



XAPP462 (v1.0) July 9, 2003

Using Digital Clock Managers (DCMs) in Spartan-3 FPGAs

Summary

Digital Clock Managers (DCMs) provide advanced clocking capabilities to Spartan™-3 FPGA applications. DCMs optionally multiply or divide the incoming clock frequency to synthesize a new clock frequency. DCMs also eliminate clock skew, thereby improving system performance. Similarly, a DCM optionally phase shifts the clock output to delay the incoming clock by a fraction of the clock period. The DCMs integrate directly with the FPGA's global low-skew clock distribution network.

Introduction

DCMs integrate advanced clocking capabilities into the Spartan-3 global clock distribution network. Consequently, Spartan-3 DCMs solve a variety of common clocking issues, especially in high-performance, high frequency applications:

- **Multiply or Divide an Incoming Clock Frequency** or synthesize a completely new frequency by a mixture of clock multiplication and division.
- **Condition a Clock**, ensuring a clean output clock with a 50% duty cycle.
- **Phase Shift** a clock signal, either by a fixed fraction of a clock period or by precise increments.
- **Eliminate Clock Skew**, either within the device or to external components, to improve overall system performance and to eliminate clock distribution delays.
- **Mirror, Forward, or Rebuffer a Clock Signal**, often to deskew and convert the incoming clock signal to a different I/O standard—for example, forwarding and converting an incoming LVTTTL clock to LVDS.
- **Any or all the above functions, simultaneously.**

Table 1: Digital Clock Manager Features and Capabilities

Feature	Description	DCM Signals
Digital Clock Managers (DCMs) per device	<ul style="list-style-type: none"> • 4, except in XC3S50 • 2 in XC3S50 	All
Digital Frequency Synthesizer (DFS) Input Frequency Range*	1 MHz to ~326 MHz	CLKIN
Delay-Locked Loop (DLL) Input Frequency Range*	24 MHz to ~326 MHz	CLKIN
Clock Input Sources	<ul style="list-style-type: none"> • Global buffer input pad • Global buffer output • General-purpose I/O (no deskew) • Internal logic (no deskew) 	CLKIN
Frequency Synthesizer Output	Multiply CLKIN by the fraction (M/D) where $M=\{2..32\}$, $D=\{1..32\}$	<ul style="list-style-type: none"> • CLKFX • CLKFX180

© 2003 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and further disclaimers are as listed at <http://www.xilinx.com/legal.htm>. All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Table 1: Digital Clock Manager Features and Capabilities (Continued)

Feature	Description	DCM Signals
Clock Divider Output	Divide CLKIN by 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, or 16	CLKDV
Clock Doubler Output	Multiply CLKIN frequency by 2	<ul style="list-style-type: none"> • CLK2X • CLK2X180
Clock Conditioning, Duty-Cycle Correction	Always provided on most outputs. Optional on CLK0, CLK90, CLK180, CLK270. 50% duty cycle \pm 100 ps*	All
Quadrant Phase Shift Outputs	0° (no phase shift), 90° ($\frac{1}{4}$ period), 180° ($\frac{1}{2}$ period), 270° ($\frac{3}{4}$ period)	<ul style="list-style-type: none"> • CLK0 • CLK90 • CLK180 • CLK270
Half-period Phase Shift Outputs	Output pairs with 0° and 180° phase shift, ideal for DDR applications	<ul style="list-style-type: none"> • CLK0, CLK180 • CLK2X, CLK2X180 • CLKFX, CLKFX180
Dynamic or Fixed Phase Shift resolution	Down to 1/256 th of a clock period (or ~30 to 50 ps)*	All
Number of Clock Outputs to General-purpose Interconnect	Up to all 9	All
Number of Clock Outputs to Global Clock Network	Any 4 of 9	All
Number of Clock Outputs to Output Pins	Up to all 9	All

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

Document Overview

This application note covers an assortment of topics related to Digital Clock Managers, not all of which are relevant to every specific FPGA application.

The “[DCM Functional Overview](#)” section provides a brief introduction to the DCM and its functions. Similarly the “[DCM Primitive](#)” section describes all the connection ports and attributes or constraints associated with a DCM. Likewise the “[DCM Wizard](#)” and the “[VHDL and Verilog Instantiation](#)” sections demonstrate the various methods to specify a DCM design.

The “[DCM Clock Requirements](#)” and the “[Input and Output Clock Frequency Restrictions](#)” sections explain the frequency requirements on the DCM clock input and the various DCM clock outputs. Similarly, the “[Clock Jitter or Phase Noise](#)” section highlights the effect jitter has on output clock quality.

Finally, the “[Eliminating Clock Skew](#)”, “[Clock Conditioning](#)”, “[Phase Shifting – Delaying the Clock by a Fraction of a Period](#)”, “[Clock Multiplication, Clock Division, and Frequency Synthesis](#)”, and “[Clock Forwarding, Mirroring, Rebuffering](#)” sections illustrate various applications using the DCM block.

DCM Locations and Clock Distribution Network Interface

As shown in [Figure 1](#), most Spartan-3 FPGAs have four DCM blocks, except for the XC3S50, which has two DCM blocks. The DCM blocks are located at the top and bottom of the block RAM/multiplier columns along the left and right edges. The XC3S50 has two DCMs, along the top and bottom of the block RAM/multiplier column along the left edge of the device.

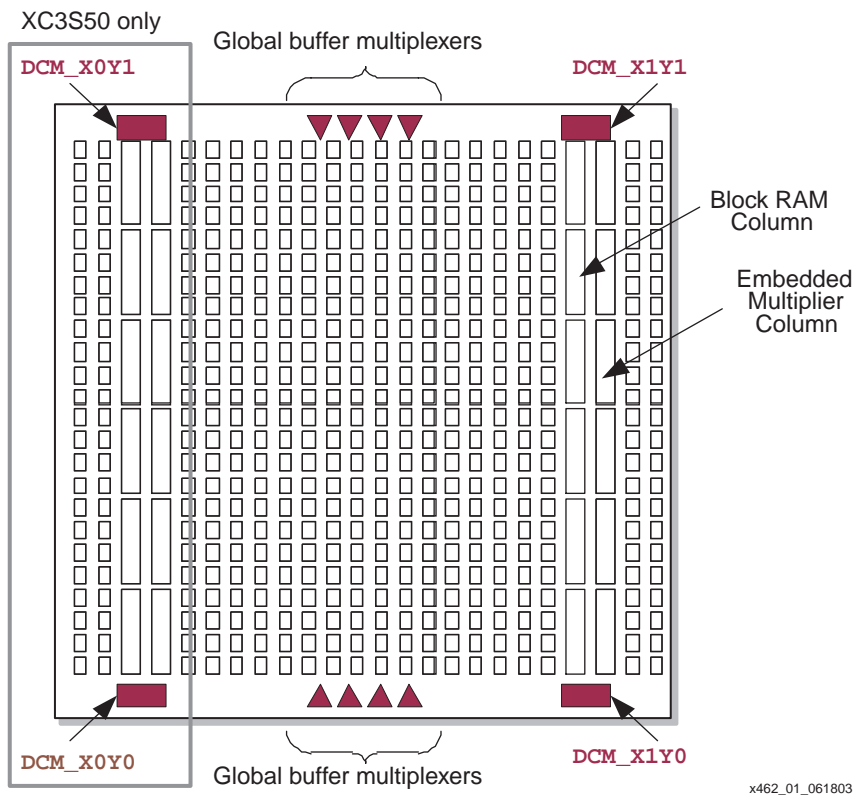


Figure 1: Location of the Four DCM Blocks on Spartan-3 FPGAs

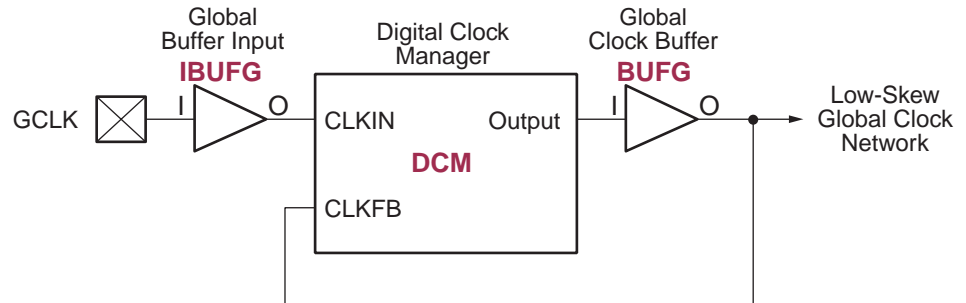
The DCM blocks have dedicated connections to the global buffer inputs and global buffer multiplexers on the same edge of the device, either top or bottom. As shown in [Figure 2](#), DCMs are an integral part of the FPGA's global clocking infrastructure. DCMs are an optional element in the clock distribution network and are available when required by the application. In [Figure 2a](#), a clock input feeds directly into the low-skew, high-fanout global clock network via a global input buffer and global clock buffer.

If the application requires some or all of the DCM's advanced clocking features, the DCM fits neatly between the global buffer input and the buffer itself, as shown in [Figure 2b](#).



x462_02a_062403

a. Global Buffer Inputs and Clock Buffers Drive a Low-Skew Global Network in the FPGA



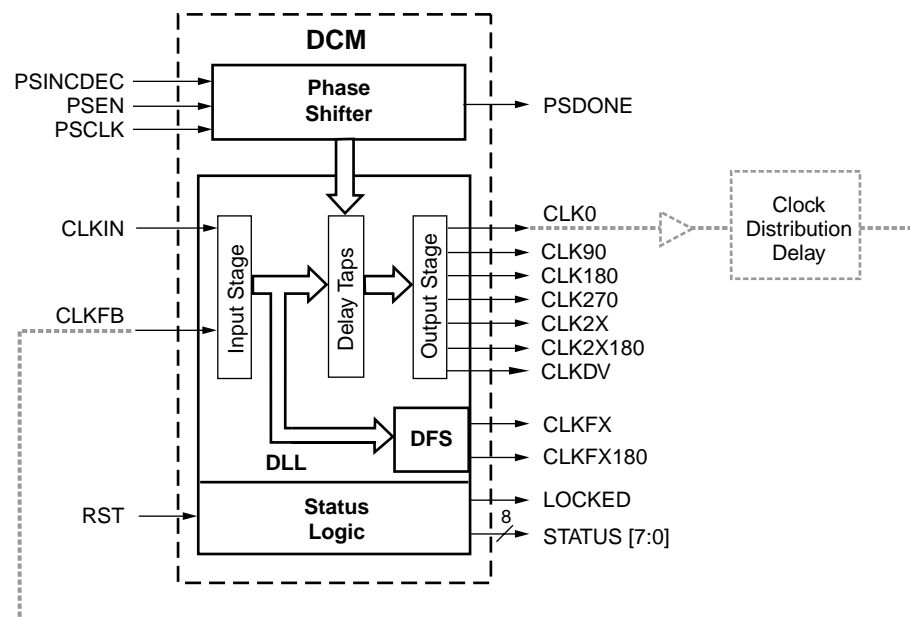
x462_02b_062403

b. A Digital Clock Manager (DCM) Inserts Directly into the Global Clock Path

Figure 2: DCMs are an Integral Part of the FPGA's Global Clock Network

DCM Functional Overview

The single entity called a Digital Clock Manager (DCM) actually consists of four distinct functional units as depicted in Figure 3 and described below. These units operate independently or in tandem.



DS099-2_07_040103

Figure 3: DCM Functional Block Diagram

Delay-Locked Loop (DLL)

The Delay-Locked Loop (DLL) unit provides an on-chip digital deskew circuit that generates zero-propagation-delay clock output signals. The deskew circuit compensates for the delay on the routing network by monitoring an output clock, either the CLK0 or the CLK2X. The DLL unit effectively eliminates the delay from the external clock input port to the individual clock loads within the device. The well-buffered global network minimizes the clock skew on the network caused by loading differences.

The input signals to the DLL unit are CLKIN and CLKFB. The output signals from the DLL are CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, and CLKDV.

The DLL unit generates the outputs for the [Clock Doubler \(CLK2X, CLK2X180\)](#), the [Clock Divider \(CLKDV\)](#) and the [Quadrant Phase Shifted Outputs](#) functions.

Digital Frequency Synthesizer (DFS)

The Digital Frequency Synthesizer (DFS) provides a wide and flexible range of output frequencies based on the ratio of two user-defined integers, a Multiplier (CLKFX_MULTIPLY) and a Divisor (CLKFX_DIVIDE). The output frequency is derived from the input clock (CLKIN) by simultaneous frequency division and multiplication. This feature can be used with or without the DLL feature of the DCM. If the DLL is not used, then there is no phase relationship between CLKIN and the DFS outputs.

The DFS unit generates the [Frequency Synthesizer \(CLKFX, CLKFX180\)](#) outputs.

Phase Shift (PS)

The Phase Shift (PS) unit controls the phase relations of the DCM's clock outputs to the CLKIN input.

The Phase Shift unit shifts the phase of all nine DCM clock output signals by a fixed fraction of the input clock period. The fixed phase shift value is set at design time and loaded into the DCM during FPGA configuration.

The Phase Shift unit also provides a digital interface for the FPGA application to dynamically advance or retard the current shift value by 1/256th of the clock period.

The input signals to the Phase Shift unit are PSINC DEN, PSEN, and PSCLK. The output signals are PSDONE and the STATUS[0] signal.

Status Logic

The Status Logic indicates the current state of the DCM via the [LOCKED](#) and [STATUS\[0\]](#), [STATUS\[1\]](#), and [STATUS\[2\]](#) output signals. The LOCKED output signal indicates whether the DCM outputs are in phase with the CLKIN input. The STATUS output signals indicate the state of the DLL and PS operations.

The [RST](#) input signal resets the DCM logic and returns it to its post-configuration state. Likewise, a reset forces the DCM to reacquire and lock to the CLKIN input.

DCM Primitive

The DCM primitive represents all the Digital Clock Manger functionality. The DCM primitive appears in [Figure 4](#), and the DCM's [Connection Ports](#) and [Attributes, Properties, or Constraints](#) are summarized below.

Symbol

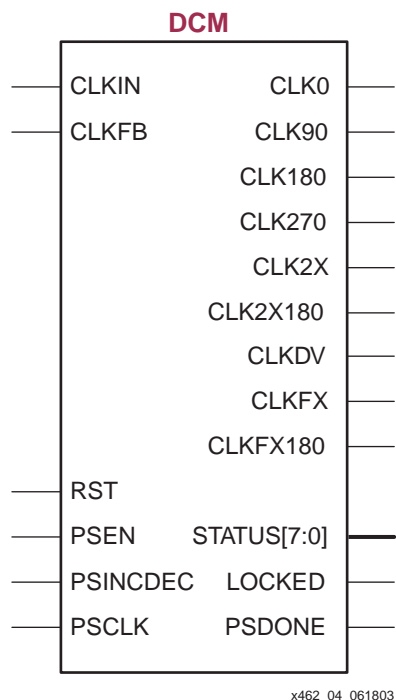


Figure 4: DCM Primitive

Connection Ports

[Table 3](#) lists the various connection ports to the Digital Clock Manager. Each port connection has a brief description, which includes the signal direction, and which DCM function units require the connection. [Table 2](#) provides the abbreviated name for each function unit used in [Table 3](#).

Table 2: Functional Unit Abbreviations for [Table 3](#)

Abbreviation	Functional Unit
DLL	Delay-Locked Loop
PS	Phase Shifter
DFS	Digital Frequency Synthesizer

Table 3: DCM Connection Ports

Port	Direction	Description	Functional Unit						
			DLL	PS	DFS				
CLKIN	Clock Input	Clock input to DCM. Always required. The CLKIN frequency and jitter must fall within the limits specified in the Spartan-3 Data Sheet. The frequency limits are controlled by the DLL_FREQUENCY_MODE and DFS_FREQUENCY_MODE attributes.	✓	✓	✓				
CLKFB	Input	Clock feedback input to DCM. The feedback input is required unless the Digital Frequency Synthesis outputs, CLKFX or CLKFX180 , are used stand-alone. The source of the CLKFB input must be the CLK0 or CLK2X output from the DCM and the CLK_FEEDBACK must be set to 1X or 2X accordingly. The feedback point ideally includes the delay added by the clock distribution network, either internally or externally. See “Feedback from a Reliable Source.”	✓	Optional	✓				
RST	Input	Asynchronous reset input. Resets the DCM logic to its post-configuration state. Causes DCM reacquire and relock to the CLKIN input. Invertible within DCM block. Non-inverted behavior shown below. See “RST Input Behavior.”	✓	✓	✓				
		<table border="1"> <tr> <td>0</td> <td>No effect.</td> </tr> <tr> <td>1</td> <td>Reset DCM block. Hold RST pulse High for at least 2 ns.</td> </tr> </table>	0	No effect.	1	Reset DCM block. Hold RST pulse High for at least 2 ns.			
0	No effect.								
1	Reset DCM block. Hold RST pulse High for at least 2 ns.								
PSEN	Input	Dynamic phase shift enable. Invertible within DCM block. Non-inverted behavior shown below. See “Dynamic Fine Phase Shifting.”		✓					
		<table border="1"> <tr> <td>0</td> <td>Disable dynamic phase shifter. Ignore inputs to phase shifter.</td> </tr> <tr> <td>1</td> <td>Enable dynamic phase shifter operations on next rising PSClk clock edge.</td> </tr> </table>	0	Disable dynamic phase shifter. Ignore inputs to phase shifter.	1	Enable dynamic phase shifter operations on next rising PSClk clock edge.			
0	Disable dynamic phase shifter. Ignore inputs to phase shifter.								
1	Enable dynamic phase shifter operations on next rising PSClk clock edge.								
PSINCDEC	Input	Increment/decrement dynamic phase shift. Invertible within DCM block. Non-inverted behavior shown below. See “Dynamic Fine Phase Shifting.”		✓					
		<table border="1"> <tr> <td>0</td> <td>Increment phase shift value on next enabled, rising PSClk clock edge.</td> </tr> <tr> <td>1</td> <td>Decrement phase shift value on next enabled, rising PSClk clock edge.</td> </tr> </table>	0	Increment phase shift value on next enabled, rising PSClk clock edge.	1	Decrement phase shift value on next enabled, rising PSClk clock edge.			
0	Increment phase shift value on next enabled, rising PSClk clock edge.								
1	Decrement phase shift value on next enabled, rising PSClk clock edge.								
PSClk	Clock Input	Clock input to dynamic phase shifter, clocked on rising edge. Invertible within DCM block. The frequency limits are controlled by the DLL_FREQUENCY_MODE attribute. See “Dynamic Fine Phase Shifting.”		✓					
CLK0	Clock Output	Same frequency as CLKIN, 0° phase shift (i.e., not phase shifted). Conditioned to 50% duty cycle when DUTY_CYCLE_CORRECTION attribute is TRUE. Either CLK0 or CLK2X is required as a feedback source for DLL functions. See “Half-Period Phase Shifted Outputs,” and “Quadrant Phase Shifted Outputs.”	✓						

Table 3: DCM Connection Ports (Continued)

Port	Direction	Description	Functional Unit		
			DLL	PS	DFS
CLK90	Clock Output	Same frequency as CLKIN, 90° phase shifted (quarter period). Not available if DLL_FREQUENCY_MODE attribute set to HIGH. Conditioned to 50% duty cycle when DUTY_CYCLE_CORRECTION attribute is TRUE. See “ Quadrant Phase Shifted Outputs. ”	✓		
CLK180	Clock Output	Same frequency as CLKIN, 180° phase shifted (half period). Conditioned to 50% duty cycle when DUTY_CYCLE_CORRECTION attribute is TRUE. See “ Half-Period Phase Shifted Outputs, ” and “ Quadrant Phase Shifted Outputs. ”	✓		
CLK270	Clock Output	Same frequency as CLKIN, 270° phase shifted (three-quarters period). Not available if DLL_FREQUENCY_MODE attribute set to HIGH. Conditioned to 50% duty cycle when DUTY_CYCLE_CORRECTION attribute is TRUE. See “ Quadrant Phase Shifted Outputs. ”	✓		
CLK2X	Clock Output	Double-frequency clock output, 0° phase shift. Not available if DLL_FREQUENCY_MODE attribute set to HIGH. When available, the CLK2X output is always 50% duty cycle. Either CLK0 or CLK2X is required as a feedback source for DLL functions. Clock Doubler (CLK2X, CLK2X180) output. See “ Half-Period Phase Shifted Outputs. ”	✓		
CLK2X180	Clock Output	Double-frequency clock output, 180° phase shifted. Not available if DLL_FREQUENCY_MODE attribute set to HIGH. When available, the CLK2X180 output is always 50% duty cycle. Clock Doubler (CLK2X, CLK2X180) output. See “ Half-Period Phase Shifted Outputs. ”	✓		
CLKDV	Clock Output	Divided clock output, controlled by the CLKDV_DIVIDE attribute. The CLKDV output has a 50% duty cycle unless the DLL_FREQUENCY_MODE attribute is HIGH and the CLKDV_DIVIDE attribute is a non-integer value. Locking time is longer when CLKDV_DIVIDE is non-integer value. Clock Divider (CLKDV) output. $F_{CLKDV} = \frac{F_{CLKIN}}{CLKDV_DIVIDE}$	✓		
CLKFX	Clock Output	Synthesized clock output, controlled by the CLKFX_MULTIPLY and CLKFX_DIVIDE attributes. Always has 50% duty cycle. If the CLKFX or CLKFX180 clock outputs are used stand-alone, then no clock feedback is required. See “ Frequency Synthesizer (CLKFX, CLKFX180), ” and “ Half-Period Phase Shifted Outputs. ” $F_{CLKFX} = F_{CLKIN} \cdot \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$			✓
CLKFX180	Clock Output	Synthesized clock output CLKFX, phase shifted by 180° (appears to be inverted version of CLKFX). Always has 50% duty cycle. If only CLKFX or CLKFX180 clock outputs are used on the DCM, then no feedback loop is required. See “ Frequency Synthesizer (CLKFX, CLKFX180), ” and “ Half-Period Phase Shifted Outputs. ”			✓

Table 3: DCM Connection Ports (Continued)

Port	Direction	Description	Functional Unit								
			DLL	PS	DFS						
STATUS[0]	Output	<p>Dynamic Phase Shift Overflow. Control output for Dynamic Fine Phase Shifting. The dynamic phase shifter has reached its minimum or maximum limit value. The limit value is either ± 255 or a lesser value if the phase shifter reached the end of the delay line. See “Dynamic Fine Phase Shifting.”</p> <table border="1"> <tr> <td>0</td> <td>The Phase Shifter has not yet reached its limit value.</td> </tr> <tr> <td>1</td> <td>Phase Shifter has reached its limit value.</td> </tr> </table>	0	The Phase Shifter has not yet reached its limit value.	1	Phase Shifter has reached its limit value.		✓			
0	The Phase Shifter has not yet reached its limit value.										
1	Phase Shifter has reached its limit value.										
STATUS[1]	Output	<p>CLKIN Input Stopped Indicator. Available only when CLKFB feedback input is connected. Held in reset until LOCKED output is asserted. Requires at least one CLKIN cycle to become active. Never asserted if CLKIN never toggles.</p> <table border="1"> <tr> <td>0</td> <td>CLKIN input is toggling.</td> </tr> <tr> <td>1</td> <td>CLKIN input is not toggling.</td> </tr> </table>	0	CLKIN input is toggling.	1	CLKIN input is not toggling.	✓	✓	✓		
0	CLKIN input is toggling.										
1	CLKIN input is not toggling.										
STATUS[2]	Output	<p>CLKFX or CLKFX180 Output Stopped Indicator. See Frequency Synthesizer (CLKFX, CLKFX180).</p> <table border="1"> <tr> <td>0</td> <td>CLKFX and CLKFX180 outputs are toggling.</td> </tr> <tr> <td>1</td> <td>CLKFX and CLKFX180 outputs are not toggling, even though LOCKED output may still be High.</td> </tr> </table>	0	CLKFX and CLKFX180 outputs are toggling.	1	CLKFX and CLKFX180 outputs are not toggling, even though LOCKED output may still be High.			✓		
0	CLKFX and CLKFX180 outputs are toggling.										
1	CLKFX and CLKFX180 outputs are not toggling, even though LOCKED output may still be High.										
STATUS[7:3]	Output	Reserved									
LOCKED	Output	<p>All DCM features have locked onto CLKIN frequency. Clock outputs are now valid, assuming CLKIN is within specified limits (as described in “DCM Clock Requirements”). See “Frequency Synthesizer (CLKFX, CLKFX180).”</p> <table border="1"> <tr> <td>0</td> <td>DCM is attempting to lock onto CLKIN frequency. DCM clock outputs are not valid.</td> </tr> <tr> <td>1</td> <td>DCM is locked onto CLKIN frequency. DCM clock outputs are valid.</td> </tr> <tr> <td>1-to-0</td> <td>DCM lost lock. Reset DCM.</td> </tr> </table>	0	DCM is attempting to lock onto CLKIN frequency. DCM clock outputs are not valid.	1	DCM is locked onto CLKIN frequency. DCM clock outputs are valid.	1-to-0	DCM lost lock. Reset DCM.	✓	✓	✓
0	DCM is attempting to lock onto CLKIN frequency. DCM clock outputs are not valid.										
1	DCM is locked onto CLKIN frequency. DCM clock outputs are valid.										
1-to-0	DCM lost lock. Reset DCM.										
PSDONE	Output	<p>Dynamic phase shift operation complete. See “Dynamic Fine Phase Shifting.”</p> <table border="1"> <tr> <td>0</td> <td>No phase shift operation active or phase shift operation in progress.</td> </tr> <tr> <td>1</td> <td>Requested phase shift operation is complete. Output High for one PSCLK cycle. Okay to provide next dynamic phase shift operation.</td> </tr> </table>	0	No phase shift operation active or phase shift operation in progress.	1	Requested phase shift operation is complete. Output High for one PSCLK cycle. Okay to provide next dynamic phase shift operation.		✓			
0	No phase shift operation active or phase shift operation in progress.										
1	Requested phase shift operation is complete. Output High for one PSCLK cycle. Okay to provide next dynamic phase shift operation.										

Attributes, Properties, or Constraints

Table 4 lists the various attributes for the Digital Clock Manager. All attributes are set at design time and programmed during configuration. Most, except for the Dynamic Fine Phase Shift function, cannot be changed by the FPGA application at run-time. To set an attribute, set <ATTRIBUTE>=<SETTING> as appropriate for the design entry tool.

Table 4: DCM Attributes

Attribute	Allowable Settings and Description						
DLL_FREQUENCY_MODE	<p>Specifies the allowable frequency range for the CLKIN input, the PSCLK input, and for the output clocks from the DCM's Delay-Locked Loop (DLL) unit. The DLL clock outputs include CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, CLKDV.</p> <table border="1"> <tr> <td>LOW</td> <td>Default. The DLL function unit operates in its low-frequency mode. All DLL-related outputs are available. The frequency for all clock inputs and outputs must fall within the low-frequency DLL limits specified in the Spartan-3 Data Sheet.</td> </tr> <tr> <td>HIGH</td> <td>The DLL function unit operates in its high-frequency mode. The Clock Doubler (CLK2X, CLK2X180) outputs are not available. The Quadrant Phase Shifted Outputs CLK90 and CLK270 are not available. The duty cycle for the CLKDV output is not 50% if the CLKDV_DIVIDE attribute is a non-integer. The frequency for all clock inputs and outputs must fall within the high-frequency DLL limits specified in the Spartan-3 Data Sheet.</td> </tr> </table>	LOW	Default. The DLL function unit operates in its low-frequency mode. All DLL-related outputs are available. The frequency for all clock inputs and outputs must fall within the low-frequency DLL limits specified in the Spartan-3 Data Sheet.	HIGH	The DLL function unit operates in its high-frequency mode. The Clock Doubler (CLK2X, CLK2X180) outputs are not available. The Quadrant Phase Shifted Outputs CLK90 and CLK270 are not available. The duty cycle for the CLKDV output is not 50% if the CLKDV_DIVIDE attribute is a non-integer. The frequency for all clock inputs and outputs must fall within the high-frequency DLL limits specified in the Spartan-3 Data Sheet.		
LOW	Default. The DLL function unit operates in its low-frequency mode. All DLL-related outputs are available. The frequency for all clock inputs and outputs must fall within the low-frequency DLL limits specified in the Spartan-3 Data Sheet.						
HIGH	The DLL function unit operates in its high-frequency mode. The Clock Doubler (CLK2X, CLK2X180) outputs are not available. The Quadrant Phase Shifted Outputs CLK90 and CLK270 are not available. The duty cycle for the CLKDV output is not 50% if the CLKDV_DIVIDE attribute is a non-integer. The frequency for all clock inputs and outputs must fall within the high-frequency DLL limits specified in the Spartan-3 Data Sheet.						
CLK_FEEDBACK	<p>Defines the frequency of the feedback clock.</p> <table border="1"> <tr> <td>1X</td> <td>Default. CLK0 feedback. Same frequency as CLKIN.</td> </tr> <tr> <td>2X</td> <td>CLK2X feedback. Double the frequency of CLKIN.</td> </tr> <tr> <td>None</td> <td>No feedback. Allowed if using only the CLKFX or CLKFX180 outputs.</td> </tr> </table>	1X	Default. CLK0 feedback. Same frequency as CLKIN.	2X	CLK2X feedback. Double the frequency of CLKIN.	None	No feedback. Allowed if using only the CLKFX or CLKFX180 outputs.
1X	Default. CLK0 feedback. Same frequency as CLKIN.						
2X	CLK2X feedback. Double the frequency of CLKIN.						
None	No feedback. Allowed if using only the CLKFX or CLKFX180 outputs.						
DUTY_CYCLE_CORRECTION	<p>Enables or disables the 50% duty-cycle correction for the CLK0, CLK90, CLK180, and CLK270 outputs from the DLL unit.</p> <table border="1"> <tr> <td>TRUE</td> <td>Default. Enable duty-cycle correction.</td> </tr> <tr> <td>FALSE</td> <td>Disable duty-cycle correction.</td> </tr> </table>	TRUE	Default. Enable duty-cycle correction.	FALSE	Disable duty-cycle correction.		
TRUE	Default. Enable duty-cycle correction.						
FALSE	Disable duty-cycle correction.						
CLKDV_DIVIDE	<p>Defines the frequency of the CLKDV output. Allowable values for CLKDV_DIVIDE include 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, 16.</p> $F_{CLKDV} = \frac{F_{CLKIN}}{CLKDV_DIVIDE}$ <p>Locking time is longer and there is more output jitter when CLKDV_DIVIDE is a non-integer value.</p>						
CLKFX_MULTIPLY	<p>Defines the multiplication factor for the frequency of the CLKFX and CLKFX180 outputs. Used in conjunction with CLKFX_DIVIDE attribute. Allowable values for CLKFX_MULTIPLY include integers ranging from 2 to 32. Default value is 4.</p> $F_{CLKFX} = F_{CLKIN} \cdot \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$						

Table 4: DCM Attributes (Continued)

Attribute	Allowable Settings and Description						
CLKFX_DIVIDE	<p>Defines the division factor for the frequency of the CLKFX and CLKFX180 outputs. Used in conjunction with CLKFX_MULTIPLY attribute. Allowable values for CLKFX_DIVIDE include integers ranging from 1 to 32. Default value is 1.</p> $F_{CLKFX} = F_{CLKIN} \cdot \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$						
PHASE_SHIFT	<p>The PHASE_SHIFT attribute is applicable only if the CLKOUT_PHASE_SHIFT attribute is set to FIXED or VARIABLE. Defines the rising-edge skew between CLKIN and all the DCM clock outputs at configuration and consequently phase shifts the DCM clock outputs.</p> <p>The skew or phase shift value is specified as an integer that represents a fraction of the clock period as expressed in the following equations. The integer value must range from -255 to 255. The default is 0. Actual allowable values depend on input clock frequency. The actual range is less when $T_{CLKIN} > FINE_SHIFT_RANGE$. The FINE_SHIFT_RANGE specification represents the total delay of all taps in the delay line. See "Fine Phase Shifting," for more information.</p>						
CLKOUT_PHASE_SHIFT	<p>Sets the phase shift mode. Together with the PHASE_SHIFT constraint, implements the Digital Phase Shifter (DPS) feature of the DCM. Affects all DCM clock outputs from both the DLL and DFS units. See "Fine Phase Shifting," for more information.</p> <table border="1" data-bbox="534 911 1523 1255"> <tbody> <tr> <td data-bbox="534 911 708 1024">NONE</td> <td data-bbox="708 911 1523 1024">Default. CLKIN and CLKFB are in phase (no skew) and phase relationship cannot be changed. Equivalent to FIXED setting with a PHASE_SHIFT value of 0.</td> </tr> <tr> <td data-bbox="534 1024 708 1108">FIXED</td> <td data-bbox="708 1024 1523 1108">Phase relationship is set at configuration by the PHASE_SHIFT attribute value and cannot be changed by the application.</td> </tr> <tr> <td data-bbox="534 1108 708 1255">VARIABLE</td> <td data-bbox="708 1108 1523 1255">Phase relationship is set at configuration by the PHASE_SHIFT attribute value but can be changed by the application using the dynamic phase shift controls, PSEN, PSCLK, PSINCDEC, and PSDONE.</td> </tr> </tbody> </table>	NONE	Default. CLKIN and CLKFB are in phase (no skew) and phase relationship cannot be changed. Equivalent to FIXED setting with a PHASE_SHIFT value of 0.	FIXED	Phase relationship is set at configuration by the PHASE_SHIFT attribute value and cannot be changed by the application.	VARIABLE	Phase relationship is set at configuration by the PHASE_SHIFT attribute value but can be changed by the application using the dynamic phase shift controls, PSEN, PSCLK, PSINCDEC, and PSDONE.
NONE	Default. CLKIN and CLKFB are in phase (no skew) and phase relationship cannot be changed. Equivalent to FIXED setting with a PHASE_SHIFT value of 0.						
FIXED	Phase relationship is set at configuration by the PHASE_SHIFT attribute value and cannot be changed by the application.						
VARIABLE	Phase relationship is set at configuration by the PHASE_SHIFT attribute value but can be changed by the application using the dynamic phase shift controls, PSEN, PSCLK, PSINCDEC, and PSDONE.						
DESKEW_ADJUST	<p>Controls the clock delay alignment between the FPGA clock input pin and the DCM output clocks. See "Skew Adjustment."</p> <table border="1" data-bbox="534 1367 1523 1535"> <tbody> <tr> <td data-bbox="534 1367 915 1451">SYSTEM_SYNCHRONOUS</td> <td data-bbox="915 1367 1523 1451">Default. All devices clocked by a common, system-wide clock source.</td> </tr> <tr> <td data-bbox="534 1451 915 1535">SOURCE_SYNCHRONOUS</td> <td data-bbox="915 1451 1523 1535">Clock is provided by the data source, i.e., source-synchronous applications.</td> </tr> </tbody> </table> <p>Do not use this setting to phase shift DCM clock outputs. Instead, use the CLKOUT_PHASE_SHIFT and PHASE_SHIFT constraints to achieve accurate phase shifting.</p>	SYSTEM_SYNCHRONOUS	Default. All devices clocked by a common, system-wide clock source.	SOURCE_SYNCHRONOUS	Clock is provided by the data source, i.e., source-synchronous applications.		
SYSTEM_SYNCHRONOUS	Default. All devices clocked by a common, system-wide clock source.						
SOURCE_SYNCHRONOUS	Clock is provided by the data source, i.e., source-synchronous applications.						

Table 4: DCM Attributes (Continued)

Attribute	Allowable Settings and Description				
DFS_FREQUENCY_MODE	<p>Specifies the allowable frequency range for the CLKFX and CLKFX180 output clocks from the DCM's Digital Frequency Synthesizer (DFS). If any DLL clock outputs are used, then the more restrictive DLL_FREQUENCY_MODE limits the CLKIN input frequency.</p> <table border="1" data-bbox="532 390 1511 747"> <tr> <td data-bbox="532 390 646 569">LOW</td> <td data-bbox="646 390 1511 569">Default. The DFS function unit operates in its low-frequency mode. The frequency for the CLKFX and CLKFX180 outputs must fall within the low-frequency DFS limits specified in the Spartan-3 Data Sheet. The frequency limits for the CLKIN input depend on if any DLL clock outputs are used.</td> </tr> <tr> <td data-bbox="532 569 646 747">HIGH</td> <td data-bbox="646 569 1511 747">The DFS function unit operates in its high-frequency mode. The frequency for the CLKFX and CLKFX180 outputs must fall within the high-frequency DFS limits specified in the Spartan-3 Data Sheet. The frequency limits for the CLKIN input depend on if any DLL clock outputs are used.</td> </tr> </table>	LOW	Default. The DFS function unit operates in its low-frequency mode. The frequency for the CLKFX and CLKFX180 outputs must fall within the low-frequency DFS limits specified in the Spartan-3 Data Sheet. The frequency limits for the CLKIN input depend on if any DLL clock outputs are used.	HIGH	The DFS function unit operates in its high-frequency mode. The frequency for the CLKFX and CLKFX180 outputs must fall within the high-frequency DFS limits specified in the Spartan-3 Data Sheet. The frequency limits for the CLKIN input depend on if any DLL clock outputs are used.
LOW	Default. The DFS function unit operates in its low-frequency mode. The frequency for the CLKFX and CLKFX180 outputs must fall within the low-frequency DFS limits specified in the Spartan-3 Data Sheet. The frequency limits for the CLKIN input depend on if any DLL clock outputs are used.				
HIGH	The DFS function unit operates in its high-frequency mode. The frequency for the CLKFX and CLKFX180 outputs must fall within the high-frequency DFS limits specified in the Spartan-3 Data Sheet. The frequency limits for the CLKIN input depend on if any DLL clock outputs are used.				
STARTUP_WAIT	<p>Controls whether the FPGA configuration signal DONE waits for the DCM to assert its LOCKED signal before going High.</p> <table border="1" data-bbox="532 863 1511 1121"> <tr> <td data-bbox="532 863 646 947">FALSE</td> <td data-bbox="646 863 1511 947">Default. DONE asserted at end of configuration without waiting for DCM to assert LOCKED.</td> </tr> <tr> <td data-bbox="532 947 646 1121">TRUE</td> <td data-bbox="646 947 1511 1121">DONE signal does not go High until the LOCKED signal goes HIGH on the associated DCM. STARTUP_WAIT does not prevent LOCKED from going High. The FPGA startup sequence must also be modified to insert a LCK (lock) cycle before the postponed cycle (see "Bitstream Generation Settings"). Either the DONE cycle or GWE cycle are typical choices.</td> </tr> </table> <p>If more than one DCM is so configured, the FPGA waits until all DCMs are locked.</p>	FALSE	Default. DONE asserted at end of configuration without waiting for DCM to assert LOCKED.	TRUE	DONE signal does not go High until the LOCKED signal goes HIGH on the associated DCM. STARTUP_WAIT does not prevent LOCKED from going High. The FPGA startup sequence must also be modified to insert a LCK (lock) cycle before the postponed cycle (see " Bitstream Generation Settings "). Either the DONE cycle or GWE cycle are typical choices.
FALSE	Default. DONE asserted at end of configuration without waiting for DCM to assert LOCKED.				
TRUE	DONE signal does not go High until the LOCKED signal goes HIGH on the associated DCM. STARTUP_WAIT does not prevent LOCKED from going High. The FPGA startup sequence must also be modified to insert a LCK (lock) cycle before the postponed cycle (see " Bitstream Generation Settings "). Either the DONE cycle or GWE cycle are typical choices.				
CLKIN_DIVIDE_BY_2	<p>Optionally divides the CLKIN in half before entering DCM block. In some applications, reduces the input clock frequency to within acceptable limits.</p> <table border="1" data-bbox="532 1287 1511 1514"> <tr> <td data-bbox="532 1287 646 1339">FALSE</td> <td data-bbox="646 1287 1511 1339">Default. CLKIN input directly feeds the DCM block.</td> </tr> <tr> <td data-bbox="532 1339 646 1514">TRUE</td> <td data-bbox="646 1339 1511 1514">Divides CLKIN frequency in half and provides roughly a 50% duty cycle clock before entering the DCM block. Helpful with high-frequency clocks to meet the DCM input clock frequency or duty-cycle requirements. Divides clock frequency in half when determining operating frequency modes and calculating phase shift limits.</td> </tr> </table>	FALSE	Default. CLKIN input directly feeds the DCM block.	TRUE	Divides CLKIN frequency in half and provides roughly a 50% duty cycle clock before entering the DCM block. Helpful with high-frequency clocks to meet the DCM input clock frequency or duty-cycle requirements. Divides clock frequency in half when determining operating frequency modes and calculating phase shift limits.
FALSE	Default. CLKIN input directly feeds the DCM block.				
TRUE	Divides CLKIN frequency in half and provides roughly a 50% duty cycle clock before entering the DCM block. Helpful with high-frequency clocks to meet the DCM input clock frequency or duty-cycle requirements. Divides clock frequency in half when determining operating frequency modes and calculating phase shift limits.				

Table 4: DCM Attributes (Continued)

Attribute	Allowable Settings and Description								
FACTORY_JF	<p>Controls how often the DCM's DLL unit adjusts its tap settings. The FACTORY_JF setting affects the jitter characteristics of the DLL element.</p> <p>The settings are automatically adjusted based on the DLL_FREQUENCY_MODE attribute.</p> <table border="1"> <thead> <tr> <th>DLL_FREQUENCY_MODE</th> <th>FACTORY_JF</th> </tr> </thead> <tbody> <tr> <td>LOW</td> <td>0xC080</td> </tr> <tr> <td>HIGH</td> <td>0xF0F0</td> </tr> </tbody> </table> <p>Do not change the default values unless otherwise recommended (see “Adjusting FACTORY_JF Setting”).</p>	DLL_FREQUENCY_MODE	FACTORY_JF	LOW	0xC080	HIGH	0xF0F0		
DLL_FREQUENCY_MODE	FACTORY_JF								
LOW	0xC080								
HIGH	0xF0F0								
LOC	<p>Specifies the physical location of the DCM, as shown in Figure 1.</p> <table border="1"> <tbody> <tr> <td>DCM_X0Y0</td> <td>Lower left DCM.</td> </tr> <tr> <td>DCM_X1Y0</td> <td>Lower right DCM. Not available in XC3S50.</td> </tr> <tr> <td>DCM_X0Y1</td> <td>Upper left DCM.</td> </tr> <tr> <td>DCM_X1Y1</td> <td>Upper right DCM. Not available in XC3S50.</td> </tr> </tbody> </table>	DCM_X0Y0	Lower left DCM.	DCM_X1Y0	Lower right DCM. Not available in XC3S50.	DCM_X0Y1	Upper left DCM.	DCM_X1Y1	Upper right DCM. Not available in XC3S50.
DCM_X0Y0	Lower left DCM.								
DCM_X1Y0	Lower right DCM. Not available in XC3S50.								
DCM_X0Y1	Upper left DCM.								
DCM_X1Y1	Upper right DCM. Not available in XC3S50.								

Compatibility with Other Xilinx FPGA Families

The Spartan-3 Digital Clock Manager (DCM) is nearly functionally identical to the DCM units found in Virtex™-II and Virtex-II Pro FPGA families. However, the Spartan-3 DCM is the third-generation in DCM design with some improved capabilities over previous FPGA families. Specifically, Spartan-3 has improved immunity to noise on the V_{CCAUX} supply compared to Virtex-II and has more flexible phase shifting than both Virtex-II and Virtex-II Pro families. The DCMs on both Virtex-II and Virtex-II Pro families have a higher output frequency limit.

The Spartan-3 DCM is a significant enhancement over the Spartan-II/IIE Delay-Locked Loop (DLL) function. A Spartan-3 DCM provides all the capabilities of the Spartan-II/IIE DLL with new capabilities such as the Frequency Synthesizer and phase shifting functions. The Spartan-3 Frequency Synthesizer multiplies an input clock by up to a factor of 32. The Spartan-II/IIE DLL has limited frequency multiplication capabilities—namely, an input clock can be doubled. Similarly, the Spartan-3 DCM has a wider divider range compared to Spartan-II DLLs.

DCM Clock Requirements

The DCM is built for maximum flexibility, but there are certain requirements on clock frequency and clock stability, both frequency variation and clock jitter.

Input Clock Frequency Range

The DCM clock input frequency depends on whether the DLL functional unit, the DFS unit, or both are utilized in the application.

[Table 5](#) shows the clock input, [CLKIN](#), frequency range for the [Digital Frequency Synthesizer \(DFS\)](#) unit. The DFS unit, if used stand-alone, has a wider frequency range than the DLL unit. If the application uses both units, then the more restrictive DLL requirements apply. The table shows the data sheet specification name and an estimated value. The actual value depends on which speed grade is required for the design and the value specified in the data sheet takes precedence over the estimate.

Table 5: Digital Frequency Synthesizer (DFS) Unit Clock Input Frequency Requirements

Function	Minimum Frequency	Maximum Frequency
Digital Frequency Synthesizer (DFS)	CLKIN_FREQ_FX_MIN ~ 1.00 MHz*	CLKIN_FREQ_FX_MAX ~326 MHz*

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

Table 6 shows the clock input, CLKIN, frequency range for the **Delay-Locked Loop (DLL)** unit. The DLL frequency restrictions apply regardless if the DLL is used stand-alone or with the DFS unit. The table shows the frequency range when the DLL unit operates in either is low- or high-frequency mode. The mode is controlled by the **DLL_FREQUENCY_MODE** attribute. Likewise, the table shows the data sheet specification name and an estimated value. The actual value depends on which speed grade is required for the design and the value specified in the data sheet takes precedence over the estimate.

Table 6: Delay-Locked Loop (DLL) Unit Clock Input Frequency Requirements

Function	DLL Frequency Mode Attribute (DLL_FREQUENCY_MODE)			
	= LOW		= HIGH	
	Minimum Frequency	Maximum Frequency	Minimum Frequency	Maximum Frequency
Delay Locked Loop (DLL)	CLKIN_FREQ_DLL_LF_MIN ~ 24 MHz*	CLKIN_FREQ_DLL_LF_MAX ~ 180 MHz*	CLKIN_FREQ_DLL_HF_MIN ~ 48 MHz*	CLKIN_FREQ_DLL_HF_MAX ~326 MHz*

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

Output Clock Frequency Range

The various DCM output clocks also have a specified frequency range. See the “[Input and Output Clock Frequency Restrictions](#)” section for more information.

Input Clock and Clock Feedback Variation

As described later in the “[A Stable, Monotonic Clock Input](#)” section, the DCM expects a stable, monotonic clock input. However, for maximum flexibility, the DCM tolerates a certain amount of clock jitter on the CLKIN input and a reasonable amount of frequency variation on both the CLKIN input and the CLKFB clock feedback input.

There are two types of jitter tolerance on the CLKIN input. Cycle-to-cycle jitter indicates how much the CLKIN input period is allowed to change from one cycle to the next. The maximum allowable cycle-to-cycle change is shown in **Table 7**, including the data sheet specification name and an estimated value.

Table 7: Maximum Allowable Cycle-to-Cycle Jitter

Functional Unit	Frequency Mode	
	Low	High
Digital Frequency Synthesizer (DFS)	CLKIN_CYC_JITT_FX_LF ~ ±300 ps*	CLKIN_CYC_JITT_FX_HF ~ ±150 ps*
Delay Locked Loop (DLL)	CLKIN_CYC_JITT_DLL_LF ~ ±300 ps*	CLKIN_CYC_JITT_DLL_HF ~ ±150 ps*

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

The other applicable type of jitter is called period jitter. Period jitter indicates the maximum range of the period variation over millions of clock cycles. Cycle-to-cycle jitter shows the

change from one clock to the next while period jitter shows the total range of changes over time. The maximum allowable period jitter appears in [Table 8](#), including the data sheet specification name, and an estimated value.

Table 8: Maximum Allowable Period Jitter

Functional Unit	Frequency Mode	
	Low	High
Digital Frequency Synthesizer (DFS)	CLKIN_PER_JITT_FX_LF ~ ±1,000 ps* (±1 ns)	CLKIN_PER_JITT_FX_HF ~ ±1,000 ps* (±1 ns)
Delay Locked Loop (DLL)	CLKIN_PER_JITT_DLL_LF ~ ±1,000 ps* (±1 ns)	CLKIN_PER_JITT_DLL_HF ~ ±1,000 ps* (±1 ns)

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

Another source of stability for the DCM is the clock feedback path used by the DLL unit. The feedback path delay variance must also be within the limit shown in [Table 9](#). This limit only applies to an external feedback path as any on-chip variance is minimal when connected to a global clock line.

Table 9: External Feedback Path Delay Variation

Description	Specification
Maximum allowable variation in off-chip CLKFB feedback path	CLKFB_DELAY_VAR_EXT ~ ±1,000 ps* (±1 ns)

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

LOCKED Output Behavior

The DCM's LOCKED output indicates when all the enabled DCM functions have locked to the CLKIN input. When the DCM asserts LOCKED, the output clocks are valid for use within the FPGA application.

[Figure 5](#) shows the behavior of the LOCKED output. The LOCKED output is Low immediately after the FPGA finishes its configuration process and is Low whenever the RST input is asserted.

After configuration, the DCM always attempts to lock, whether the CLKIN signal is valid yet or not. If the input clock is not yet stable, the FPGA circuit should assert the RST input until the CLKIN input stabilizes. The DLL unit uses both the CLKIN input and the CLKFB feedback input to determine when locking is complete, that is, when the rising edges of CLKIN and CLKFB are in phase. The DFS unit monitors CLKIN to determine if a valid frequency is present on CLKIN. To achieve lock, the DCM may need to sample several thousand clock cycles.

The DCM asserts its LOCKED output High upon locking onto CLKIN. The DCM clock outputs are then valid and available for use within the FPGA application. The DCM timing section of the Spartan-3 Data Sheet provides worst-case locking times. In general, the DLL unit outputs lock faster with increasing clock frequency. The DFS unit outputs require significantly longer to lock, depending on the multiply and divide factors. Smaller multiply and divide factors result in faster lock times.

To guarantee that the system clock is established before the FPGA completes its configuration process, the DCM can optionally delay the completion of the configuration process until after the DCM locks. The STARTUP_WAIT attribute activates this feature.

Until LOCKED is High, there is no guarantee how the DCM clock outputs behave. The DCM output clocks are not valid until LOCKED is High and before that time can exhibit glitches, spikes, or other spurious behavior.

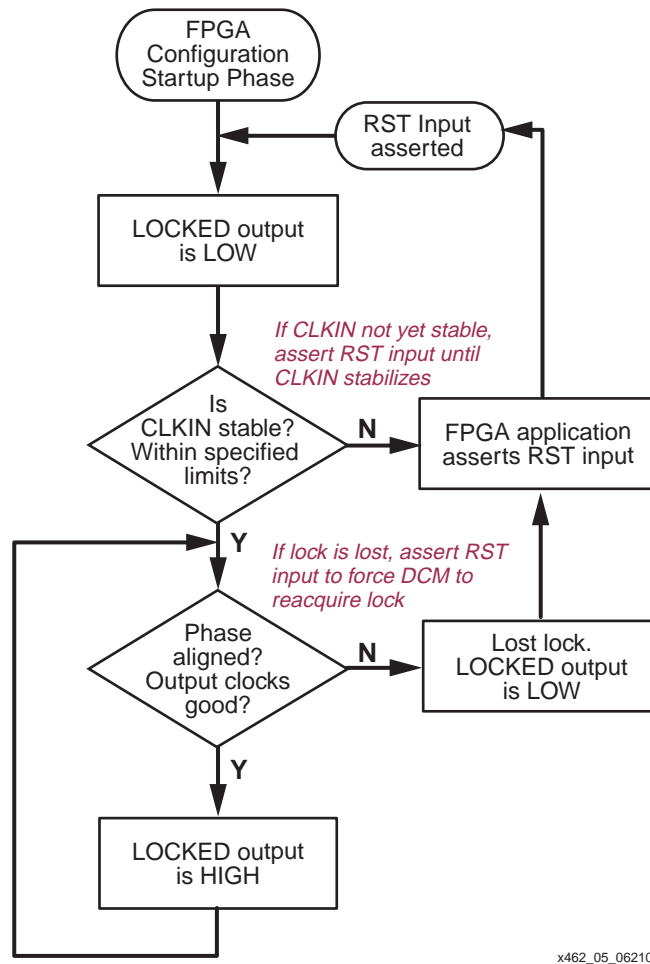


Figure 5: Functional Behavior of LOCKED Output

While the CLKIN input stays within the specified limits, the DCM continues to adjust its internal delay taps to maintain lock. However, if the CLKIN input strays well beyond the specified limits, then the DCM loses lock and deasserts the LOCKED output.

Once the DCM loses lock, it does not automatically attempt to reacquire lock. When the DCM loses lock—i.e., LOCKED was High, then goes Low—the FPGA application must take the appropriate action. For example, once lock is lost, resetting the DCM via the RST input forces the DCM to reacquire lock.

RST Input Behavior

The asynchronous RST input forces the DCM to its post-configuration state. Use the RST pin when reconfiguring the FPGA or when changing the input clock frequency beyond the allowable range. The active-High RST pin either must connect to a dynamic signal or must be tied to ground. The RST input must be asserted for 2 ns or longer.

If the input clock frequency is not yet stable after configuration, assert RST until the clock stabilizes. When using external feedback, hold the DCM in reset immediately after configuration. [Figure 20, page 30](#) shows an example reset technique using an SRL16 shift register primitive.

If the DCM loses lock—i.e., the LOCKED output was High then goes Low—then the FPGA application must assert RST to force the DCM to reacquire the input clock frequency.

If the DCM LOCKED output is High, then the LOCKED signal deactivates within four source clock cycles after RST is asserted. Asserting RST forces the DCM to reacquire lock.

Asserting RST also resets the DCM's delay tap position to zero. Due to the tap position changes, glitches may occur on the DCM clock output pins. Similarly, RST affects the duty cycle on the clock outputs.

Asserting RST also resets the present variable phase shift value back to the value specified by the PHASE_SHIFT attribute.

DCM Wizard

To simplify applications using DCMs, the Xilinx ISE development software includes a software wizard that provides step-by-step instructions for configuring a DCM. As shown in [Figure 6](#), DCM Wizard generates a vendor-specific logic synthesis file instantiating the DCM in either VHDL or Verilog syntax. Similarly, DCM Wizard generates a user constraints (UCF) file for the specific implementation. Finally, all the user specifications are saved in a Xilinx Architecture Wizard (XAW) settings file.

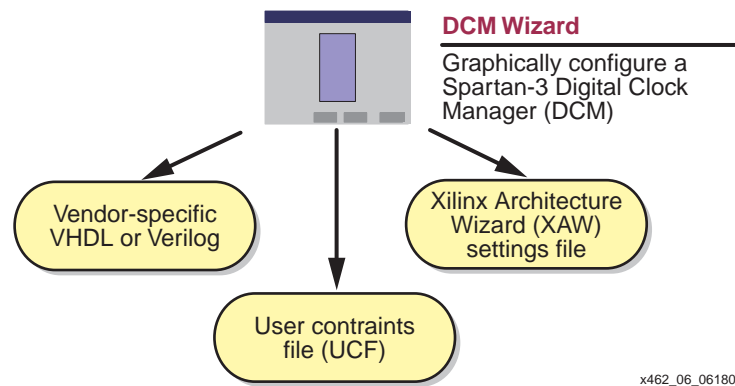


Figure 6: DCM Wizard Provides a Graphical Interface for Configuring Digital Clock Managers

Invoking DCM Wizard

There are multiple methods to invoke DCM Wizard, either from the Windows Start button or from within the Xilinx ISE Project Navigator software.

From Windows Start Button

To invoke DCM Wizard from the Windows Start button, click **Start** → **Programs** → **Xilinx ISE 5** → **Accessories** → **Architecture Wizard**. The setup window shown in [Figure 7](#) appears.

- Specify the name of the Xilinx Architecture Wizard (.xaw) file that holds the option settings for this DCM.
- Optionally, click **Browse** and select a directory location for the * .xaw file.
- Select the logic synthesis language for the output file, either VHDL or Verilog.
- Choose the targeted logic synthesis tool. DCM Wizards creates vendor-specific output for the specified synthesis tool.
- Select the targeted Spartan-3 device.

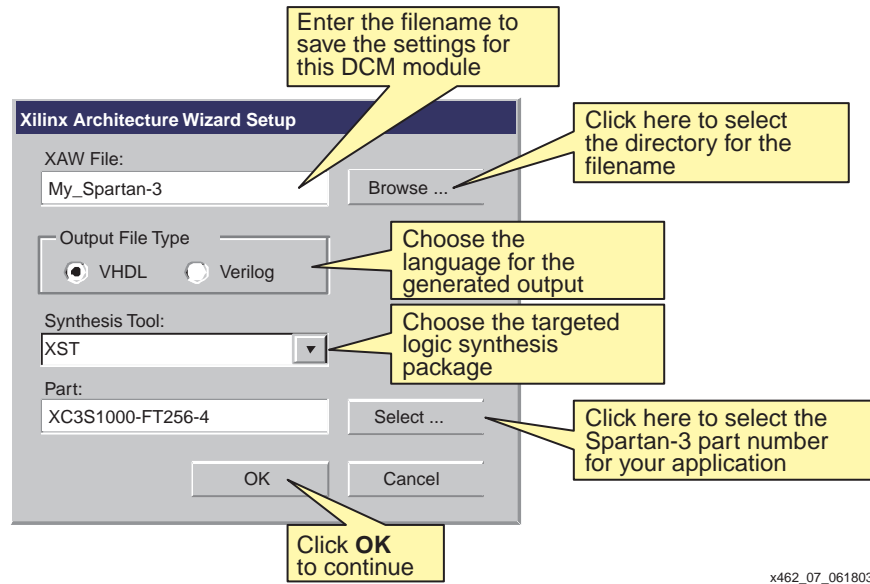


Figure 7: Set Up the Architecture Wizard

From within Project Navigator

Optionally, invoke DCM Wizard from within Project Navigator, either from the menu bar or from within the “Sources in Project” window. From the menu bar, select **Project → New Source**. Alternately, right-click in the “Sources in Project” window and choose **New Source**.

Select **Architecture Wizard** from the available list, as shown in Figure 8. Enter the file name for the Xilinx Architecture Wizard (* .xaw) file, and select the directory where the file will be saved. Click **Next >** to continue.

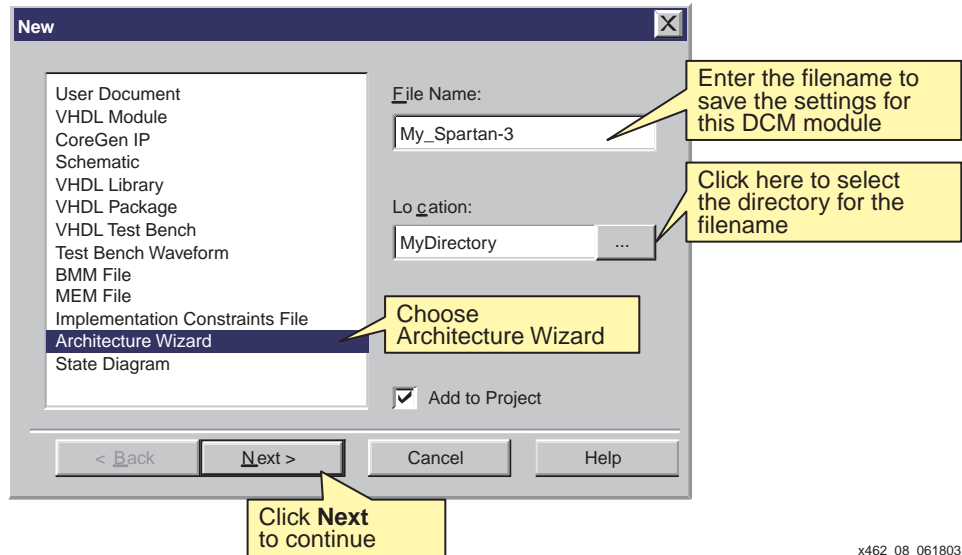


Figure 8: Configuring a New Architecture Wizard in the Project Navigator

Wizard Selection

The previous procedures are common to any of the ISE Architecture Wizards. Spartan-3 FPGAs supports the DCM Wizard, as shown in [Figure 9](#). Click **OK** to continue.

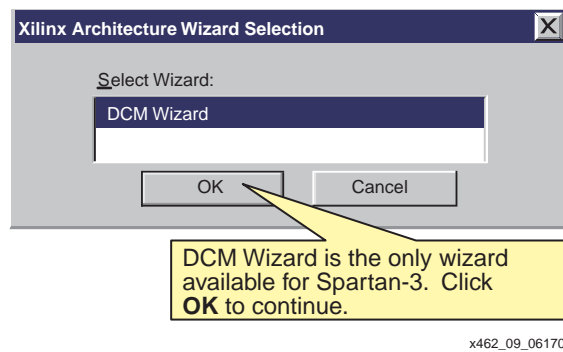


Figure 9: DCM Wizard is the only Wizard Available for Spartan-3 FPGAs

General Setup

Specify most of the DCM's options using the DCM Wizard General Setup panel, as shown in [Figure 10](#). The text in ovals shows the DCM primitive attribute name for the corresponding setting.

- Enter the name for this specific DCM instance. This name is used within the Verilog or VHDL output file.
- To select the outputs and functions used in the final application, check the option boxes next to the desired DCM clock outputs. Checking the output boxes enables related option settings below.
- Enter the frequency of the CLKIN clock input. Either specify the frequency in MHz, or specify the clock period in nanoseconds. The specified value also sets DCM's [DLL_FREQUENCY_MODE](#) attribute.
- Specify whether the CLKIN source is internal or external to the FPGA. If **External**, then DCM Wizard automatically inserts a global buffer input (IBUFG) primitive. If **Internal**, then the source signal is provided as a top-level input within the generated HDL source file.
- If the CLKDV output box is checked, then specify the **Divide by Value** for the Clock Divider circuit. This setting defines the DCM's [CLKDV_DIVIDE](#) attribute.
- Specify the feedback path to the DCM. If only the [CLKFX](#) or [CLKFX180](#) outputs are used, then select **None**. Otherwise, feedback is required. If the feedback is from within the FPGA, choose **Internal**. If the feedback loop is from outside the FPGA, choose **External**. Furthermore, specify the source of the DCM feedback, either from CLK0 (**1X**) or from CLK2X (**2X**). This setting defines the DCM's [CLK_FEEDBACK](#) attribute.

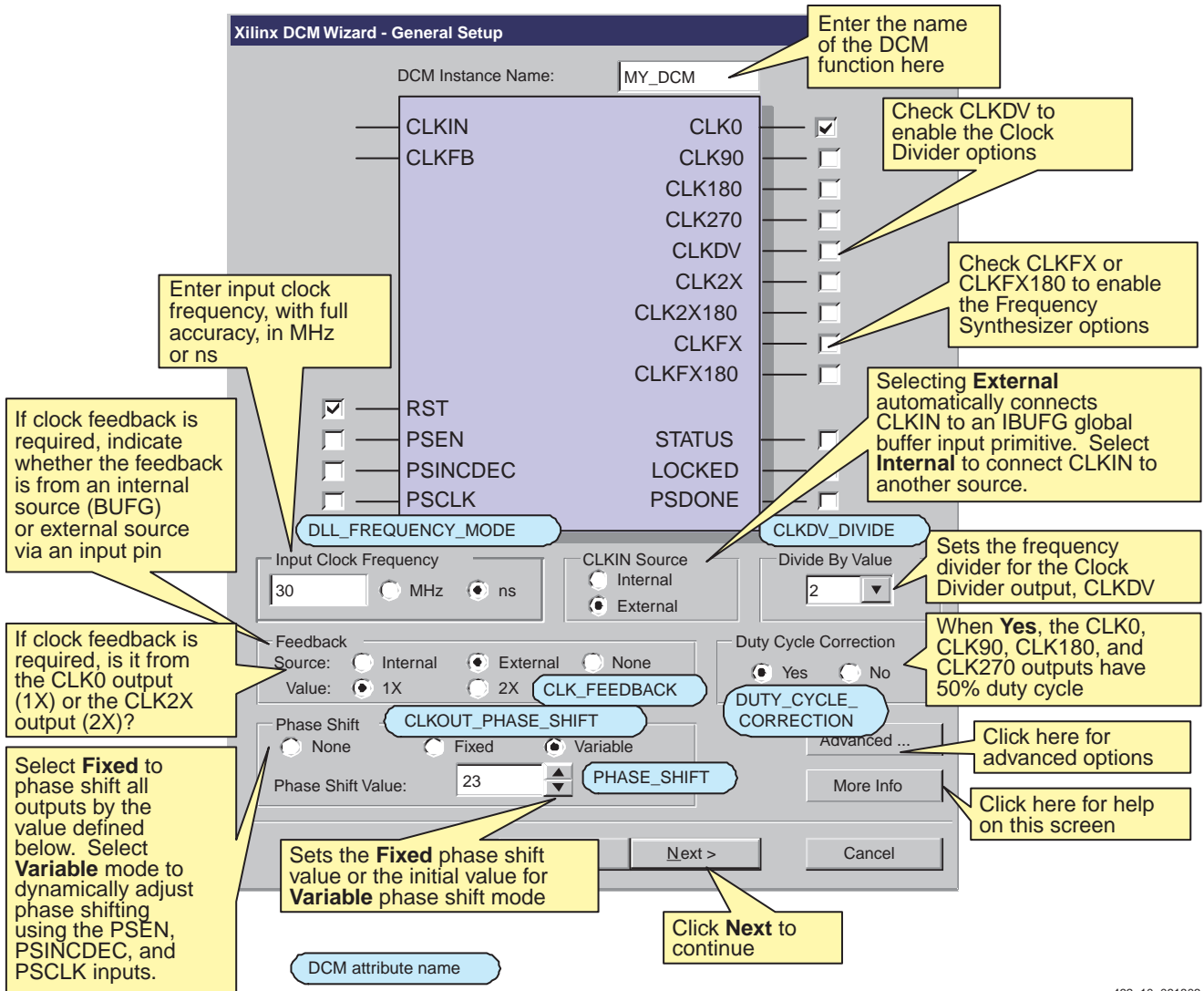


Figure 10: A Majority of DCM Options are Set in the General Setup Panel

- Specify whether to phase shift all DCM outputs. By default, there is no phase shifting (**None**). If phase shifting is required by the application, choose whether the phase shift value is **Fixed** or **Variable**. Selecting **Variable** also enables the dynamic phase shift controls, **PSEN**, **PSINCDEC**, **PSCLK**, and **PSDONE**. This setting defines the DCM's **CLKOUT_PHASE_SHIFT** attribute. For both **Fixed** and **Variable** modes, specify the related **Phase Shift Value**, which provides either the fixed phase shift value or the initial value for the dynamic phase shift. This setting defines the DCM's **PHASE_SHIFT** attribute.
- To open the **Advanced Options** window, click **Advanced**.
- When finished, click **Next >** to continue to the **Clock Buffers** panel.

Advanced Options

Various advanced DCM options are grouped together in the Advanced Options window, shown in Figure 11:

- By default, the DCM has no effect on the FPGA's configuration process. Click **Yes** to have the FPGA wait for the DCM to assert its **LOCKED** output before asserting the **DONE** signal at the end of configuration. This setting defines the DCM's **STARTUP_WAIT**

attribute. If set to **Yes**, additional bitstream generation option changes are required, as described in the “[Setting Configuration Logic to Wait for DCM LOCKED Output](#)” section.

- If the CLKIN input frequency is too high for a particular DCM feature, click **Yes** under **Divide Input Clock by 2** to reduce the input frequency by half with roughly a 50% duty cycle before entering the DCM block. This setting defines the DCM's [CLKIN_DIVIDE_BY_2](#) attribute.
- If required for source-synchronous data transfer applications, modify the **DCM Deskew Adjust** value to **SOURCE_SYNCHRONOUS**. Do not use any values other than **SOURCE_SYNCHRONOUS** or **SYSTEM_SYNCHRONOUS** without first consulting Xilinx. This setting defines the DCM's [DESKEW_ADJUST](#) attribute. See “[Skew Adjustment](#).”
- Click **OK** when finished to apply any changes and return to the [General Setup](#) window.

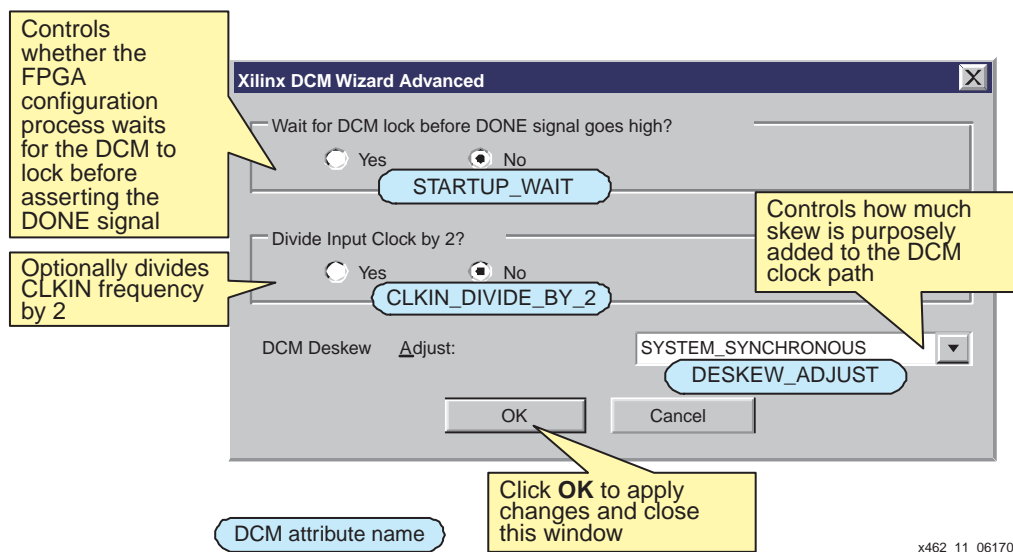


Figure 11: DCM Advanced Options Panel

Clock Buffers

Define the clock buffer output type for each DCM clock output, shown in [Figure 12](#). By default, DCM Wizard automatically assigns all outputs to a global buffer (BUFG). However, there are only four global buffers along each the top or bottom edge of the device, shared by two DCMs. In the XC3S50, there is a single DCM along the top or bottom edge that optionally connects to all four global buffers along the edge.

- To assign clock buffer types for each DCM clock output, click **Customize** under **Clock Buffer Settings**.
- For each DCM clock output, select a **Clock Buffer** output type using the drop-down list. [Table 10](#) lists the available Clock Buffer options.
- If using an **Enabled Buffer** output type, either specify a signal name for the buffer enable (CE) input or use the automatically generated name.
- If using a **Clock Mux** output type, either specify a signal name for the select (S) input or use the automatically generated name.
- When finished, click **Next >** or **Finish** to continue. The **Next >** option only appears if the [CLKFX](#) or [CLKFX180](#) outputs were selected in the [General Setup](#) panel. Otherwise, click **Finish** to generate the HDL output (see “[Generating HDL Output](#)”).

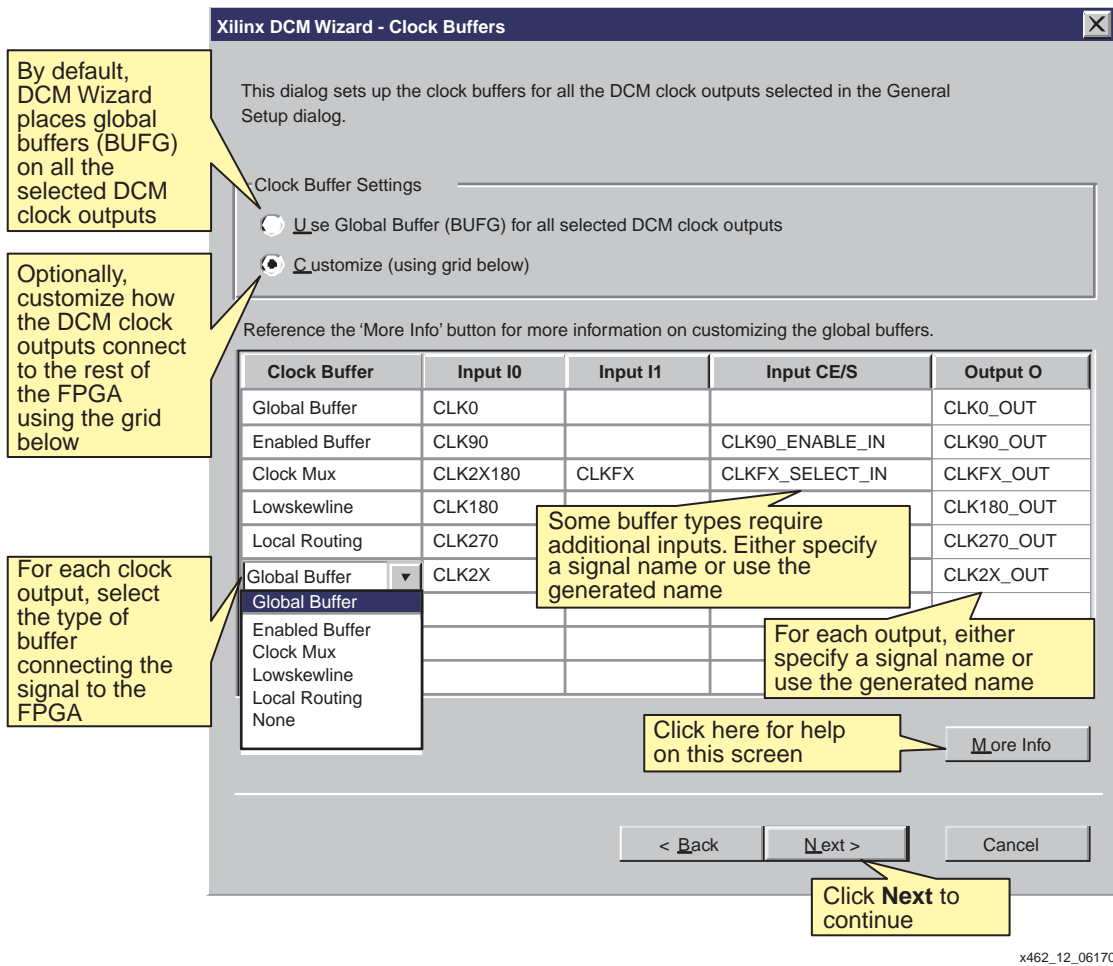


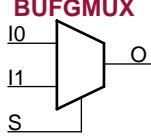
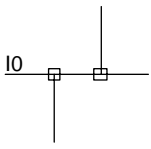
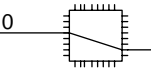
Figure 12: DCM Wizard Provides a Variety of Buffer Options for each DCM Output

Table 10: Settings for Clock Buffer Output Types

Clock Buffer Selection	Diagram	Description
Global Buffer		Connect to one of four global buffers (BUFG) along the same edge as the DCM.
Enabled Buffer		Connect to one of the four global buffers configured as an enable clock buffer (BUFGCE). The CE input enables the buffer when High. When CE is Low, the buffer output is zero.

CE	O
0	0
1	IO

Table 10: Settings for Clock Buffer Output Types

Clock Buffer Selection	Diagram	Description						
Clock Mux		<p>Connect to one of the four global buffers configured as a clock multiplexer (BUFGMUX). The S input selects the clock source.</p> <table border="1" data-bbox="922 352 1123 504"> <thead> <tr> <th>S</th> <th>O</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>I0</td> </tr> <tr> <td>1</td> <td>I1</td> </tr> </tbody> </table>	S	O	0	I0	1	I1
S	O							
0	I0							
1	I1							
Lowskewline		Connect to low-skew programmable interconnect.						
Local Routing		Connect to local interconnect, skew not critical.						
None		Disable DCM output.						

Clock Frequency Synthesizer

The Clock Frequency Synthesizer panel, shown in [Figure 13](#), only appears if the [CLKFX](#) or [CLKFX180](#) outputs were selected in the [General Setup](#) panel.

Here, specify either the desired output frequency or enter the specific values for the multiply and divide factors. The frequency limits—or delay limits if CLKIN was specified in ns—appear under **Valid Ranges for Selected Speed Grade**. The range is displayed for both possible values of the [DFS_FREQUENCY_MODE](#) attribute. The range is tighter if the DCM uses any of the DLL-related clock outputs.

- Click **Use output frequency** and enter the requested value, in as much precision as possible, either in megahertz (**MHz**) or in nanoseconds (**ns**). Click **Calculate** to compute the values for the [CLKFX_MULTIPLY](#) and [CLKFX_DIVIDE](#) attributes. If no solution is available using the possible multiply and divide values, DCM Wizard issues an error message asking for another output frequency value. If a solution exists, then the multiply and divide values, plus the resulting jitter values (see [“Clock Jitter or Phase Noise”](#)) appear under **Generated Output**.
- Optionally, click **Use Multiply (M) and Divide (D) values** and enter the desired values. Click **Calculate** to calculate the resulting output frequency and jitter, displayed under **Generated Output**.
- Finally, click **Finish** to generate the HDL output (see [“Generating HDL Output”](#)).

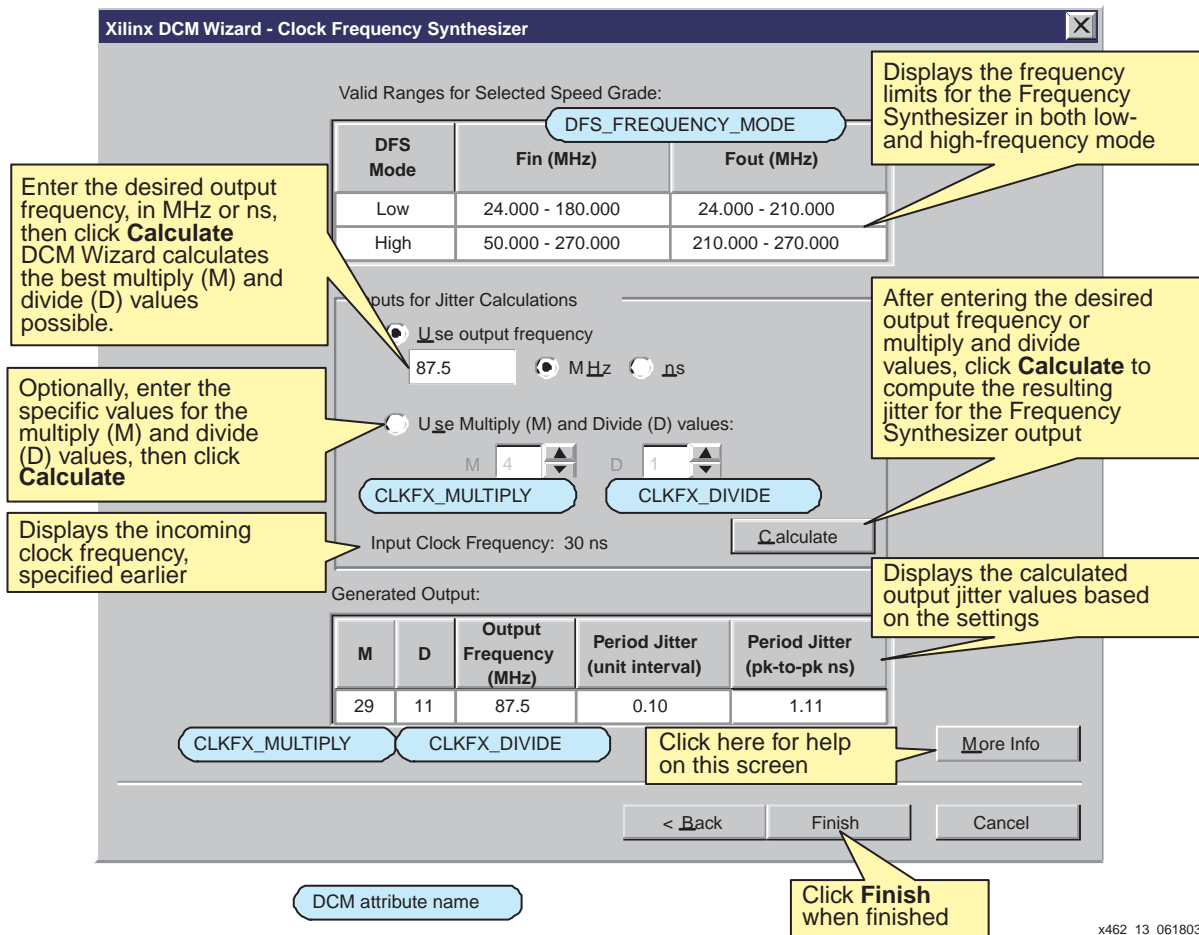


Figure 13: Set the Multiply and Divide Values for the Digital Frequency Synthesizer and Calculate the Resulting Jitter

Generating HDL Output

After entering all the parameters and clicking **Finish**, DCM Wizard automatically generates the requested VHDL or Verilog HDL output file, as shown in Figure 14. DCM Wizard also generates a User Constraints File (UCF) based on the settings.

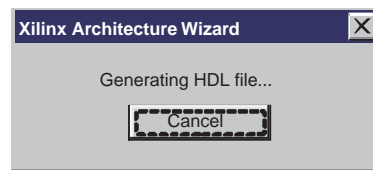


Figure 14: DCM Wizard Generates Either a VHDL or Verilog HDL Output File

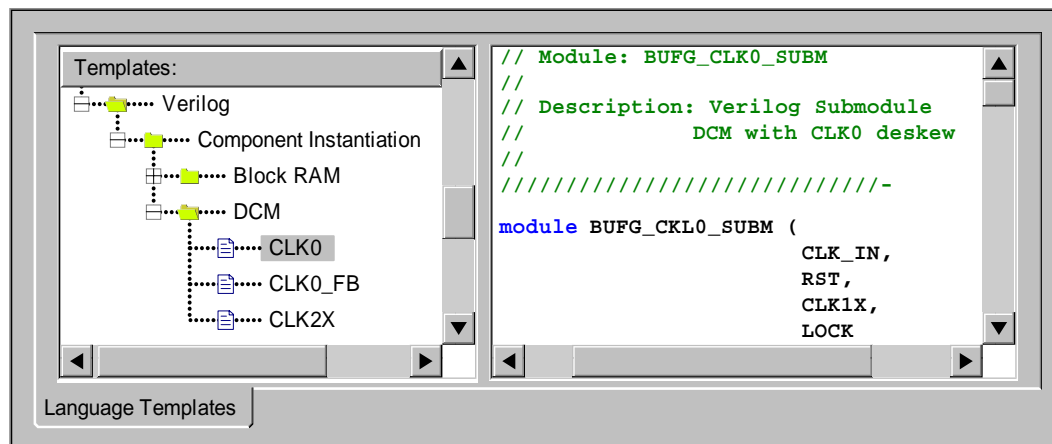
VHDL and Verilog Instantiation

DCM Wizard is the easiest method to create a VHDL or Verilog HDL description of a DCM. However, Verilog and VHDL source examples are also available.

Language Templates within Project Navigator

There are DCM language templates available within the ISE 5.2i and later Project Navigator. To select a DCM template, select **Edit → Language Templates** from the Project Navigator menu. From the Templates tree shown in Figure 15, expand either the **Verilog** or **VHDL** folder, then

the **Component Instantiation** folder, then the **DCM** folder. Under the DCM folder, select the desired DCM source file. The selected source file appears in the adjacent window.



x462_15_061803

Figure 15: DCM Designs in Project Navigator Language Templates

Use the file either as a reference or cut the content of the window into a new source file.

VHDL and Verilog Reference Files

The same VHDL and Verilog source files are also available for download from the Xilinx FTP site from the following locations.

- VHDL DCM Reference Files
ftp://ftp.xilinx.com/pub/applications/xapp/xapp462_vhdl.zip
- Verilog DCM Reference Files
ftp://ftp.xilinx.com/pub/applications/xapp/xapp462_verilog.zip

Eliminating Clock Skew

One of the fundamental functions of a DCM is to eliminate clock skew. Eliminating clock skew is important for most designs that operate at 50 MHz or more. Furthermore, the concepts involved in clock skew elimination also apply to many of the other applications of a DCM.

What is Clock Skew?

Clock skew inherently exists in every synchronous system. The pristine clock edge generated by the clock source arrives at different times at different points in the system—either within a single device or on the clock inputs to the different devices connected to the clock. This difference in arrival times is defined as clock skew.

Figure 16 illustrates clock skew in an example system. A clock source drives the clock input to an FPGA. The clock enters through an input pin on the FPGA, is distributed within the FPGA using the internal low-skew global clock network, and arrives at a flip-flop within the FPGA. Each element in the clock path delays the arrival of the clock edge at the flip-flop. Consequently, the clock input at the flip-flop—Point (B)—is delayed, or skewed compared to the original clock source at Point (A). In this example, this clock skew or difference in arrival time for this path is called Δb .

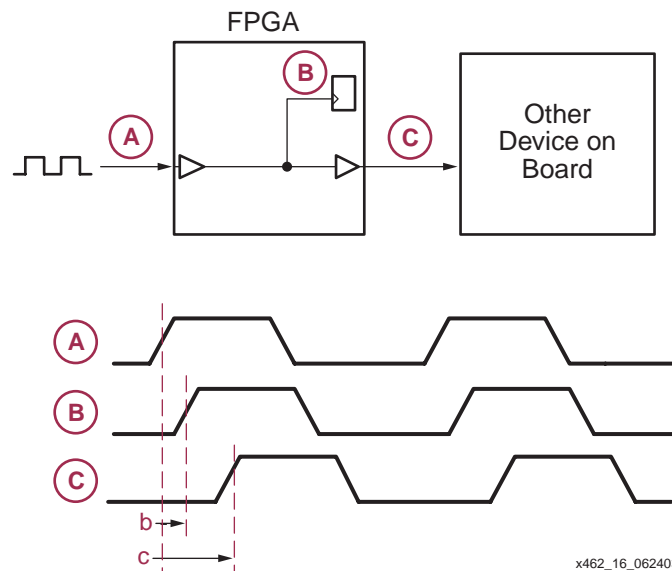


Figure 16: Clock Skew Inherently Exists in Every Synchronous System

Similarly, the clock source is rebuffered in the FPGA and drives another device on the board. In this case, again the clock source enters the FPGA via an input pin, is distributed via the global clock network, feeds an output pin on the FPGA, and finally connects to the other device via a trace on the printed circuit board (PCB). Because there is more total delay in this clock path, the resulting skew, Δc , is also larger.

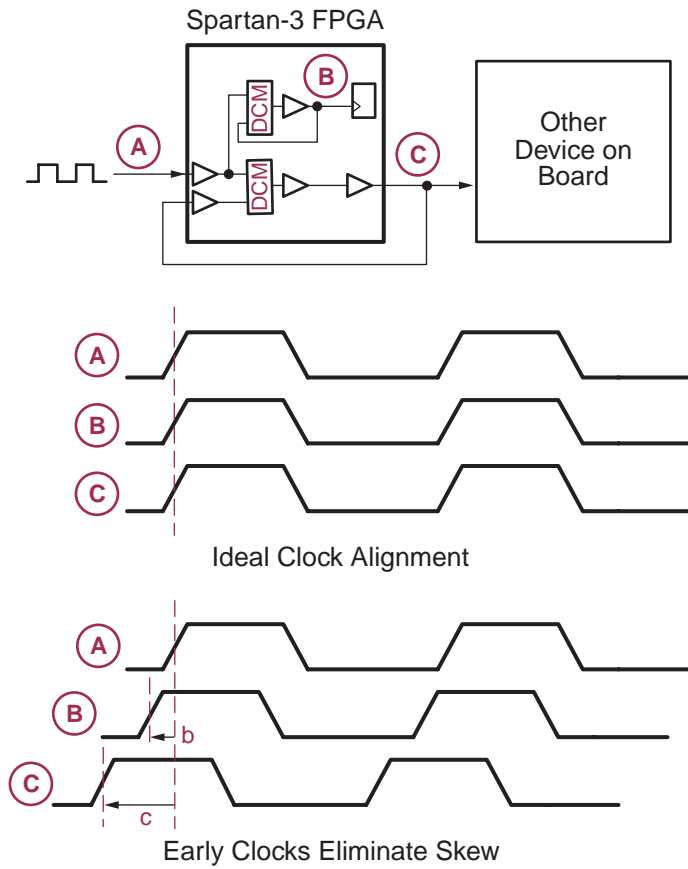
Clock Skew: The Performance Thief

Clock skew potentially reduces the overall performance of the design by increasing setup times and lengthening clock-to-output delays—both of which increase the clock cycle time. Similarly, clock skew might require lengthy hold times on some devices. Otherwise, unreliable operation might result.

Make it Go Away!

Is there a way to eliminate clock skew? Fortunately, a Digital Clock Manager (DCM) provides such capabilities. Figure 17 shows the same example design as Figure 16, except this time implemented in a Spartan-3 FPGA. Two DCMs eliminate the clock skew: One DCM eliminates the skew for clocked items within the FPGA, the other DCM eliminates the skew when clocking the other device on the board. The result is practically ideal alignment between the clock at Points (A), (B), and (C)!

How is clock skew elimination accomplished? Remember, clock skew is caused by the delay in the clock path. In Figure 17, the clock at Point (B) was skewed by Δb and the clock at Point (C) was skewed by Δc . What if there was a way to provide Point (B) with an early version of the clock, advanced by Δb and a way to provide Point (C) with an early version of the clock, advanced by Δc ? The result would be that all clocks would arrive at their destinations with perfect clock edge alignment. Such perfect alignment reduces setup times, shortens clock-to-output delays, and increases overall system performance.



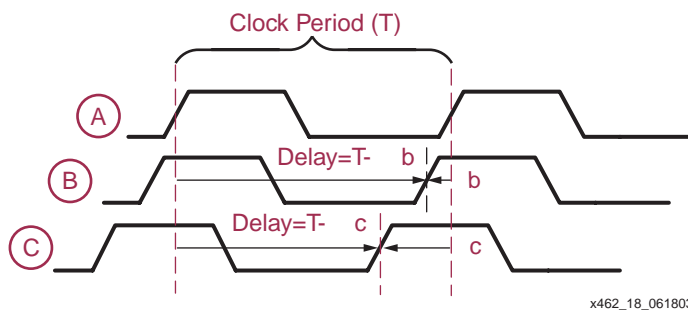
x462_17_062403

Figure 17: Eliminating Clock Skew in a Spartan-3 FPGA Design

Predicting the Future by Closely Examining the Past

Even though Spartan-3 FPGAs employ highly advanced digital logic, unfortunately they cannot predict the future. However, a DCM applies its knowledge of the past behavior of the clock to predict the future. Most input clocks to a system have a never-changing, monotonic frequency. Consequently, the input clock has a nearly constant period, T .

Because it is impossible to insert a negative delay to counteract the clock skew, the DCM *delays* the clocks enough so that they appear to be advanced in time. How is this accomplished? The clock cycle is repetitive and has a fixed period, T . As shown in Figure 18, the clock at Point (B) appears to be advanced in time by the delay Δb . In reality however, the clock is delayed by $(T - \Delta b)$. Similarly, the clock at Point (B) is delayed by $(T - \Delta c)$.



x462_18_061803

Figure 18: Delaying a Fixed Frequency Clock Appears to Predict the Future

The clock period, T , is easy to derive knowing the frequency of the incoming monotonic clock signal. But what are the clock skew delays Δb and Δc ? With careful analysis, they can be determined after examining the behavior of multiple systems under different conditions. In reality, this is impractical. Furthermore, the values of Δb and Δc are different between devices and vary with temperature and voltage on the same device.

Instead of attempting to determine the Δb and Δc delays in advance, the Spartan-3 DCM employs a Delay-Locked Loop (DLL) that constantly monitors the delay via a feedback loop, as shown in [Figure 17](#). In this particular example, two DCMs are required—one to compensate for the clock skew to internal signals and another to compensate for the skew to external devices, each with their own clock feedback loop. The DLL constantly adapts to subtle changes caused by temperature and voltage.

Locked on Target

In order to determine and insert the correct delay, the DCM samples up to several thousand clock cycles. Once the DCM inserts the correct delay, the DCM asserts its **LOCKED** output signal.

Do not use the DCM clock outputs until the DCM asserts its **LOCKED** signal. Until the DCM locks onto the input clock signal, the output clocks are invalid. While the DCM attempts to lock onto the clock signal, the output clocks can exhibit glitches, spikes, or other spurious movements.

In an application, the **LOCKED** signal qualifies the output clock. Think of **LOCKED** as a “clock signal good” indicator.

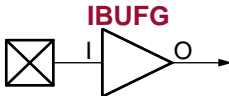
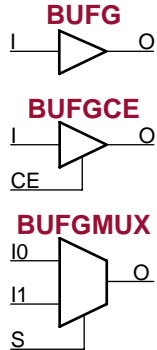
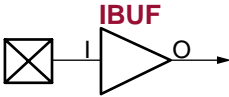

A Stable, Monotonic Clock Input

To operate properly, the DCM requires a stable, monotonic clock input. Consequently, the DCM can predict future clock periods and adjust the output clock timing appropriately. Once locked, the DCM tolerates clock period variations up to the value specified in the Spartan-3 Data Sheet. See “[DCM Clock Requirements](#)” section.

Should the input clock vary well outside the specified limits, the DCM loses lock and the **LOCKED** output switches Low. If the DCM loses lock, reset the DCM to reacquire lock. If the input clock stays within the specified limits, then the output clocks always are valid when the **LOCKED** output is High. However, it is possible for the clock to stray well outside the limits, for the **LOCKED** output to stay High, and for either the **CLKDV** or **CLKFX** outputs to be invalid. In short, a stable, monotonic clock input guarantees problem-free designs.

The recommended input path to a DCM's **CLKIN** input is via one of the four global buffer inputs (**IBUFG**) along the same half of the device. Using the **IBUFG** path, the delay from the pad, through the global buffer, to the DCM is eliminated from the deskewed output. Other paths are possible, however, as shown in [Table 11](#). The signal driving the **CLKIN** input can also originate a general-purpose input pin (**IBUF** primitive) via general-purpose interconnect, from in global buffer input (**IBUFG**), or from a global buffer multiplexer (**BUFGMUX**, **BUFGCE**). Similarly, an LVDS clock input may provide the clock signal. The deskew logic is characterized for a single-ended clock input such as LVCMOS or LVTTTL. Differential signals may incur a slight amount of phase error due to I/O timing. See the [Spartan-3 Data Sheet, Module 3](#) for specific I/O timing differences.

Table 11: CLKIN Input Sources

CLKIN Source	Description
Via global buffer input 	A global buffer input, IBUFG, is the preferred source for an external clock to the DCM. The delay from the pad, through the global buffer, to the CLKIN input is characterized, and this delay is removed from the deskewed clock output.
Global Clock Buffer 	A global clock buffer, using either a BUFG, BUFGCE, or BUFGMUX primitive, is a preferred source for an internally generated clock to the DCM. The delay through the global buffer is characterized, and this delay is removed from the deskewed clock output.
Via general-purpose I/O 	Any user-I/O pin, IBUF, becomes an alternate source for an external clock. The pad-to-DCM delay cannot be predetermined due to the numerous potential input paths, and consequently, the delay is not compensated by the DCM.
Derived from internal logic 	Logic within the FPGA also may be the clock source. Again, the logic-to-DCM delay cannot be predetermined as it is not compensated by the DCM.

Feedback from a Reliable Source

In order to lock in on the proper delay, the DCM monitors both the incoming clock and a feedback clock, tapped after the clock distribution delay. There are no restrictions on the total delay in the clock feedback path. If required, the DLL effectively delays the output clock by multiple clock periods. Consequently, a DCM can compensate for either internal or external delays, but the clock feedback must connect to the correct feedback point.

Removing Skew from an Internal Clock

To eliminate skew within the FPGA, the feedback tap is the same clock as that seen by the clocked elements within the FPGA, shown in [Figure 19](#). The feedback clock is typically the CLK0 output (no phase shift) from the DCM, connected to the output of a global clock buffer (BUFG) or a global clock multiplexer (BUFGMUX or BUFGCE primitive) on the same edge of the device. Alternatively, the DCM's CLK2X output (no phase shift, frequency doubled) may be used instead of the CLK0 output.

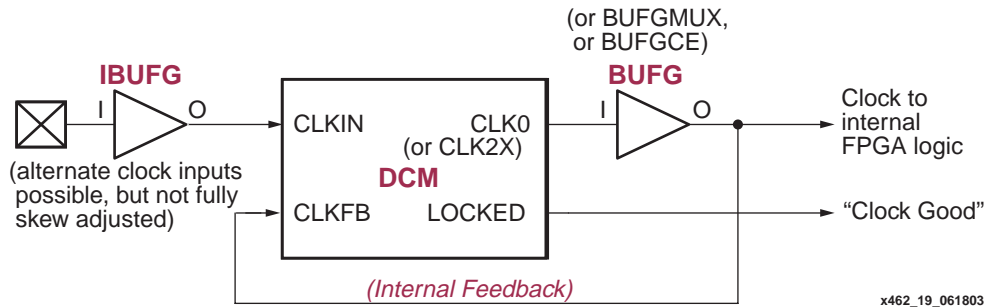


Figure 19: Eliminating Skew on Internal Clock Signals

Removing Skew from an External Clock

Constructing the DCM feedback for an external clock is slightly more complex. Ideally, the clock feedback originates from the point where the signal feeds any external clocked inputs, after any long printed-circuit board traces or external clock rebuffering, as shown in Figure 20.

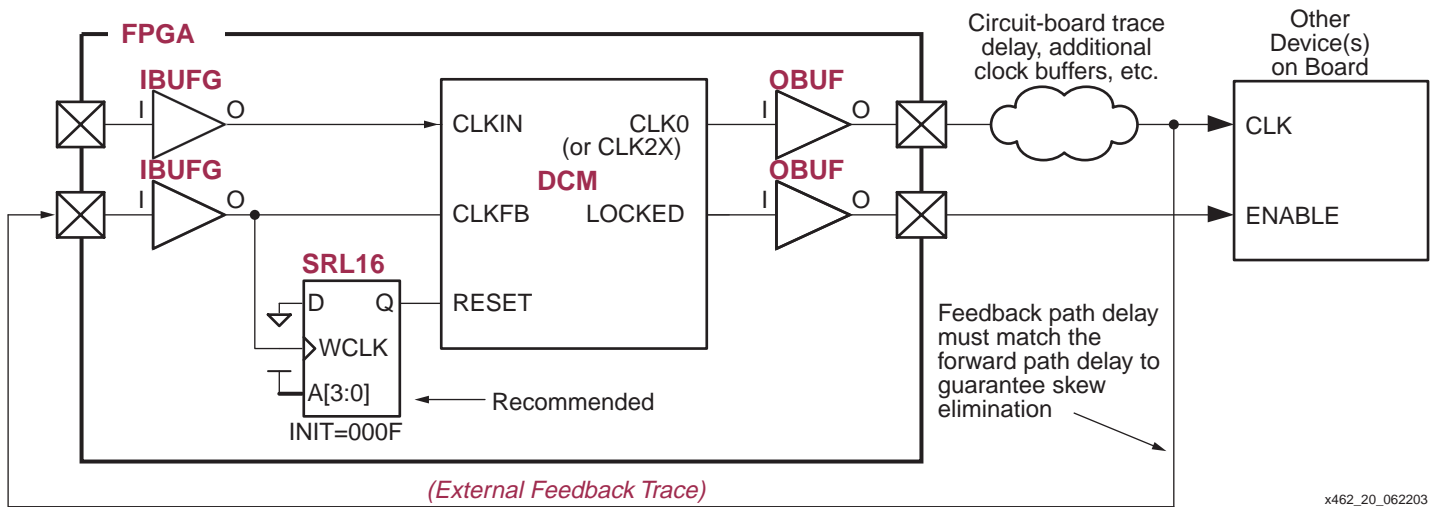


Figure 20: Eliminating Skew on External Clock Signals

The LOCKED signal indicates when the DCM achieves lock, qualifying the clock signal. The LOCKED signal can enable external devices or an inverted version can connect to an active-Low chip enable.

Reset DCM After Configuration

When using external feedback, apply a reset pulse to the DCM immediately after configuration to ensure consistent locking. An SRL16 primitive, initialized with 0x000F, supplies the necessary reset pulse, as shown in Figure 20. See “RST Input Behavior.”

Why Reset?

Why is this extra reset pulse required? For an optimum locking process, a DCM configured with external feedback requires both the CLKIN and either the CLK0 or CLK2X signals to be present and stable when the DCM begins to lock. During the configuration process, the external feedback, CLKFB, is not available because the FPGA’s I/O buffers are not yet active.

At the end of configuration, the DCM begins the capture process once the device enters the startup sequence. Because the FPGA’s global 3-state signal (GTS) still is asserted at this time, any output pins remain in a 3-state (high-impedance, floating) condition. Consequently, the CLKFB signal is in an unknown logic state.

When CLKFB eventually appears after the GTS is deasserted, the DCM proceeds to capture. However, without the reset pulse, the DCM might not lock at the optimal point, which potentially introduce slightly more jitter and greater clock cycle latency through the DCM.

Without the reset, another possible issue might occur if the CLKFB signal, while in the 3-state condition, cross-couples with another signal on the board due to a printed-circuit board signal integrity problem. The DCM might sense this invalid cross-coupled signal as CLKFB and use it to proceed with a lock. This possibly prevents the DCM from properly locking once the GTS signal deasserts and the true CLKFB signal appears.

What is a Delay-Locked Loop?

Two basic types of circuits remove clock delay:

- Delay-Locked Loops (DLLs) and
- Phase-Locked Loops (PLLs)

In addition to their primary function of removing clock distribution delay, DLLs and PLLs typically provide additional functionality such as frequency synthesis, clock conditioning, and phase shifting.

Delay-Locked Loop (DLL)

As shown in [Figure 21](#), a DLL in its simplest form consists of a tapped delay line and control logic. The delay line produces a delayed version of the input clock CLKIN. The clock distribution network routes the clock to all internal registers and to the clock feedback CLKFB pin. The control logic continuously samples the input clock as well as the feedback clock to properly adjust the delay line. Delay lines are constructed either using a voltage controlled delay or as a series of discrete delay elements. For best, ruggedly stable performance, the Spartan-3 DLL uses an all-digital delay line.

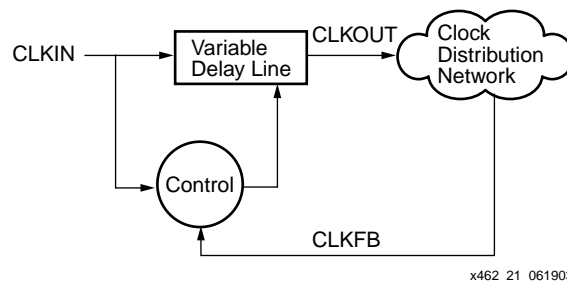


Figure 21: Delay-Locked Loop (DLL) Block Diagram

A DLL works by inserting delay between the input clock and the feedback clock until the two rising edges align, effectively delaying the feedback clock by almost an entire period—minus the clock distribution delay, of course. In DLL and PLL parlance, the feedback clock is 360° out of phase, which means that they appear to be exactly in phase again.

After the edges from the input clock line up with the edges from the feedback clock, the DLL “locks”, and the two clocks have no discernible difference. Thus, the DLL output clock compensates for the delay in the clock distribution network, effectively removing the delay between the source clock and its loads. Voila!

Phase-Locked Loop (PLL)

While designed for the same basic function, a PLL uses a different architecture to accomplish the task. As shown in [Figure 22](#), the fundamental difference between the PLL and DLL is that instead of a delay line, the PLL uses a voltage-controlled oscillator, which generates a clock signal that approximates the input clock CLKIN. The control logic, consisting of a phase detector and filter, adjusts the oscillator frequency and phase to compensate for the clock distribution delay. The PLL control logic compares the input clock to the feedback clock CLKFB and adjusts the oscillator clock until the rising edge of the input clock aligns with the feedback clock. The PLL then “locks.”

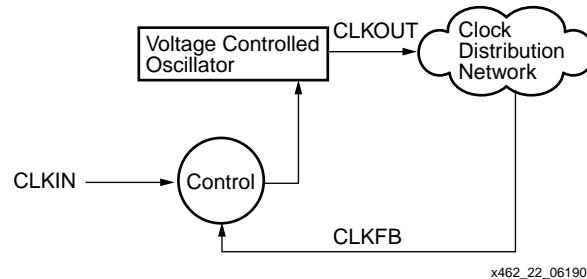


Figure 22: Phase-Locked Loop (PLL) Block Diagram

Implementation

A DLL or PLL is assembled using either analog or digital circuitry; each approach has its own advantages. An analog implementation with careful circuit design produces a DLL or PLL with a finer timing resolution. Additionally, analog implementations sometimes consume less silicon area.

Conversely, digital implementations offer advantages in noise immunity, lower power consumption and better jitter performance. Digital implementations also provide the ability to stop the clock, facilitating power management. Analog implementations can require additional power supplies, require close control of the power supply, and pose problems in migrating to new process technologies.

DLL vs. PLL

When choosing between a PLL or a DLL for a particular application, understand the differences in the architectures. The oscillator used in the PLL inherently introduces some instability, which degrades the performance of the PLL when attempting to compensate for the delay of the clock distribution network. Conversely, the unconditionally stable DLL architecture excels at delay compensation and clock conditioning. On the other hand, the PLL typically has more flexibility when synthesizing a new clock frequency.

Skew Adjustment

Most of this section discusses how to remove skew and how to phase align an internal or external clock to the clock source. In actuality, the DCM purposely adds a small amount of skew via an advanced attribute called [DESKEW_ADJUST](#). In DCM Wizard, the [DESKEW_ADJUST](#) attribute is controlled via the [Advanced Options](#) window.

There are two primary applications for this attribute, [SYSTEM_SYNCHRONOUS](#) and [SOURCE_SYNCHRONOUS](#). The overwhelming majority of applications use the default [SYSTEM_SYNCHRONOUS](#) setting. The purpose of each mode is described below.

System Synchronous

In a Source Synchronous system, all devices within a data path share a common clock source, as shown in [Figure 23](#). This is the traditional and most-common system configuration. The [SYSTEM_SYNCHRONOUS](#) option, which is the default value, adds a small amount of clock delay so that there is zero hold time when capturing data. Hold time is essentially the timing difference between the best-case data path and the worst-case clock path. The DCM's clock skew elimination function advances the clock, essentially dramatically shortening the worst-case clock path. However, if the clock path is advance so far that the clock appears before the

data, then hold time results. The SYSTEM_SYNCHRONOUS setting injects enough additional skew on the clock path to guarantee zero hold times, but at the expense of a slightly longer clock-to-output time.

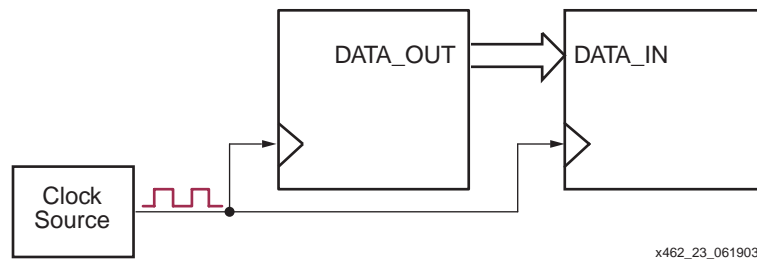


Figure 23: System-Synchronous Applications are Clocked by a Single, System-Wide Clock Source

Source Synchronous

SOURCE_SYNCHRONOUS mode is an advanced setting, used primarily in high-speed data communications interfaces. In Source Synchronous applications, both the data and the clock are derived from the same clock source, as shown in Figure 24. The transmitting device sends the both data and clock to the receiving device. The receiving device then adjusts the clock timing for best data reception. High-speed Dual-Data Rate (DDR) and LVDS connections are examples of such systems.

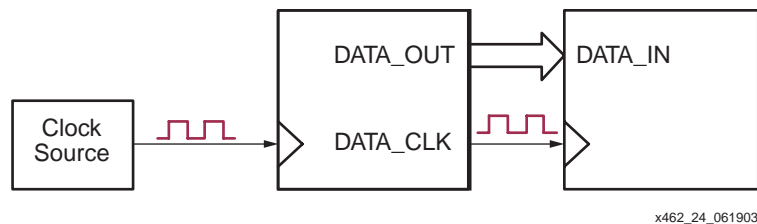


Figure 24: In Source-Synchronous Applications, the Data Clock is Provided by the Data Source

The SOURCE_SYNCHRONOUS setting essentially zeros out any phase difference between the incoming clock and the deskewed output clock from the DCM. The FPGA application must then adjust the clock timing using either the Fixed or Dynamic Fine Phase Shift mode. The following application notes provide additional information on Source Synchronous design and using dynamic phase alignment:

- XAPP268: *Dynamic Phase Alignment*
<http://www.xilinx.com/xapp/xapp268.pdf>
- XAPP622: *SDR LVDS Transmitter/Receiver*
<http://www.xilinx.com/xapp/xapp622.pdf>

Similarly, the following application note delves into more details on system-level timing. Although the application note is written for the Virtex-II and Virtex-II Pro FPGA architectures, most of the concepts apply directly to Spartan-3 FPGAs.

- XAPP259: *System Interface Timing Parameters*
<http://www.xilinx.com/xapp/xapp259.pdf>

Timing Comparisons

Figure 25 compares the effect of both SYSTEM_SYNCHRONOUS and SOURCE_SYNCHRONOUS settings using a Dual-Data Rate (DDR) application. In DDR applications, two data bits appear on each data line—one during the first half-period of the clock, the second during the second half-period.

In SYSTEM_SYNCHRONOUS mode, a small amount of skew is purposely added to the DCM clock path so that there is zero hold time.

In SOURCE_SYNCHRONOUS mode, no additional skew is inserted to the DCM clock path. However, the FPGA application must insert additional skew or phase shifting so that the clock appears at the ideal location in the data window.

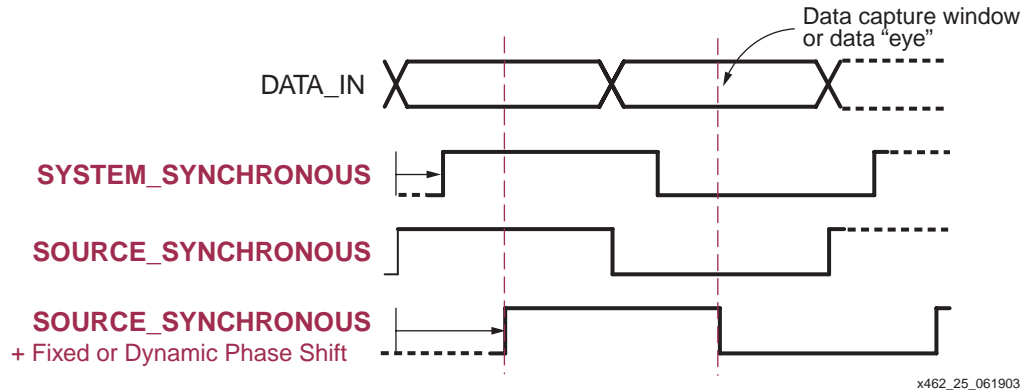


Figure 25: Comparing SYSTEM_SYNCHRONOUS and SOURCE_SYNCHRONOUS Timing in a Dual-Data Rate (DDR) Application

Clock Conditioning

Clock conditioning is a function where an incoming clock with a duty cycle other than 50% is reshaped to have a 50% duty cycle. Figure 26 shows an example where an incoming clock, with roughly a 40% High time and a 60% Low time (40%/60% duty cycle), is reshaped into a nearly perfect 50% duty cycle—nearly perfect because there is some residual duty-cycle distortion specified by the CLKOUT_DUTY_CYCLE_DLL and CLKOUT_DUTY_CYCLE_FX values in the Spartan-3 Data Sheet. The distortion is estimated at less than 150 ps.

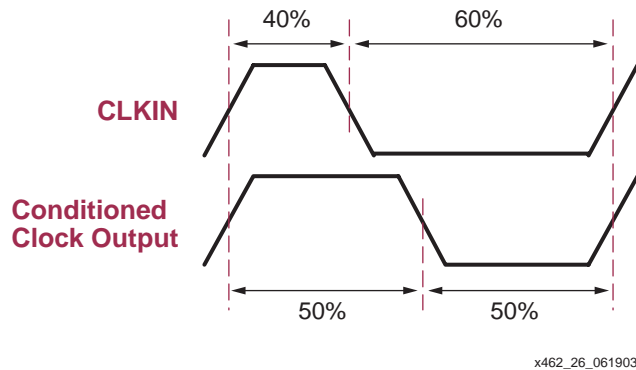


Figure 26: DCM Duty-Cycle Correction Feature Provides 50% Duty Cycle Outputs

Clocks with 50% duty cycle are mandatory for high-speed communications interfaces such as LVDS or Dual-Data Rate (DDR) and for clock forwarding or clock mirroring applications. See “Dual-Data Rate (DDR) Clocking Example.”

The DCM automatically conditions most clock outputs so that they have a 50% duty cycle. Other clock outputs are optionally conditioned, depending either on the operating conditions or on attribute settings, as shown in Table 12.

Table 12: Conditioned Clock Output with 50% Duty Cycle

DCM Clock Output	50% Duty Cycle Output							
CLK0 CLK90	When DUTY_CYCLE_CORRECTION attribute set to TRUE							
CLK180 CLK270	<table border="1"> <thead> <tr> <th colspan="2">DLL_FREQUENCY_MODE Attribute</th> </tr> <tr> <th>LOW</th> <th>HIGH</th> </tr> </thead> <tbody> <tr> <td>When DUTY_CYCLE_CORRECTION attribute set to TRUE</td> <td>Outputs not available</td> </tr> </tbody> </table>		DLL_FREQUENCY_MODE Attribute		LOW	HIGH	When DUTY_CYCLE_CORRECTION attribute set to TRUE	Outputs not available
DLL_FREQUENCY_MODE Attribute								
LOW	HIGH							
When DUTY_CYCLE_CORRECTION attribute set to TRUE	Outputs not available							
CLK2X CLK2X180	<table border="1"> <thead> <tr> <th colspan="2">DLL_FREQUENCY_MODE Attribute</th> </tr> <tr> <th>LOW</th> <th>HIGH</th> </tr> </thead> <tbody> <tr> <td>Always</td> <td>Outputs not available</td> </tr> </tbody> </table>		DLL_FREQUENCY_MODE Attribute		LOW	HIGH	Always	Outputs not available
DLL_FREQUENCY_MODE Attribute								
LOW	HIGH							
Always	Outputs not available							
CLKDV	<table border="1"> <thead> <tr> <th colspan="2">DLL_FREQUENCY_MODE Attribute</th> </tr> <tr> <th>LOW</th> <th>HIGH</th> </tr> </thead> <tbody> <tr> <td>Always</td> <td>When CLKDV_DIVIDE attribute is an integer value</td> </tr> </tbody> </table>		DLL_FREQUENCY_MODE Attribute		LOW	HIGH	Always	When CLKDV_DIVIDE attribute is an integer value
DLL_FREQUENCY_MODE Attribute								
LOW	HIGH							
Always	When CLKDV_DIVIDE attribute is an integer value							
CLKFX CLKFX180	Always							

The [Quadrant Phase Shifted Outputs](#), CLK0, CLK90, CLK180, and CLK270 have optional clock conditioning, controlled by the DUTY_CYCLE_CORRECTION attribute. By default, the DUTY_CYCLE_CORRECTION attribute is set to TRUE, meaning that these outputs are conditioned to a 50% duty cycle. Setting this attribute to FALSE disables the clock-conditioning feature, in which case the effected clock outputs have roughly the same duty cycle as the incoming clock. Exact replication of the CLKIN duty cycle is not guaranteed.

Phase Shifting – Delaying the Clock by a Fraction of a Period

A DCM also optionally phase shifts an incoming clock, effectively delaying the clock by a fraction of the clock period.

The DCM supports four different types of phase shifting. Each type may be used independently, or in conjunction with other phase shifting modes. The phase shift capabilities for each clock output appear in [Table 13](#).

1. [Half-Period Phase Shifted Outputs](#), most with conditioned 50% duty cycle. A pair of outputs provides a rising edge at 0° and 180° phase shift—or, at the beginning and half-period points during the clock period.
2. [Quadrant Phase Shifted Outputs](#) of 0° (CLK0), 90° (CLK90), 180° (CLK180), and 270° (CLK270), with optional 50% duty-cycle conditioning.
3. [Fixed Fine Phase Shifting](#) of all DCM clock outputs with a resolution of 1/256th of a clock cycle.
4. [Dynamic Fine Phase Shifting](#) of all DCM clock outputs from within the FPGA application, again with a resolution of 1/256th of a clock cycle.

Table 13: Phase Shift Capabilities by Clock Output

Clock Output	Half-Period	Quadrant	Fixed or Dynamic
CLK0	✓	✓	✓
CLK90		✓	✓
CLK180	✓	✓	✓
CLK270		✓	✓
CLK2X	✓		✓
CLK2X180	✓		✓
CLKDV			✓
CLKFX	✓		✓
CLKFX180	✓		✓

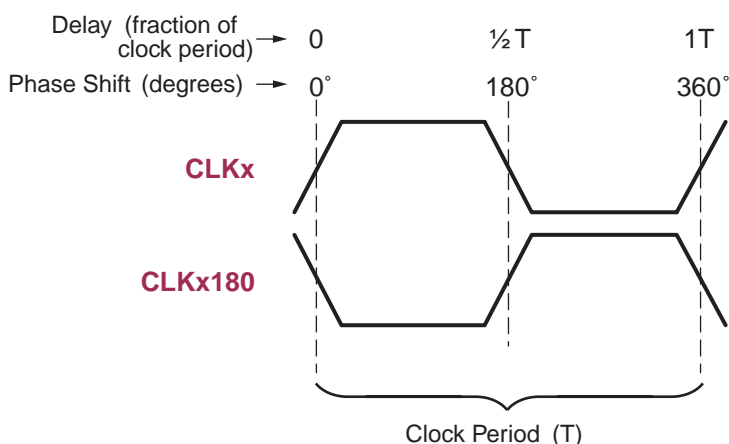
Half-Period Phase Shifted Outputs

The Half-Period Phase Shift outputs provide a non-shifted clock output, and the equivalent clock output but shifted by half a period (180° phase shift). The Half-Period Phase Shift outputs appear in pairs, as shown in Table 14.

Table 14: Half-Period Phase Shifted Outputs

Output Pairs		Comment
No Phase Shift	180° Phase Shift	
CLK0	CLK180	Same frequency as CLKIN input. 50% duty cycle corrected by default and controlled by the DUTY_CYCLE_CORRECTION attribute.
CLK2X	CLK2X180	Outputs from the Clock Doubler (CLK2X, CLK2X180) . Twice the frequency of the CLKIN input, always with 50% duty cycle.
CLKFX	CLKFX180	Outputs from the Frequency Synthesizer (CLKFX, CLKFX180) . Output frequency depends on Frequency Synthesizer attributes. Always with 50% duty cycle.

The Half-Period Phase Shift outputs are ideal for duty-cycle critical applications such as high-speed Dual-Data Rate (DDR) designs and clock mirrors. The Half-Period Phase Shift output pairs provide two clocks, one with a rising edge at the beginning of the clock period, and another rising edge precisely aligned at half the clock period, as shown in Figure 27.



x462_27_061903

Figure 27: Half-Period Phase Shift Outputs

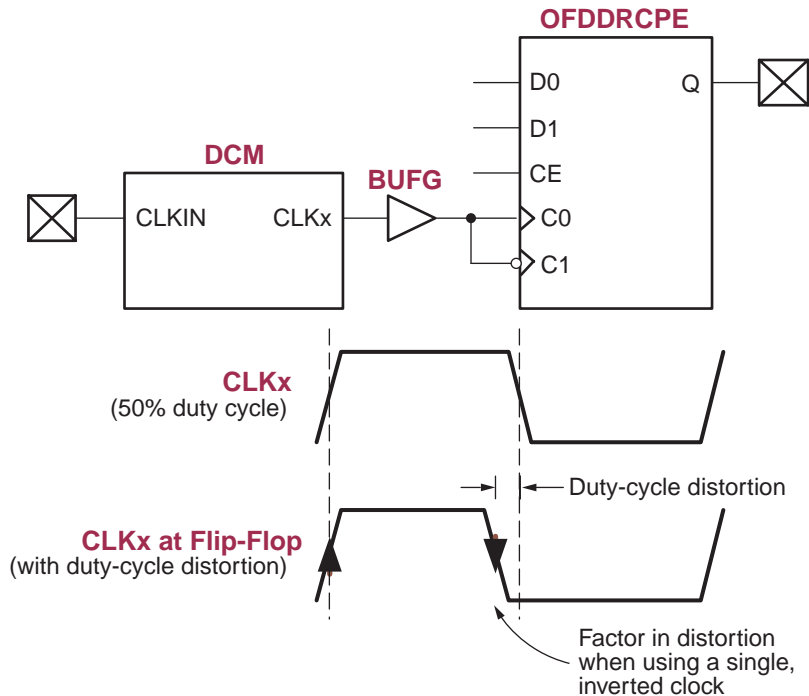
Half-Period Phase Shift Outputs Reduce Duty-Cycle Distortion

When the DCM clock outputs are duty-cycle corrected to 50%, it appears that the 180° phase-shifted clock is just an inverted version on the non-shifted clock. For low-frequency applications, this is essentially true.

However, at very high operating frequencies, duty-cycle distortion—due to differences in rise and fall times of individual transistors—becomes relevant within the FPGA device. Starting with a 50% clock cycle, such distortion causes differences between the clock High and clock Low times, which is consistent from cycle to cycle.

Dual-Data Rate (DDR) Clocking Example

In [Figure 28](#), a single DCM clock output, CLKx, drives both clocks on a Dual-Data Rate (DDR) output flip-flop. One DDR clock input uses the clock output as is, the other input inverts the clock within the DDR flip-flop. The CLKx output from the DCM has a 50% duty cycle, but after traveling through the FPGA's clock network, the duty cycle becomes slightly distorted. In this exaggerated example, the distortion truncates the clock High time and elongates the clock Low time. Consequently, the C1 clock input triggers slightly before half the clock period. At lower frequencies, this distortion is usually negligible. However, high-performance DDR-based systems require precise clocking.



x462_28_061903

Figure 28: Dual-Data Rate (DDR) Output Using Both Edges of a Single Clock Induces Duty-Cycle Distortion

[Figure 29](#) shows a slightly modified circuit compared to [Figure 28](#). In this case, the DCM provides both a non-shifted and a 180° phase-shifted output to the DDR output flip-flop. The CLKx clock signal precisely triggers the DDR flip-flop's C0 input at the start of the clock period. Similarly, the CLKx180 clock signal precisely triggers the DDR flip-flop's C1 input halfway through the clock period. The cost of this approach is an additional global buffer and global clock line, but it potentially reduces the potential duty-cycle distortion by approximately 300 ps (this value is an estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value).

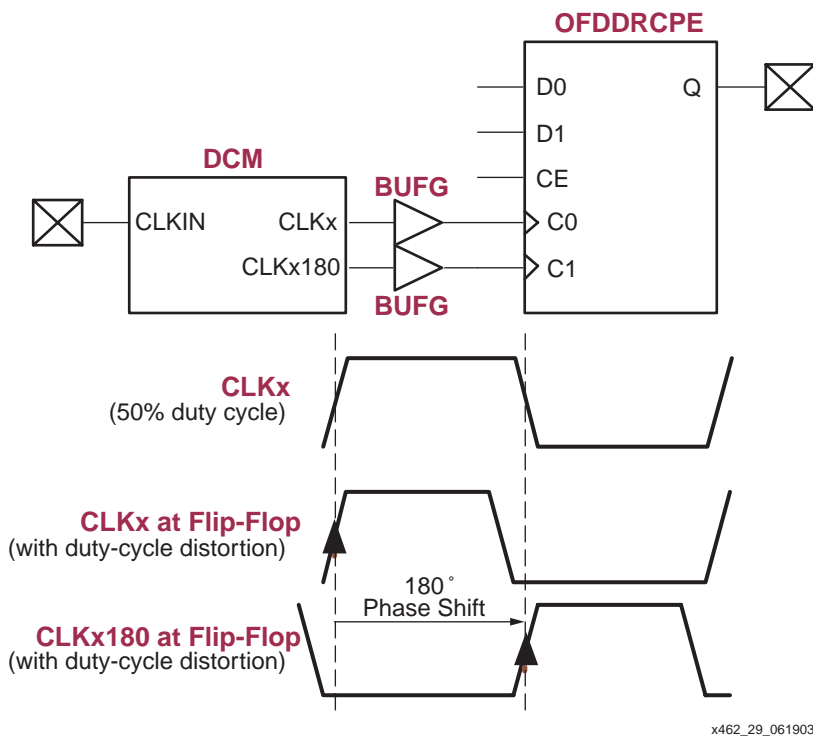


Figure 29: Using Half-Period Phase Shift Outputs Reduces Potential Duty-Cycle Distortion

Table 15 shows the specified duty-cycle distortion values as measured using DDR output flip-flops and LVDS outputs. There may be additional distortion on other output types caused by asymmetrical rise and fall times, which can be simulated using IBIS.

Table 15: Duty-Cycle Distortion Parameters

Parameter	Description	Estimated Value
T _{D_{CD}_CLK0}	Duty-cycle distortion when local inversion provides negative-edge clock to DDR element in an I/O block. See Figure 28.	~400 ps*
T _{D_{CD}_CLK180}	Duty-cycle distortion when DCM CLKx180 output provides clock to DDR element in an I/O block. See Figure 29.	~60 ps*

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

Quadrant Phase Shifted Outputs

The Quadrant Phase Shift outputs shift the CLKIN input, each by a quarter period, as shown in Figure 30 and Table 17. Because the Quadrant Phase Shift outputs require a feedback path back to the CLKFB input, the CLK0 output is phase aligned to the rising edge of the CLKIN input. The CLK90 output is phase shifted 90° from the CLKIN input, and so forth.

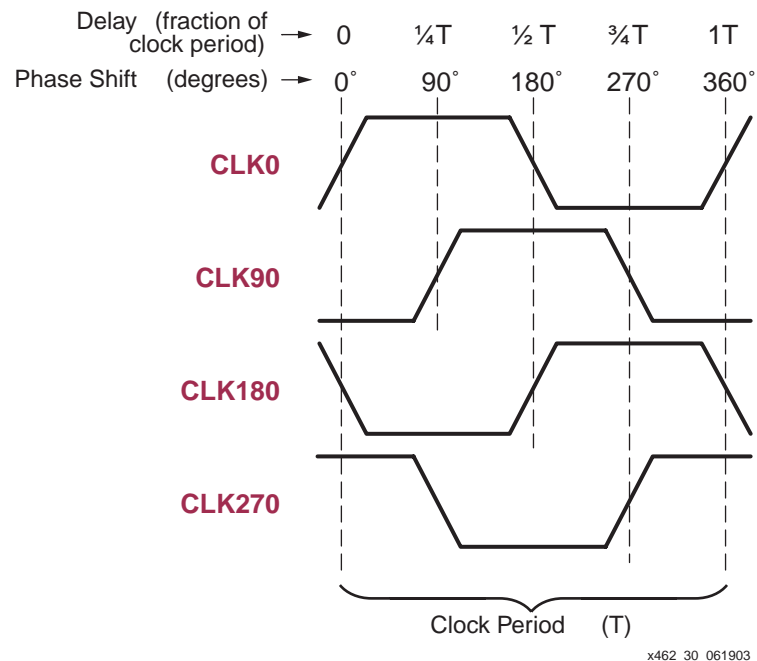


Figure 30: Quadrant Phase Shift Outputs Shift CLKIN, Each by a Quarter Period (Shown with Duty-Cycle Correction Enabled)

Output Availability Depends on DLL Frequency Mode

The availability of the Quadrant Phase Shift outputs depends on the DLL's frequency mode, controlled by the `DLL_FREQUENCY_MODE` attribute. All four Quadrant Phase Shift outputs are available in low-frequency mode (`DLL_FREQUENCY_MODE=LOW`), as shown in Table 16. Only the CLK0 and CLK180 outputs are available in both modes.

Table 16: Quadrant Phase Shift Output Availability by DLL Frequency Mode

Output	DLL_FREQUENCY_MODE	
	LOW	HIGH
CLK0	✓	✓
CLK90	✓	
CLK180	✓	✓
CLK270	✓	

Optional 50/50 Duty Cycle Correction

As a group, the Quadrant Phase Shift outputs are optionally conditioned to a 50% duty cycle, controlled by the `DUTY_CYCLE_CORRECT` attribute. When TRUE, which is the default, all four outputs have a 50% duty cycle. When FALSE, the outputs do not have the same duty cycle as the CLKIN input. See the “Clock Conditioning” section for more information.

Four Phases, Delayed Clock Edges, Phased Pulses

One view of the Quadrant Phase Shift outputs is that each provides a rising clock that is delay one quarter period from the preceding pulse, as shown in Table 17. These outputs provide flexible timing for such applications as memory interfaces and peripheral control.

With the duty-cycle correction option enabled (`DUTY_CYCLE_CORRECTION = TRUE`), there are other ways to view the outputs. For example, the outputs also provide falling-edge clocks

separated by a quarter phase. Again, see [Table 17](#). Similarly, each output produces a High-going pulse, and a Low-going pulse, both half a period wide. For example, the CLK90 output shown in [Figure 30](#) produces a High-going pulse, centered within the CLK0 clock period.

Table 17: Quadrant Phase Shift Outputs and Characteristics (DUTY_CYCLE_CORRECTION=TRUE)

DCM Output	Phase Shift	Delayed by Period Fraction	Rising Edge	Falling Edge	Comment
CLK0	0°	0	0	½T	Deskewed input clock, no phase shift
CLK90	90°	¼T	¼T	¾T	High-going pulse, ½T wide, in middle of period
CLK180	180°	½T	½T	0T	Inverted CLK0, rising clock edge in middle of period
CLK270	270°	¾T	¾T	¼T	Low-going pulse, ½T wide, in middle of period

Fine Phase Shifting

The DCM provides additional controls over clock skew using fine phase shifting. Fine-phase adjustment affects all nine DCM output clocks simultaneously. The fine phase shift capability requires the DCM's DLL functional unit. Consequently, clock feedback via the CLKFB input is required.

Physically, the fine phase shift control adjusts the phase relationship between the rising edges of the CLKIN and CLKFB inputs. The net effect, however, is the all DCM outputs are phase shifted with relation to the CLKIN input.

By default, fine phase shifting is disabled (`CLKOUT_PHASE_SHIFT=NONE`), meaning that the clock outputs are phase aligned with CLKIN. In this case, there is no skew between the input clock, CLKIN, and the feedback clock, measured at the appropriate feedback point (see [“Feedback from a Reliable Source”](#) section). When fine phase shifting is enabled, the output clock edges can be phase shifted so that they are advanced or are retarded compared to the CLKIN input, as shown in [Figure 31](#).

There are two fine phase shift modes as described below. Both are commonly used in high-speed data communications applications. See the [“Source Synchronous”](#) section.

1. **Fixed Fine Phase Shift** mode sets the phase shift value at design time. The phase shift value is loaded during the FPGA configuration process and cannot be changed by the application.
2. **Dynamic Fine Phase Shift** mode has an initial phase shift value, similar to Fixed Fine Phase Shift, which is set during FPGA configuration. However, the phase shift value can be changed by the application after the DCM's LOCKED output goes High.

Fixed Fine Phase Shifting

In Fixed Fine Phase Shift mode, the phase shift value is specified at design time and set during the FPGA configuration process. The application cannot change the value during run time.

Two attributes control this mode. The `CLKOUT_PHASE_SHIFT` attribute is set to FIXED, and the `PHASE_SHIFT` attribute controls the amount of phase shift. If `PHASE_SHIFT` is 0, then the output clocks and the CLKIN input are phase aligned, as shown in [Figure 31](#). If `PHASE_SHIFT` is a negative integer, then the clock output(s) are phase shifted before CLKIN. If `PHASE_SHIFT` is a positive integer, then the clock output(s) are phase shifted after CLKIN.

Fixed Fine Phase Shift Range

The `PHASE_SHIFT` attribute is always an integer value, ranging between -255 and +255. However, the actual limits may be lower depending on the CLKIN input frequency, as described below.

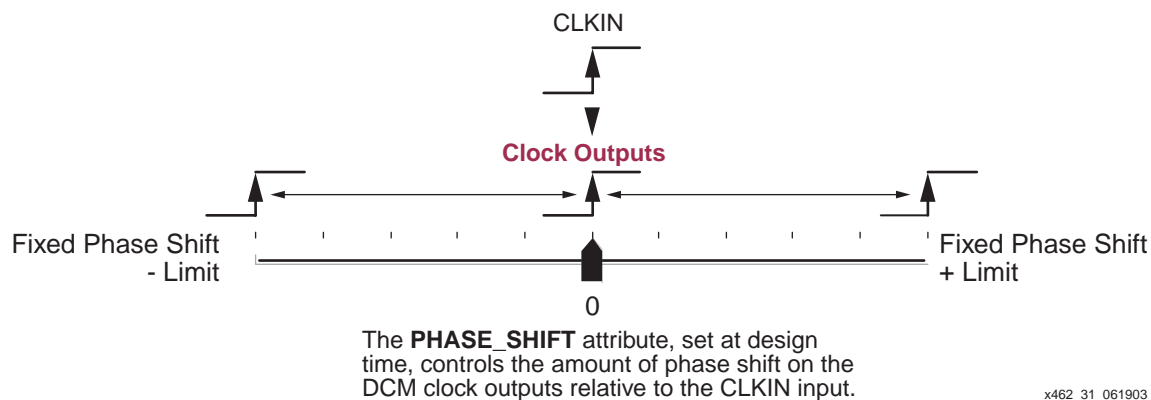


Figure 31: Fixed-Value Fine Phase Shift Control

The minimum and maximum limit of the PHASE_SHIFT attribute depend on two values.

1. The period of the CLKIN input, T_{CLKIN} , measured in nanoseconds.
2. The value of the FINE_SHIFT_RANGE specification for the Spartan-3 device and speed grade, found in the Spartan-3 Data Sheet, Module 3. FINE_SHIFT_RANGE is the total delay achievable by the phase shift delay line, which is a function of the number of delay taps used in the circuit. The actual delay line may be longer than FINE_SHIFT_RANGE, but only the delay up to FINE_SHIFT_RANGE is guaranteed.

Using these two values, calculate the SHIFT_DELAY_RATIO using Equation 1. The limits for the PHASE_SHIFT attribute are different, depending on whether the result is less than or if it is greater than or equal to one.

$$SHIFT_DELAY_RATIO = \frac{FINE_SHIFT_RANGE}{T_{CLKIN}} \quad \text{Eq. 1}$$

SHIFT_DELAY_RATIO < 1

If the clock period is longer than the specified FINE_SHIFT_RANGE, then the SHIFT_DELAY_RATIO < 1, meaning that maximum fine phase shift is limited by FINE_SHIFT_RANGE. When SHIFT_DELAY_RATIO < 1, then the PHASE_SHIFT limits are set according to Equation 2:

$$PHASE_SHIFT_{LIMITS} = \pm \left[INTEGER \left(256 \cdot \frac{FINE_SHIFT_RANGE}{T_{CLKIN}} \right) \right] \quad \text{Eq. 2}$$

For example, assume that F_{CLKIN} is 75 MHz ($T_{CLKIN} = 13.33$ ns) and FINE_SHIFT_RANGE is 10.00 ns⁽¹⁾. In this case, the PHASE_SHIFT value is limited to ± 191 .

Consequently, the phase shift value when SHIFT_DELAY_RATIO < 1 is shown by Equation 3. To determine the phase shift resolution, set PHASE_SHIFT = 1.

$$T_{PhaseShift} = \left(\frac{PHASE_SHIFT}{|PHASE_SHIFT_{LIMITS}|} \right) \cdot FINE_SHIFT_RANGE \quad \text{Eq. 3}$$

1. Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

SHIFT_DELAY_RATIO ≥ 1

By contrast, if the clock period is shorter than the specified FINE_SHIFT_RANGE, then the SHIFT_DELAY_RATIO ≥ 1, meaning that maximum fine phase shift is limited to ±255.

$$PHASE_SHIFT_{LIMITS} = \pm 255 \tag{Eq. 4}$$

Consequently, the phase shift value when SHIFT_DELAY_RATIO ≥ 1 is shown by Equation 5. To determine the phase shift resolution, set PHASE_SHIFT = 1.

$$T_{PhaseShift} = \left(\frac{PHASE_SHIFT}{256} \right) \cdot T_{CLKIN} \tag{Eq. 5}$$

Minimum Phase Shift Size

The minimum phase shift size is controlled by the greater of two limiting factors.

1. The minimum delay-line tap resolution, listed as the DCM_TAP_MIN specification in the Spartan-3 Data Sheet (estimated at ~30ps), or
2. 1/256th of the clock period.

Other Design Considerations

In Fixed Phase Shift mode, the Dynamic Phase Shift control inputs must be tied to GND, which DCM Wizard does automatically.

DCM Wizard

To use Fixed Phase Shift mode, click **Fixed** in the Phase Shift section of DCM Wizard’s **General Setup** panel, shown in Figure 32. This action sets the CLKOUT_PHASE_SHIFT attribute to FIXED.

Enter the **Phase Shift Value**, which must be an integer within the limits described above. This action sets the PHASE_SHIFT attribute value. DCM Wizard checks that the phase shift value is within the limits.

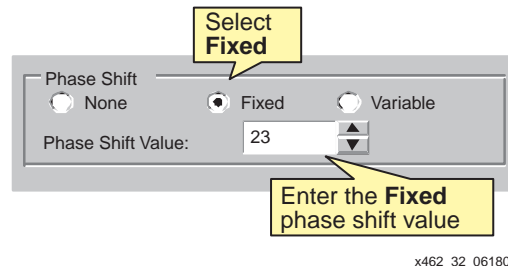


Figure 32: Selecting Fixed Fine Shift Mode

Dynamic Fine Phase Shifting

In Dynamic Fine Phase Shift mode, the initial skew or phase shift is still controlled by the PHASE_SHIFT attribute during configuration, just as it is for Fixed Fine Shift mode. However, in dynamic mode, the FPGA application can adjust the current phase shift location after the DCM’s LOCKED output goes High using the Dynamic Fine Phase Shift control inputs, PSEN, PSCLK, and PSINCDEC.

Operation

Use the phase shift control inputs to adjust the current phase shift value, as shown in Figure 33. The rising edge of PSCLK synchronizes all dynamic phase shift operations. A valid operation starts by asserting the PSEN enable input for one and only one PSCLK clock period. Asserting PSEN for more than one rising PSCLK clock edge may cause undesired behavior.

The value on the PSINCDEC increment/decrement control input determines the phase shift direction. When **PSINCDEC** is High, the present dynamic phase shift value is incremented by one unit. Similarly, when **PSINCDEC** is Low, the present dynamic phase shift value is decremented by one unit.

The actual phase shift operation timing varies and the operation completes when the DCM asserts the **PSDONE** output High for a single **PSCLK** clock period. Between enabling **PSEN** until **PSDONE** is asserted, the DCM output clocks slide, bit by bit, from their original phase shift value to their new phase shift value. During this time, the DCM remains locked on the incoming clock and continues to assert its **LOCKED** output.

When or after **PSDONE** is asserted High, the next phase shift operation can be initiated.

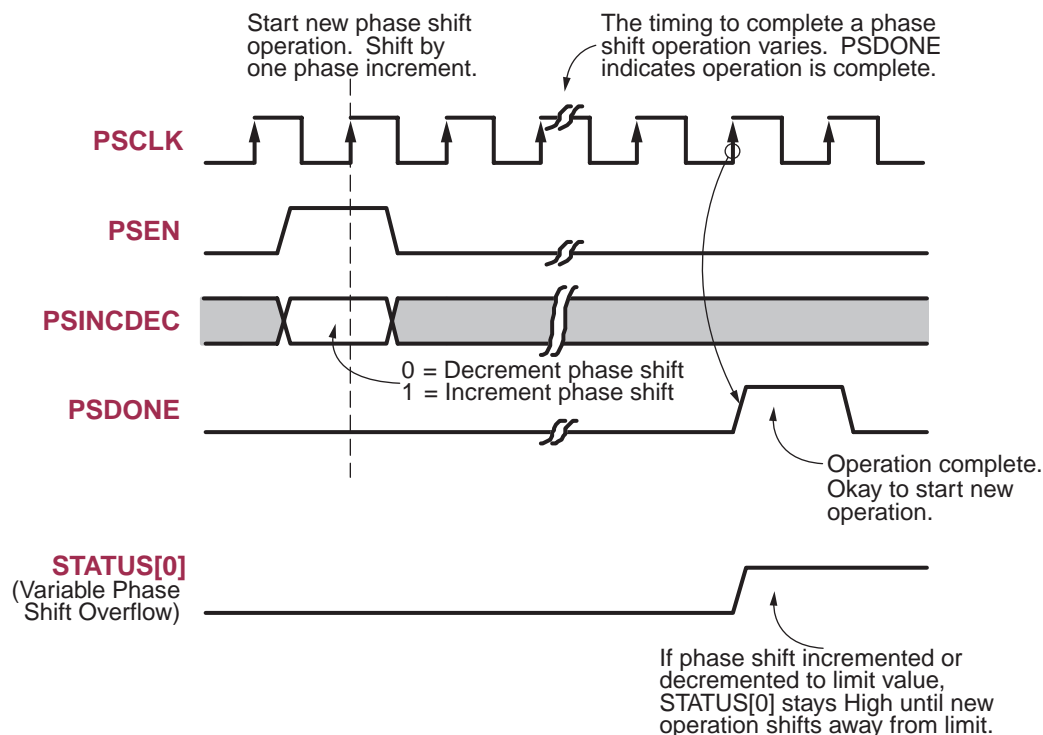
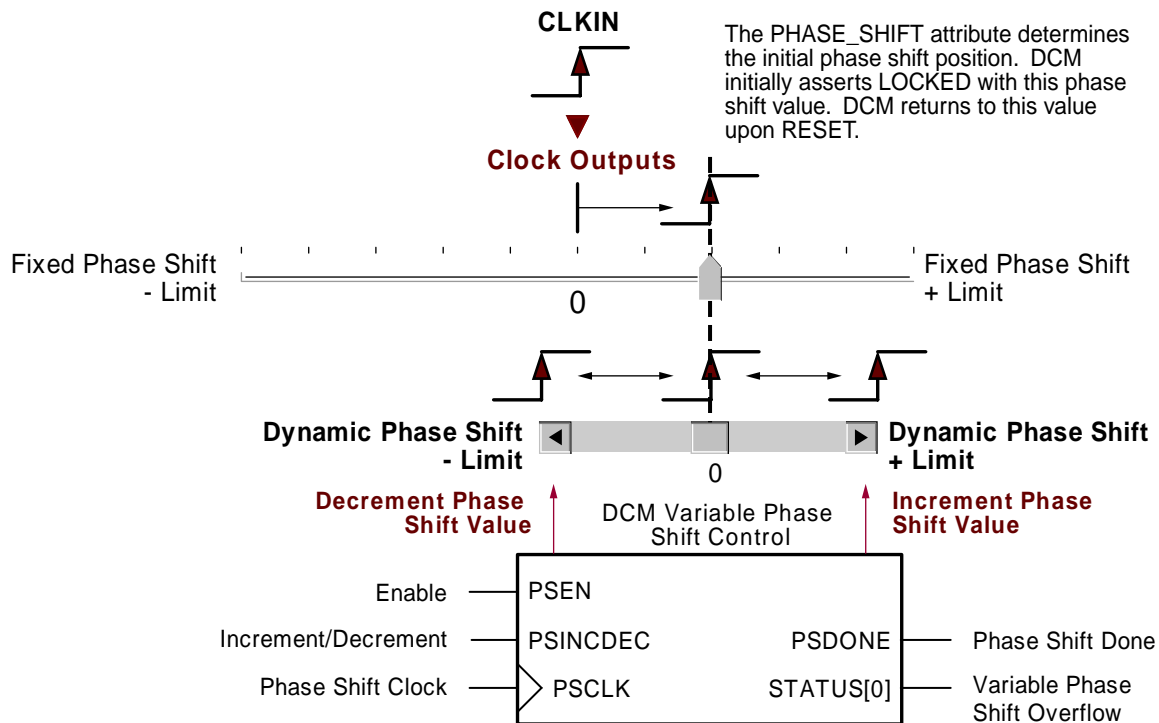


Figure 33: Dynamic Fine Phase Shift Control Interface

To enable Dynamic Fine Phase Shift mode, set the **CLKOUT_PHASE_SHIFT** attribute to **VARIABLE**. The **PHASE_SHIFT** attribute value sets the initial phase shift location, established after FPGA configuration. The FPGA application can dynamically adjust the skew or phase shift on the DCM's output clocks after the DCM's **LOCKED** output goes High. If the DCM is reset, the **PHASE_SHIFT** value reverts to its initial configuration value.



x462_34_061903

Figure 34: Dynamic Phase Shift Controls

Dynamic Fine Phase Shift Range

Just as the PHASE_SHIFT has a minimum and maximum phase shift limit, so does the Dynamic Phase Shift, as shown in Figure 34. Similarly, the limit depends on the ratio of the FINE_SHIFT_RANGE versus the input clock period, as calculated by the SHIFT_DELAY_RATIO equation above.

SHIFT_DELAY_RATIO < 2

If the specified FINE_SHIFT_RANGE value is less than twice the clock period (SHIFT_DELAY_RATIO < 2), then the maximum dynamic fine phase shift value is limited by FINE_SHIFT_RANGE, the maximum delay tap value. When SHIFT_DELAY_RATIO < 2, then the dynamic phase shift limits are set according to Equation 6.

$$DynamicPhaseShift_{LIMITS} = \pm \left[\text{INTEGER} \left(128 \cdot \frac{FINE_SHIFT_RANGE}{T_{CLKIN}} \right) \right] \text{ Eq. 6}$$

For example, assume that F_{CLKIN} is 75 MHz (T_{CLKIN} = 13.33 ns) and FINE_SHIFT_RANGE is 10.00 ns⁽¹⁾. In this case, the Dynamic Phase Shift value is limited to ±96.

1. Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for correct specified value.

When $SHIFT_DELAY_RATIO < 2$, the dynamic phase shift value is shown by [Equation 7](#). To determine the dynamic phase shift resolution, set Dynamic Phase Shift = 1.

$$T_{PhaseShift} = \left(\frac{DynamicPhaseShift}{|DynamicPhaseShift_{LIMITS}|} \right) \cdot FINE_SHIFT_RANGE \quad \text{Eq. 7}$$

$SHIFT_DELAY_RATIO \geq 2$

If the incoming clock period is less than or equal to half the specified $FINE_SHIFT_RANGE$, then the $SHIFT_DELAY_RATIO \geq 2$, meaning that maximum fine phase shift is limited to ± 255 .

$$DynamicPhaseShift_{LIMITS} = \pm 255 \quad \text{Eq. 8}$$

Consequently, the phase shift value when $SHIFT_DELAY_RATIO \geq 2$ is shown by [Equation 9](#). To determine the phase shift resolution, set $PHASE_SHIFT = 1$.

$$T_{PhaseShift} = \frac{DynamicPhaseShift}{256} \cdot T_{CLKIN} \quad \text{Eq. 9}$$

Controls

As shown in [Figure 33, page 43](#) and [Figure 34, page 44](#), the DCM's Dynamic Phase Shift control signals allow the FPGA application to adjust the present phase relationship between the $CLKIN$ input and the DCM clock outputs. [Table 18](#) shows the detailed relationship between control inputs, the current and next phase relationship, how the operation affects the delay tap, and the control outputs.

Table 18: Dynamic Phase Shifter Control (assumes no internal inversion)

PSEN	PSINC-DEC	PSCLK	Current Phase Shift	Next Phase Shift	Delay Line	PSDONE	STATUS[0] (Overflow)	Operation
0	X	X	X	No change	No change	?	?	Dynamic phase shift disabled.
1	0	↑	> -Limit	Current – 1	Current – 1	1*	0	Decrement phase shift and phase pointer.
1	0	↑	≤ -Limit and > -255	Current – 1	No Change	1*	1	End of delay line. No phase shift change. Phase pointer decremented.
1	0	↑	-255	-255	No Change	1*	1	End of delay line. No phase shift change. Phase pointer at limit.
1	1	↑	< +Limit	Current + 1	Current + 1	1*	0	Increment phase shift and phase pointer.
1	1	↑	≥ +Limit and < +255	Current + 1	No Change	1*	1	End of delay line. No phase shift change. Phase pointer incremented.
1	1	↑	+255	+255	No Change	1*	1	End of delay line. No phase shift change. Phase pointer at limit.

Notes:

- X = don't care.
- ? = indeterminate, depends on current application state.
- 1* = PSDONE asserted High for one PSCLK period.
- Limit = minimum delay line position.
- +Limit = maximum delay line position.
- Assert PSEN for only one PSCLK cycle.

When PSEN is Low, the Dynamic Phase Shifter is disabled and all other inputs are ignored. All present shift values and the delay line position remain unchanged.

If the delay line has not reached its limits (-Limit or -255 when decrementing, +Limit or +255 when incrementing), then the FPGA application can change the existing phase shift value by asserting PSEN High and the appropriate increment/decrement value on PSINCDEC before the next rising edge of PSCLK. The phase shift value increments or decrements as instructed. At the end of the operation, PSDONE goes High for a single PSCLK period indicating that the phase shift operation is complete. STATUS[0] remains Low because no phase shift overflow condition occurred.

When the DCM is incremented beyond +255 or below -255, the delay line position remains unchanged at its limit value of +255 or -255 and no phase change occurs. STATUS[0] goes High, indicating a dynamic phase shift overflow. When a new phase shift operation changes the value in the opposition direction—i.e., away from the limit value—STATUS[0] returns Low.

If the phase shift does not reach +255 or -255, but the phase shift exceeds the delay-line range—indicated by +Limit and -Limit in Table 18—then no phase change occurs. However, STATUS[0] again goes High. The STATUS[0] output indicates when the delay tap reaches the end of the delay line. In the FPGA application, however, use the limit value calculated using either Equation 6 or Equation 8. The calculated delay limit is a guaranteed value. A specific device, due to processing, voltage, or temperature, may have a longer line delay, but this cannot be guaranteed from device to device. The phase shift value—but not the delay line positions—continues to increment or decrement until it reaches its +255 or -255 limit. When a new phase shift operation changes the value in the opposition direction—i.e., away from the limit value—the STATUS[0] signal returns Low. The phase shift value is incremented or decremented back to a value that corresponds to a valid absolute delay in the delay line.

DCM Wizard

The Dynamic Phase Shift options are part of the DCM Wizard's General Setup panel, shown in Figure 35. To enable dynamic fine phase shifting, select Variable, as shown in Figure 35. Enter an initial Phase Shift Value in the text box provided. The initial value behaves exactly like the Fixed Fine Phase Shifting mode described above.

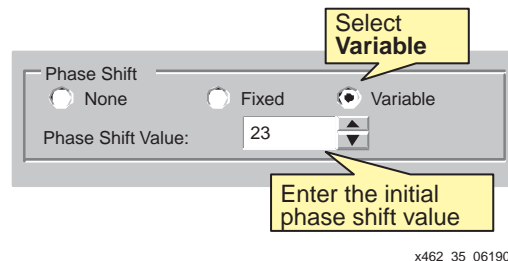


Figure 35: Selecting Dynamic Fine Phase Shift Mode in DCM Wizard

Choosing Variable mode also enables the Dynamic Phase Shift control signals, PSEN, PSINCDEC, PSCLK, and PSDONE, as shown in Figure 36. Check the input and output boxes to use these controls in the FPGA application. Also, check the STATUS output box to enable the STATUS[0] signal. STATUS[0] indicates when the dynamic phase shifter reaches its maximum or minimum limit value.

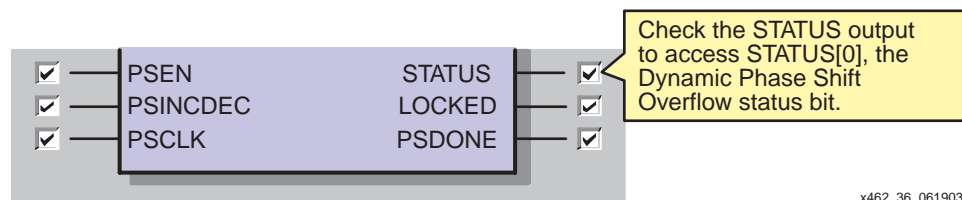


Figure 36: Check the Dynamic Phase Shift Control Outputs in DCM Wizard

Example Applications

See application note XAPP268 for an example of how to use the Dynamic Phase Shift function to perform dynamic phase alignment.

- XAPP268: *Dynamic Phase Alignment*
<http://www.xilinx.com/xapp/xapp268.pdf>

Clock Multiplication, Clock Division, and Frequency Synthesis

A DCM provides flexible methods for generating new clock frequencies—one of the most common DCM applications. Spartan-3 DCMs provide up to three independent frequency synthesis functions, listed below, and in Figure 37, and summarized in Table 19. An application may use one or all three functions simultaneously. Detailed descriptions for each function follows.

1. A **Clock Doubler (CLK2X, CLK2X180)** that doubles the frequency of the input clock.
2. A **Clock Divider (CLKDV)** that reduces the input frequency by a fixed divider value.
3. A **Frequency Synthesizer (CLKFX, CLKFX180)** for generating a completely new frequency from an incoming clock frequency.

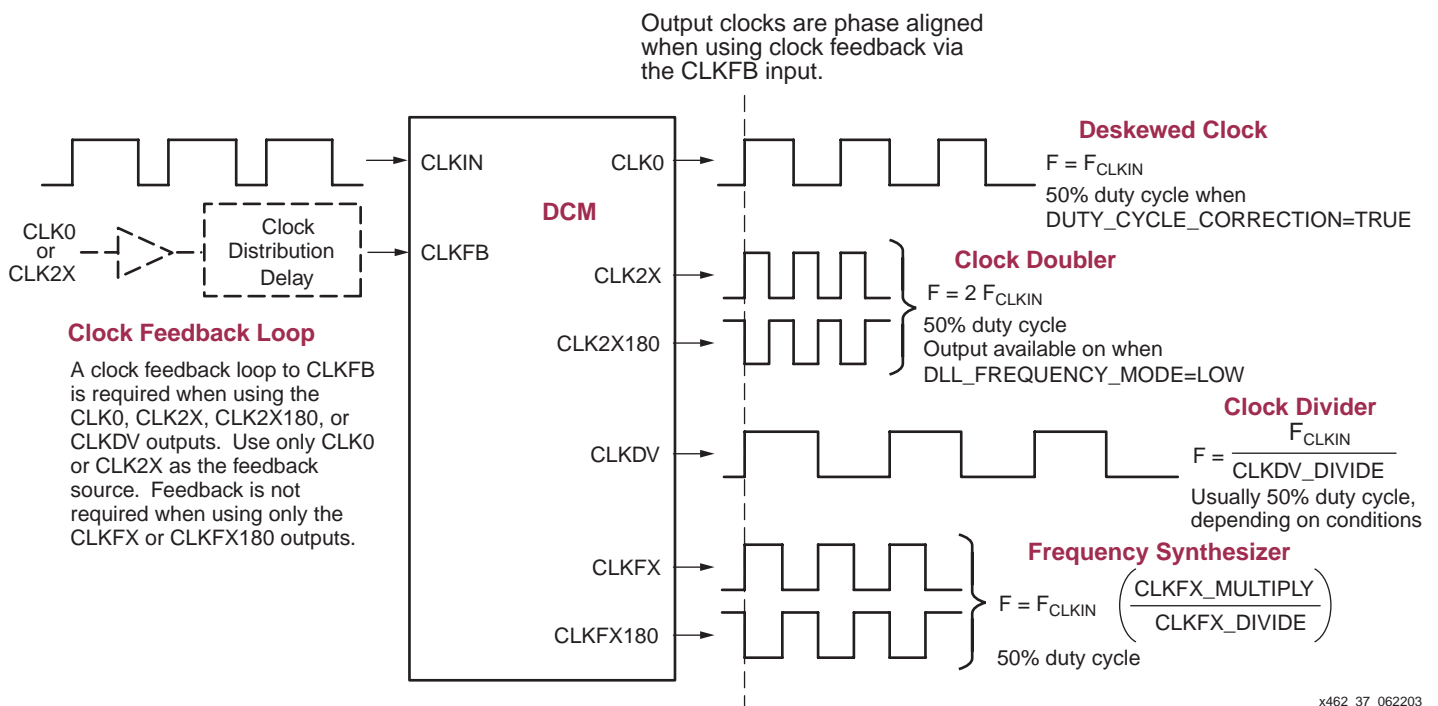


Figure 37: Clock Synthesis Options

All the frequency synthesis outputs, except CLKDV, always have a 50/50 duty cycle. CLKDV usually has a 50% duty cycle except when dividing by a non-integer value at high frequency, as shown in Table 23. The **Clock Doubler (CLK2X, CLK2X180)** circuit is not available at high frequencies.

All the DCM clock outputs, except **CLKFX** and **CLKFX180**, are generated by the DCM's **Delay-Locked Loop (DLL)** unit and consequently require some form of clock feedback to the CLKFB pin. The DCM's **Digital Frequency Synthesizer (DFS)** unit generates the **CLKFX** and **CLKFX180** clock outputs. If the application uses only the **CLKFX** or **CLKFX180** outputs, then the feedback path may be eliminated, which also extends the DCM's operating range. The **Frequency Synthesizer** has a feedback path within the DCM, based on CLKIN.

Table 19: DCM Frequency Synthesis Options

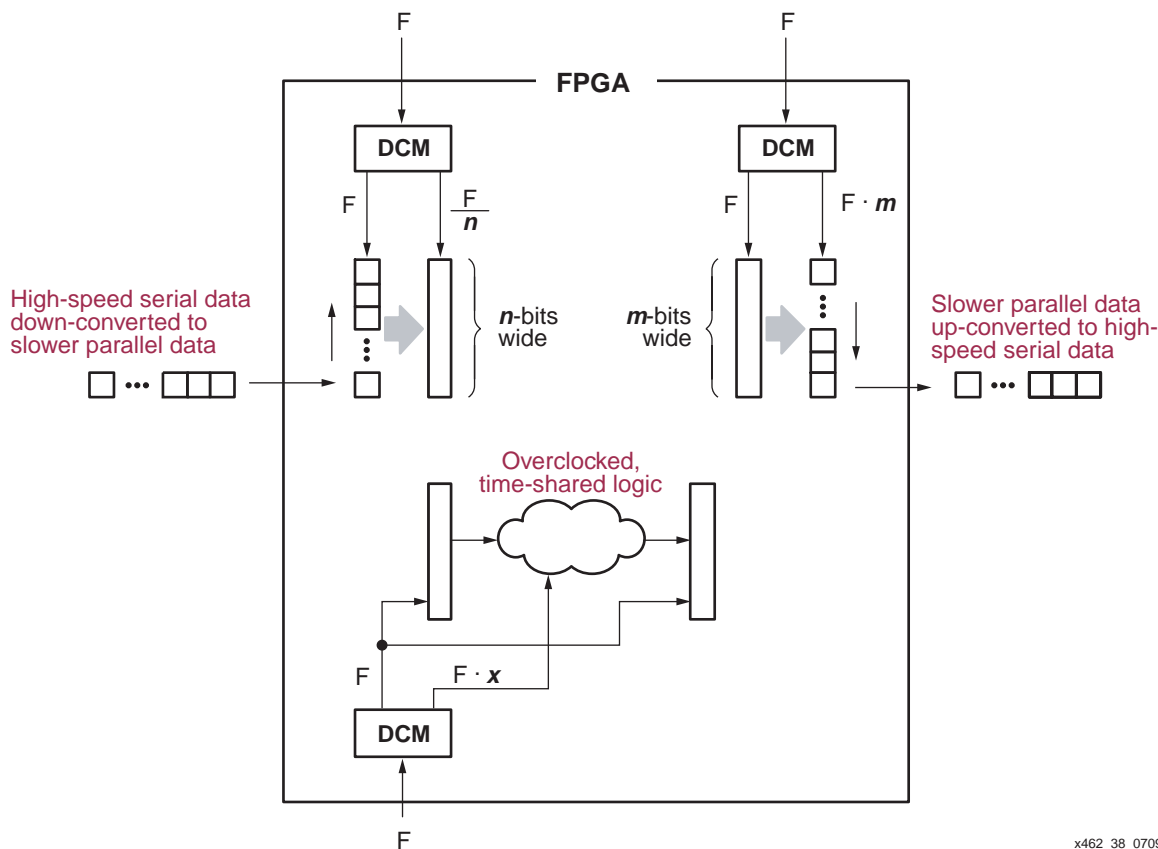
Function	DCM Output(s)	Frequency	DCM Functional Unit	Feedback Required?	50% Duty Cycle?
Deskewed Clock	CLK0	F_{CLKIN}	DLL	Yes	When DUTY_CYCLE_CORRECTION = TRUE
Clock Doubler	CLK2X CLK2X180	$2 \cdot F_{CLKIN}$	DLL	Yes	Always
Clock Divider	CLKDV	$\frac{F_{CLKIN}}{CLKDV_DIVIDE}$	DLL	Yes	Always except when dividing by non-integer value in high-frequency mode
Frequency Synthesizer	CLKFX CLKFX180	$F_{CLKIN} \cdot \left(\frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE} \right)$	DFS	Optional. No feedback extends clock input frequency limits.	Always

If clock feedback is used, then all the output clocks are phase aligned. Obviously, full clock-edge alignment across all the DCM outputs occurs only occasionally because some of the outputs are divided clock values. For example, the CLKDV output is aligned to CLKIN and CLK0 every CLKDV_DIVIDE cycles. Similarly, the CLK2X output is aligned to CLK0 every other clock cycle. The CLKFX output is aligned to CLKIN every CLKFX_DIVIDE cycles of CLKIN and every CLKFX_MULTIPLY cycles of CLKFX.

Frequency Synthesis Applications

The potential applications for frequency synthesis are almost boundless. Some example applications include the following.

- Generating a completely new clock frequency for the FPGA and external logic using an available clock frequency on the board.
- Generate a high-frequency internal clock from a slower external clock source to reduce system EMI.
- Dividing a high-speed serial data clock to process data in parallel within the FPGA, as shown in [Figure 38](#).
- Multiplying a parallel data clock before converting to a high-speed serial data format, also shown in [Figure 38](#).
- Multiplying an input clock to overclock internal logic to reduce resources by time-sharing logic when implementing moderately fast functions.



x462_38_070903

Figure 38: Common Applications of Frequency Synthesis

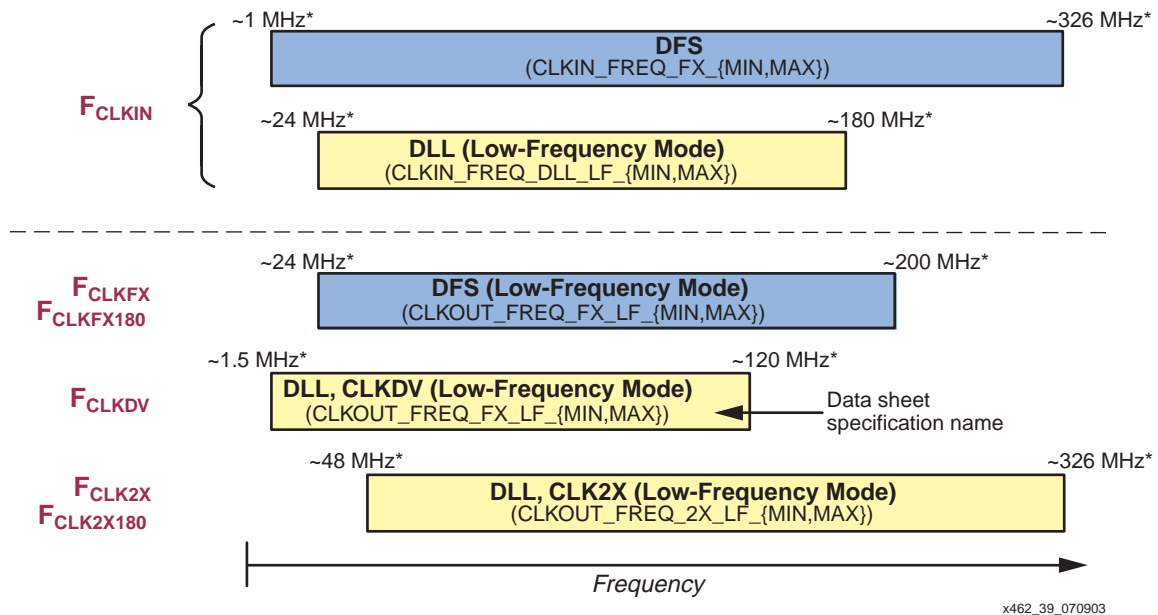
Input and Output Clock Frequency Restrictions

The input and output clock frequency restrictions for frequency synthesis depend on which DCM clock outputs are used. For example, the [CLKFX](#) and [CLKFX180](#) outputs only use the DCM's Digital Frequency Synthesis (DFS) unit. All the other clock outputs use the DCM's Delay-Locked Loop (DLL) unit. The DLL unit has tighter frequency restrictions than the DFS. Consequently, operating the DFS unit without the DLL allows a wider frequency operating range. When using both the DFS and DLL units, the DLL frequency range limits the application.

Also, both the DLL and DFS have a low- and a high-frequency operating mode and the mode settings determine the allowable frequency operating range.

A valid DCM design requires that the CLKIN frequency be within the operating range specified in the Spartan-3 Data Sheet, Module 3. Likewise, the output frequency for any of the clock outputs used must fall within their respective specified operating range.

[Figure 39](#) shows how the various clock input and clock output specifications line up by frequency range. Only the low-frequency operating modes are shown. The data sheet specification for each name is provided within the shaded boxes. [Table 20](#) provides example DCM applications and how the frequency restrictions apply.



* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

Figure 39: Input and Output Clock Frequency Restrictions (Low-Frequency Mode)

Table 20: DCM Frequency Restriction Examples

Input Frequency	Output Frequency	Comments
1.2 MHz	12.8 MHz	Not possible in a single DCM. F_{CLKIN} is within acceptable range for DFS unit, but F_{CLKFX} requires at least a 24MHz output frequency.
1.2 MHz	32.4 MHz	Possible in a single DCM using DFS unit. Set $CLKFX_MULTIPLY=27$. F_{CLKFX} is within the DFS output frequency range.
25 MHz	2.5 MHz 30 MHz	Possible in a single DCM using both the DFS and DLL units. Use the CLKDV output for a 2.5MHz signal, setting $CLKDV_DIVIDE=10$. Use the CLKFX output for a 30MHz signal, setting $CLKFX_MULTIPLY=6$ and $CLKFX_DIVIDE=5$. All input and output frequencies are within appropriate ranges.

Clock Doubler (CLK2X, CLK2X180)

The Clock Doubler unit doubles the frequency of the incoming CLKIN input, as summarized in [Table 21](#). The Clock Doubler is part of the DLL functional unit and requires a clock feedback path back to CLKFB from either the CLK0 or CLK2X output. The outputs from the Clock Doubler are CLK2X and CLK2X180. Both outputs are always conditioned to a 50% duty cycle. Both have the same output frequency but CLK2X180 is 180° phase shifted from CLK2X, essentially inverting the CLK2X output. Having both phases is essential for high-performance Dual-Data Rate (DDR) or clock forwarding applications.

The CLK2X and CLK2X180 outputs are only available when the [DLL_FREQUENCY_MODE](#) attribute is LOW. If required by the application, reduce the CLKIN input frequency using the optional divide-by-two feature (see ["Advanced Options"](#)).

Table 21: Clock Doubler Summary

DCM Output(s)	CLK2X CLK2X180		
Output Frequency	$2 \cdot F_{CLKIN}$		
DCM Functional Unit	Delay-Locked Loop (DLL)		
Feedback Required?	Yes		
50% Duty Cycle?	Yes		
Controlling Attributes			
DLL_FREQUENCY_MODE	The CLK2X and CLK2X180 outputs are only valid when DLL_FREQUENCY_MODE = LOW		
CLKIN	The CLKIN frequency limits are determined by the DLL_FREQUENCY_MODE attribute. The Clock Doubler outputs are not available in high-frequency mode but may be required for other DCM clock outputs.		
	DLL_FREQUENCY_MODE	Minimum Frequency	Maximum Frequency
	LOW	CLKIN_FREQ_DLL_LF_MIN (~24 MHz)*	CLKIN_FREQ_DLL_LF_MAX (~180 MHz)*
	HIGH	CLKIN_FREQ_DLL_HF_MIN (~48 MHz)*	CLKIN_FREQ_DLL_HF_MAX (~326 MHz)*
CLK2X CLK2X180	The CLKDV frequency limits are determined by the DLL_FREQUENCY_MODE attribute.		
	DLL_FREQUENCY_MODE	Minimum Frequency	Maximum Frequency
	LOW	CLKOUT_FREQ_2X_LF_MIN (48 MHz)*	CLKOUT_FREQ_2X_LF_MAX (~325 MHz)*
	HIGH	Not available	Not available

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

Clock Divider (CLKDV)

The Clock Divider unit, summarized in [Table 22](#), divides the incoming CLKIN frequency by the value specified by the CLKDV_DIVIDE attribute, set at design time. The Clock Divider unit is part of the DLL functional unit and requires a clock feedback path back to CLKFB from either the CLK0 or CLK2X output.

Table 22: Clock Divider Summary

DCM Output(s)	CLKDV		
Output Frequency	$\frac{F_{CLKIN}}{CLKDV_DIVIDE}$		
DCM Functional Unit	Delay-Locked Loop (DLL)		
Feedback Required?	Yes, using either CLK0 or CLK2X output from DCM		
50% Duty Cycle?	Yes, except when <code>DLL_FREQUENCY_MODE=HIGH</code> and <code>CLKDV_DIVIDE</code> is a non-integer value		
Controlling Attributes			
<code>DLL_FREQUENCY_MODE</code>	CLKDV is available in both modes. Potentially affects duty cycle of output (see “ CLKDV Clock Conditioning ”), depending on divider value.		
<code>CLKDV_DIVIDE</code>	Controls the output frequency per the equation above. Legal values include 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, and 16. The DLL locks faster on integer values than on non-integer values. Likewise, integer values result in lower output jitter.		
Frequency Constraints			
CLKIN	The CLKIN frequency limits are determined by the <code>DLL_FREQUENCY_MODE</code> attribute.		
	DLL_FREQUENCY_MODE	Minimum Frequency	Maximum Frequency
	LOW	CLKIN_FREQ_DLL_LF_MIN (24 MHz)*	CLKIN_FREQ_DLL_LF_MAX (~180 MHz)*
	HIGH	CLKIN_FREQ_DLL_HF_MIN (48 MHz)*	CLKIN_FREQ_DLL_HF_MAX (~325 MHz)*
CLKDV	The CLKDV frequency limits are determined by the <code>DLL_FREQUENCY_MODE</code> attribute.		
	DLL_FREQUENCY_MODE	Minimum Frequency	Maximum Frequency
	LOW	CLKOUT_FREQ_DV_LF_MIN (1.5 MHz)*	CLKOUT_FREQ_DV_LF_MAX (~120 MHz)*
	HIGH	CLKOUT_FREQ_DV_HF_MIN (3.0 MHz)*	CLKOUT_FREQ_DV_HF_MAX (~240 MHz)*

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

CLKDV Clock Conditioning

The CLKDV output is conditioned to a 50% duty cycle unless the `DLL_FREQUENCY_MODE` attribute is set to HIGH and `CLKDV_DIVIDE` is a non-integer value. Under these conditions, the CLKDV duty cycle is shown in Table 23. A Spartan-3 DCM requires CLKIN to have at least a 60%/40% (or 40%/60%) or better duty cycle. Consequently, the CLKDV output, divided by 1.5 in high-frequency mode cannot provide a clock input to a second cascaded DCM.

Table 23: CLKDV Duty Cycle with `DLL_FREQUENCY_MODE=HIGH`

CLKDV_DIVIDE Attribute	Duty Cycle	High Time/ Total Cycle
Integer	50.000%	1/2
1.5	33.333%	1/3
2.5	40.000%	2/5
3.5	42.857%	3/7
4.5	44.444%	4/9
5.5	45.454%	5/11
6.5	46.154%	6/13
7.5	46.667%	7/15

CLKDV Jitter Depends on Frequency Mode and Integer or Non-Integer Value

Similarly, integer values for the `CLKDV_DIVIDE` attribute result in half the output jitter and faster DLL locking times.

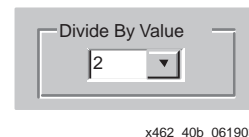
Table 24: CLKDV Output Jitter

CLKDV_DIVIDE	CLKDV Output Period Jitter
Integer Value	CLKOUT_PER_JITT_DV1 (± 150 ps)*
Non-integer Value	CLKOUT_PER_JITT_DV2 (± 300 ps)*

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

DCM Wizard

The Clock Divider controls are in DCM Wizard's [General Setup](#) window. Check the CLKDV output box, shown in Figure 40a. Then, choose the Clock Divider's **Divide by Value** using the drop-down list, shown in Figure 40b.



a. Check the CLKDV output box

b. Select the Divide by Value from the Drop-Down List

Figure 40: Specifying the Clock Divider in DCM Wizard

Frequency Synthesizer (CLKFX, CLKFX180)

The Frequency Synthesizer provides the most flexible means to multiply, divide, or multiply and divide an input frequency. As shown in Table 25, the two Frequency Synthesizer outputs are CLKFX and CLKFX180. The CLKFX180 output has the same frequency as CLKFX but is phase shifted 180°, or half a clock period. Because both Frequency Synthesizer outputs have 50% duty cycles, CLKFX180 appears to be an inverted version of CLKFX.

Two attributes, set at design time, control the synthesized output frequency, as shown in the equation in Table 25. The CLKIN clock input is multiplied the fraction formed by CLKFX_MULTIPLY as the numerator and CLKFX_DIVIDE as the denominator. For example, to create a 155MHz output using a 75MHz CLKIN input, the Frequency Synthesizer multiplies CLKIN by the fraction 31/15. Note that it does not multiply CLKIN by 31 first, then divide by the result by 15. Multiplying CLKIN by 31 would result in a 2.325GHz output frequency—well outside the frequency range of the Spartan-3 DCM.

The multiplier and divider values should be reduced to their simplest form, which results in faster lock times. For example, reduce the fraction 6/8 to 3/4.

Frequency synthesis always requires some form of clock feedback. However, the DFS unit has an internal feedback loop based on CLKIN and does not require a separate loop on CLKFB if used without the DLL unit.

The CLKFX output is phase aligned with the CLKIN input every CLKFX_DIVIDE cycles of CLKIN and every CLKFX_MULTIPLY cycles of CLKFX. For example, if CLKFX_MULTIPLY = 3 and CLKFX_DIVIDE = 5, then the CLKFX output is phase aligned with the CLKIN input every five CLKIN cycles and every three CLKFX cycles. After the DCM asserts its LOCKED output, the DFS unit is resynchronized to the CLKIN input at each concurrence and phase alignment is nearly perfect at these edges.

Table 25: Frequency Synthesizer Summary

DCM Output(s)	CLKFX CLKFX180 (same as CLKFX, phase shifted 180°)
Output Frequency	$F_{CLKIN} \cdot \frac{CLKFX_MULTIPLY}{CLKFX_DIVIDE}$
DCM Functional Unit	Digital Frequency Synthesizer (DFS)
Feedback Required?	No. Uses internal feedback based on CLKIN. Optionally can use CLKFB input if required for Delay-Locked Loop (DLL) functions.
50% Duty Cycle?	Yes, always.
Controlling Attributes	
DFS_FREQUENCY_MODE	Affects frequency limits on CLKIN and the CLKFX, CLKFX180 outputs.
DLL_FREQUENCY_MODE	Only affects the Frequency Synthesizer if the application uses any DLL outputs. Potentially reduces the CLKIN frequency to the more restrictive DLL limits. If only the CLKFX or CLKFX180 outputs are used, then DFS_FREQUENCY_MODE alone defines the frequency limits.
CLKFX_MULTIPLY	Controls the output frequency per the equation above. Legal values include integer values ranging from 2 to 32. Default value is 4.
CLKFX_DIVIDE	Controls the output frequency per the equation above. Legal values include integer values ranging from 1 to 32. Default value is 1.

Table 25: Frequency Synthesizer Summary (Continued)

Frequency Constraints			
CLKIN	The CLKIN frequency limits are determined by the DFS_FREQUENCY_MODE attribute unless the application uses any outputs from the Delay-Locked Loop (DLL) unit. If the DLL unit is used, then the more restrictive DLL clock limits apply.		
	Minimum Frequency		Maximum Frequency
	CLKIN_FREQ_FX_MIN (~1.0 MHz)*		CLKIN_FREQ_FX_MAX (~326 MHz)*
CLKFX CLKFX180	The CLKFX and CLKFX180 output frequency limits are determined by the DFS_FREQUENCY_MODE attribute.		
	DFS_FREQUENCY_MODE	Minimum Frequency	Maximum Frequency
	LOW	CLKIN_FREQ_FX_LF_MIN (~24 MHz)*	CLKIN_FREQ_FX_LF_MAX (~210 MHz)*
	HIGH	CLKIN_FREQ_FX_HF_MIN (~210 MHz)*	CLKIN_FREQ_FX_HF_MAX (~325 MHz)*

* Estimate only. See the [Spartan-3 Data Sheet, Module 3](#) for the correct specified value.

DCM Wizard

To enable the Frequency Synthesizer in DCM Wizard, check the [CLKFX](#), [CLKFX180](#), or both clock outputs in the [General Setup](#) window, as shown in [Figure 41](#).

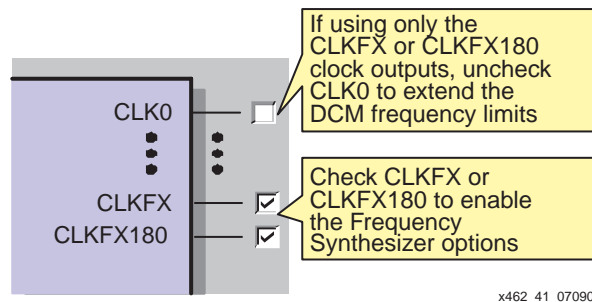


Figure 41: Enabling Frequency Synthesizer in DCM Wizard

If using the CLKFX or CLKFX180 clock outputs stand-alone, then optionally extend the frequency limits by disabling any DLL clock outputs and any feedback.

- By default, the CLK0 output is always checked. If just using CLKFX or CLKFX180, uncheck CLK0.
- Disable DCM feedback by selecting **None**, as shown in [Figure 42](#). Without feedback, the CLKFX and CLKFX180 frequency range is extended to both lower and higher frequencies.

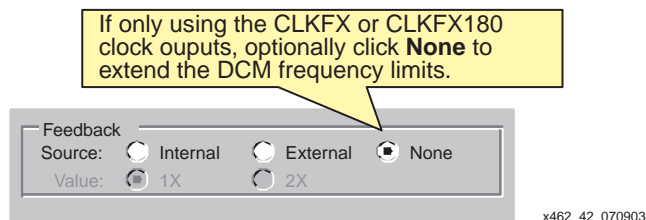


Figure 42: Select No Feedback (None) to Extend Frequency Synthesizer Frequency Limits

Finally, enter the desired output frequency or the Multiply and Divide values, as described in the DCM Wizard [Clock Frequency Synthesizer](#) panel section.

Clock Forwarding, Mirroring, Rebuffering

Because DCMs provide advanced clock control features and Spartan-3 I/O pins support a variety of I/O voltage standards, Spartan-3 FPGAs commonly are used to rebuffer or mirror clock signals, often changing the input clock from one voltage standard to another. Likewise, the DCM conditions an incoming clock signal so that it has a 50% duty cycle.

[Figure 20](#) shows a simple example where a DCM conditions an incoming clock to a 50% duty cycle, and then either forwards the clock at the same frequency using the CLK0 output, or doubles the frequency using the DCM CLK2X output. Similarly, the input and output clocks are phase aligned once the DCM asserts its LOCKED output. The clock feedback path to CLKFB monitors and eliminates the clock distribution delay at the external clock feedback point.

If a 50/50 duty cycle is important on the output clock, make sure that the output I/O standard can switch fast enough to preserve the 50% duty cycle. Verify the duty cycle performance using IBIS simulation on the output signal. Some I/O standards have asymmetric rise and fall times that distort the duty cycle higher frequencies, as can be seen in the IBIS simulation. The DCI versions of HSTL, SSTL, and LVCMOS I/O standards have better symmetry.

To guarantee a 50/50 duty cycle above 100 MHz, the DCM's duty cycle correction capability is mandatory, even if the CLKIN source provides a clean 50% duty cycle. Consequently, the DUTY_CYCLE_CORRECTION attribute must equal TRUE when using the CLK0, CLK90, CLK180, or CLK270 outputs for clock forwarding. The other DCM clock outputs are normally always clock corrected to a 50% duty cycle (see [“Clock Conditioning”](#)).

For best duty-cycle performance—especially at 200 MHz and greater—use a circuit similar to that shown in [Figure 43](#). Use both the CLKx and CLKx180 outputs from the DCM to drive the C0 and C1 inputs, respectively, on a Dual-Data Rate (DDR) output flip-flop. Connect the D0 input of the DDR flip-flop to V_{CC} and the D1 input to GND. Each DCM output drives a separate global buffer, which minimizes duty-cycle distortion. At higher frequencies, it is best not to distribute just one clock and invert one phase locally within the DDR flip-flop, as this adds approximately 150 ps of duty-cycle distortion.

At frequencies of 250 MHz or higher, distribute clocks using a differential signaling standard, such as LVDS. In [Figure 43](#), for example, both the CLKIN clock input and the clock output use LVDS. Additionally, the clock feedback path uses LVDS. For optimal performance, both the clock input and the clock feedback paths require differential global buffer inputs (IBUFGDS), which unfortunately consumes all the global buffer inputs along one edge of the device. However, this solution provides the best-quality clock forwarding solution at high frequencies.

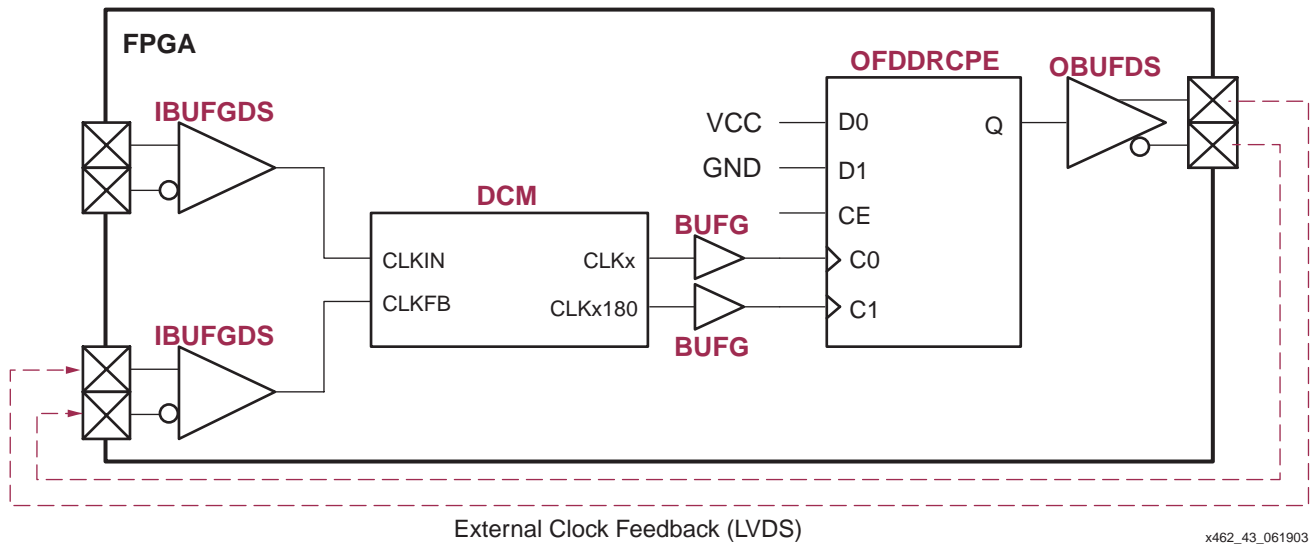


Figure 43: High-Frequency (250+ MHz) LVDS Clock Forwarding Circuit with 50% Duty Cycle

Clock Jitter or Phase Noise

All clocks, including the most expensive, high-precision sources, exhibit some amount of clock jitter or phase noise. The Spartan-3 Digital Clock Managers have their own jitter characteristics, as described in this section. When operating at low frequencies—20 MHz, for example—the effects of jitter usually can be ignored. However, when operating at high frequencies—200 MHz, for example, especially in dual-data rate (DDR) applications—clock jitter becomes a relevant design factor. Clock jitter directly subtracts from the time available to the FPGA application by effectively reducing the available time between active clock edges.

What is Clock Jitter?

Clock jitter is the variation of a clock edge from its ideal position in time, as illustrated in Figure 44. The heavy line shows the ideal position on the clock signal. On each clock edge, there is some amount of variation between the actual clock edge and its ideal location. The difference between the maximum and minimum variations is called peak-to-peak jitter. Jitter is only relevant on the active clock edge. For example, in single-data rate (SDR) applications, data is clocked at each rising clock edge and the specified jitter only subtracts from the total clock period. In dual-data rate (DDR) application, data is clocked at the start of each period and halfway into the period. Therefore, jitter affects each half period.

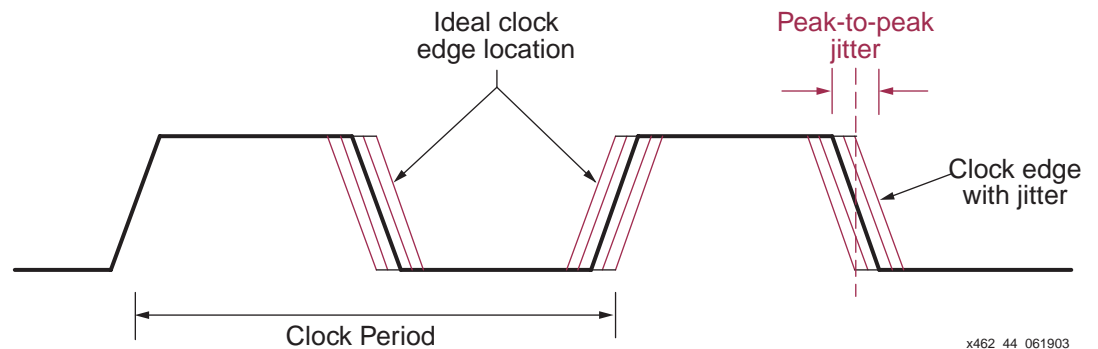


Figure 44: Jitter in Clock Signals

What Causes Clock Jitter?

Clock jitter is unavoidable and exists in all systems. Clock jitter is caused by the various sources of noise or by signal imperfections within the system. In fact, jitter is the manifestation of noise in the time domain. The incoming clock source, for example, has its own jitter characteristics due to random thermal or mechanical vibration noise from the crystal. A large number of simultaneous switching outputs (SSOs) adds substrate noise that slightly changes internal switching thresholds and therefore adds jitter. Similarly, an improperly designed power supply or insufficient decoupling also contributes to jitter. Other sources of clock jitter include cross talk from adjacent signals, poor termination, ground bounce, and electromagnetic interference (EMI).

This application note only discusses the jitter behavior of Spartan-3 Digital Clock Managers (DCMs) and how to improve overall jitter performance within the FPGA.

Understanding Clock Jitter Specifications

Clock jitter is specified in a variety of manners, and the various specifications show different aspects of the same phenomenon.

Cycle-to-Cycle Jitter

Cycle-to-cycle jitter, also called adjacent cycle jitter, indicates the maximum clock period variance from one clock cycle to the next, as shown in [Figure 45](#). In this simple example, the maximum change from one cycle to the next is +100 ps and -100 ps, or put simply, ± 100 ps. Although the clock period may change by larger absolute amounts when measured over millions of clock cycles, the clock period never changes by more than ± 100 ps from one clock cycle to the next.

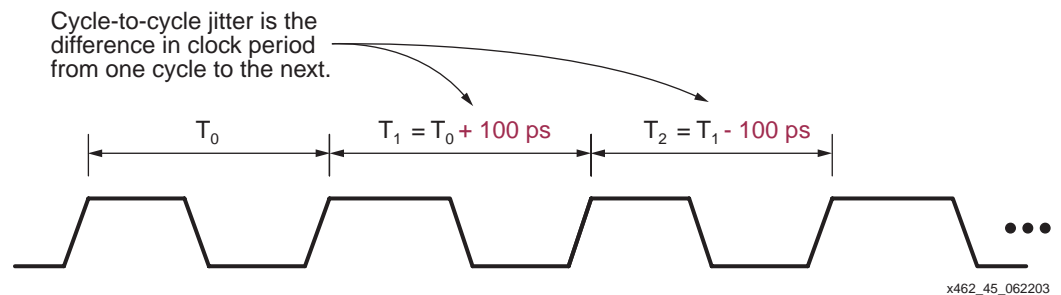


Figure 45: Cycle-to-Cycle Jitter Example

Cycle-to-cycle is an important measure of the quality of a clock output or oscillator but has little use in analyzing the timing of an application.

Period Jitter

Period jitter is the summation of all the cycle-to-cycle jitter values over millions of clock cycles. Peak jitter indicates the earliest and the latest transition times compared to the ideal clock transition time over consecutive clocks.

Period jitter for Digital Clock Managers is random and is expressed as peak-to-peak jitter. Conceptually, the position of the clock transition is a probabilistic distribution or histogram, centered around the ideal, desired clock position, as shown in [Figure 46](#). The actual distribution may not appear purely Gaussian and may be bimodal. Regardless, most actual clock transitions occur near the desired ideal position. However, measured over millions of clock cycles, some clock transitions occur far from the desired position.

The statistical distance from the desired position is measured in standard deviations, also called σ (sigma). Because the DCM is an all-digital design, it is highly stable and Xilinx specifies jitter deviation to $\pm 7\sigma$ or peak-to-peak jitter to 14σ . As a point of reference, $\pm 7\sigma$ guarantees that

99.9999999974% of the jitter values are less than the specified worst-case jitter value. A 14σ peak-to-peak jitter, $\pm 7\sigma$ jitter deviation, equates to a maximum bit error rate (BER) of 1.28×10^{-12} .

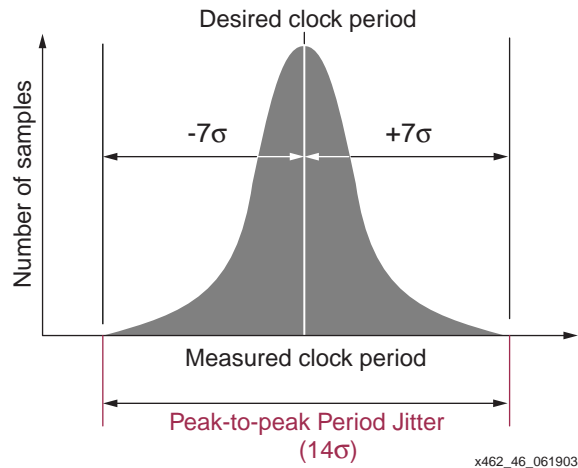


Figure 46: Peak-to-Peak Period Jitter Example

Unit Interval (UI)

Another method to specify jitter is as a fraction of the Unit Interval (UI). One UI represents the time equivalent to one bit time, irrespective of frequency. In single-data rate (SDR) applications where either the rising or the falling clock edge captures data, one UI equals one clock period. In dual-data rate (DDR) applications where data is clocked at twice the clock rate, one UI equals half the clock period.

The peak-to-peak jitter amplitude, quantified in UIs, is the fraction of the peak-to-peak jitter value compared to the total bit period time.

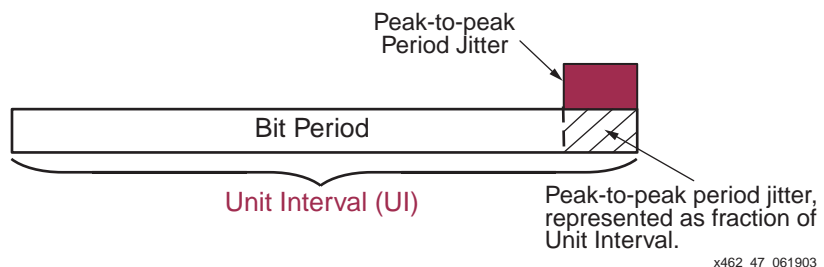


Figure 47: Period Jitter Specified as a Fraction of a Unit Interval

Calculating Total Jitter

The Spartan-3 Data Sheet specifies the output jitter from the DCM clock outputs, except for [CLKFX/CLKFX180](#). The Digital Frequency Synthesizer (DFS) jitter is calculated based on the multiplier and divider settings.

The clock outputs from the DLL unit—i.e., every clock output except CLKFX and CLKFX180—have a worst-case specified jitter listed in the data sheet. This specified value includes the jitter added by the DLL unit. The DLL unit does not remove jitter, so the total jitter on the DLL clock output includes the jitter on the input clock, CLKIN, plus the specified value from the data sheet.

The DFS clock outputs, CLKFX and CLKFX180, remove some amount of incoming clock jitter, so the calculated output jitter is the total jitter.

Adding Input Jitter to DLL Output Jitter

When adding the input jitter and the DLL output jitter, use a root-mean-square (RMS) calculation, similar to noise calculations.

Peak-to-Peak

$$JITTER_{TOTAL} = \sqrt{(JITTER_{INPUT})^2 + (JITTER_{SPEC})^2} \quad \text{Eq. 10}$$

Peak-to-Peak Deviation

$$JITTER_{TOTAL} = \pm \left[\sqrt{\frac{(JITTER_{INPUT})^2 + (JITTER_{SPEC})^2}{2}} \right] \quad \text{Eq. 11}$$

where

- JITTER_{INPUT} = The input period jitter, measured at the clock input pin of the FPGA
- JITTER_{SPEC} = The DLL clock output period jitter, as specified in the Spartan-3 Data Sheet for the appropriate output port
- JITTER_{TOTAL} = The expected total output period jitter

Example

Assume that an input clock has 150 ps peak-to-peak period jitter, optionally expressed as ±75 ps. The incoming clock is duty-cycle corrected, using the same frequency, on the CLK0 DCM output.

In this case, JITTER_{INPUT} = 150 ps. The value for JITTER_{SPEC} is the Spartan-3 Data Sheet specification called CLKOUT_JITT_PER_0, which is estimated here as ±100 ps, or 200 ps peak-to-peak.

$$JITTER_{TOTAL} = \sqrt{(150 \text{ ps})^2 + (200 \text{ ps})^2} = 250 \text{ ps}$$

Consequently, the total jitter on the DCM output is 250 ps peak-to-peak or ±125 ps.

Calculating Jitter for Cascaded DCMs

Figure 48 shows an example application where multiple DCMs are cascaded together to create various output frequencies. The jitter at any point depends on:

- the incoming jitter from the previous sources and
- which DCM output is used.

Each DCM output has slightly different jitter characteristics, as specified in the data sheet. Also, the CLKFX and CLKFX180 outputs from the DFS unit remove some amount of input jitter and DCM Wizard calculates their jitter values (see “Clock Frequency Synthesizer”).

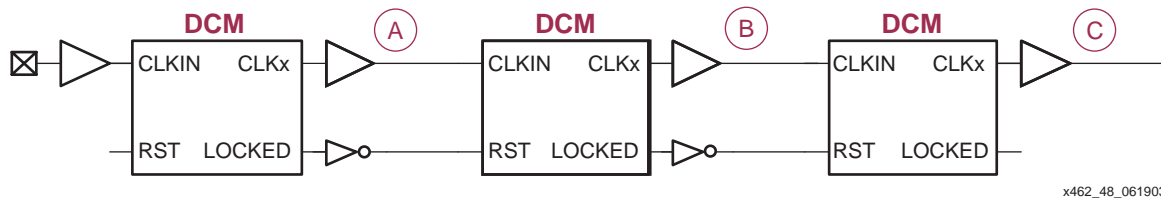


Figure 48: Calculating Jitter for Cascaded DCMs Depends on which DCM Outputs are Used

Consequently, the jitter at any point in the cascaded DCM chain depends on the factors described above. The following examples illustrate how to calculate total jitter at the various points in the circuit.

Example 1: All DCMs Use DLL Outputs

In this example, assume that the input clock has 150 ps (± 75 ps) of period jitter.

Assume that DCM (A) uses the CLK2X output. Use the Spartan-3 Data Sheet specification called CLKOUT_PER_JITT_2X for the DCM output jitter, estimated here as 400 ps (± 200 ps). Calculate the total period jitter on clock (A) using Equation 10.

$$JITTER_{TOTAL(A)} = \sqrt{(150ps)^2 + (400ps)^2} = 427ps = \pm 214ps$$

Assume that DCM (B) uses the CLKDV output with an integer divider value. Use the Spartan-3 Data Sheet specification called CLKOUT_PER_JITT_DV1 for the DCM output jitter, estimated here as 300 ps (± 150 ps). Calculate the total period jitter on clock (B) using Equation 10. Because there are now three elements involved—the input jitter, the jitter from DCM (A), and the jitter from DCM (B)—expand the RMS equation appropriately.

$$JITTER_{TOTAL(B)} = \sqrt{(150ps)^2 + (400ps)^2 + (300ps)^2} = 522ps = \pm 261ps$$

Finally, assume that DCM (C) phase shifts the output from DCM (B) by 90°. Use the Spartan-3 Data Sheet specification called CLKOUT_PER_JITT_90 for the DCM output jitter, estimated here as 300 ps (± 150 ps). Calculate the total period jitter on clock (C) using Equation 10. Because there are now four elements involved—the input jitter, the jitter from DCM (A), the jitter from DCM (B), and the jitter from DCM (C)—expand the RMS equation appropriately.

$$JITTER_{TOTAL(C)} = \sqrt{(150ps)^2 + (400ps)^2 + (300ps)^2 + (300ps)^2} = 602ps = \pm 301ps$$

Example 2: Some DCMs Use the CLKFX or CLKFX180 Outputs

This example is similar to Example 1: All DCMs Use DLL Outputs above except that some DCMs use the CLKFX or CLKFX180 outputs from the DCM's DFS unit.

In this example, assume that the 75MHz input clock has 150 ps (± 75 ps) of period jitter.

As in Example 1, assume again that DCM (A) uses the CLK2X output. The resulting output jitter is the same as that shown in the following equation.

In this example, assume that DCM (B) synthesizes a 90MHz clock using the 150MHz clock generated by DCM (A). Per the DCM Wizard (see "Clock Frequency Synthesizer"), set the attributes CLKFX_MULTIPLY=3 and CLKFX_DIVIDE=5. DCM Wizard also specifies the worst-case output period jitter as 700 ps.

$$JITTER_{TOTAL(B)} = 700ps = \pm 350ps$$

Finally, assume again that DCM (C) phase shifts the output from DCM (B) by 90°. Use the Spartan-3 Data Sheet specification called CLKOUT_PER_JITT_90 for the DCM output jitter,

estimated here as 300 ps (± 150 ps). Calculate the total period jitter on clock (C) using the following equation. Because the preceding DCM used the CLKFX output, the total incoming jitter is set at 700 ps, worst-case. Use the RMS equation to calculate the resulting output jitter as shown below.

$$JITTER_{TOTAL(C)} = \sqrt{(700ps)^2 + (300ps)^2} = 762ps = \pm 381ps$$

Cascaded DCM Design Recommendations

When cascading DCMs, be sure that the LOCKED output of the preceding DCM controls the cascaded DCM's RST input, as shown in [Figure 48](#). The cascaded DCM should not attempt to lock to the input clock until the preceding DCM asserts its LOCKED output, indicating that the clock is stable.

When cascading DCMs, place the most jitter-critical clock output on the first DCM in the cascaded chain.

Jitter Effect on System Performance

Clock jitter, along with other effects, adversely affects system performance by reducing the effective bit period. The bit period available to the FPGA application is the total bit period, T_{BIT} , minus the following effects, as shown in the following equation. In single-data rate (SDR) applications, the clock period and the bit period are equal. However, in dual-data-rate (DDR) applications, the bit period is half the clock period.

$$T_{AVAILABLE} = T_{BIT} - t_{TOTAL_JITTER} - t_{DUTY_CYCLE_DISTORTION}$$

where

T_{BIT} = Bit period time

t_{TOTAL_JITTER} = Total clock jitter. Includes the clock input jitter plus any DCM output jitter or cascaded DCM output jitter.

$t_{DUTY_CYCLE_DISTORTION}$ = Duty cycle distortion specification. Only required for dual-data rate (DDR) applications; otherwise zero. Either data sheet specification CLKOUT_DUTY_CYCLE_DLL or CLKOUT_DUTY_CYCLE_FX depending on which DCM clock output is used.

If the total jitter is specified as a positive value instead of a deviation from the clock period—e.g., 200 ps instead of ± 100 ps—subtract half the positive value—i.e., 100 ps. The bit period is only shortened by the negative deviation. The positive deviate adds to the bit period, adding more timing slack.

Example

Assume that an incoming clock signal enters the FPGA at 75 MHz and that the clock source has ± 100 ps of jitter. The application clocks data on the rising edge of an internally generated 150 MHz, or a total bit period, T_{BIT} , of 6.67 ns. How long is the available bit period, $T_{AVAILABLE}$, after considering the effects of jitter?

The CLK2X output from the Clock Doubler generates a 150MHz clock from the 75MHz clock input. The Clock Doubler output, CLK2X, has ± 200 ps (estimated) of worst-case jitter according to the CLKOUT_PER_JITT_2X specification in the Spartan-3 Data Sheet. Adding

the DCM's ± 200 ps of jitter to the clock source's ± 100 ps of jitter using root-mean square (RMS), the total jitter, t_{TOTAL_JITTER} , is ± 0.223 ns.

$$t_{TOTAL_JITTER} = \sqrt{(\pm 100\text{ps})^2 + (\pm 200\text{ps})^2} = \pm 223.60\text{ps} = \pm 0.223\text{ns}$$

Because data is only clocked on the rising clock edge, there are no duty-cycle distortion effects and $t_{DUTY_CYCLE_DISTORTION} = 0$.

Therefore, the total available clock period, $T_{AVAILABLE}$ is reduced down to 6.444 ns from a total bit period of 6.667 ns. Effectively, this forces the logic to operate at 155.1831 MHz instead of 150 MHz.

$$T_{AVAILABLE} = 6.667\text{ns} - 0.223\text{ns} = 6.444\text{ns}$$

Recommended Design Practices to Minimize Clock Jitter

In higher-performance applications, clock jitter steals valuable bit period time. Adhere to the following recommendations to minimize the amount of system-wide clock jitter.

Properly Design the Power Distribution System

A properly designed power distribution system (PDS), including proper power-plane decoupling, reduces system jitter by creating a stable power environment. Application note XAPP623 discusses recommended design practices for PDS design.

- XAPP623: *Power Distribution System (PDS) Design: Using Bypass/Decoupling Capacitors*
<http://www.xilinx.com/xapp/xapp623.pdf>

Properly Design the Printed Circuit Board

Design the printed circuit board for expected operating frequency range and application environment.

- WP174: *Methodologies for Efficient FPGA Integration into PCBs*
http://www.xilinx.com/publications/whitepapers/wp_pdf/wp174.pdf
- PCB Checklist
http://support.xilinx.com/xlnx/xil_prodcat_product.jsp?title=si_pcbcheck

Obey Simultaneous Switching Output (SSO) Recommendations

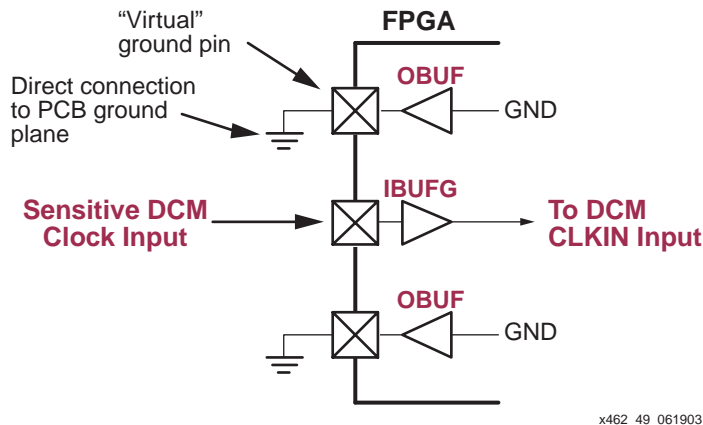
To avoid signal-related corruption of clock inputs to or clock outputs from a DCM, be sure to follow the Simultaneous Switching Output (SSO) recommendations outlined in the Spartan-3 Data Sheet.

Whenever possible, avoid placing DCM inputs or output near heavily switching I/Os, especially those with large output voltage swings or with high current drive.

Place Virtual Ground Pins Around DCM Input and Output Connections

On sensitive, high frequency DCM inputs or outputs, use additional user-I/O pin to create extra connections to the PCB ground—i.e., create virtual ground pins. Place these virtual ground pins on the I/O pads adjacent to the sensitive DCM signal. Make sure that the I/O pads are on adjacent pads on the FPGA die level, not just on adjacent pins or balls on the package. Adjacent balls on BGA packages do not necessarily connect to adjacent pads on the FPGA. These techniques reduce the internal voltage drop and improve the jitter.

To create a “virtual ground”, configure an IOB as an output driving GND (Low logic level) and connect the IOB externally directly to the ground plane, as shown in [Figure 49](#).



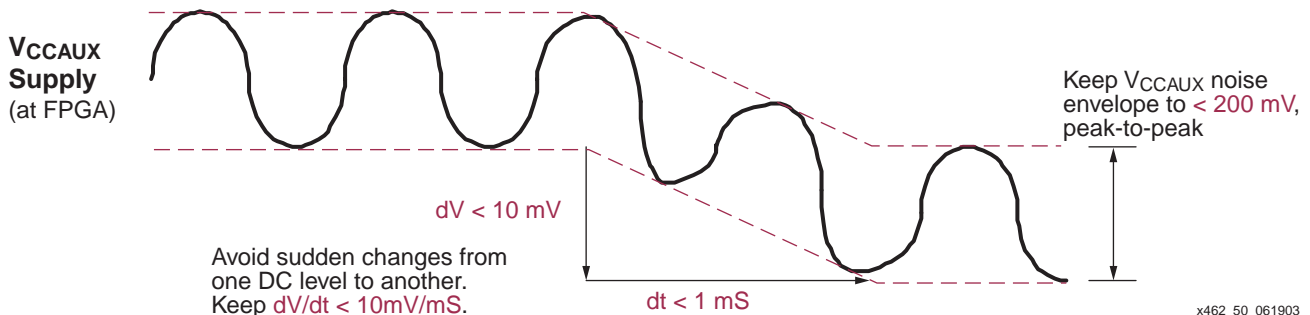
x462_49_061903

Figure 49: Place Virtual Ground Pins Adjacent to Sensitive DCM Input or Output Clock Signals

V_{CCAUX} Considerations for Improving Jitter Performance

The Digital Clock Managers are powered by the V_{CCAUX} supply input. Any excessive noise on the V_{CCAUX} supply input to the FPGA adversely affects the DCM's characteristics, especially its jitter performance. For best DCM performance, please follow these recommendations.

1. Limit changes in on the V_{CCAUX} power supply or ground potentials to less than 10 mV in any 1 ms interval, as shown in Figure 50. This recommendation allows the DCM to properly track out the change.
2. Limit the noise at the power supply to be within 200 mV peak-to-peak, as shown in Figure 50.



x462_50_061903

Figure 50: Recommended V_{CCAUX} Supply Considerations Avoid Voltage Droop

3. If V_{CCAUX} and V_{CCO} are of the same power plane, every V_{CCAUX}/V_{CCO} pin must be properly decoupled or bypassed (see "Properly Design the Power Distribution System"). Separate the V_{CCAUX} supply from any V_{CCO} supplies if Guidelines 1 and 2 above cannot be maintained.
4. The CLK2X output is especially affected by the power or ground shift. Consequently, the CLKFX output, using CLKFX_MULTIPLY=2 and CLKFX_DIVIDE=1, may provide a better quality output when all IOBs and CLBs are switching. The CLKFX circuitry updates the tap every three input clocks in the DFS mode, as opposed to the slower update rate for the CLK2X output.

Adjusting FACTORY_JF Setting

A well-designed, stable, properly decoupled power supply is the best overall solution to reducing clock skew and jitter within the FPGA. However, increasing the FACTORY_JF attribute setting to 0xFFFF may improve jitter performance on a problem board. When

FACTORY_JF=FFFF, the DCM updates its tap settings approximately every twenty input clocks. The frequency-based default settings update the tap settings much more slowly.

Increasing the FACTORY_JF setting may introduce a small amount of jitter (~30 ps) because the DCM frequently updates its delay line, which is why FACTORY_JF is not set to the maximum value by default. If the power supply is unstable, the phase error introduced may be much bigger than the extra jitter introduced; therefore, increasing the FACTORY_JF setting may improve the design.

Miscellaneous Topics

Bitstream Generation Settings

There are two bitstream generation (BitGen) options related to the DCM:

- **-g lck_cycle:** This option causes the FPGA configuration startup sequence to wait until all instantiated DCMs assert their LOCKED outputs.
- **-g DCMSshutdown:** This option resets the DCM logic if the "SHUTDOWN" configuration command is loaded into the configuration logic, as during either partial reconfiguration or during full reconfiguration via the JTAG port.

Setting Bitstream Generation Options in Project Navigator

If using the ISE 5.2i Project Navigator graphical interface, set the bitstream generation options by right-mouse clicking **Generate Programming File** in the Processes for Current Source panel, as shown in [Figure 51](#). Select **Properties** from the resulting menu.

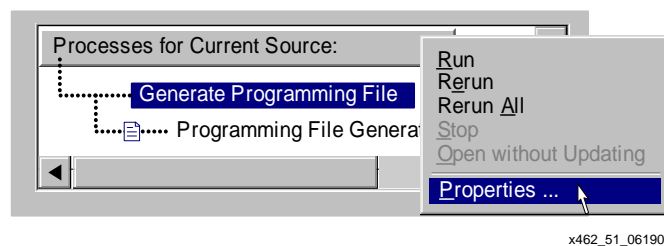


Figure 51: Setting Bitstream Generator (BitGen) Options within Project Navigator

See "BitGen Switches and Options" for more information.

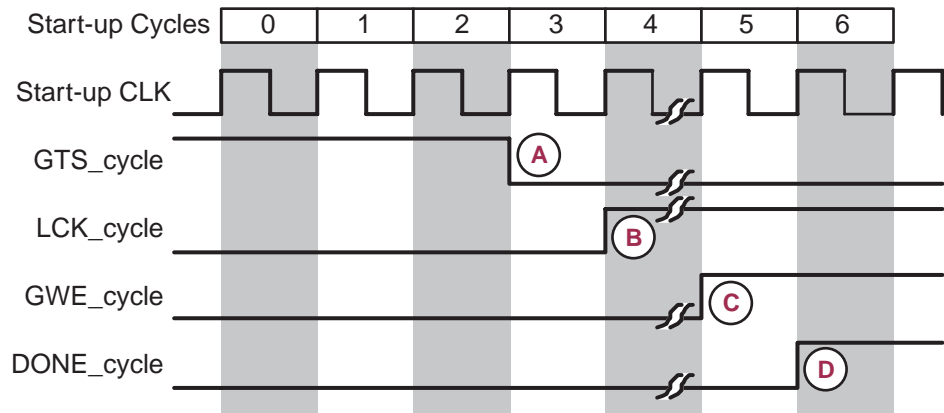
Setting Bitstream Generation Options via Command Line or Script

To see the available options, type the following in a command window:

```
bitgen -help spartan3
```

Setting Configuration Logic to Wait for DCM LOCKED Output

The DCM's STARTUP_WAIT attribute signals the FPGA's configuration start-up logic to wait for the DCM to assert its LOCKED output before the FPGA asserts its DONE output. Two actions are required at design time, however. First, set the STARTUP_WAIT attribute to TRUE on each of the DCMs that must be locked before configuration completes. Then, modify the bitstream generation options so that the events shown in [Figure 52](#) happen within the six-clock start-up cycle. Sufficient configuration clock cycles must be provided after the DCM locks to allow the device to complete the configuration start-up sequence.



x462_52_062403

Figure 52: Start-up Logic Interaction with DCM LOCKED Output

- A. Release the FPGA's internal Global Three-State (GTS_cycle) signal, enabling all I/O signals.
- B. Set the cycle where the start-up logic waits for the DCM(s) to assert LOCKED after the GTS_cycle. The DCMs require some form of external input—a clock and possibly a feedback signal—before the DCM can lock on the clock signal.
- C. After achieving valid DCM lock, assert the FPGA's internal Global Write Enable (GWE_cycle) signal.
- D. Finally, assert the internal DONE signal.

Figure 53 shows these same option settings from within Project Navigator.

Property Name	Value
FPGA Startup Clock	CLK
Enable Internal Done Pipe	<input type="checkbox"/>
Done (Output Events)	6
Enable Outputs (Output Events)	3
Release Write Enable (Output Events)	5
Release DLL (Output Events)	4
Match Cycle (Output Events)	Default (No Wait)
Drive Done Pin High	<input type="checkbox"/>

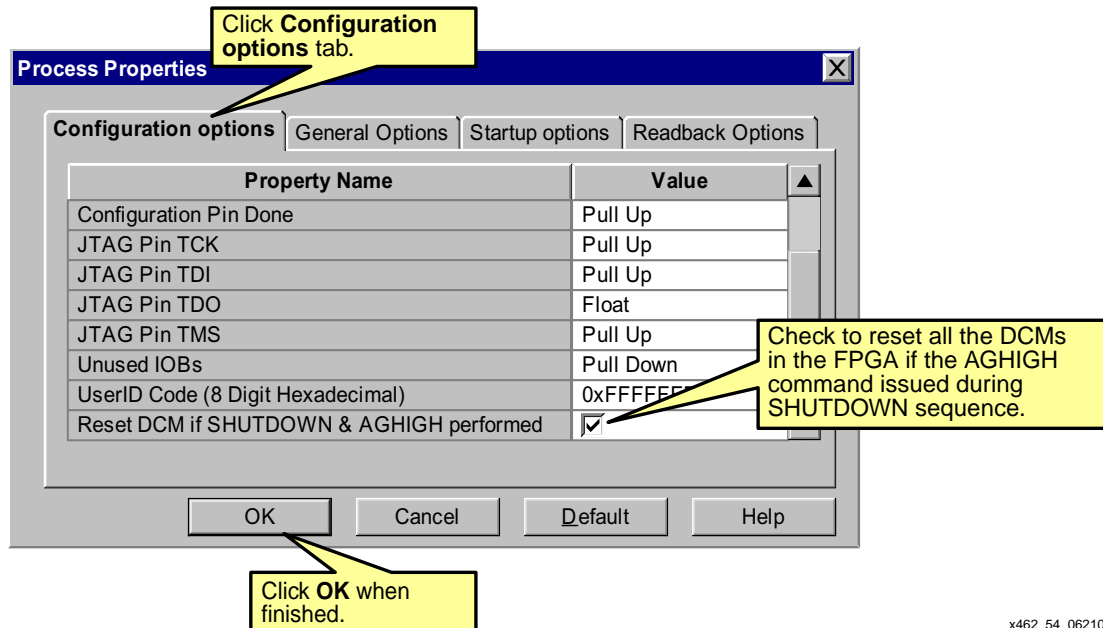
x462_53_061903

Figure 53: BitGen Options

The specific start-up phase timing and the timing of both the GWE_cycle and DONE_cycle are flexible. However, if using the STARTUP_WAIT attribute on a DCM, the GTS_cycle must always happen before the LCK_cycle. Otherwise, the DCM never locks and configuration never completes!

Reset DCM During Partial Reconfiguration or During Full Reconfiguration via JTAG

Another bitstream option resets all the DCMs in the FPGA application during reconfiguration via the SelectMAP interface or during full or partial reconfiguration via the JTAG port. If the option is enabled, the DCMs are reset when the AGHIGH configuration command is issued during the SHUTDOWN command sequence. It is imperative to reset the DCMs when reconfiguring through JTAG. Change the bitstream generator options in Project Navigator (see “[Setting Bitstream Generation Options in Project Navigator](#)”). Click **Configuration options**, then check the **Reset DCM if SHUTDOWN & AGHIGH performed** option as shown in [Figure 54](#).



x462_54_062103

Figure 54: Configuration Option Allows DCM Reset During Reconfiguration Process

Momentarily Stopping CLKIN

To reduce overall system noise while taking precision analog measurements, it is possible to momentarily stop the clock inputs to the DCM without adversely affecting the remainder of the FPGA application. This is possible, in part, because the DCM is an all-digital, stable system. The DCM must first lock to the input clock and assert the LOCKED output. If the DCM is not reset, it is possible to momentarily stop the CLKIN input clock with little impact to the deskew circuit, provided that these guidelines are followed:

- The clock must not be stopped for more than 100 ms to minimize the effect of device cooling, which would change the tap delays.
- The clock should be stopped during a Low phase, and when restored, must generate a full High half-period.

Although the above conditions do technically violate the clock input jitter specifications, the DCM LOCKED output stays High and remains High when the clock is restored. Consequently, the High on LOCKED does not necessarily mean that a valid clock is available. The above conditions technically do violate the clock input jitter specifications but work within the limits described above.

When CLKIN is stopped, an additional one to eight output clock cycles are still generated as the DCM's digital delay line is flushed. Similarly, once CLKIN is restarted, output clocks are not generated for one to four clocks cycles as the delay line is filled. The delay line usually fills within two or three clocks.

Likewise, it is also possible to phase shift the input clock. This phase shift propagates to the output one to four clocks after the original shift with no disruption to the DCM control.

Figure 55 shows an example where the CLKIN input clock is momentarily stopped. The figure also illustrates the corresponding effect on the CLK2X clock output.

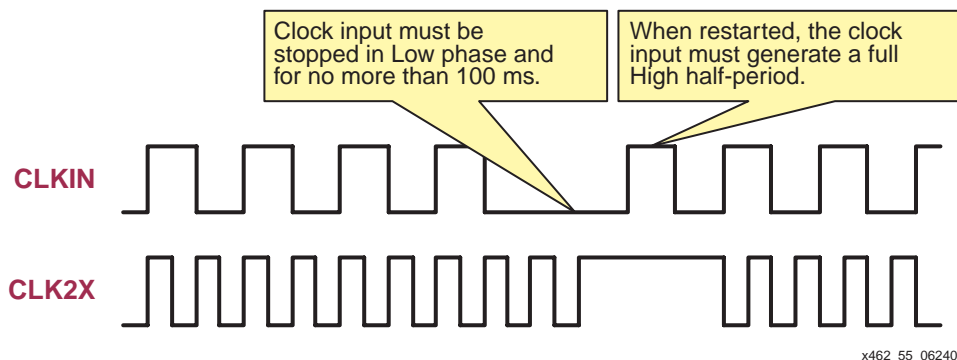


Figure 55: Momentarily Stopping CLKIN Clock Input

Related Materials and References

- Spartan-3 Data Sheet (Module 2). "Digital Clock Manager". Description of features and capabilities. <http://www.xilinx.com/bvdocs/publications/ds099-2.pdf>
- Spartan-3 Data Sheet (Module 3). Timing and Jitter Specifications. <http://www.xilinx.com/bvdocs/publications/ds099-3.pdf>
- Libraries Guide, for ISE 5.2i by Xilinx, Inc. DCM Primitive description. <http://toolbox.xilinx.com/docsan/xilinx5/pdf/docs/lib/lib.pdf>
- Architecture Wizard. Free recorded E-learning session. <http://www.xilinx.com/support/training/ise5-wizard.htm>
- XAPP259: System Interface Timing Parameters. <http://www.xilinx.com/xapp/xapp259.pdf>
- XAPP268: Dynamic Phase Alignment. <http://www.xilinx.com/xapp/xapp268.pdf>
- XAPP622: SDR LVDS Transmitter/Receiver. <http://www.xilinx.com/xapp/xapp622.pdf>
- Development System Reference Guide. Chapter 15, "BitGen". Description of bitstream generation program and options. Pages 335-367. <http://toolbox.xilinx.com/docsan/xilinx5/pdf/docs/dev/dev.pdf>

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/09/03	1.0	Initial Xilinx release.