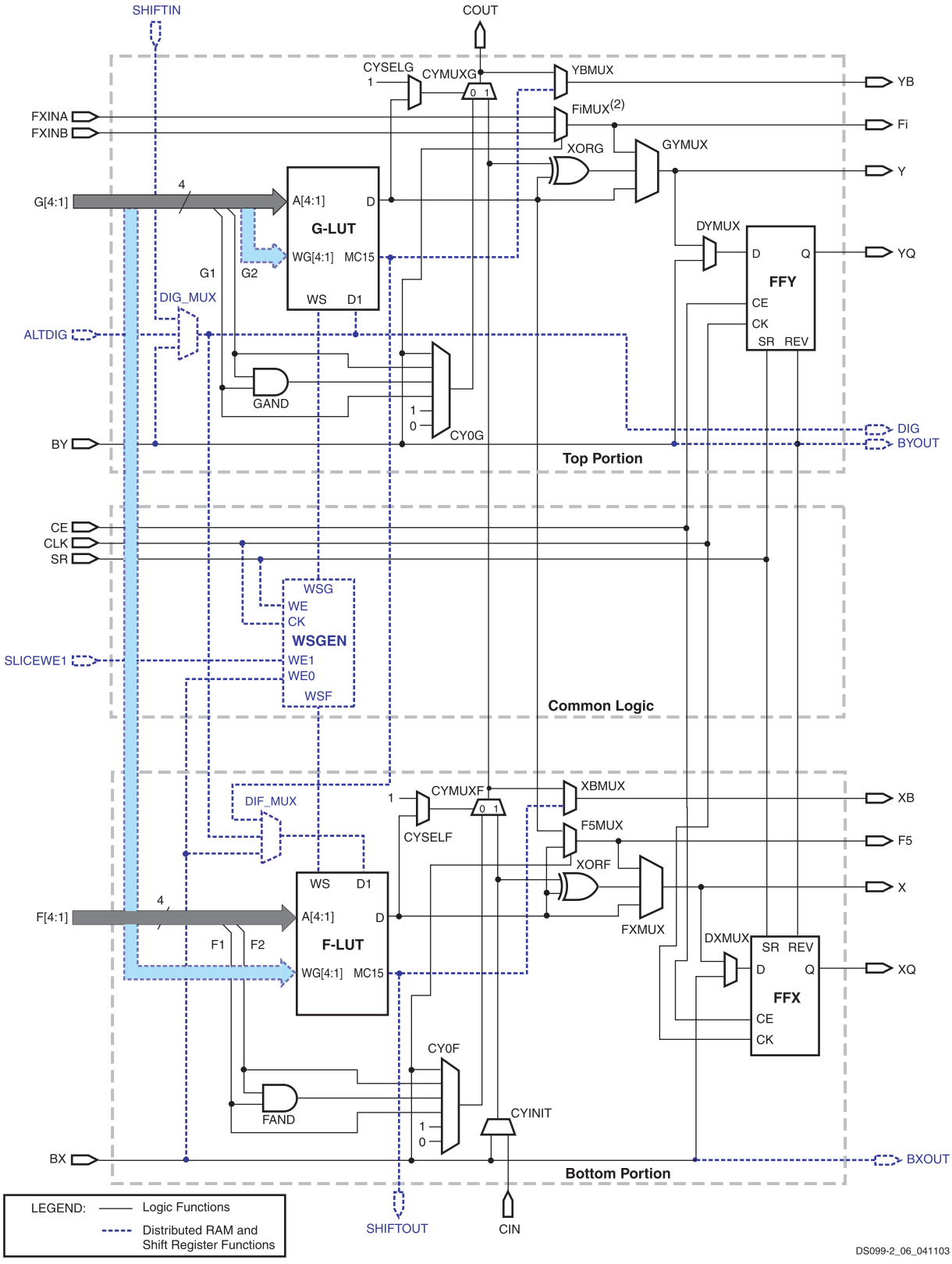
Elements Within a Slice

All four slices have the following elements in common: two logic function generators, two storage elements, wide-function multiplexers, carry logic, and arithmetic gates, as shown in Figure 6. Both the left-hand and right-hand slice pairs use these elements to provide logic, arithmetic, and ROM functions. Besides these, the left-hand pair supports two additional functions: storing data using Distributed RAM and shifting data with 16-bit registers. Figure 6 is a diagram of the left-hand slice; therefore, it represents a superset of the elements and connections to be found in all slices. See **Function Generator**, page 12 for more information.

The RAM-based function generator — also known as a Look-Up Table or LUT — is the main resource for implementing logic functions. Furthermore, the LUTs in each left-hand slice pair can be configured as Distributed RAM or a 16-bit shift register. For information on the former, see **XAPP464**: *Using Look-Up Tables as Distributed RAM in Spartan-3 FPGAs*; for information on the latter, refer to **XAPP465**: *Using Look-Up Tables as Shift Registers (SRL16) in Spartan-3 FPGAs*. The function generators located in the upper and lower portions of the slice are referred to as the "G" and "F", respectively.

The storage element, which is programmable as either a D-type flip-flop or a level-sensitive latch, provides a means for synchronizing data to a clock signal, among other uses. The storage elements in the upper and lower portions of the slice are called FFY and FFX, respectively.

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 lower portion of the slice and FXMUX in the upper portion. Depending on the slice, FXMUX takes on the name F6MUX, F7MUX, or F8MUX. For more details on the multiplexers, see **XAPP466**: *Using Dedicated Multiplexers in Spartan-3 FPGAs*.



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. In this position, the upper right-hand slice has an F8MUX, and the upper left-hand slice has an F7MUX. The lower right-hand and left-hand slices both have an F6MUX.

Figure 6: **Simplified Diagram of the Left-Hand SLICEM**

The carry chain, together with various dedicated arithmetic logic gates, support fast and efficient implementations of math operations. The carry chain enters the slice as CIN and exits as COUT. Five multiplexers control the chain: CYINIT, CY0F, and CYMUXF in the lower portion as well as CY0G and CYMUXG in the upper portion. 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).

Main Logic Paths

Central to the operation of each slice are two nearly identical data paths, distinguished using the terms *top* and *bottom*. 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. 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 function generator "F" (or "G") that performs logic operations. The function generator's 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 (DYMUX) which feeds the data input, "D", of the FFY (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, then exit the slice as BXOUT (or BYOUT) and return to interconnect.
2. Bypass the LUT, then pass through a storage element via the D input before exiting as XQ (or YQ).
3. Control the wide function multiplexer F5MUX (or F6MUX).
4. Via multiplexers, serve as an input to the carry chain.
5. Drives the DI input of the LUT.
6. BY can control the REV inputs of both the FFY and FFX storage elements. See Storage Element Section.
7. Finally, the DIG_MUX multiplexer can switch BY onto to the DIG line, which exits the slice.

Other slice signals shown in Figure 6, page 11 are discussed in the sections that follow.

Function Generator

Each of the two LUTs (F and G) in a slice have four logic inputs (A1-A4) and a single output (D). This permits any four-variable Boolean logic operation to be programmed into them. Furthermore, wide function multiplexers can be used to effectively combine LUTs within the same CLB or across different CLBs, making logic functions with still more input variables possible.

The LUTs in both the right-hand and left-hand slice-pairs not only support the logic functions described above, but also can function as ROM that is initialized with data at the time of configuration.

The LUTs in the left-hand slice-pair (even-numbered columns such as X0 in Figure 5) of each CLB support two additional functions that the right-hand slice-pair (odd-numbered columns such as X1) do not.

First, it is possible to program the "left-hand LUTs" as distributed RAM. This type of memory affords moderate amounts of data buffering anywhere along a data path. One left-hand LUT stores 16 bits. Multiple left-hand LUTs can be combined in various ways to store larger amounts of data. A dual port option combines two LUTs so that memory access is possible from two independent data lines. A Distributed ROM option permits pre-loading the memory with data during FPGA configuration.

Second, it is possible to program each left-hand LUT as a 16-bit shift register. Used in this way, each LUT can delay serial data anywhere from one to 16 clock cycles. The four left-hand LUTs of a single CLB can be combined to produce delays up to 64 clock cycles. The SHIFTIN and SHIFTOUT lines cascade LUTs to form larger shift registers. It is also possible to combine shift registers across more than one CLB. The resulting programmable delays can be used to balance the timing of data pipelines.

Block RAM Overview

All Spartan-3 devices support block RAM, which is organized as configurable, synchronous 18Kbit blocks. Block RAM stores relatively large amounts of data more efficiently than the distributed RAM feature described earlier. (The latter is better suited for buffering small amounts of data anywhere along signal paths.) This section describes basic Block RAM functions. For more information, see **XAPP463**: *Using Block RAM in Spartan-3 FPGAs*.

The aspect ratio — i.e., width vs. depth — of each block RAM is configurable. Furthermore, multiple blocks can be cascaded to create still wider and/or deeper memories.

A choice among primitives determines whether the block RAM functions as dual- or single-port memory. A name of the form RAM16_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 w_A and w_B, respectively. Thus, a RAM16_S9_S18 is a dual-port RAM with a 9-bit-wide Port A and an 18-bit-wide Port B. A name of the form RAM16_S[w] identifies the single-port primitive, where the integer w specifies the total data path width of the lone port. A RAM16_S18 is a single-port RAM with an 18-bit-wide port. Other memory functions — e.g., FIFOs, data path width conversion, ROM, etc. — are readily available using the CORE Generator™ system, part of the Xilinx development software.

Arrangement of RAM Blocks on Die

The XC3S50 has one column of block RAM. The Spartan-3 devices ranging from the XC3S200 to XC3S2000 have two columns of block RAM. The XC3S4000 and XC3S5000 have four columns. The position of the columns on the die is shown in **Figure 1 in Module 1**. For a given device, the total available RAM blocks are distributed equally among the columns. Table 8 shows the number of RAM blocks, the data storage capacity, and the number of columns for each device.

Table 8: **Number of RAM Blocks by Device**

Device	Total Number of RAM Blocks	Total Addressable Locations (bits)	Number of Columns
XC3S50	4	73,728	1
XC3S200	12	221,184	2
XC3S400	16	294,912	2
XC3S1000	24	442,368	2
XC3S1500	32	589,824	2
XC3S2000	40	737,280	2
XC3S4000	96	1,769,472	4
XC3S5000	104	1,916,928	4

Block RAM and multipliers have interconnects between them that permit simultaneous operation; however, since the multiplier shares inputs with the upper data bits of block RAM, the maximum data path width of the block RAM is 18 bits in this case.

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 RAM block, which has a maximum capacity of 18,432 bits — or 16,384 bits when no parity lines are used. 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 7: (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, and (4) data transfer from Port B to Port A.

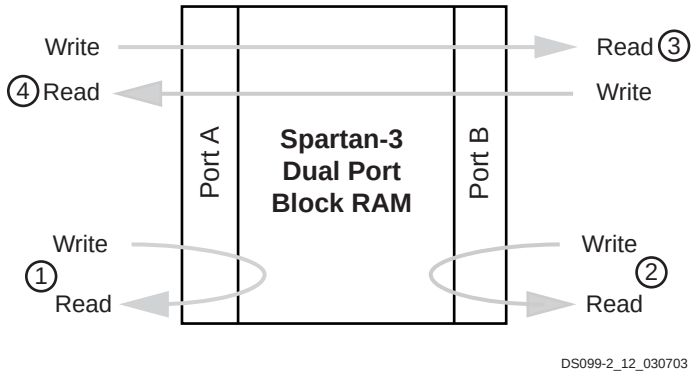
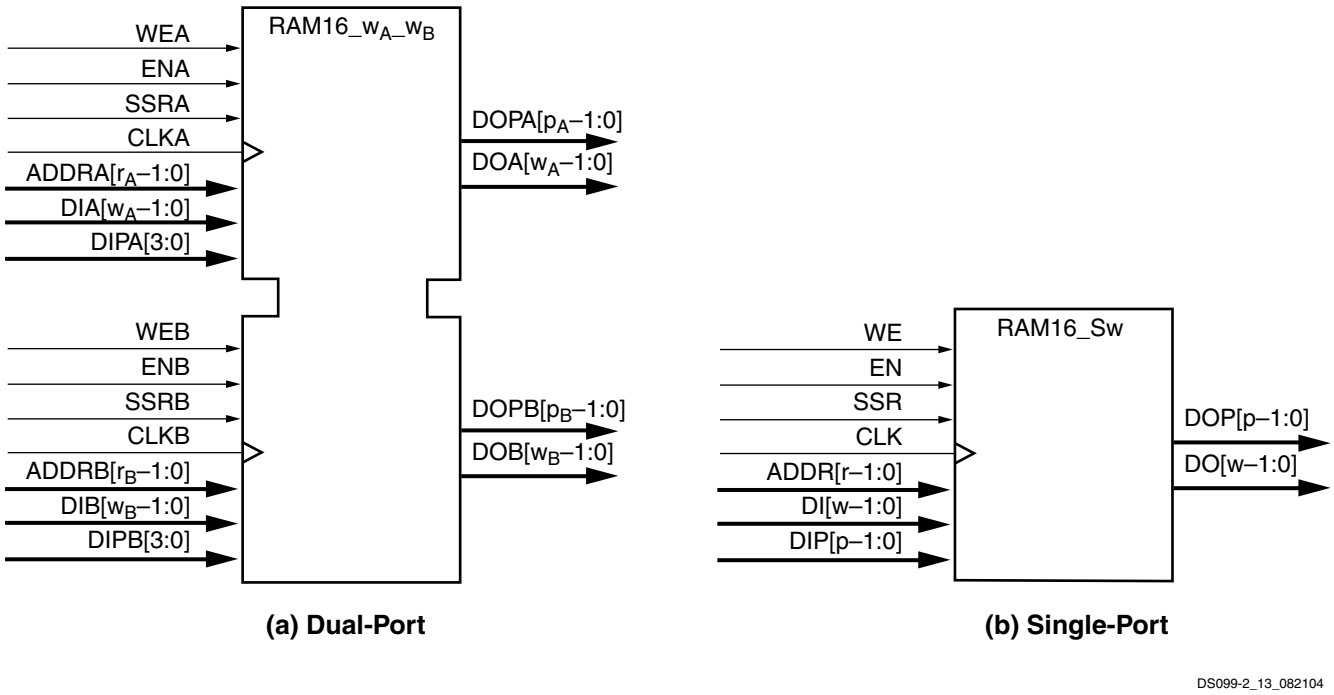


Figure 7: **Block RAM Data Paths**

Block RAM Port Signal Definitions

Representations of the dual-port primitive RAM16_S[w_A]_S[w_B] and the single-port primitive RAM16_S[w] with their associated signals are shown in Figure 8a and Figure 8b, respectively. These signals are defined in Table 9.



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 8: **Block RAM Primitives**

Table 9: **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).
Data Input Bus	DIA	DIB	Input	Data at the DI input bus is written to the addressed memory location addressed on an enabled active CLK edge. It is possible to configure a port's total data path width (w) to be 1, 2, 4, 9, 18, or 36 bits. This selection applies to both the DI and DO paths of a given port. Each port is independent. For a port assigned a width (w), the number of addressable locations will be 16,384/(w-p) where "p" is the number of parity bits. Each memory location will have a width of "w" (including parity bits). See the DIP signal description for more information of parity.
Parity Data Input(s)	DIPA	DIPB	Input	Parity inputs represent additional bits included in the data input path to support error detection. 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 10.

Table 9: **Block RAM Port Signals *(Continued)***

Signal Description	Port A Signal Name	Port B Signal Name	Direction	Function
Data Output Bus	DOA	DOB	Output	Basic data access occurs whenever WE is inactive. The DO outputs mirror the data stored in the addressed memory location. Data access with WE asserted is also possible if one of the following two attributes is chosen: WRITE_FIRST accesses data before the write takes place. READ_FIRST accesses data after the write occurs. A third attribute, NO_CHANGE, latches the DO outputs upon the assertion of WE. It is possible to configure a port's total data path width (w) to be 1, 2, 4, 9, 18, or 36 bits. This selection applies to both the DI and DO paths. See the DI signal description.
Parity Data Output(s)	DOPA	DOPB	Output	Parity inputs represent additional bits included in the data input path to support error detection. 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 10.
Write Enable	WEA	WEB	Input	When asserted together with EN, this input enables the writing of data to the RAM. In this case, the data access attributes WRITE_FIRST, READ_FIRST or NO_CHANGE determines if and how data is updated on the DO outputs. See the DO signal description. When WE is inactive with EN asserted, read operations are still possible. In this case, a transparent 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 synchronize Block RAM functions as follows: the writing of data to the DI inputs (when WE is also asserted), the updating of data at the DO outputs as well as the setting/resetting of the DO output latches. When de-asserted, the above functions are disabled.
Set/Reset	SSRA	SSRB	Input	When asserted, this pin forces the DO output latch to the value that the SRVAL attribute is set to. A Set/Reset operation on one port has no effect on the other ports functioning, nor does it disturb the memory's data contents. 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.

Port Aspect Ratios

On a given port, it is possible to select a number of different possible widths (w – p) for the DI/DO buses as shown in Table 10. These two buses always have the same width. This data bus width selection is independent for each port. If the data bus width of Port A differs from that of Port B, the Block RAM automatically performs a bus-matching function. When data are written to a port with a narrow bus, then read from a port with a wide bus, the latter port will effectively combine "narrow" words to form "wide" words. Similarly, when data are written into a port with a wide bus, then read from a port with a narrow bus, the latter port will divide

"wide" words to form "narrow" words. When the data bus width is eight bits or greater, extra parity bits become available. The width of the total data path (w) is the sum of the DI/DO bus width and any parity bits (p).

The width selection made for the DI/DO bus determines the number of address lines according to the relationship expressed below:

$$r = 14 - [\log(w-p)/\log(2)] \quad \textbf{(1)}$$

In turn, the number of address lines delimits the total number (n) of addressable locations or depth according to the following equation:

$$n = 2^r \quad \textbf{(2)}$$

The product of w and n yields the total block RAM capacity. Equations (1) and (2) show that as the data bus width increases, the number of address lines along with the number of addressable memory locations decreases. Using the permissible DI/DO bus widths as inputs to these equations provides the bus width and memory capacity measures shown in Table 10.

Table 10: **Port Aspect Ratios for Port A or B**

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

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.

The waveforms for the write operation are shown in the top half of the Figure 9, Figure 10, and Figure 11. 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.

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 transparent output latch to the DO outputs. The timing for basic data access is shown in the portions of Figure 9, Figure 10, and Figure 11 during which WE is Low.

Data can also be accessed on the DO outputs when asserting the WE input. This is accomplished using two different attributes:

Choosing the WRITE_FIRST attribute, 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 9 during which WE is High.

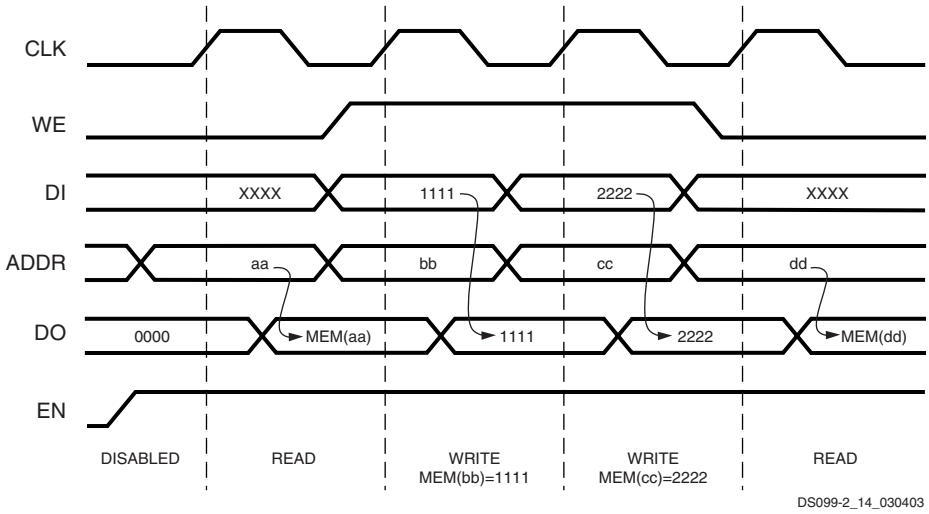


Figure 9: **Waveforms of Block RAM Data Operations with WRITE_FIRST Selected**

Choosing the READ_FIRST attribute, data already stored in the addressed location pass to the DO outputs before that location is over-written with new data from the DI inputs on an enabled active CLK edge. READ_FIRST timing is shown in the portion of Figure 10 during which WE is High.

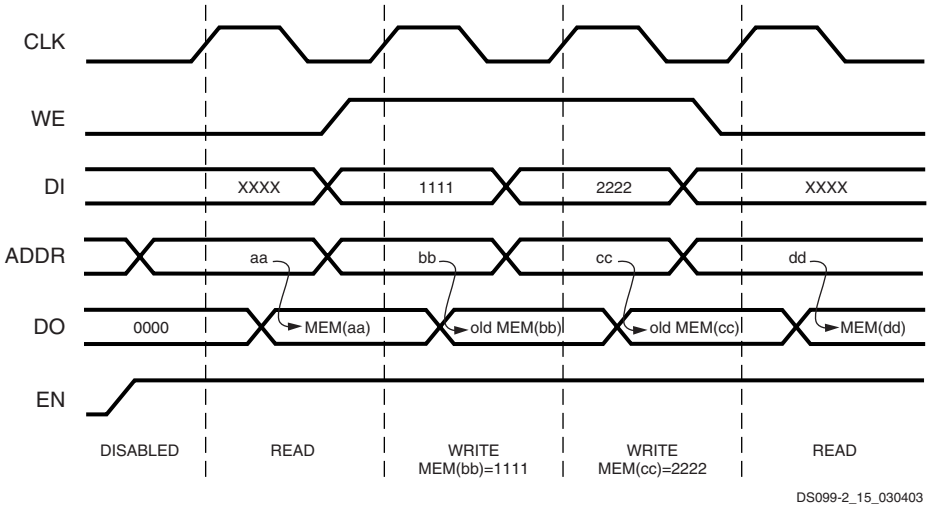


Figure 10: **Waveforms of Block RAM Data Operations with READ_FIRST Selected**

Choosing a third attribute called NO_CHANGE puts the DO outputs in a latched state when asserting WE. Under this condition, the DO outputs will retain the data driven just before WE was asserted. NO_CHANGE timing is shown in the portion of Figure 11 during which WE is High.

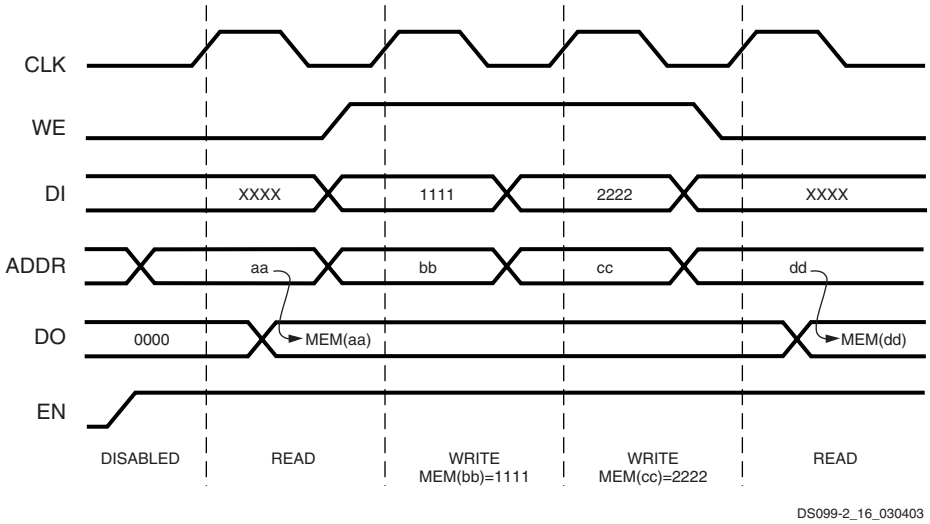


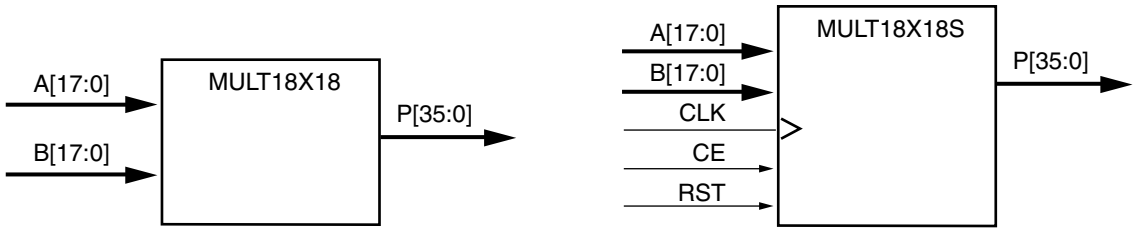
Figure 11: **Waveforms of Block RAM Data Operations with NO_CHANGE Selected**

Dedicated Multipliers

All Spartan-3 devices provide embedded multipliers that accept two 18-bit words as inputs to produce a 36-bit product. This section provides an introduction to multipliers. For further details, see **XAPP467**: *Using Embedded Multipliers in Spartan-3 FPGAs*.

The input buses to the multiplier accept data in two's-complement form (either 18-bit signed or 17-bit unsigned). One such multiplier is matched to each block RAM on the die. The close physical proximity of the two ensures efficient data handling. Cascading multipliers permits multiplicands more than three in number as well as wider than 18-bits. The multiplier is placed in a design using one of two primitives: an asynchronous version called MULT18X18 and a version with a register at the outputs called MULT18X18S, as shown in Figure 12a and Figure 12b, respectively. The signals for these primitives are defined in Table 11.

The CORE Generator system produces multipliers based on these primitives that can be configured to suit a wide range of requirements.



(a) Asynchronous 18-bit Multiplier

(b) 18-bit Multiplier with Register at Outputs

Figure 12: **Embedded Multiplier Primitives**

Table 11: **Embedded Multiplier Primitives Descriptions**

Signal Name	Direction	Function
A[17:0]	Input	Apply one 18-bit multiplicand to these inputs. The MULT18X18S primitive requires a setup time before the enabled rising edge of CLK.
B[17:0]	Input	Apply the other 18-bit multiplicand to these inputs. The MULT18X18S primitive requires a setup time before the enabled rising edge of CLK.
P[35:0]	Output	The output on the P bus is a 36-bit product of the multiplicands A and B. In the case of the MULT18X18S primitive, an enabled rising CLK edge updates the P bus.
CLK	Input	CLK is only an input to the MULT18X18S primitive. The clock signal applied to this input when enabled by CE, updates the output register that drives the P bus.
CE	Input	CE is only an input to the MULT18X18S primitive. Enable for the CLK signal. Asserting this input enables the CLK signal to update the P bus.
RST	Input	RST is only an input to the MULT18X18S primitive. Asserting this input resets the output register on an enabled, rising CLK edge, forcing the P bus to all zeroes.

Notes:
1. The control signals CLK, CE and RST have the option of inverted polarity.

Digital Clock Manager (DCM)

Spartan-3 devices provide flexible, complete control over clock frequency, phase shift and skew through the use of the DCM feature. 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. For further information, see **XAPP462**: *Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs*.

Each member of the Spartan-3 family has four DCMs, except the smallest, the XC3S50, which has two DCMs. The DCMs are located at the ends of the outermost Block RAM column(s). See **Figure 1 in Module 1**. The Digital Clock Manager is placed in 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 may lie in the signal path leading from the clock output of the DCM to its feedback input.
- **Frequency Synthesis:** Provided with an input clock 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.

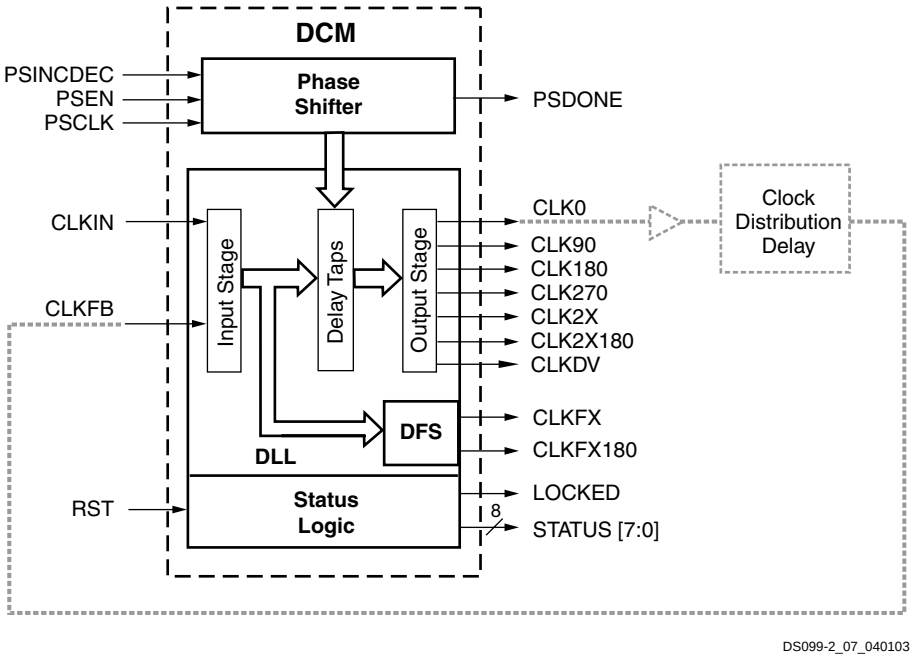


Figure 13: **DCM Functional Blocks and Associated Signals**

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 13.

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 14.

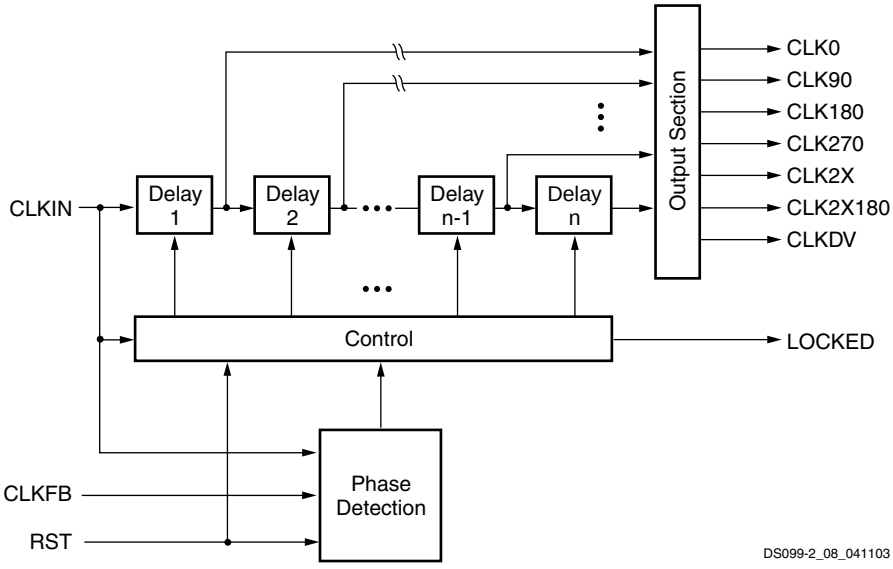


Figure 14: **Simplified Functional Diagram of DLL**

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 12. The clock outputs drive simultaneously; however, the High Frequency mode only supports a subset of the outputs available in the Low Frequency mode. See **DLL Frequency Modes**, page 23. Signals that initialize and report the state of the DLL are discussed in **The Status Logic Component**, page 28.

Table 12: **DLL Signals**

Signal	Direction	Description	Mode Support	
			Low Frequency	High Frequency
CLKIN	Input	Accepts original clock signal.	Yes	Yes
CLKFB	Input	Accepts either CLK0 or CLK2X as feed back signal. (Set CLK_FEEDBACK attribute accordingly).	Yes	Yes
CLK0	Output	Generates clock signal with same frequency and phase as CLKIN.	Yes	Yes
CLK90	Output	Generates clock signal with same frequency as CLKIN, only phase-shifted 90°.	Yes	No
CLK180	Output	Generates clock signal with same frequency as CLKIN, only phase-shifted 180°.	Yes	Yes
CLK270	Output	Generates clock signal with same frequency as CLKIN, only phase-shifted 270°.	Yes	No
CLK2X	Output	Generates clock signal with same phase as CLKIN, only twice the frequency.	Yes	No
CLK2X180	Output	Generates clock signal with twice the frequency of CLKIN, phase-shifted 180° with respect to CLKIN.	Yes	No
CLKDV	Output	Divides the CLKIN frequency by CLKDV_DIVIDE value to generate lower frequency clock signal that is phase-aligned to CLKIN.	Yes	Yes

The clock signal supplied to the CLKIN input serves as a reference waveform, with which the DLL seeks to align the feedback signal at the CLKFB 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 13. Each attribute is described in detail in the sections that follow:

Table 13: **DLL Attributes**

Attribute	Description	Values
CLK_FEEDBACK	Chooses either the CLK0 or CLK2X output to drive the CLKFB input	NONE, 1X, 2X
DLL_FREQUENCY_MODE	Chooses between High Frequency and Low Frequency modes	LOW, HIGH
CLKIN_DIVIDE_BY_2	Halves the frequency of the CLKIN signal just as it enters the DCM	TRUE, FALSE
CLKDV_DIVIDE	Selects 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

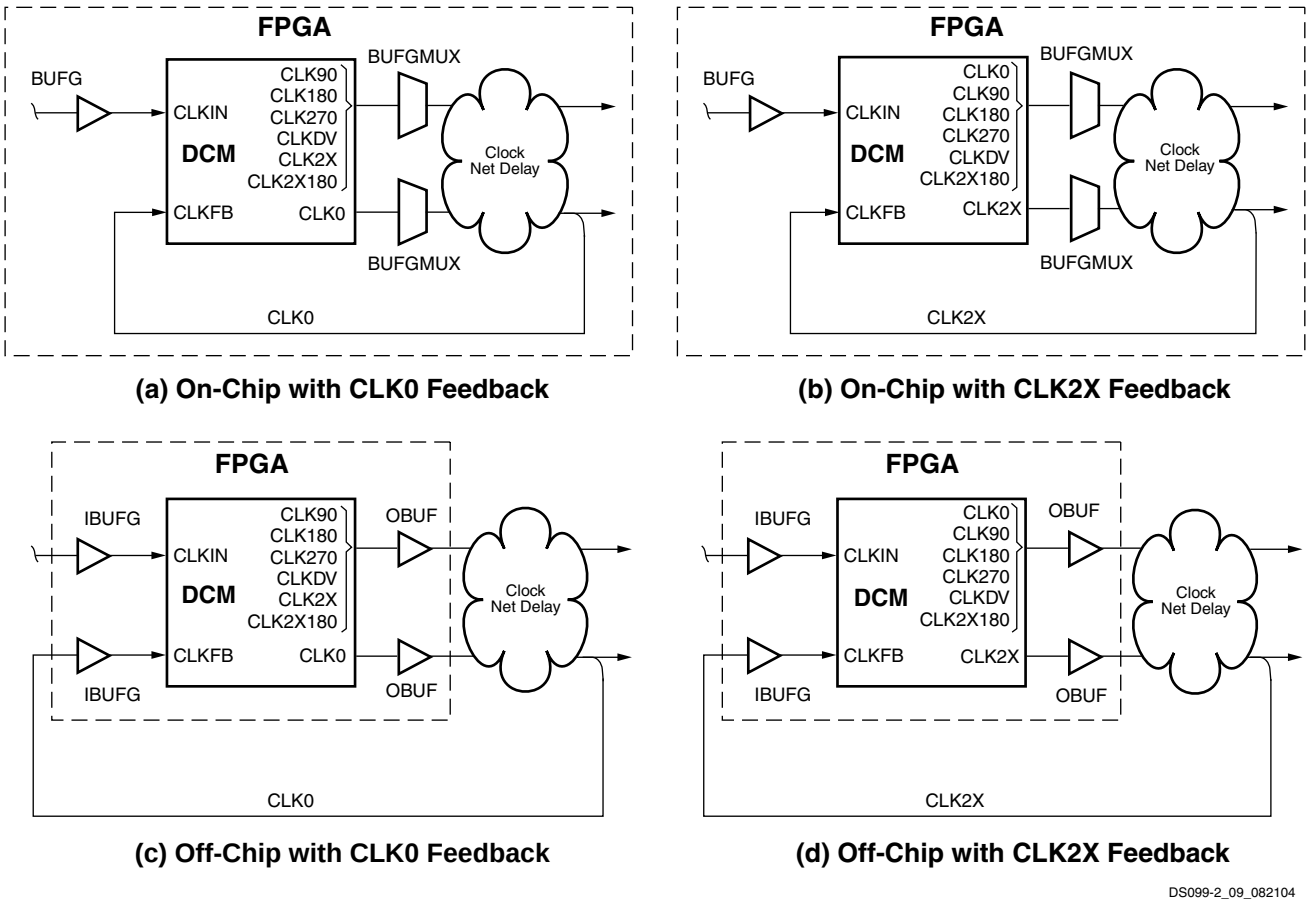
An external clock source enters the FPGA using a Global Clock Input Buffer (IBUFG), which directly accesses the global clock network or an Input Buffer (IBUF). Clock signals within the FPGA drive a global clock net using a Global Clock Multiplexer Buffer (BUFGMUX). The global clock net connects directly to the CLKIN input. The internal and external connections are shown in Figure 15a and Figure 15c, 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 the four BUFGMUX buffers on the same die edge (top or bottom). 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 — i.e., only the DFS is used — then there is no 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 15a through Figure 15d.



Notes:
1. In the Low Frequency mode, all seven DLL outputs are available. In the High Frequency mode, only the CLK0, CLK180, and CLKDV outputs are available.

Figure 15: **Input Clock, Output Clock, and Feedback Connections for the DLL**

In the on-chip synchronization case (Figure 15a and Figure 15b), 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 15a, the feedback loop is created by routing CLK0 (or CLK2X, in Figure 15b) to a global clock net, which in turn drives the CLKFB input.

In the off-chip synchronization case (Figure 15c and Figure 15d), 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 15c, the feedback loop is formed by feeding CLK0 (or CLK2X, in Figure 15d) 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.

DLL Frequency Modes

The DLL supports two distinct operating modes, High Frequency and Low Frequency, with each specified over a different clock frequency range. The DLL_FREQUENCY_MODE attribute chooses between the two modes. When the attribute is set to LOW, the Low Frequency mode permits all seven DLL clock outputs to operate over a low-to-moderate frequency range. When the attribute is set to HIGH, the High Frequency mode allows the CLK0, CLK180 and CLKDV outputs to operate at the highest possible frequencies. The remaining DLL clock outputs are not available for use in High Frequency mode.

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 12.

Their relative timing in the Low Frequency Mode is shown in Figure 16. The CLK90, CLK180 and CLK270 outputs are not available when operating in the High Frequency mode. (See the description of the DLL_FREQUENCY_MODE attribute in Table 13.) For control in finer increments than 90°, see the **Phase Shifter (PS)**, page 26 section.

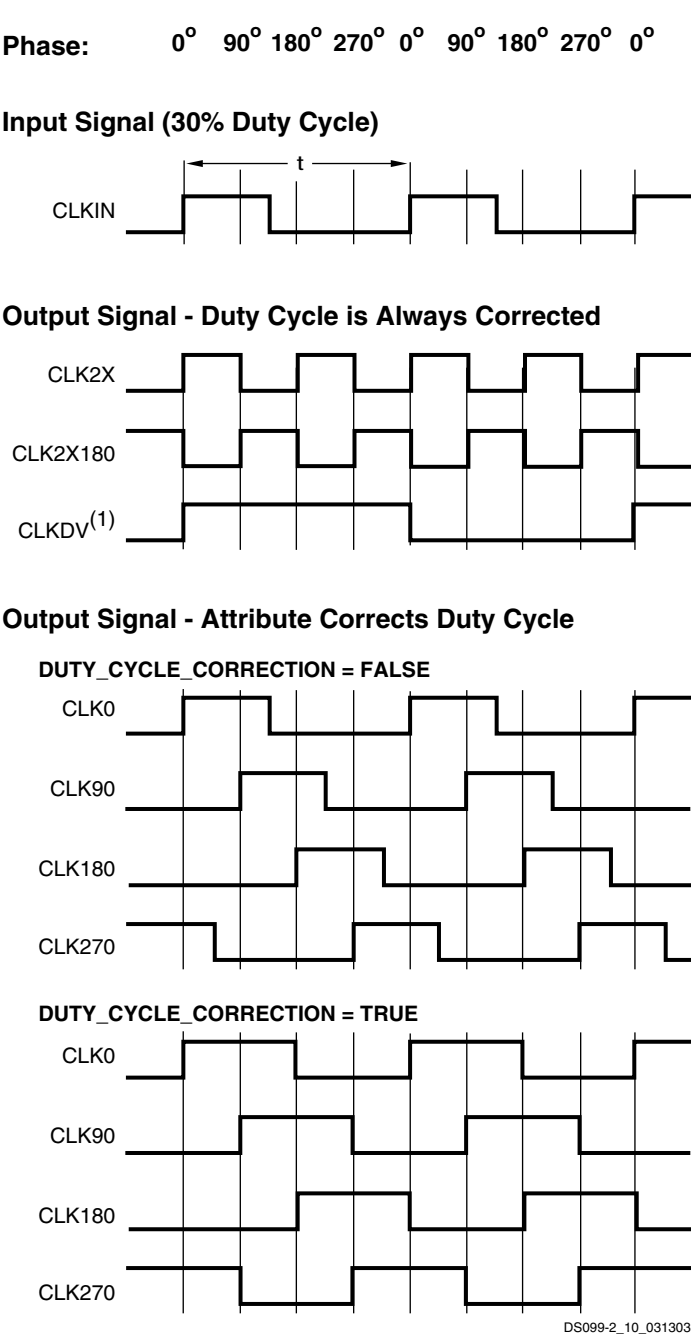
Basic Frequency Synthesis Outputs of the DLL Component

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 various values as described in Table 13. The basic frequency synthesis outputs are described in Table 12. Their relative timing in the Low Frequency Mode is shown in Figure 16.

The CLK2X and CLK2X180 outputs are not available when operating in the High Frequency mode. (See the description of the DLL_FREQUENCY_MODE attribute in Table 14.)

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 16 compares the characteristics of the DLL's output signals to those of the CLKIN signal.



Notes:
1. The DLL attribute CLKDV_DIVIDE is set to 2.

Figure 16: **Characteristics of the DLL Clock Outputs**

1. The CLK2X output generates a 25% duty cycle clock at the same frequency as the CLKIN signal until the DLL has achieved lock.
2. The duty cycle of the CLKDV outputs may differ somewhat from 50% (i.e., the signal will be 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 15.

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 14.

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 14.
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 would be 5/3 that of the input clock signal.

DFS Frequency Modes

The DFS supports two operating modes, High Frequency and Low Frequency, with each specified over a different clock frequency range. The DFS_FREQUENCY_MODE attribute chooses between the two modes. When the attribute is set to LOW, the Low Frequency mode permits the two DFS outputs to operate over a low-to-moderate frequency range. When the attribute is set to HIGH, the High Frequency mode allows both these outputs to operate at the highest possible frequencies.

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 will 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 will occur 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 will result in the multiplication of clock frequency by 3/2, the latter value-pair will enable the DLL to lock more quickly.

Table 14: **DFS Attributes**

Attribute	Description	Values
DFS_FREQUENCY_MODE	Chooses between High Frequency and Low Frequency modes	Low, High
CLKFX_MULTIPLY	Frequency multiplier constant	Integer from 2 to 32
CLKFX_DIVIDE	Frequency divisor constant	Integer from 1 to 32

Table 15: **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 15a and Figure 15c, respectively. This is similar to what has already been described for the DLL component. See the **DLL Clock Output and Feedback Connections**, page 22 section.

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 1/256 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 16, 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 17a 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 component 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

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. 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:

$$T_{PS} = (\text{PHASE\_SHIFT}/256)*T_{CLKIN} \quad \textbf{(4)}$$

Both the Fixed Phase and Variable Phase operating modes employ this calculation. If the PHASE_SHIFT value is zero, then CLKFB and CLKIN will be in phase, the same as when the PS component is disabled. When the PHASE_SHIFT value is positive, the CLKFB signal will be shifted later in time with respect to CLKIN. If the attribute value is negative, the CLKFB signal will be 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 17b 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.

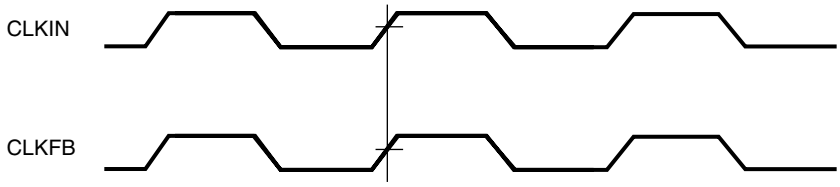
Table 16: **PS Attributes**

Attribute	Description	Values
CLKOUT_PHASE_SHIFT	Disables 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(1)

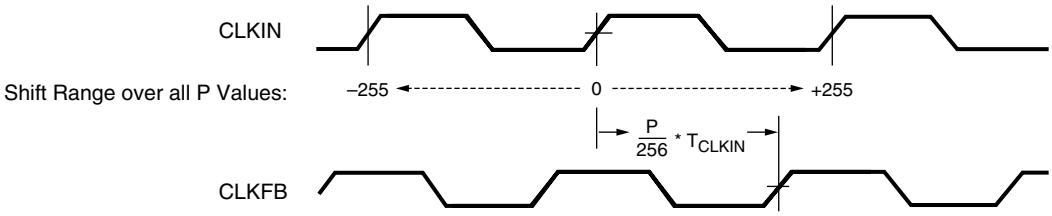
Notes:

1. The practical range of values will be less when T_{CLKIN} > FINE_SHIFT_RANGE in the Fixed Phase mode, also when T_{CLKIN} > (FINE_SHIFT_RANGE)/2 in the Variable Phase mode. the FINE_SHIFT_RANGE represents the sum total delay of all taps.

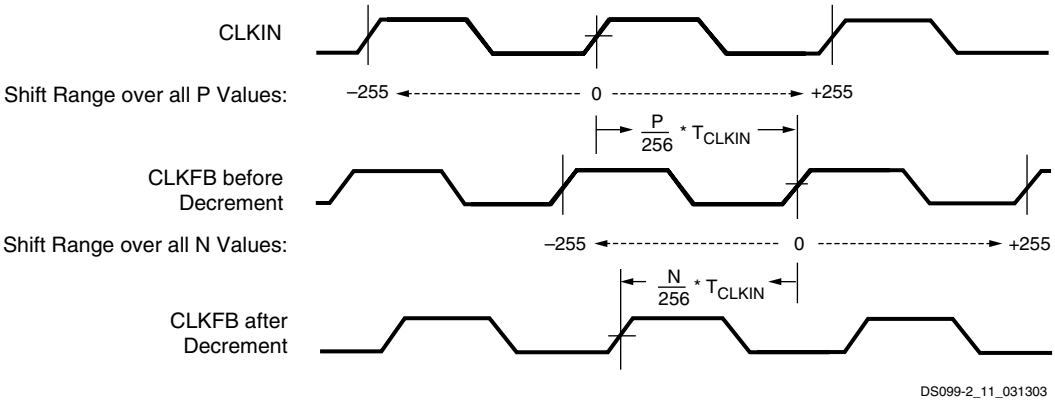
a. CLKOUT_PHASE_SHIFT = NONE



b. CLKOUT_PHASE_SHIFT = FIXED



c. CLKOUT_PHASE_SHIFT = VARIABLE



Notes:

1. P represents the integer value ranging from –255 to +255 to which the PHASE_SHIFT attribute is assigned.
2. N is an integer value ranging from –255 to +255 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}

   A positive value for N indicates a net increment; a negative value indicates a net decrement.

Figure 17: **Phase Shifter Waveforms**

Table 17: **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

The Variable Phase Mode

The "Variable Phase" mode dynamically adjusts the fine phase shift over time using three inputs to the PS component, namely PSEN, PSCLK and PSINCDEC, as defined in Table 17.

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 1/256 of a CLKIN cycle to T_{PS}. Similarly, for each enabled PSCLK cycle that PSINCDEC is Low, the PS component subtracts 1/256 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 17c 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 18.

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, RST must be tied to GND.

The eight bits of the STATUS bus are defined in Table 19.

Table 18: **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 19: **DCM STATUS Bus**

Bit	Name	Description
0	Phase Shift Overflow	A value of 1 indicates a phase shift overflow when one of two conditions occur: • Incrementing (or decrementing) TPS beyond 255/256 of a CLKIN cycle. • The DLL is producing its maximum possible phase shift (i.e., all delay taps are active).(1)
1	CLKIN Activity	A value of 1 indicates that the CLKIN signal is not toggling. A value of 0 indicates toggling. This bit functions only when the CLKFB input is connected.(2)
2	Reserved	-
3	Reserved	-
4	Reserved	-
5	Reserved	-
6	Reserved	-
7	Reserved	-

Notes:
1. The DLL phase shift with all delay taps active is specified as the parameter FINE_SHIFT_RANGE.
2. If only the DFS clock outputs are used, but none of the DLL clock outputs, this bit will not go High when the CLKIN signal stops.

Table 20: **Status Attributes**

Attribute	Description	Values
STARTUP_WAIT	Delays transition from configuration to user mode until lock condition is achieved.	TRUE, FALSE

Stabilizing DCM Clocks Before User Mode

It is possible to delay the completion of device configuration until after the DLL has achieved a lock condition using the STARTUP_WAIT attribute described in Table 20. This option ensures that the FPGA does not enter user mode — i.e., begin functional operation — until all system clocks generated by the DCM are stable. In order to achieve the delay, it is necessary to set the attribute to TRUE as well as set the BitGen option LCK_cycle to one of the six cycles making up the Startup phase of configuration. The selected cycle defines the point at which configuration will halt until the LOCKED output goes High.

Global Clock Network

Spartan-3 devices have eight Global Clock inputs called GCLK0 - GCLK7. These inputs provide access to a low-capacitance, low-skew network that is well-suited to carrying high-frequency signals. The Spartan-3 clock network is shown in Figure 18. GCLK0 through GCLK3 are placed at the center of the die's bottom edge. GCLK4 through GCLK7 are placed at the center of the die's top edge. It is possible to route each of the eight Global Clock inputs to any CLB on the die.

Eight Global Clock Multiplexers (also called BUFGMUX elements) are provided that accept signals from Global Clock inputs and route them to the internal clock network as well as DCMs. Four BUFGMUX elements are placed at the center of the die's bottom edge, just above the GCLK0 - GCLK4 inputs. The remaining four BUFGMUX elements are placed at the center of the die's top edge, just below the GCLK4 - GCLK7 inputs.

Each BUFGMUX element is a 2-to-1 multiplexer that can receive signals from any of the four following sources:

1. One of the four Global Clock inputs on the same side of the die — top or bottom — as the BUFGMUX element in use.
2. Any of four nearby horizontal Double lines.
3. Any of four outputs from the DCM in the right-hand quadrant that is on the same side of the die as the BUFGMUX element in use.
4. Any of four outputs from the DCM in the left-hand quadrant that is on the same side of the die as the BUFGMUX element in use.

Each BUFGMUX can switch incoming clock signals to two possible destinations:

1. The vertical spine belonging to the same side of the die — top or bottom — as the BUFGMUX element in use. The two spines — top and bottom — each comprise four vertical clock lines, each running from one of the BUFGMUX elements on the same side towards the center of the die. At the center of the die, clock signals reach the eight-line horizontal spine, which spans the

width of the die. In turn, the horizontal spine branches out into a subsidiary clock interconnect that accesses the CLBs.

2. The clock input of either DCM on the same side of the die — top or bottom — as the BUFGMUX element in use.

A Global clock input is placed in a design using either a BUFGMUX element or the BUFG (Global Clock Buffer) element. For the purpose of minimizing the dynamic power dissipation of the clock network, the Xilinx development software automatically disables all clock line segments that a design does not use.

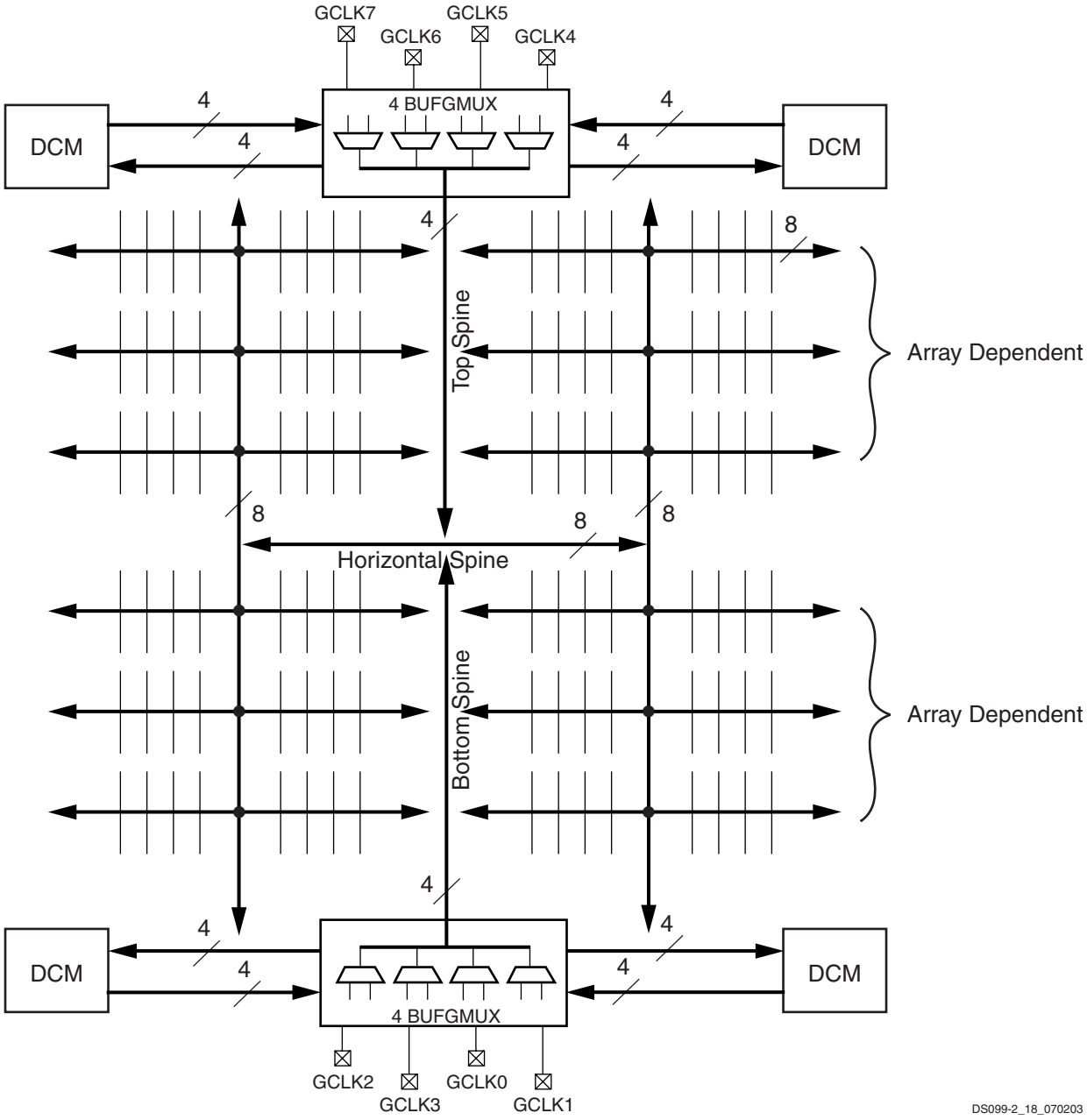


Figure 18: **Spartan-3 Clock Network (Top View)**

Interconnect

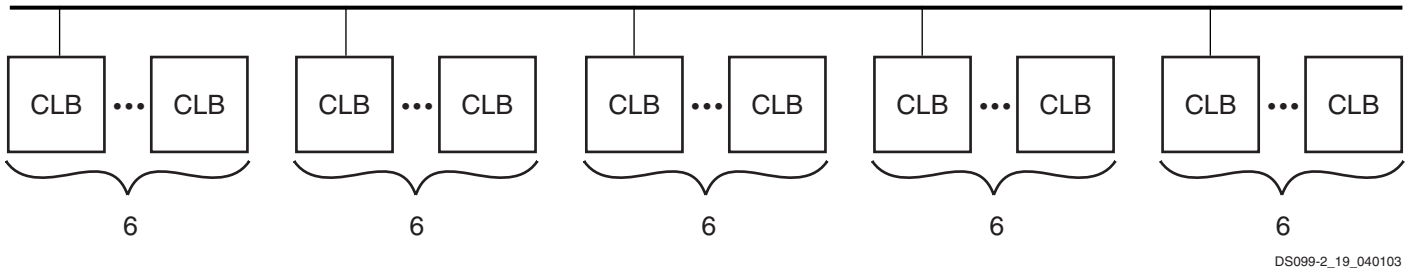
Interconnect (or routing) passes signals among the various functional elements of Spartan-3 devices. There are four kinds of interconnect: Long lines, Hex lines, Double lines, and Direct lines.

Long lines connect to one out of every six CLBs (see Figure 19a). Because of their low capacitance, these lines are well-suited for carrying high-frequency signals with minimal loading effects (e.g. skew). If all eight Global Clock Inputs are already committed and there remain additional clock signals to be assigned, Long lines serve as a good alternative.

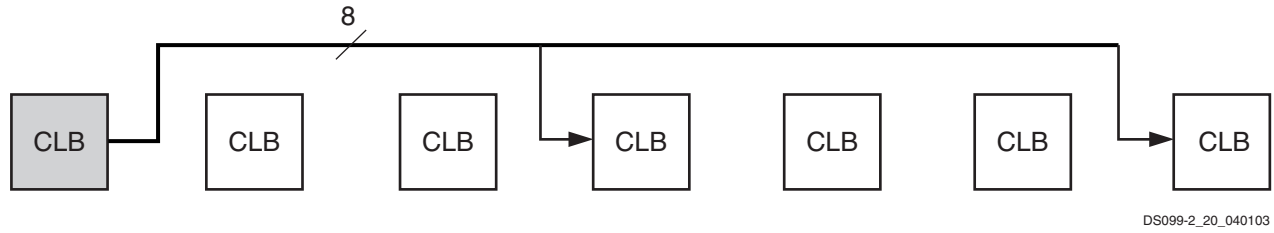
Hex lines connect one out of every three CLBs (see Figure 19b). These lines fall between Long lines and Double lines in terms of capability: Hex lines approach the high-frequency characteristics of Long lines at the same time, offering greater connectivity.

Double lines connect to every other CLB (see Figure 19c). Compared to the types of lines already discussed, Double lines provide a higher degree of flexibility when making connections.

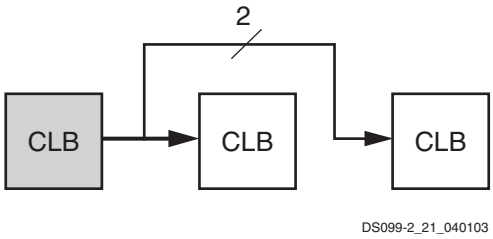
Direct lines afford any CLB direct access to neighboring CLBs (see Figure 19d). These lines are most often used to conduct a signal from a "source" CLB to a Double, Hex, or Long line and then from the longer interconnect back to a Direct line accessing a "destination" CLB.



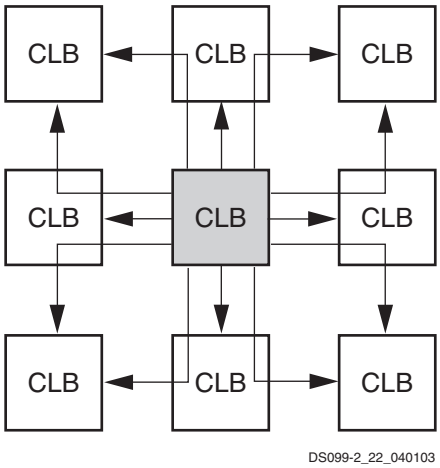
(a) Long Line



(b) Hex Line



(c) Double Line



(d) Direct Lines

Figure 19: **Types of Interconnect**

Configuration

Spartan-3 devices are configured by loading application specific configuration data into the internal configuration memory. Configuration is carried out using a subset of the device pins, some of which are "Dedicated" to one function only, while others, indicated by the term "Dual-Purpose", can be re-used as general-purpose User I/Os once configuration is complete.

Depending on the system design, several configuration modes are supported, selectable via mode pins. The mode pins M0, M1, and M2 are Dedicated pins. The mode pin settings are shown in Table 21.

Table 21: **Spartan-3 Configuration Mode Pin Settings**

Configuration Mode(1)	M0	M1	M2	Synchronizing Clock	Data Width	Serial DOUT(2)
Master Serial	0	0	0	CCLK Output	1	Yes
Slave Serial	1	1	1	CCLK Input	1	Yes
Master Parallel	1	1	0	CCLK Output	8	No
Slave Parallel	0	1	1	CCLK Input	8	No
JTAG	1	0	1	TCK Input	1	No

Notes:
1. The voltage levels on the M0, M1, and M2 pins select the configuration mode.
2. The daisy chain is possible only in the Serial modes when DOUT is used.

An additional pin, HSWAP_EN, is used in conjunction with the mode pins to select whether user I/O pins have pull-ups during configuration. By default, HSWAP_EN is tied High (internal pull-up) which shuts off the pull-ups on the user I/O pins during configuration. When HSWAP_EN is tied Low, user I/Os have pull-ups during configuration. Other Dedicated pins are CCLK (the configuration clock pin), DONE, PROG_B, and the boundary-scan pins: TDI, TDO, TMS, and TCK. Depending on the configuration mode chosen, CCLK can be an output generated by the FPGA, or an input accepting an externally generated clock.

A persist option is available which can be used to force the configuration pins to retain their configuration function even after device configuration is complete. If the persist option is not selected then the configuration pins with the exception of CCLK, PROG_B, and DONE can be used as user I/O in normal operation. The persist option does not apply to the boundary-scan related pins. The persist feature is valuable in applications that readback configuration data after entering the User mode.

Table 22 lists the total number of bits required to configure each FPGA as well as the PROMs suitable for storing those bits. See **DS123**: *Platform Flash In-System Programmable Configuration PROMs* data sheet for more information.

Table 22: **Spartan-3 Configuration Data**

Device	File Sizes	Xilinx Platform Flash PROM	
		Serial Configuration	Parallel Configuration
XC3S50	439,264	XCF01S	XCF08P
XC3S200	1,047,616	XCF01S	XCF08P
XC3S400	1,699,136	XCF02S	XCF08P
XC3S1000	3,223,488	XCF04S	XCF08P
XC3S1500	5,214,784	XCF08P	XCF08P
XC3S2000	7,673,024	XCF08P	XCF08P
XC3S4000	11,316,864	XCF16P	XCF16P
XC3S5000	13,271,936	XCF16P	XCF16P

The Standard Configuration Interface

Configuration signals belong to one of two different categories: Dedicated or Dual-Purpose. Which category determines which of the FPGA's power rails supplies the signal's driver and, thus, helps describe the electrical at the pin.

The Dedicated configuration pins include PROG_B, HSWAP_EN, TDI, TMS, TCK, TDO, CCLK, DONE, and M0-M2. These pins use the V_{CCAUX} lines for power.

The Dual-Purpose configuration pins comprise INIT_B, DOUT, BUSY, RDWR_B, CS_B, and DIN/D0-D7. Each of these pins, according to its bank placement, uses the V_{CCO} lines for either Bank 4 (VCCO_4) or Bank 5 (VCCO_5). All the signals used in the serial configuration modes rely on VCCO_4 power. Signals used in the parallel configuration modes and Readback require from VCCO_5 as well as from VCCO_4.

Both the Dedicated and Dual-Purpose signals described above constitute the configuration interface. In the standard case, this interface is 2.5V-LVCMOS-compatible. This means that 2.5V is applied to the V_{CCAUX}, VCCO_4, and VCCO_5 lines (this last in the parallel or Readback case only). One need only apply 2.5 Volts to these V_{CCO} lines from power-on to the end of configuration. Upon entering the User mode, it is possible to switch to supply voltage permitting signal swings other than 2.5V.

3.3V-Tolerant Configuration Interface

It is possible to achieve 3.3V-tolerance at the configuration interface simply by adding a few external resistors. This approach may prove useful when it is undesirable to switch the VCCO_4 and VCCO_5 voltages from 2.5V to 3.3V after configuration.

The 3.3V-tolerance is implemented as follows (a similar approach can be used for other supply voltage levels):

First, to power the Dual-Purpose configuration pins, apply 3.3V to the VCCO_4 and (as needed) the VCCO_5 lines. This scales the output voltages and input thresholds associated with these pins so that they become 3.3V-compatible.

Second, to power the Dedicated configuration pins, apply 2.5V to the V_{CCAUX} lines (the same as for the standard interface). In order to achieve 3.3V-tolerance, the Dedicated inputs will require series resistors that limit the incoming current to 10mA or less. The Dedicated outputs will need pull-up resistors to ensure adequate noise margin when the FPGA is driving a High logic level into another device's 3.3V receiver. Choose a power regulator or supply that can tolerate reverse current on the V_{CCAUX} lines.

Configuration Modes

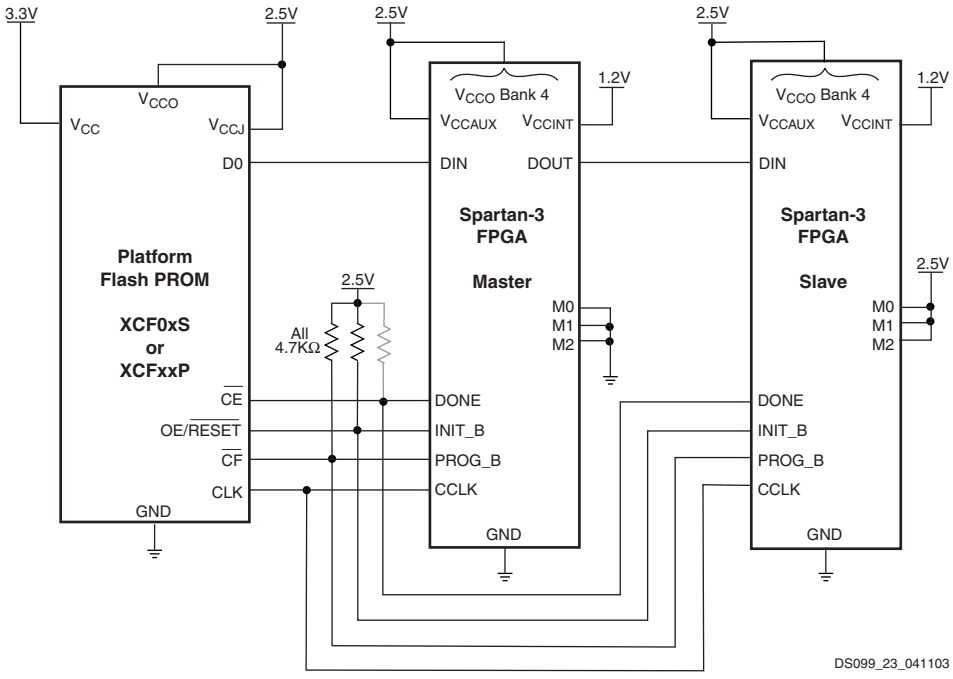
Spartan-3 supports the following five configuration modes:

- Slave Serial mode
- Master Serial mode
- Slave Parallel mode
- Master Parallel mode
- Boundary-Scan (JTAG) mode (IEEE 1532/IEEE 1149.1)

Slave Serial Mode

In Slave Serial mode, the FPGA receives configuration data in bit-serial form from a serial PROM or other serial source of configuration data. The FPGA on the far right of Figure 20 is set for the Slave Serial mode. The CCLK pin on the FPGA is an input in this mode. The serial bitstream must be set up at the DIN input pin a short time before each rising edge of the externally generated CCLK.

Multiple FPGAs can be daisy-chained for configuration from a single source. After a particular FPGA has been configured, the data for the next device is routed internally to the DOUT pin. The data on the DOUT pin changes on the falling edge of CCLK.



Notes:

1. There are two ways to use the DONE line. First, one may set the BitGen option DriveDone to "Yes" only for the last FPGA to be configured in the chain shown above (or for the single FPGA as may be the case). This enables the DONE pin to drive High; thus, no pull-up resistor is necessary. DriveDone is set to "No" for the remaining FPGAs in the chain. Second, DriveDone can be set to "No" for all FPGAs. Then all DONE lines are open-drain and require the pull-up resistor shown in grey. In most cases, a value between 3.3KΩ to 4.7KΩ is sufficient. However, when using DONE synchronously with a long chain of FPGAs, cumulative capacitance may necessitate lower resistor values (e.g. down to 330Ω) in order to ensure a rise time within one clock cycle.
2. For information on how to program the FPGA using 3.3V signals and power, see **3.3V-Tolerant Configuration Interface**.

Figure 20: **Connection Diagram for Master and Slave Serial Configuration**

Slave Serial mode is selected by applying <111> to the mode pins (M0, M1, and M2). A pull-up on the mode pins makes slave serial the default mode if the pins are left unconnected.

Master Serial Mode

In Master Serial mode, the CCLK pin is an output pin. The FPGA just to the right of the PROM in Figure 20 is set for Master Serial mode. It is the FPGA that drives the configuration clock on the CCLK pin to a Xilinx Serial PROM which in turn feeds bit-serial data to the DIN input. The FPGA accepts this data on each rising CCLK edge. After the FPGA has been loaded, the data for the next device in a daisy-chain is presented on the DOUT pin after the falling CCLK edge.

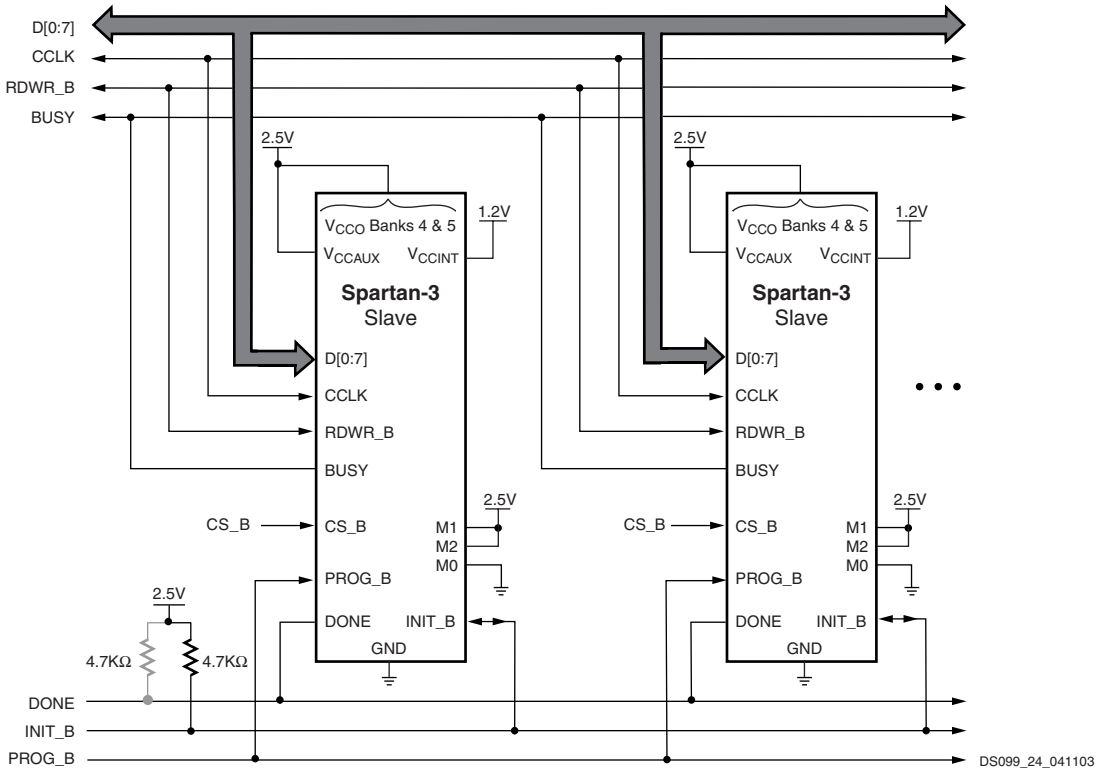
The interface is identical to slave serial except that an internal oscillator is used to generate the configuration clock (CCLK). A wide range of frequencies can be selected for CCLK which always starts at a default frequency of 6 MHz. Configuration bits then switch CCLK to a higher frequency for the remainder of the configuration.

Slave Parallel Mode

The Parallel modes support the fastest configuration. Byte-wide data is written into the FPGA with a BUSY flag controlling the flow of data. An external source provides 8-bit-wide data, CCLK, an active-Low Chip Select (CS_B) signal and an active-Low Write signal (RDWR_B). If BUSY is asserted (High) by the FPGA, the data must be held until BUSY goes Low. Data can also be read using the Slave Parallel mode. If RDWR_B is asserted, configuration data is read out of the FPGA as part of a readback operation.

After configuration, it is possible to use any of the Multipurpose pins (DIN/D0-D7, DOUT/BUSY, INITB, CS_B, and RDWR_B) as User I/Os. To do this, simply set the BitGen option *Persist* to *No* and assign the desired signals to multipurpose configuration pins using the Xilinx development software. Alternatively, it is possible to continue using the configuration port (e.g. all configuration pins taken together) when operating in the User mode. This is accomplished by setting the *Persist* option to *Yes*.

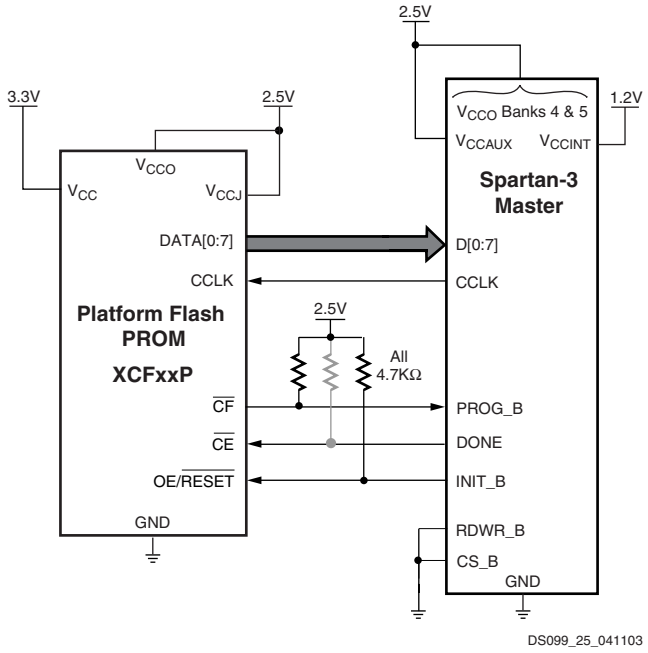
Multiple FPGAs can be configured using the Slave Parallel mode and can be made to start-up simultaneously. Figure 21 shows the device connections. To configure multiple devices in this way, wire the individual CCLK, Data, RDWR_B, and BUSY pins of all the devices in parallel. The individual devices are loaded separately by deasserting the CS_B pin of each device in turn and writing the appropriate data.



Notes:

1. There are two ways to use the DONE line. First, one may set the BitGen option DriveDone to "Yes" only for the last FPGA to be configured in the chain shown above (or for the single FPGA as may be the case). This enables the DONE pin to drive High; thus, no pull-up resistor is necessary. DriveDone is set to "No" for the remaining FPGAs in the chain. Second, DriveDone can be set to "No" for all FPGAs. Then all DONE lines are open-drain and require the pull-up resistor shown in grey. In most cases, a value between 3.3KΩ to 4.7KΩ is sufficient. However, when using DONE synchronously with a long chain of FPGAs, cumulative capacitance may necessitate lower resistor values (e.g. down to 330Ω) in order to ensure a rise time within one clock cycle.
2. If the FPGAs use different configuration data files, configure them in sequence by first asserting the CS_B of one FPGA then asserting the CS_B of the other FPGA.
3. For information on how to program the FPGA using 3.3V signals and power, see **3.3V-Tolerant Configuration Interface**.

Figure 21: **Connection Diagram for Slave Parallel Configuration**



Notes:

1. There are two ways to use the DONE line. First, one may set the BitGen option DriveDone to "Yes" only for the last FPGA to be configured in the chain shown above (or for the single FPGA as may be the case). This enables the DONE pin to drive High; thus, no pull-up resistor is necessary. DriveDone is set to "No" for the remaining FPGAs in the chain. Second, DriveDone can be set to "No" for all FPGAs. Then all DONE lines are open-drain and require the pull-up resistor shown in grey. In most cases, a value between 3.3KΩ to 4.7KΩ is sufficient. However, when using DONE synchronously with a long chain of FPGAs, cumulative capacitance may necessitate lower resistor values (e.g. down to 330Ω) in order to ensure a rise time within one clock cycle.

Figure 22: **Connection Diagram for Master Parallel Configuration**

Master Parallel Mode

In this mode, the device is configured byte-wide on a CCLK supplied by the FPGA. Timing is similar to the Slave Parallel mode except that CCLK is supplied by the FPGA. The device connections are shown in Figure 22.

Boundary-Scan (JTAG) Mode

In Boundary-Scan mode, dedicated pins are used for configuring the FPGA. The configuration is done entirely through the IEEE 1149.1 Test Access Port (TAP). FPGA configuration using the Boundary-Scan mode is compliant with the IEEE 1149.1-1993 standard and the new IEEE 1532 standard for In-System Configurable (ISC) devices.

Configuration through the boundary-scan port is always available, independent of the mode selection. Selecting the Boundary-Scan mode simply turns off the other modes.

Configuration Sequence

The configuration of Spartan-3 devices is a three-stage process that occurs after Power-On Reset or the assertion of PROG_B. POR occurs after the V_{CCINT}, V_{CCAUX}, and V_{CCO} Bank 4 supplies have reached their respective maximum input threshold levels (see **Table 6 in Module 3**). After POR, the three-stage process begins.

First, the configuration memory is cleared. Next, configuration data is loaded into the memory, and finally, the logic is activated by a start-up process. A flow diagram for the configuration sequence of the Serial and Parallel modes is shown in Figure 23. The flow diagram for the Boundary-Scan configuration sequence appears in Figure 24.

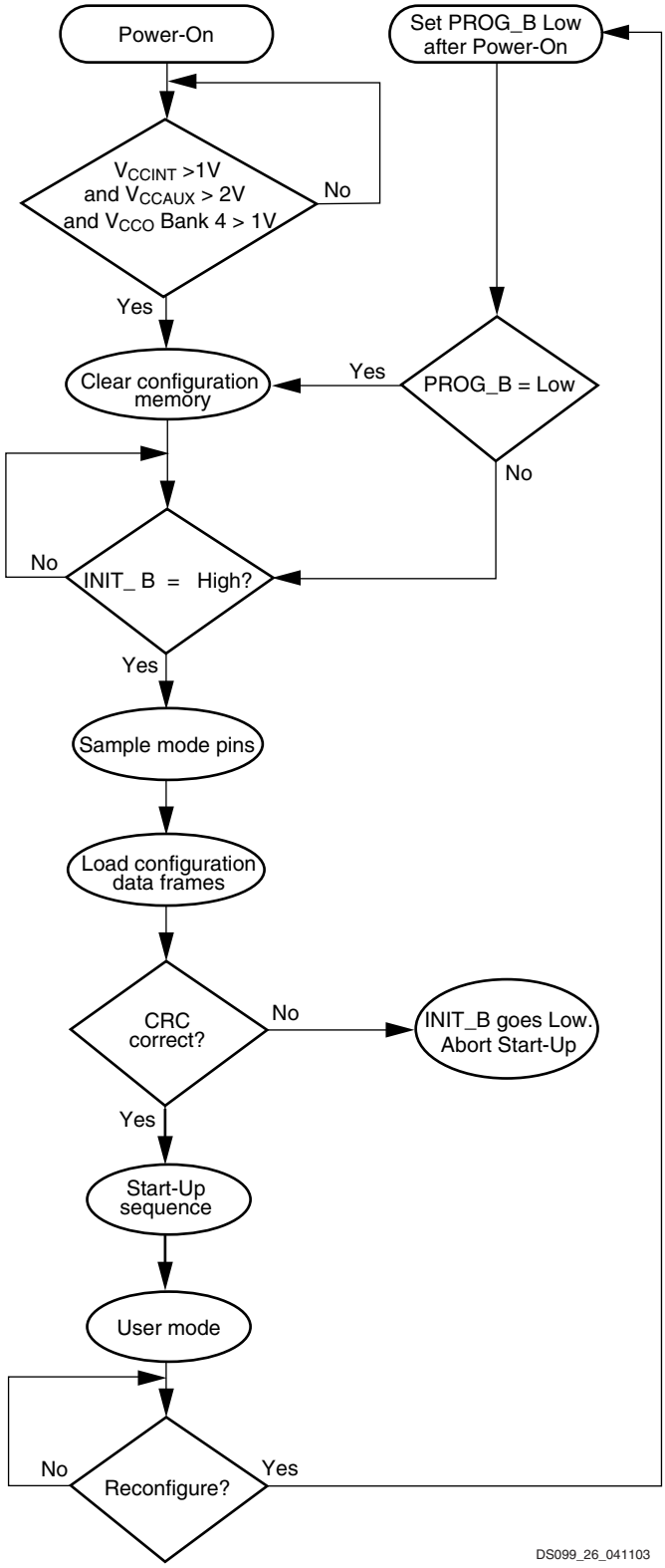


Figure 23: **Configuration Flow Diagram for the Serial and Parallel Modes**

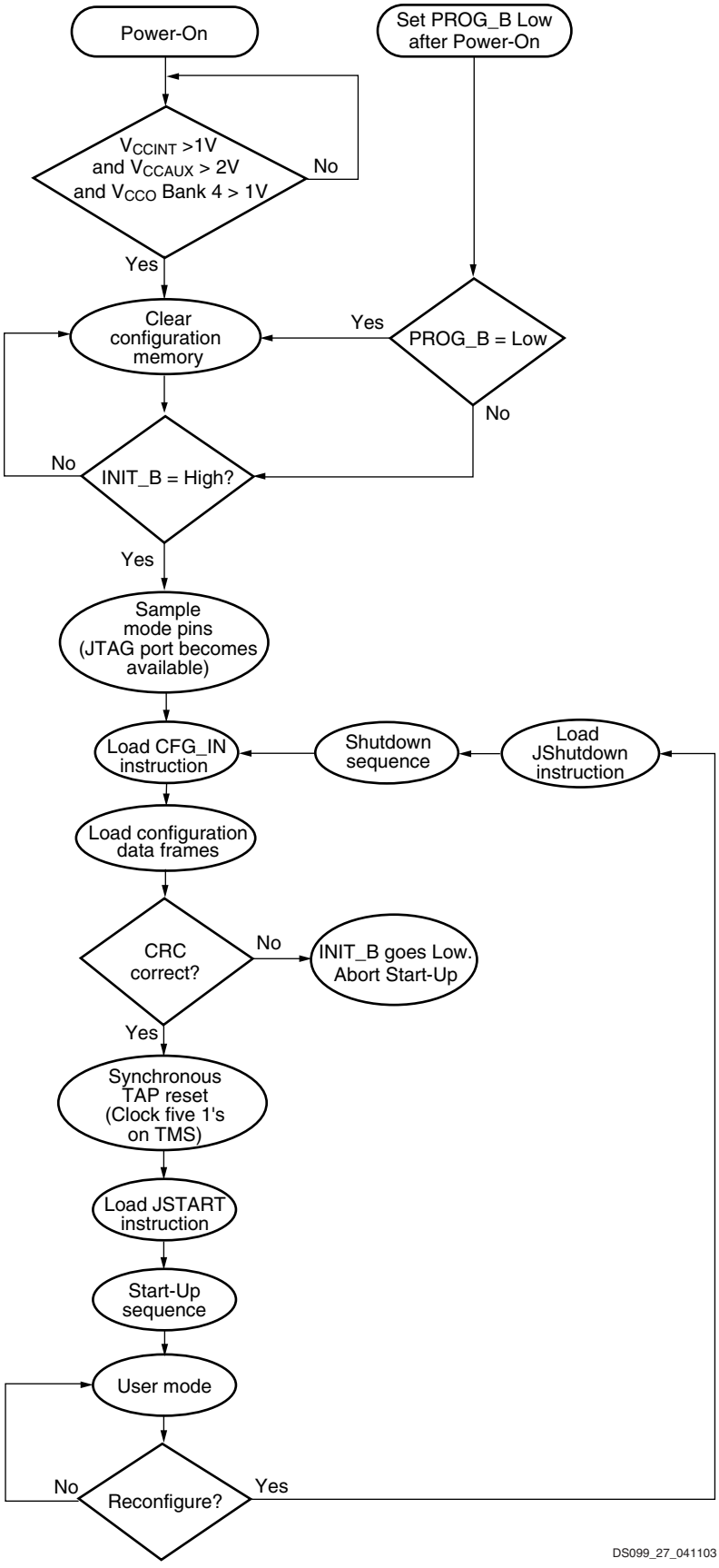
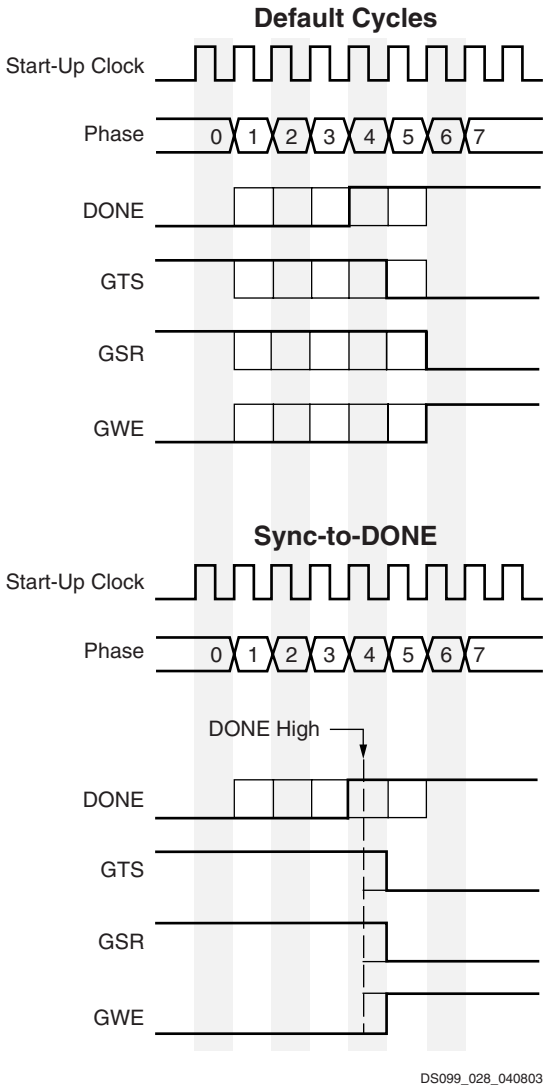


Figure 24: **Boundary-Scan Configuration Flow Diagram**

Configuration is automatically initiated after power-on unless it is delayed by the user. INIT_B is an open-drain line that the FPGA holds Low during the clearing of the configuration memory. Extending the time that the pin is Low causes the configuration sequencer to wait. Thus, configuration is delayed by preventing entry into the phase where data is loaded.

The configuration process can also be initiated by asserting the PROG_B pin. The end of the memory-clearing phase is signaled by the INIT_B pin going High. At this point, the configuration data is written to the FPGA. The FPGA holds the Global Set/Reset (GSR) signal active throughout configuration, keeping all flip-flops on the device in a reset state. The completion of the entire process is signaled by the DONE pin going High.



Notes:

1. The BitGen option StartupClk in the Xilinx development software selects the CCLK input, TCK input, or a user-designated global clock input (the GCLK0 - GCLK7 pins) for receiving the clock signal that synchronizes Start-Up.

Figure 25: **Default Start-Up Sequence**

The default start-up sequence, shown in Figure 25, serves as a transition to the User mode. The default start-up sequence is that one CCLK cycle after DONE goes High, the Global Three-State signal (GTS) is released. This permits device outputs to which signals have been assigned to become active. One CCLK cycle later, the Global Write Enable (GWE) signal is released. This permits the internal storage elements to begin changing state in response to the design logic and the user clock.

The relative timing of configuration events can be changed via the BitGen options in the Xilinx development software. In addition, the GTS and GWE events can be made dependent on the DONE pins of multiple devices all going High, forcing the devices to start synchronously. The sequence can also be paused at any stage, until lock has been achieved on any DCM.

Readback

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

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

Revision History

Date	Version No.	Description
04/11/03	1.0	Initial Xilinx release
05/19/03	1.1	Added Block RAM column, DCMs, and multipliers to XC3S50 descriptions.
07/11/03	1.2	Explained the configuration port *Persist* option in **Slave Parallel Mode** section. Updated Figure 2 and **Double-Data-Rate Transmission** section to indicate that DDR clocking for the XC3S50 is the same as that for all other Spartan-3 devices. Updated description of I/O voltage tolerance in **ESD Protection** section. In Table 6, changed input termination type for DCI version of the LVCMOS standard to *None*. Added additional flexibility for making DLL connections in Figure 15 and accompanying text. In the **Configuration** section, inserted an explanation of how to choose power supplies for the configuration interface, including guidelines for achieving 3.3V-tolerance.
08/24/04	1.3	Showed inversion of 3-state signal (Figure 1). Clarified description of pull-up and pull-down resistors (Table 2 and page 4). Added information on operating block RAM with multipliers to page 13. Corrected output buffer name in Figure 15. Corrected description of how DOUT is synchronized to CCLK (page 33).

The Spartan-3 Family Data Sheet

DS099-1, ***Spartan-3 FPGA Family: Introduction and Ordering Information*** (Module 1)

DS099-2, *Spartan-3 FPGA Family: Functional Description* (Module 2)

DS099-3, *Spartan-3 FPGA Family: DC and Switching Characteristics* (Module 3)

DS099-4, ***Spartan-3 FPGA Family: Pinout Descriptions*** (Module 4)