



WP257 (v1.0) March 21, 2007

What Are PERIOD Constraints?

By: Chris Zeh

A fundamental timing constraint is the PERIOD constraint. This paper discusses the overall purpose of PERIOD constraints and the specific paths that are covered by PERIOD constraints. Additionally, examples of timing reports are included with the common application of the PERIOD constraints.

© 2006 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.

Why Use a PERIOD Constraint?

The PERIOD constraint or "Clock Period Specification" is a fundamental timing and synthesis constraint. They define each clock within the design, cover all synchronous paths within each clock domain and cross-clock domain paths between related clock domains. The PERIOD constraint defines the duration of the clock and can be configured to have different duty cycles. The PERIOD constraint is preferred over FROM:TO constraints, because the PERIOD constraint covers a majority of the paths in the design and decreases the runtime of the implementation tools. The Clock Period Specification:

- Defines the timing between synchronous elements (FFS, RAMS, LATCHES, HSIOs, CPUs, DSPS, and PADS) clocked by a specific clock net that is terminated at a registered clock pin, as shown in Figure 1
- Defines the timing between related clock domains based upon the destination clock domain

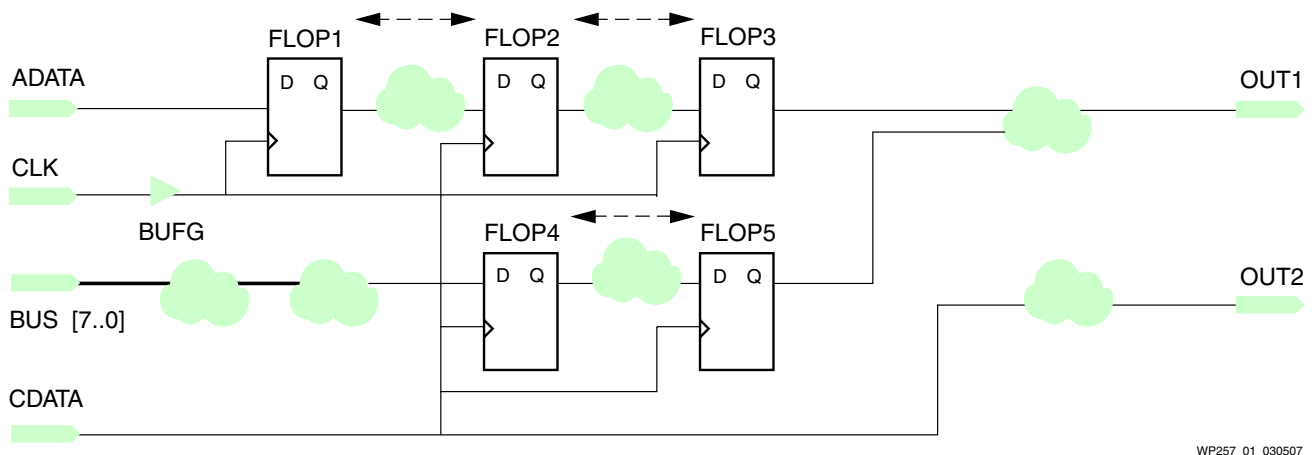


Figure 1: PERIOD Constraints Cover Register to Register Paths

The PERIOD constraint on a clock net analyzes all delays on all paths that terminate at a pin with a setup/hold analysis relative to the clock net. A typical analysis includes the data paths of:

- Clock-to-out delay
- Routing & Logic delay
- Setup/hold delay
- Clock Skew between the source and destination synchronous elements
- Clock Phase – DCM Phase and Negative Edge Clocking
- Clock Duty Cycles

Using the PERIOD constraint has the following advantages:

- Includes clock path delay in the clock skew analysis for global and local clocks
- Includes local clock inversion
- Includes setup and hold time analysis
- Includes the phase relationship between related clocks
 - ◆ Related/Derived clocks can be a function of another clock (* and /)
- Includes DCM Jitter, Duty-Cycle Distortion, and DCM Phase Error for Virtex™-4, DCM Jitter, PLL Jitter, Duty-Cycle Distortion, and DCM Phase Error for Virtex™-5, and new families as Clock Uncertainty

- Includes User Defined System and Clock Input Jitter as Clock Uncertainty
- Includes unequal clock duty cycles (non 50%)
- Includes clock phase, including DCM phase and negative edge clocking

Related TIMESPEC PERIOD Constraints

Xilinx recommends that a PERIOD constraint be associated with every clock in the design. The preferred method for defining Period constraints is with the TIMESPEC Period Constraint because it allows the user to define derived clock relationships with other TIMESPEC Period constraints.

An example of this complex derivative relationship is done automatically through the DLL/DCM/PLL/BUFR/PMCD component outputs. The derived relationship is defined with one TIMESPEC PERIOD in terms of another TIMESPEC PERIOD. When a data path goes from one clock domain to another clock domain, and the PERIOD constraints are related, the timing tools will do a cross-clock domain analysis. This is very common with the outputs from the DCM/DLL/PLL components.

During cross-clock domain analysis of related PERIOD constraints, the PERIOD constraint on the destination element will cover the data path.

In Figure 2, TS_PERIOD#1 is related to TS_PERIOD#2, and the data path, in red, in the figure is analyzed by TS_PERIOD#2.

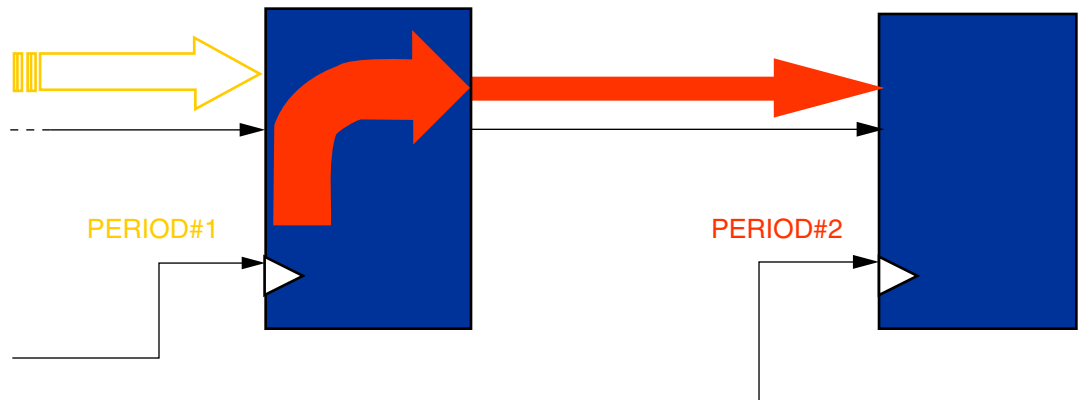


Figure 2: Related PERIOD Constraints

When PERIOD constraints are related to each other, the tools have the ability to determine the inter-clock domain path requirements as shown in Figure 2. An example of the constraint syntax is show below. The TS_Period_2 constraint value is a multiple of the TS_Period_1 TIMESPEC.

- TIMESPEC TS_Period_1 = PERIOD “clk1_in_grp” 20 ns HIGH 50%;
- TIMESPEC TS_Period_2 = PERIOD “clk2_in_grp” TS_Period_1 * 2;

If the two PERIOD constraints are not related in this method, the cross-clock domain data paths will not be covered or analyzed by any PERIOD constraint.

In Figure 3, CLKA and CLKB are not related or asynchronous to each other, so the data paths between register four and register five is not analyzed by either PERIOD constraint.

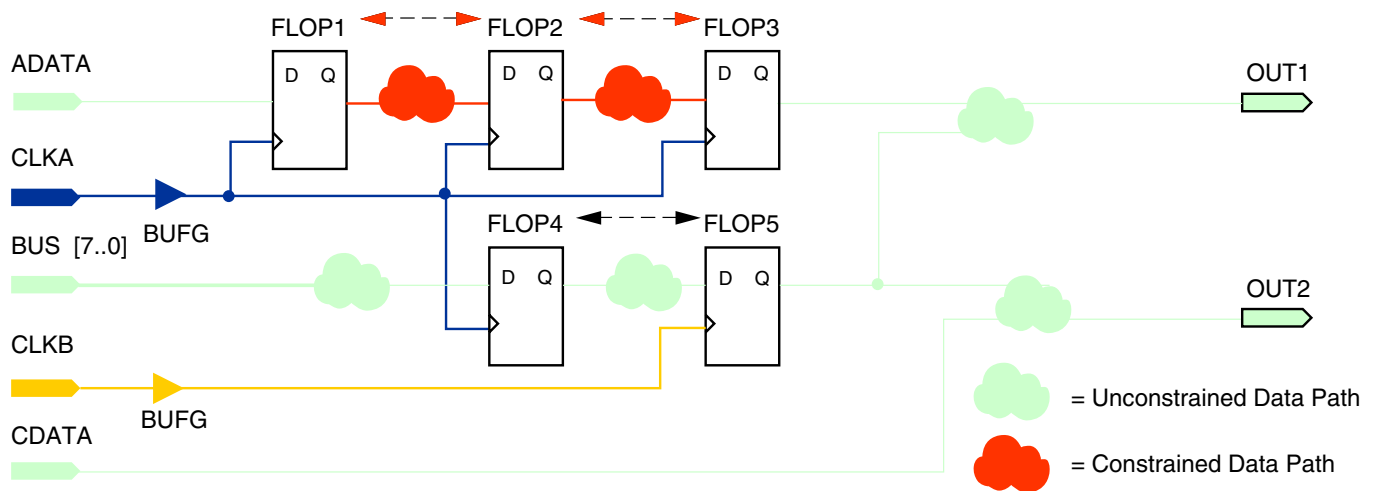


Figure 3: Unrelated Clock Domain

DLL/DCM/PLL/BUFR/PMCD Manipulation

When TIMESPEC PERIOD specification on the input pad clock net is traced or translated through the DCM/DLL/PLL/BUFR/PMCD component, the derived or output clocks are constrained with new PERIOD constraints.

When a NET PERIOD constraint is applied to the input pad clock net, this constraint will not get translated through the DCM/PLL/DLL/BUFR/PMCD component. This can result in zero items or paths analyzed for these constraints.

Additionally, the NET PERIOD is analyzed only during MAP –timing, PAR and Timing Analysis. When MAP –timing and PAR call the timing tools, the timing tools do the DLL/DCM/PLL/BUFR/PMCD manipulation for placement and routing, but not for the timing analysis reports.

When a TIMESPEC PERIOD constraint is traced into an input pin on a DLL/DCM/PLL/BUFR/PMCD component, NGDBuild or the translate process will transform the original TIMESPEC PERIOD constraint into new TIMESPEC PERIOD constraints based upon the derived output clocks. A message included in the NGDBuild report (design.bld) will indicate this transformation. MAP, PAR (Place & Route), and Timing Analyzer will use the new derived clock TIMESPEC PERIOD constraints which are located in the Physical Constraints File (PCF) file. The original TIMESPEC PERIOD is unchanged during this transformation and used as a reference for the new TIMESPEC PERIOD constraints.

Note: Constraints Editor will see only the original PERIOD constraint and not the newly transformed PERIOD constraints.

During transformation, each clock output pin of the DLL/DCM/PLL/BUFR/PMCD component will get a new TIMESPEC PERIOD constraints and corresponding destination element timing group. The new TIMESPEC PERIOD constraints are based upon the manipulation of the DLL/DCM/PLL/BUFR/PMCD component. The transformation takes into account the phase

relationship factor of the outputs and also performs the appropriate multiplication or division of the PERIOD requirement value.

The transformation occurs when the TIMESPEC PERIOD constraint is traced into the CLKIN pin of the DLL/DCM/PLL/BUFR/PMCD component and the following conditions are met:

- The group associated with the PERIOD constraint is used in exactly ONE PERIOD constraint
- The group associated with the PERIOD constraint is NOT used in any other timing constraints, including FROM:TO (Multicycle) or OFFSET constraints
- The group associated with the PERIOD constraint is NOT referenced or related to any other user group definition

If the above conditions are met, the "TIMESPEC "TS_clk20" = PERIOD "clk20_grp" 20 ns HIGH 50 %;" constraint is translated into the following constraints based upon the clock structure in [Figure 4](#):

- CLK0: TS_clk20_0=PERIOD clk20_0 TS_clk20*1.000000 HIGH 50.000000%
- CLK90: TS_clk20_90=PERIOD clk20_90 TS_clk20*1.000000 PHASE + 5.000000 nS HIGH 50.000000%

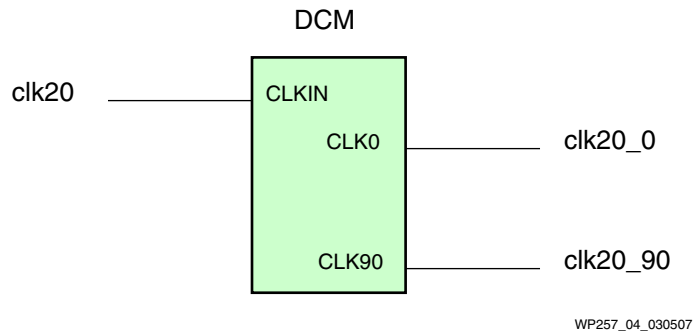


Figure 4: New PERIOD Constraints on DCM Outputs

The following message will also appear in the NGDDBuild (design.bld) report:

```
INFO:XdmHelpers:851 - TNM " clk20_grp ", used in period
specification "TS_clk20", was traced into DCM instance "my_dcm".
The following new TNM groups and period specifications were
generated at the DCM output(s):
clk0: TS_clk20_0=PERIOD clk20_0 TS_clk20*1.000000 HIGH 50.000000%
clk90: TS_clk20_90=PERIOD clk20_90 TS_clk20*1.000000 PHASE +
5.000000 nS HIGH 50.000000%
```

If the CLKIN_DIVIDE_BY_2 attribute is set to TRUE for the DCM in [Figure 4](#), the translated PERIOD constraints will be adjusted accordingly. The following constraints are the result of this attribute:

- CLK0: TS_clk20_0=PERIOD clk20_0 TS_clk20*2.000000 HIGH 50.000000%
- CLK90: TS_clk20_90=PERIOD clk20_90 TS_clk20*2.000000 PHASE + 5.000000 nS HIGH 50.000000%

If the above conditions are not met, from previous page, the PERIOD constraint will not be placed on the output or derived clocks of the DLL/DCM/PLL/BUFR/PMCD component and an error or warning message will be reported in the NGDDBuild report.

Example Error Message:

```
"ERROR:NgdHelpers:702 - The TNM "PAD_CLK" drives the CLKIN pin of
CLKDLL "$I1". This TNM cannot be traced through the CLKDLL because
it is not used in exactly one PERIOD specification. This TNM is used
in the following user groups and/or specifications:
TS_PAD_CLK=PERIOD PAD_CLK 20000.000000 pS HIGH 50.000000%
TS_01=FROM PAD_CLK TO PADS 20000.000000 pS"
```

The original TIMESPEC PERIOD constraint will be reported in the timing report and will have “0 items analyzed”.

The newly created TIMESPEC PERIOD constraints will contain all of the paths associated with the DCM/DLL/PLL/BUFR/PMCD component. If the PERIOD constraint is not translated and then traces only to the DLL/DCM/PLL/BUFR/PMCD component, the timing report will report “0 items analyzed” and no other PERIOD constraints will be reported. If the PERIOD constraint traces to other synchronous elements, then the analysis will include only those synchronous elements.

PHASE Keyword

The PHASE keyword is used in the relationship between related clocks. The timing analysis tools use this relationship for the OFFSET constraints and cross-clock domain paths analysis. The PHASE keyword can be entered in the UCF/NCF or through the translation of the DCM/DLL/PLL components during NGDBuild.

Note: If the phase-shifted value of DCM/PLL/DLL component is changed in FPGA Editor, the change is not reflected in the PCF file.

The timing analysis tools use the PHASE keyword value in the PCF to emulate the DLL/DCM/PLL phase shift value. In order to see the change that was made in FPGA Editor, the PCF needs to also be modified manually with the corresponding change.

DLL/DCM Manipulation With PHASE

Table 1 displays the new DCM/DLL/PLL component output clock net derived TIMESPEC PERIOD constraints, based upon the original PERIOD (TS_CLKIN) constraints, and TS_CLKIN is expressed as a time value. If TS_CLKIN is expressed as a frequency value, the multiply and divide operations are reversed. If the DCM attributes FIXED_PHASE_SHIFT or VARIABLE_PHASE_SHIFT are used, the amount of the phase-shifted value will be included in the PHASE keyword value. The phase shifting amount on the DCM is not reflected in [Table 1](#).

Table 1: Transformation of PERIOD Constraint through DCM

Output Pin	PERIOD Value	PHASE Shift value
CLK0	TS_CLKIN * 1	None
CLK90	TS_CLKIN * 1	PHASE + (clk0_period * ¼)
CLK180	TS_CLKIN * 1	PHASE + (clk0_period * ½)
CLK270	TS_CLKIN * 1	PHASE + (clk0_period * ¾)
CLK2x	TS_CLKIN / 2	None
CLK2x180	TS_CLKIN / 2	PHASE + (clk2x_period * ½)

Table 1: Transformation of PERIOD Constraint through DCM

Output Pin	PERIOD Value	PHASE Shift value
CLKDV	$TS_CLKIN * clkdv_divide$ (clkdv_divide = value of CLKDV_DIVIDE property (default = 2.0))	None
CLKFX	$TS_CLKIN / clkfx_factor$ (clkfx_factor = value of CLKFX_MULTIPLY property (default = 4.0) divided by value of CLKFX_DIVIDE property (default = 1.0))	None
CLKFX180	$TS_CLKIN / clkfx_factor$ (clkfx_factor = value of CLKFX_MULTIPLY property (default = 4.0) divided by value of CLKFX_DIVIDE property (default = 1.0))	PHASE + (clkfx_period * ½)

Gated Clocks

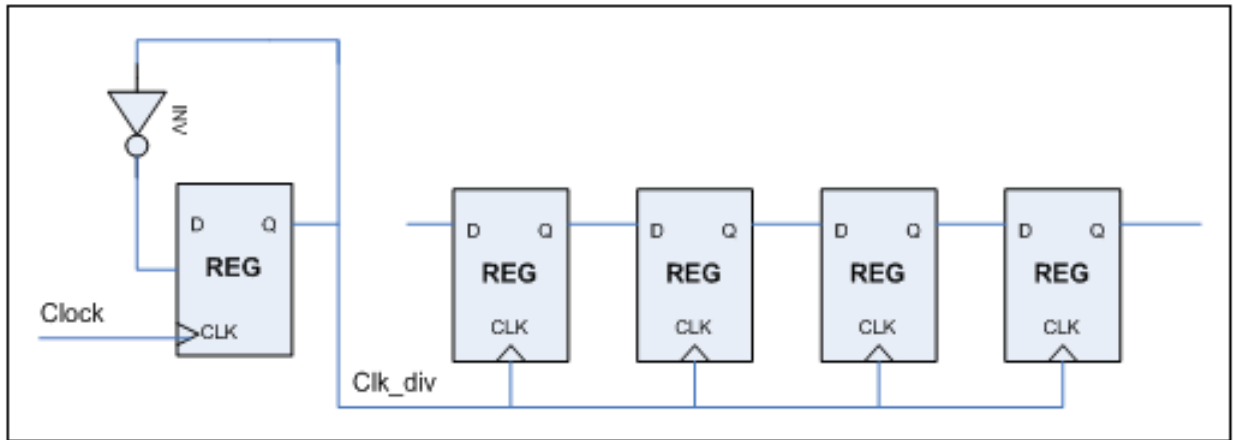
The PERIOD constraint does not analyze gated or internally derived clocks correctly. If the clock is gated or goes through a LUT (Look-Up-Table), the timing analysis will trace back through each input of the LUT to the source (synchronous elements or pads) of the signals and report the corresponding clock skew.

The end result of a clock derived from a LUT is that the clock skew is very large, depending on the levels of logic or number of LUTs.

If the clock has been divided by using internal logic and not by a DCM, the PERIOD constraint on the clock pin of the “Divide down Flip-Flop” will not trace through this flip-flop to the Clk_div signal, as shown in [Figure 5](#).

The timing analysis will not include the downstream synchronous elements, which are driven by the new clock signal.

The new clock, derived from the “Divide down Flip-Flop” will be on local routing, unless a global buffer is used. If a PERIOD constraint is placed on the output of the "Divide down Flip-Flop", which is "clk_div" signal in [Figure 5](#), and is related back to the original PERIOD constraint, the timing analysis will include the downstream synchronous elements. To ensure the relationship and the cross-clock domain analysis is correct, the difference between the divided clock and the original clock needs to be included in the PERIOD constraint with the PHASE keyword. The clock skew can be large, depending on the relationship between the two clocks.



WP257_05_030507

Figure 5: Gated Clock with Divide Down Flip-Flop

What Paths Are Covered By PERIOD Constraints?

The PERIOD constraint only covers paths between synchronous elements. Synchronous elements include the following elements:

- Flip Flops
- Latches
- Distributed RAM
- Block RAM
- Distributed ROM
- ISERDES
- OSERDES
- PPC405
- MULT18X18
- DSP48
- MGTs (GT, GT10, GT11, GTP)
- SRL16
- EMAC
- FIFO (16, 18, & 36)
- PCIE
- TEMAC

Pads are not included in this analysis and NGDBuild will issue a warning if you have pad elements in the PERIOD time group. Analysis between unrelated or asynchronous clock domains is also not included.

Priorities

During the analysis of the design, the timing analysis tools determine which paths of the design are analyzed or covered by which constraint.

This determination is based upon the constraint prioritization or which constraint appears later in the PCF file.

If the design has two PERIOD constraints that cover the same paths, the later PERIOD constraint in the PCF file will cover or analyze these paths. The previous PERIOD constraints will have “0 items analyzed” in the timing report. In order to force the timing analysis tools to use the previous PERIOD constraints, instead of the later one, use the PRIORITY keyword on the PERIOD constraints. In addition to the PRIORITY keyword, the use of a multi-cycle or FROM:TO constraints can be used to cover these paths.

The priority between constraint types is as follows, from the highest, to the lowest:

- Timing Ignore (TIG) constraint
- Multi-cycle or FROM:TO constraint
- OFFSET constraints
- PERIOD constraints

In order to prioritize within a constraint type or to avoid a conflict between two timing constraints that cover the same path, you must use the PRIORITY keyword and a value. The value for the PRIORITY can range from -255 to +255 and the lower the value, the higher the priority. The value does not affect which paths will be placed and routed first. It only affects which constraint will cover and analyze the path with two timing constraints of equal priority.

Use the following syntax to define the priority of a timing constraint:

```
TS_Normal_timing_syntax PRIORITY integer;
```

This can be applied only to TIMESPEC constraints with TS identifiers (e.g., TS03). This situation can occur when two clock signals from the DCM drive the same BUFGMUX, as shown in [Figure 6](#).

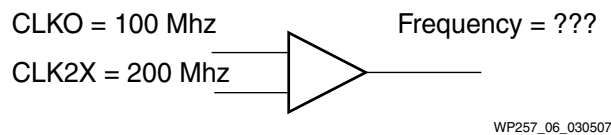


Figure 6: PRIORITY with a BUFGMUX Component

An example of a PERIOD constraint with the PRIORITY keyword:

- TIMESPEC “TS_Clk0” = PERIOD “clk0_grp” 10 ns HIGH 50% PRIORITY 2;
- TIMESPEC “TS_Clk2X” = PERIOD “clk2x_grp” TS_Clk0 / 2 PRIORITY 1;

Setup/Hold Analysis

The PERIOD constraint analysis includes the setup and hold analysis on synchronous elements. The setup analysis ensures that the data changes at the destination synchronous element prior to the clock arrival.

Additionally, the data must become valid at its input pins at least a setup time before the arrival of the active clock edge at its pin.

The equation for the setup analysis is the data path delay plus the synchronous element setup time minus the clock path skew.

$$\text{Setup Time} = \text{Data Path Delay} + \text{Synchronous Element Setup Time} - \text{Clock Path Skew}$$

The clock uncertainty increases the clock path delay, so the needed setup margin is larger. This is shown in Figure 7.

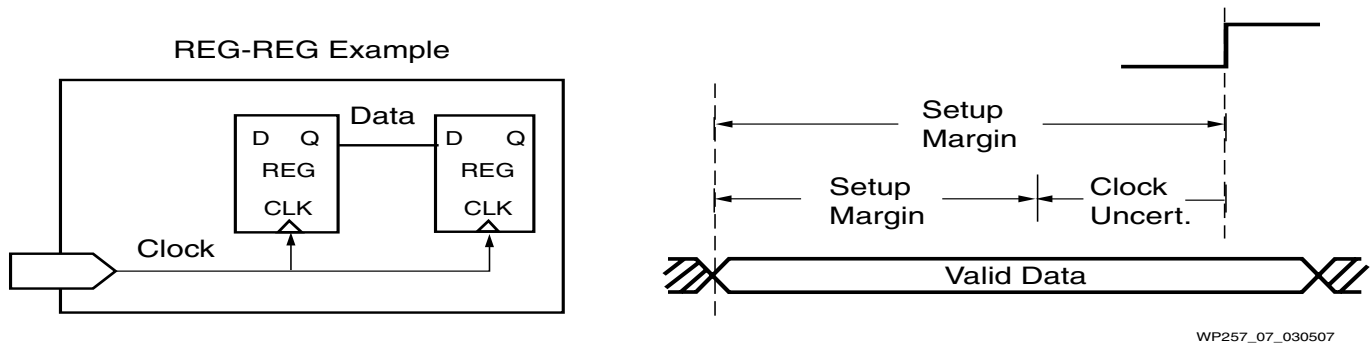


Figure 7: Reduced Setup Margin by Clock Uncertainty

The hold analysis ensures that the data changes at the destination synchronous element after the clock arrival.

Additionally, the data must stay valid at its input pins at least a hold time after the arrival of the active clock edge at its pin.

The equation for the hold analysis is the clock path skew plus the synchronous element hold time minus the data path delay. A hold time violation occurs when the positive clock skew is greater than the data path delay.

$$\text{Hold Time} = \text{Clock Path Skew} + \text{Synchronous Element Hold Time} - \text{Data Path Delay}$$

The clock uncertainty increases the clock path delay, so the needed hold margin is larger. This is shown in Figure 8.

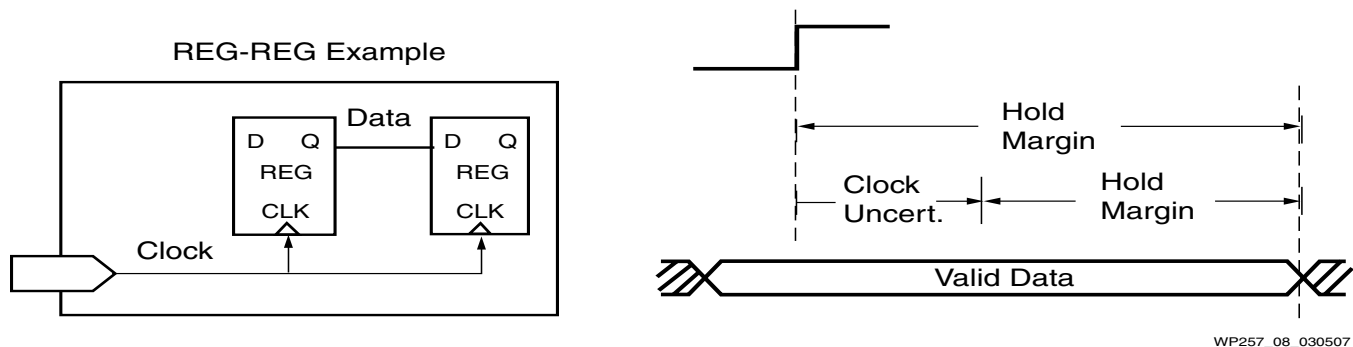
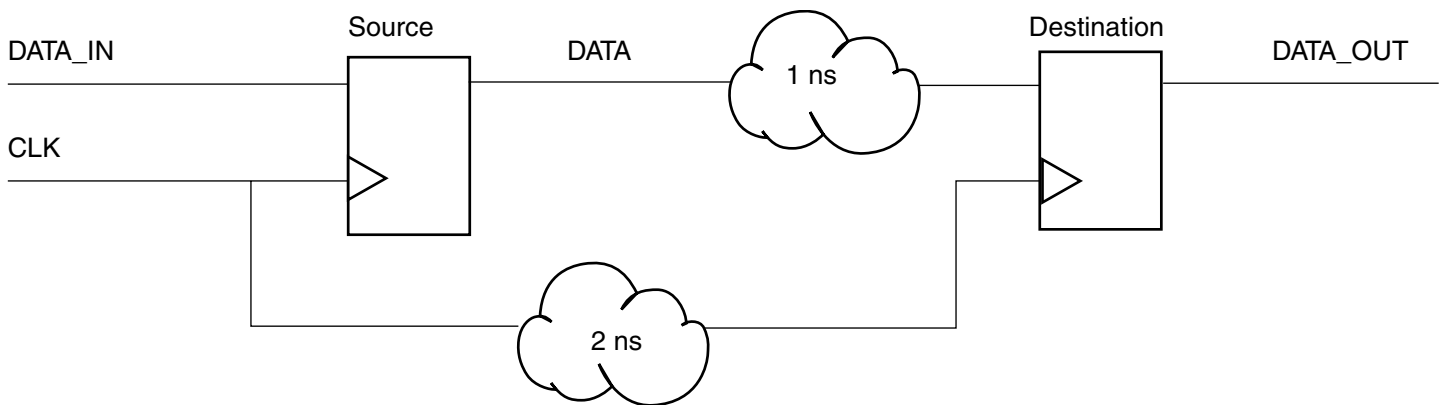


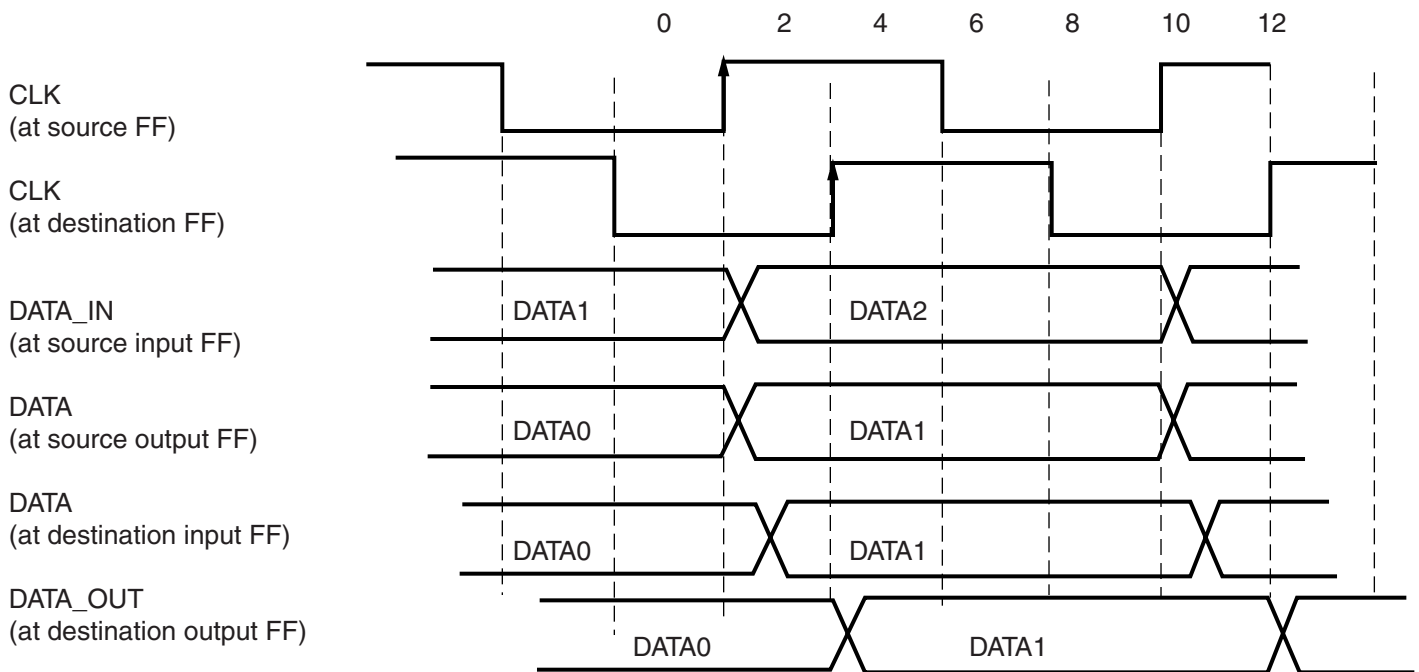
Figure 8: Increased Hold Margin by Clock Uncertainty

Both equations also include the “Clock-to-Out” time of the source synchronous element as a portion of the data path delay. In [Figure 9](#) and [Figure 10](#), the positive clock skew is greater than the data path delay; therefore, timing analysis will issue a hold violation.



WP257_09_030507

Figure 9: Hold Violation (Clock Skew > Data Path)



WP257_10_030507

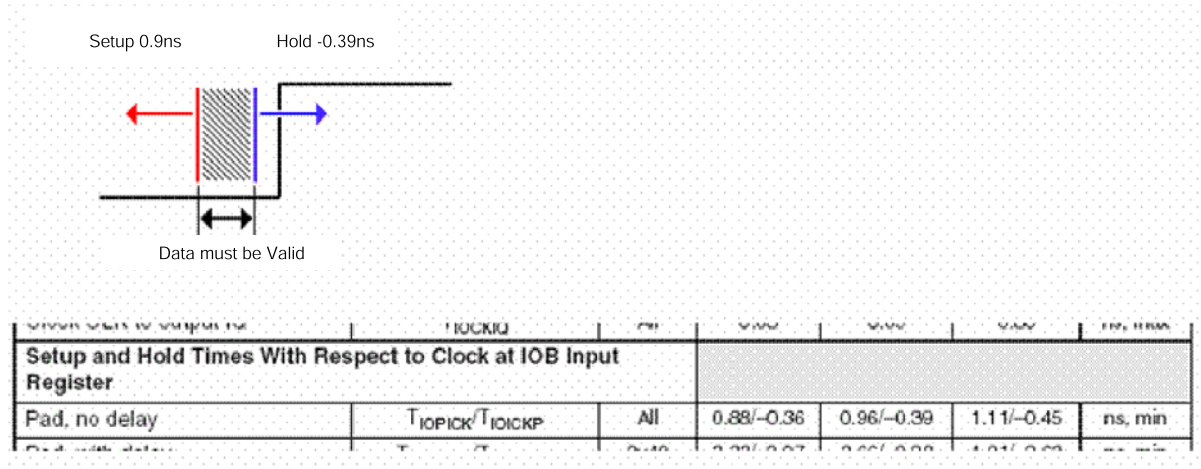
Figure 10: Hold Violation Waveform

The timing report will not list the hold paths, unless the path causes a hold violation.

To enable the reporting of the hold paths for each constraint, use the “-fastpaths” switch in tree or “Report Fast Paths Option” in Timing Analyzer. An example of setup and hold times from the data sheet are shown in Figure 11.

Historically, the setup and hold analysis in the timing report is smaller than the values in the device data sheet.

The values in the data sheet cover every pin and synchronous element, but the timing report is specific to your design for a specific pin or synchronous element.



WP257_11_030507

Figure 11: Setup/Hold times from Data Sheet

Clock Skew

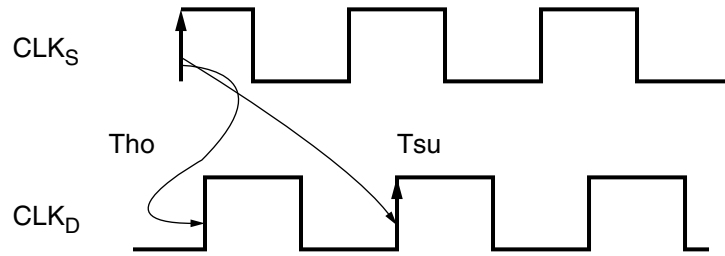
Clock skew analysis is included in both a setup and hold analysis. Clock skew is calculated based upon the clock path delay to the destination synchronous element minus the clock path delay to the source synchronous element.

Note: The majority of designs with a large clock skew find that the skew can be attributed to one of the following:

- ◆ One or both clocks using local routing
- ◆ One or both clocks are gated
- ◆ A DCM drives one clock and not the other clock

Clock skew is not the same as Phase. Phase is the difference in the clock arrival times, indicated by the source clock arrival time and the destination clock arrival time in the timing report. The arrival of the clocks is reported as clock arrival times, which is based upon the PHASE keyword in the PERIOD constraint. Clock skew is not accounted for in clock arrival times.

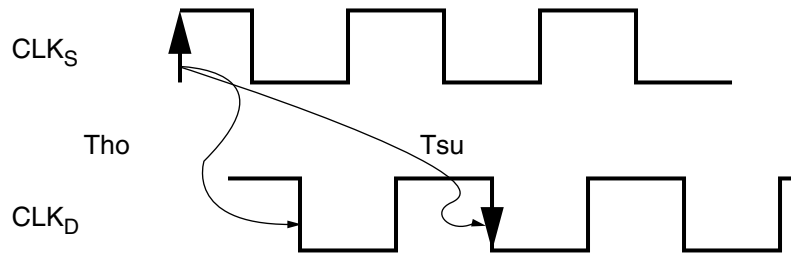
In the rising-to-rising setup/hold analysis shown in Figure 12, the positive clock skew greatly increases the chance of a hold violation and helps the setup calculation. During setup analysis, positive clock skew is truncated to zero. Positive clock skew is used during the hold analysis for this path.



WP257_12_03050:

Figure 12: Rising to Rising Setup/Hold Analysis

In the rising-to-falling setup/hold analysis shown in Figure 13 the positive clock skew is less, but the “ T_{ho} ” window is smaller and minimizes the chance for a hold violation. So, a two-phase clock is less likely to have a hold violation and can handle more positive clock skew than a single-phase clock path.



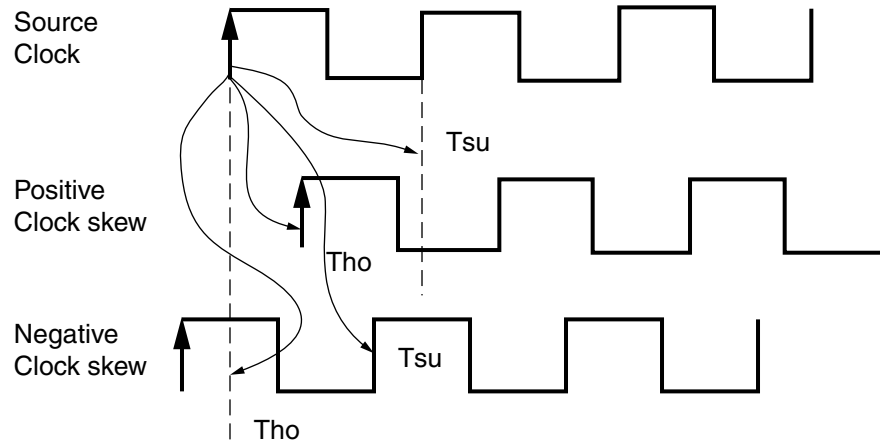
WP257_13_03050:

Figure 13: Rising to Falling Setup/Hold Analysis

During setup analysis, positive clock skew is truncated to zero and during hold analysis, the negative clock skew is truncated to zero.

During analysis of setup and hold, the negative clock skew and positive clock skew, respectively, decrease the margin on the PERIOD constraint requirement, as shown in Figure 14.

To determine how the timing analysis tools calculated the total clock skew for a path, use the “Analyze -> Against User Specified Paths” command in Timing Analyzer. Select the source and destination of the path in question, and do analysis from the clock source to the two elements in the path.



- Tsu and Tho represent the active edge the setup/hold violation calculation is done on, respectively.
- The dashed lines show the positive and negative clock skew being truncated to zero for the setup and hold checks, respectively.

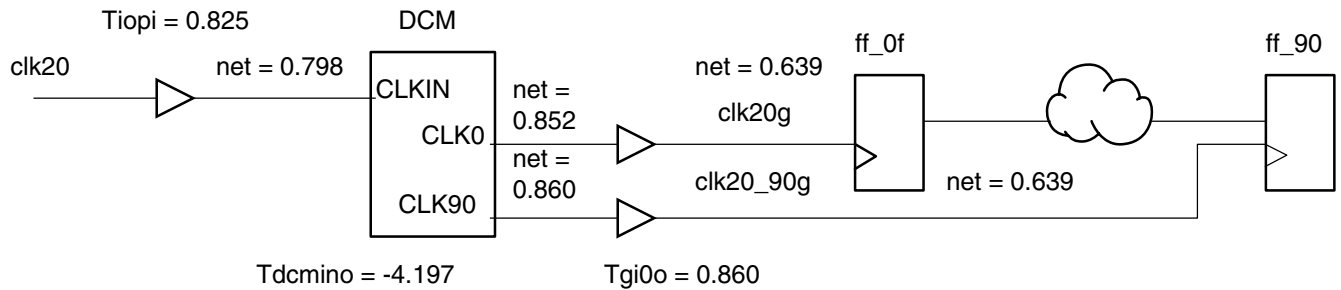
WP257_14_030507

Figure 14: Positive and Negative Clock Skew

The report will display the clock path to the source and the clock path to the destination. Review the paths to determine if the design has one of the causes of clock skew which were previously mentioned. The timing analysis tools subtract the clock path delays to produce the clock skew, as reported in the timing report.

The DLY file, produced by PAR, can also be used to determine the values used to calculate the clock skew value that was reported.

When calculating the clock path delay, the timing analysis tool traces the clock path to a common driver. In [Figure 15](#), the common driver of the clock path is at the DCM. If the tools cannot find a common driver, the analysis will start at the clock pads. In [Figure 15](#), the clock path delay from the DCM to the destination element is $(0.860 + 0.860 + 0.639) = 2.359$, and the clock path delay from the DCM to the source element is $(0.852 + 0.860 + 0.639) = 2.351$. The total clock skew is $2.359 - 2.351 = 0.008$ ns.



WP257_15_030507

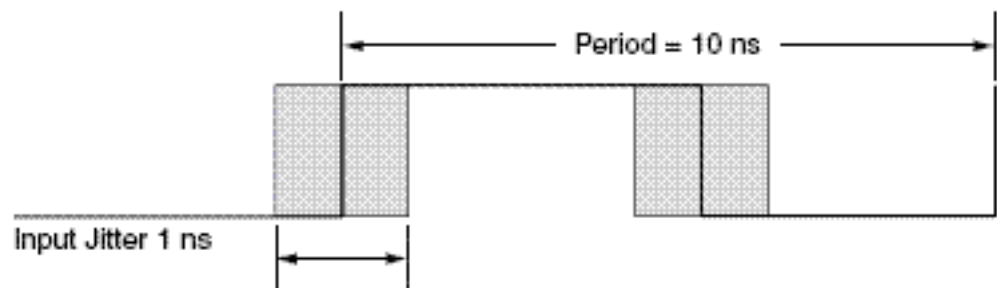
Figure 15: Clock Skew Example

Clock Uncertainty

In addition to the clock skew affecting the margin on the PERIOD constraint requirement, clock uncertainty also affects it.

Clock uncertainty is used to increase the timing accuracy by accounting for system, board level, and DCM clock jitter.

The SYSTEM_JITTER constraint and INPUT_JITTER keyword on the PERIOD constraint inform the timing analysis tools that the design has external jitter affecting the timing of this design, as shown in Figure 16.



WP257_16_030507

Figure 16: Input Jitter on Clock Signal

During the analysis for Virtex-4 and new families, the DCM Jitter, DCM Phase Error, and DCM Duty Cycle Distortion/Jitter are also included in the clock uncertainty. The individual components that make up clock uncertainty are reported in 9.1i, but only the total is reported in 8.x. The timing analysis tools calculate the clock uncertainty for the source and destination of a data path and combine them together to form the total clock uncertainty.

The following is the equation for DCM Clock Uncertainty:

- $Clock\ Uncertainty = [\sqrt{(INPUT_JITTER^2 + SYSTEM_JITTER^2)} + DCM_Discrete_Jitter]/2 + DCM_Phase_Error$
 - ◆ DCM Discrete Jitter and DCM Phase Error are provided in the speed files for Virtex-4 and newer devices.
- Example:

- ◆ INPUT_JITTER: $200\text{ps}^2 = 40000\text{ps}$
- ◆ SYSTEM_JITTER: $150\text{ps}^2 = 22500\text{ps}$
- ◆ DCM Discrete Jitter: 120ps
- ◆ DCM Phase Error: 0ps
- ◆ Clock Uncertainty: 185ps

An example of a PERIOD constraint with the INPUT_JITTER keyword:

- TIMESPEC “TS_Clk0” = PERIOD “clk0” 4 ns HIGH 60% INPUT_JITTER 200 ps PRIORITY 1;

An example of the SYSTEM_JITTER constraint:

- SYSTEM_JITTER = 150 ps;

Clock jitter consists of both random and discrete jitter components. Because the INPUT_JITTER and SYSTEM_JITTER are random jitter sources, and typically follow a Gaussian distribution, the combination of the two is added in a quadratic manner to represent the worst case combination.

Because the DCM Jitter is a discrete jitter value, it is added directly to the clock uncertainty.

In the analysis of clock uncertainty, all jitter components, both random and discrete, are specified as peak-peak values. Peak-peak values represent the total +/- range the arrival time of a clock signal will vary in the presence of jitter. In a worst-case analysis only the delay variation that causes a decrease in timing slack is used. For this reason, only the peak jitter value, or ½ the peak-to-peak value, is used for each setup and hold timing check.

The phase error component of clock uncertainty is a value representing the phase variation between two clock signals. Because this value is discrete, and represents the actual phase difference between the DCM clocks, it is added directly to the clock uncertainty value.

The following is the equation for PLL Clock Uncertainty:

- Clock Uncertainty = $[\sqrt{(\text{INPUT_JITTER}^2 + \text{SYSTEM_JITTER}^2 + \text{PLL_Discrete_Jitter}^2)}] / 2 + \text{PLL Phase_Error}$
 - ◆ PLL Discrete Jitter and PLL Phase Error are provided in the speed files for Virtex-5 devices.

In the analysis of clock uncertainty all jitter components, both random and discrete, are specified as peak-peak values. Peak-peak values represent the total +/- range the arrival time of a clock signal will vary in the presence of jitter. In a worst-case analysis only the delay variation that causes a decrease in timing slack is used.

For this reason, only the peak jitter value, or ½ the peak-to-peak value, is used for each setup and hold timing check.

The phase error component of clock uncertainty is a value representing the phase variation between two clock signals. Because this value is discrete, and represents the actual phase difference between the PLL clocks, it is added directly to the clock uncertainty value.

Asynchronous Reset Paths

The analysis of asynchronous reset paths, including the recovery time and reset pin to output time, is not included in the PERIOD constraint analysis by default.

In order to see these paths, a path tracing control (PTC) needs to be enabled, which is “ENABLE = REG_SR_R;”, for recovery time, and “ENABLE = REG_SR_O”, for output time.

These path tracing controls enable the path from the asynchronous reset pin through the synchronous element and the reset recovery time of the synchronous element.

What Is the PERIOD Constraint?

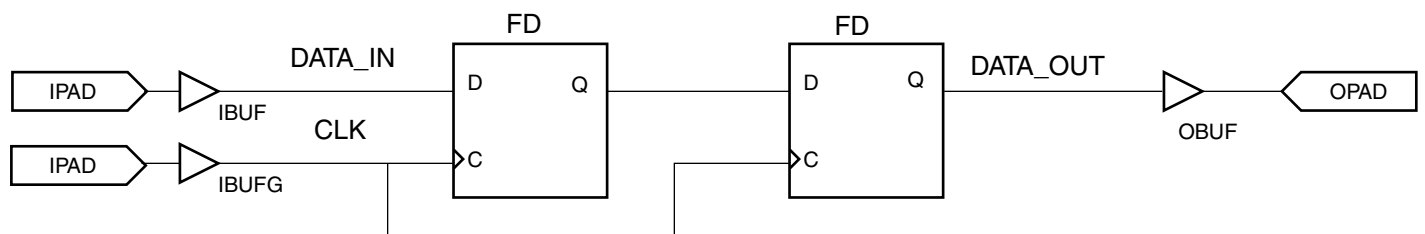
PERIOD constraints constrain those data paths from synchronous elements to synchronous elements. The most common examples are single clock domain, two-phase clock domain, and multiple clock domains. A timing report example is provided for each common type of path a PERIOD constrain might cover in your design.

Single Clock Domain

A single clock domain is easy to understand and analyze. All of the synchronous elements are on the same clock domain and are analyzed on the rising-edge of the clock or all elements are analyzed on the falling-edge of the clock. The clock source is driven by the same clock source, which can be a PAD or DCM/DLL/PLL/PMCD component with only one output.

The timing analysis tool will report the active edges of the clock driver and the corresponding time for the data path between the synchronous elements.

A simple design is shown in [Figure 17](#), and the PERIOD constraint is analyzed from the User Constraints File (UCF).



WP257_17_030507

Figure 17: Single Clock Domain Schematic

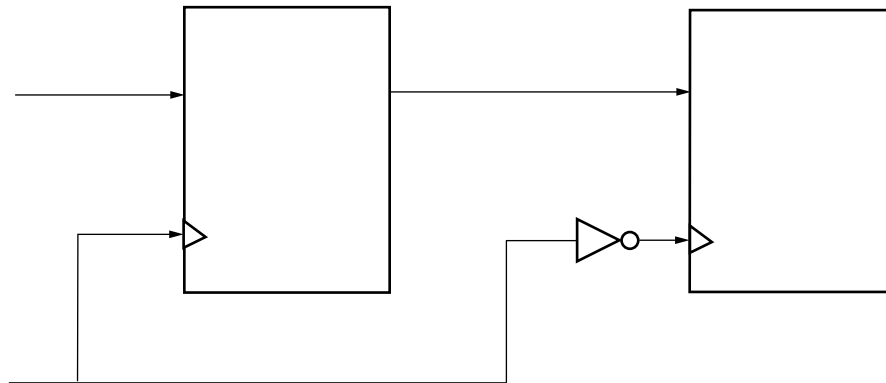
Timing Report Example

```
Slack (setup path):      3.904ns (requirement - (data path - clock path skew +
uncertainty))
Source:                  IntA\_1 (FF)
Destination:            XorA\_1 (FF)
Requirement:            8.000ns
Data Path Delay:        4.036ns (Levels of Logic = 1)
Clock Path Skew:        0.000ns
Source Clock:           clk0 rising at 0.000ns
Destination Clock:      clk0 rising at 8.000ns
Clock Uncertainty:      0.060ns
```

<u>Data Path: IntA_1 to XorA_1</u>			
Location	Delay type	Delay(ns)	Logical Resource(s)
ILOGIC_X1Y35.Q1	Tickq	0.566	IntA_1
SLICE_X46Y70.F1	net (fanout=3)	3.255	IntA<1>
SLICE_X46Y70.CLK	Tfck	0.215	Mxor_xor0000 Result<1>1 XorA_1
Total		4.036ns	(0.781ns logic, 3.255ns route) (19.4% logic, 80.6% route)

Two-Phase Clock Domain

The analysis of data path that use both edges of the clock, as in [Figure 18](#), is known as a two-phase clock. The clock is driven by the same clock source, which can be a PAD or DCM/DLL/PLL/PMCD component with only one output, but the design uses both edges of the clock. The timing analysis tool will report the active edges of the clock driver and the corresponding time for the data path between the synchronous elements. During analysis the requirement time is reduced by half the original requirement, as shown in [Figure 19](#).



WP257_18_030507

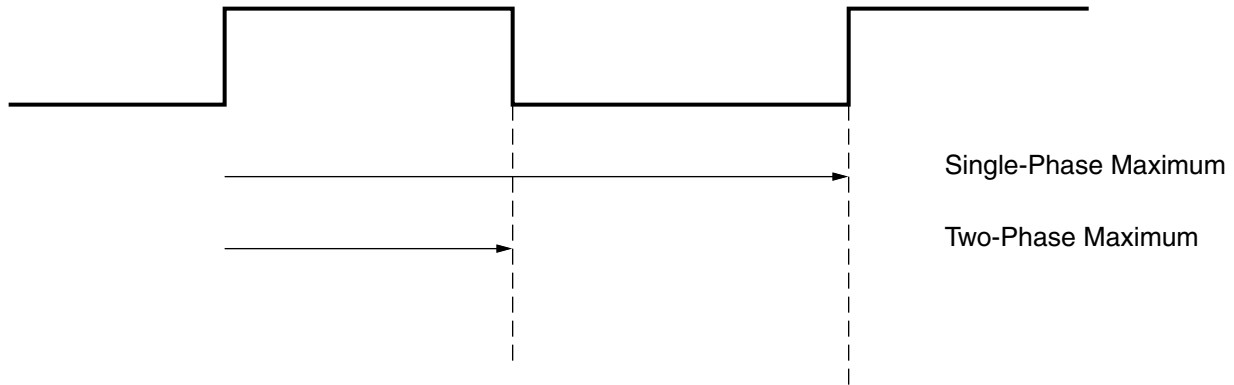
Figure 18: Two-Phase Clock

Since the timing report is sorted by the slack value, the worst slack valued path is listed first in the report for each constraint.

When the worst slack value path does not match the Minimum Period value, this is usually caused by the slack value on a two-phase path is not the largest or worst slack value.

The corresponding path to the Minimum Period value is down in the list of paths for the PERIOD constraint. Since the timing tools take the original requirement and reduce it by half

for the two phase clock, the total delay for a two-phase clock is then doubled to give a full period equivalent. This full period equivalent is then used as the Minimum Period value.



WP257_19_030507

Figure 19: Relationship between Single-Phase and Two-Phase Clocks

A two-phase clock example has a requirement of half the PERIOD constraint. If the PERIOD constraint is set to 6 ns and the timing analysis cuts the original requirement in half, to 3ns, for the two-phase data paths. If the two-phase data path has a worst-case delay as 1.309 ns, the full period equivalent is 2.618 ns. The slack on this two-phase data path is 1.691 ns = 3 ns – 1.309 ns. If a full-phase data path delay is 2 ns, which corresponds to a slack value of 4 ns, the two-phase data path will not be first in the timing report, but the Minimum Period value will be 2.618ns.

Timing Report Example

```
Slack (setup path): -1.096ns (requirement - (data path - clock path skew +
uncertainty))
Source: IntA\_1 (FF)
Destination: XorA\_1 (FF)
Requirement: 3.000ns
Data Path Delay: 4.036ns (Levels of Logic = 1)
Clock Path Skew: 0.000ns
Source Clock: clk0 rising at 0.000ns
Destination Clock: clk0 falling at 3.000ns
Clock Uncertainty: 0.060ns

Data Path: IntA\_1 to XorA\_1
-----
Location          Delay type          Delay(ns)          Logical Resource(s)
-----
ILOGIC_X1Y35.Q1    Tickq              0.566              IntA\_1
SLICE_X46Y70.F1    net (fanout=3)     3.255              IntA<1>
SLICE_X46Y70.CLK   Tfck               0.215              Mxor\_xor0000 Result<1>1
                   XorA\_1
-----
Total              4.036ns (0.781ns logic, 3.255ns route)
                   (19.4% logic, 80.6% route)
```

Multiple Clock Domains

A cross-clock domain path is a path that has two different clocks for the source and destination synchronous elements. One clock drives the source and a different clock drives the destination. If the source-clock-PERIOD constraint is related to the destination-clock-PERIOD constraint, the destination-clock-PERIOD constraint will cover the cross-clock-domain analysis.

Xilinx recommends relating the clocks via PERIOD constraints, so that the analysis will include the cross-clock domain paths properly.

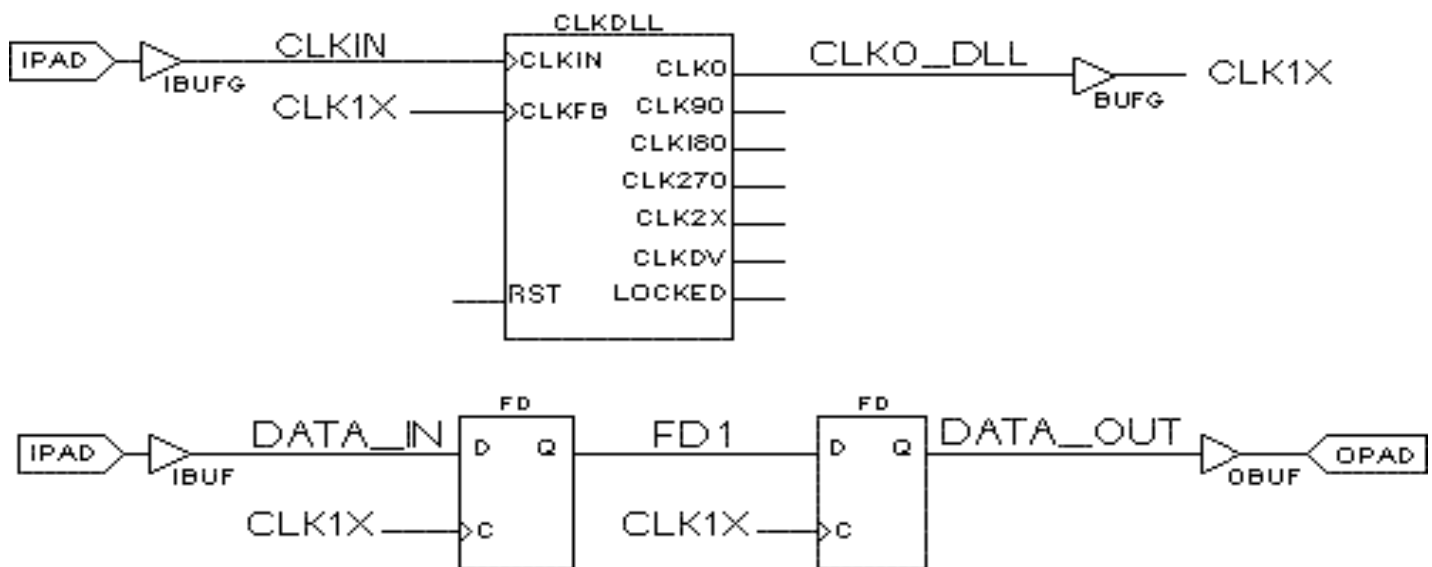
If the clocks are not related, the cross-clock domain paths will not be analyzed and Xilinx recommends using a FROM:TO or multicycle constraint to either flag it as a false path or multi-cycle path.

Clocks From DCM Outputs

The clock signals produced by a DCM/DLL/PLL/PMCD are related to each other, so the PERIOD constraints should also be related. This can be done one of two ways. One way is to let NGDBuild create new PERIOD constraints based upon the input clock signal PERIOD constraint. The second way is to manually create PERIOD constraints based upon the output clock signals of the DCM/DLL/PLL/PMCD and manually relate the PERIOD constraints.

Clk0 Clock Domain

Since the clocks produced by the DCM/PLL/DLL/PMCD are related, the timing tools take this relationship into consideration during analysis. The clock is driven by the same clock source from DCM/DLL/PLL/PMCD component. The timing analysis tool will report the active edges of the clock and the corresponding time for the data path between the synchronous elements. The example in Figure 20 shows a CLK0 clock circuit with a simple design. This clock domain will have the same requirement and phase shifting as the original requirement.



WP_20_030507

Figure 20: Clk0 DCM Output Schematic

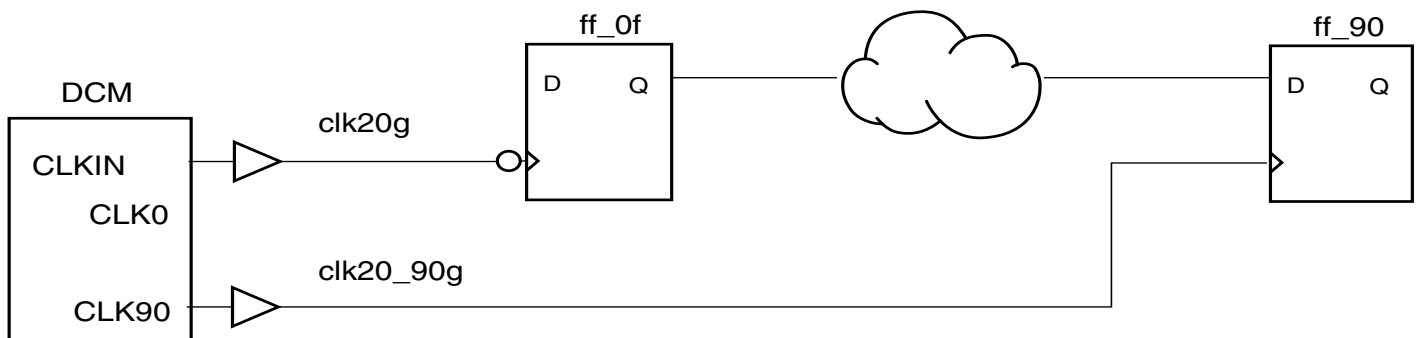
Timing Report Example

```
Slack (setup path):      3.904ns (requirement - (data path - clock path skew +
uncertainty))
Source:                  IntA\_1 (FF)
Destination:            XorA\_1 (FF)
Requirement:            8.000ns
Data Path Delay:        4.036ns (Levels of Logic = 1)
Clock Path Skew:        0.000ns
Source Clock:           clk0 rising at 0.000ns
Destination Clock:      clk0 rising at 8.000ns
Clock Uncertainty:      0.060ns
```

<u>Data Path: IntA_1 to XorA_1</u>			
Location	Delay type	Delay(ns)	Logical Resource(s)
ILOGIC_X1Y35.Q1	Tickq	0.566	IntA_1
SLICE_X46Y70.F1	net (fanout=3)	3.255	IntA<1>
SLICE_X46Y70.CLK	Tfck	0.215	Mxor_xor0000 Result<1>1 XorA_1
Total		4.036ns	(0.781ns logic, 3.255ns route) (19.4% logic, 80.6% route)

Clk90 Clock Domain

Since the clocks produced by the DCM/PLL/DLL/PMCD are related, the timing tools take this relationship into consideration during analysis. The clocks are driven by the same clock source, which can be different outputs of the DCM/DLL/PLL/PMCD component. The timing analysis tool will report the active edges of the clock and the corresponding time for the data path between the synchronous elements. The example in Figure 21 shows a CLK0 and CLK90 signals and the phase difference is 90 degrees.



WP257_21_030507

Figure 21: Clock Phase between DCM Outputs

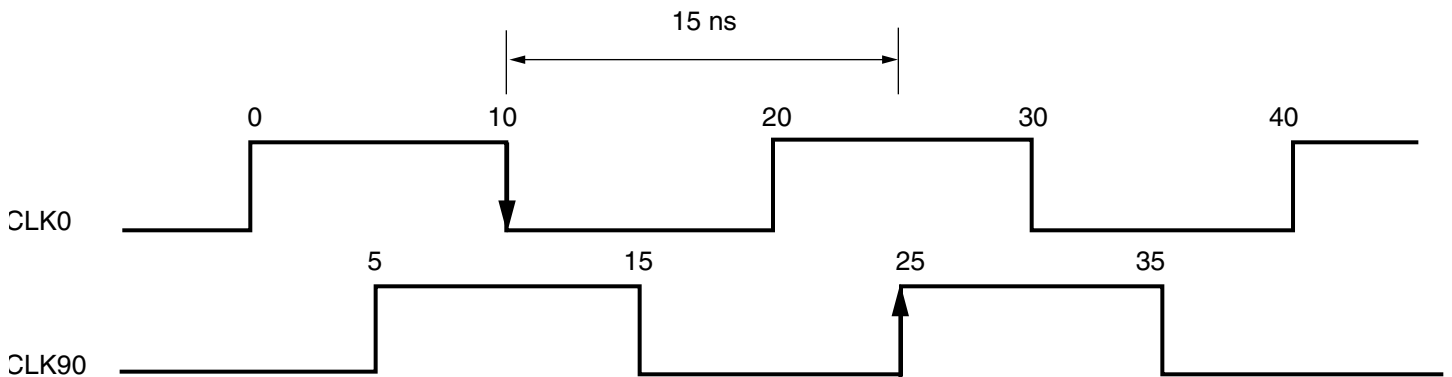
Another cause of the Minimum PERIOD value being different than the first path listed in the timing report is a cross-clock domain analysis of phase-shifted clocks.

If the phase difference between the two clock domains is 90 degrees, the total data delay will be multiplied by four to get to a full period value.

If the data path was 1.5ns for this clock90 constraint, the equivalent full period value would be 6 ns.

In addition, the data path goes from a falling-edge of CLK0 clock signal to the rising-edge of CLK90 clock signal, and the timing analysis includes the two-phase information from CLK0 to

do the analysis, as shown in Figure 19. The original PERIOD constraint was set to 20 ns, but this cross-clock domain analysis has the new requirement of 15 ns, to compensate for the phase difference between the two clocks, as shown in Figure 22.



WP257_22_030507

Figure 22: Clock Edge Relationship

Timing Report Example

```
Slack (setup path): 5.398ns (requirement - (data path - clock path skew +
uncertainty))
Source: IntB 2 (FF)
Destination: XorB 2 (FF)
Requirement: 8.000ns
Data Path Delay: 2.542ns (Levels of Logic = 1)
Clock Path Skew: 0.000ns
Source Clock: clk0 falling at 2.000ns
Destination Clock: clk90 rising at 10.000ns
Clock Uncertainty: 0.060ns
```

Data Path: [IntB 2](#) to [XorB 2](#)

Location	Delay type	Delay(ns)	Logical Resource(s)
ILOGIC_X1Y86.Q1	Tickq	0.566	IntB 2
SLICE_X35Y80.G1	net (fanout=3)	1.745	IntB<2>
SLICE_X35Y80.CLK	Tgck	0.231	Mxor_xor0001 Result<2>1 XorB 2
Total		2.542ns	(0.797ns logic, 1.745ns route) (31.4% logic, 68.6% route)

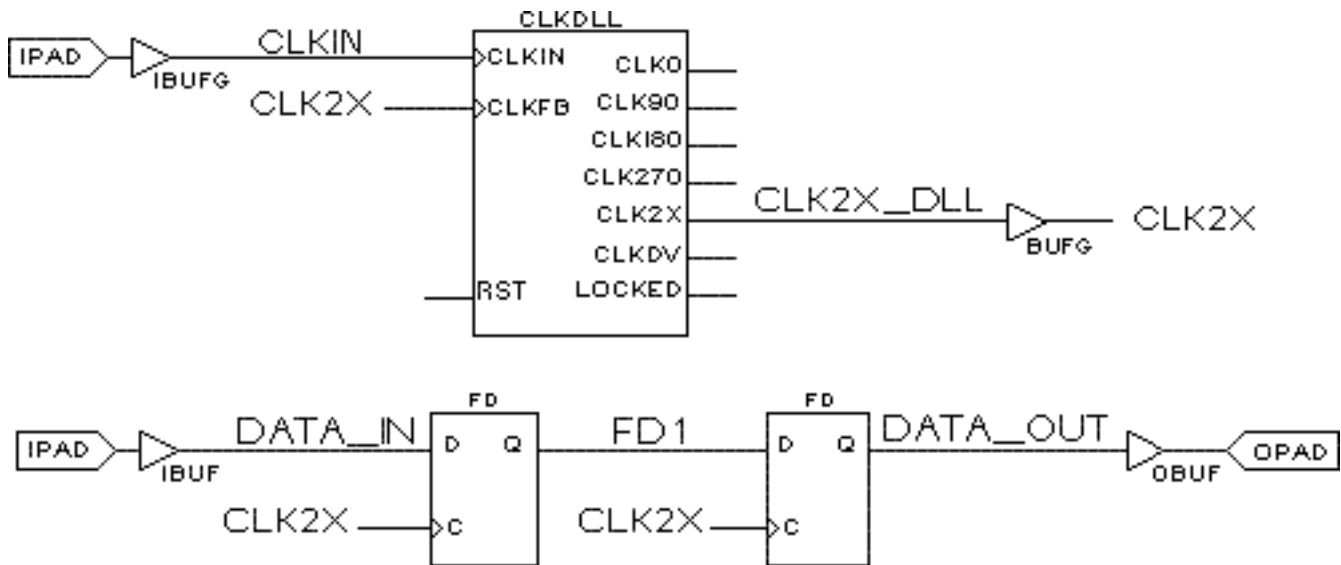
```
Slack (setup path): 13.292ns (requirement - (data path - clock path skew +
uncertainty))
Source: IntC 2 (FF)
Destination: XorB 2 (FF)
Requirement: 15.000ns
Data Path Delay: 2.594ns (Levels of Logic = 1)
Clock Path Skew: -0.086ns
Source Clock: clk0 falling at 10.000ns
Destination Clock: clk90 rising at 25.000ns
Clock Uncertainty: 0.200ns
```

Data Path: [IntC 2](#) to [XorB 2](#)

Location	Delay type	Delay(ns)	Logical Resource(s)
ILOGIC_X2Y97.Q1	Tickq	0.566	IntC 2
SLICE_X35Y80.G4	net (fanout=3)	1.797	IntC<2>
SLICE_X35Y80.CLK	Tgck	0.231	Mxor_xor0001 Result<2>1 XorB 2
Total		2.594ns	(0.797ns logic, 1.797ns route) (30.7% logic, 69.3% route)

Clk2x Clock Domain

Since the clocks produced by the DCM/PLL/DLL/PMCD are related, the timing tools take this relationship into consideration during analysis. A simple design of a CLK2X clock domain is illustrated in [Figure 23](#). The clock is driven by the same clock source, which is an output of a DCM/DLL/PLL/PMCD component. The timing analysis tool will report the active edges of the clock and the corresponding time for the data path between the synchronous elements. This clock domain will have half the requirement of the original requirement, but the phase shifting is the same as the phase shifting of the original requirement.



WP257_23_030507

Figure 23: Clk2x DCM Output Schematic

Timing Report Example

```
Slack (setup path): -1.663ns (requirement - (data path - clock path skew +
uncertainty))
Source: IntA 3 \(FF\)
Destination: OutB 3 \(FF\)
Requirement: 2.000ns
Data Path Delay: 3.443ns (Levels of Logic = 0)
Clock Path Skew: -0.020ns
Source Clock: clk2x rising at 0.000ns
Destination Clock: clk2x falling at 2.000ns
Clock Uncertainty: 0.200ns
```

[Timing Improvement Wizard](#)

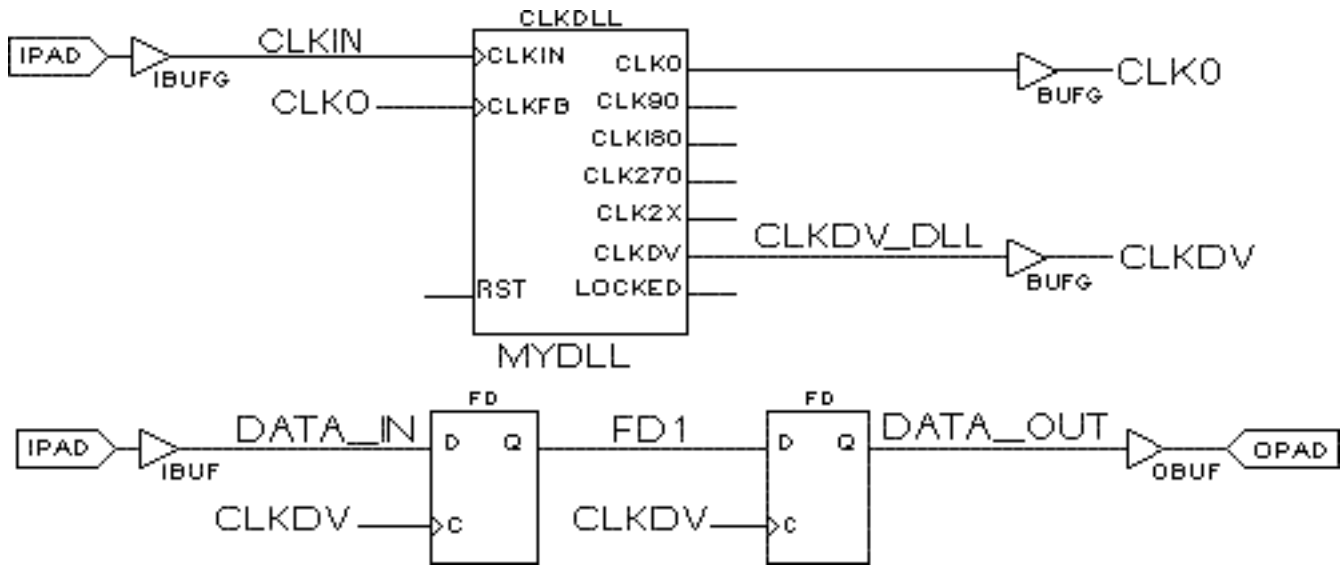
[Data Path: IntA 3 to OutB 3](#)

Location	Delay type	Delay(ns)	Logical Resource(s)
ILOGIC_X2Y96.Q1	Tickq	0.566	IntA 3
OLOGIC_X2Y23.SR	net (fanout=3)	1.580	IntA<3>
OLOGIC_X2Y23.CLK	Tosrck	1.297	OutB 3
Total		3.443ns	(1.863ns logic, 1.580ns route) (54.1% logic, 45.9% route)

CLKDV/CLKFX Clock Domain

Since the clocks produced by the DCM/PLL/DLL/PMCD are related, the timing tools take this relationship into consideration during analysis. The CLKDV and CLKFX outputs can be used to make clock signals that are derivatives of the original input clock signal, as shown in Table 1. The clock is driven by two different outputs of the DCM/DLL/PLL/PMCD component. The

timing analysis tool will report the active edges of the clock and the corresponding time for the data path between the synchronous elements. The simple design of a CLKDV clock domain is shown in Figure 24. This clock domain will have twice the requirement as the original requirement, but the phase shifting is the same as the phase shifting of the original requirement.



WP257_24_030507

Figure 24: CLKDV DCM Output Schematic

Timing Report Example

```
Slack (setup path):      1.909ns (requirement - (data path - clock path skew +
uncertainty))
Source:                  XorC 7 (FF)
Destination:            OutC 7 (FF)
Requirement:            4.000ns
Data Path Delay:        1.810ns (Levels of Logic = 0)
Clock Path Skew:        0.000ns
Source Clock:           clk0 rising at 0.000ns
Destination Clock:     clkdv rising at 4.000ns
Clock Uncertainty:      0.281ns
Timing Improvement Wizard
Data Path: XorC 7 to OutC 7
```

Location	Delay type	Delay(ns)	Logical Resource(s)
SLICE_X37Y76.XQ	Tcko	0.340	XorC 7
OLOGIC_X2Y76.D1	net (fanout=1)	0.843	XorC<7>
OLOGIC_X2Y76.CLK	Todck	0.627	OutC 7
Total		1.810ns	(0.967ns logic, 0.843ns route) (53.4% logic, 46.6% route)

PERIOD Constraint Syntax

The syntax is conversational and easy to understand. For more details, please reference the *Constraints Guide*, under PERIOD constraints.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
3/21/07	1.0	Initial Xilinx release.