Advanced Static Timing Analysis Using SmartTime

Table of Contents

Introduction . . . 1
Overview of SmartTime Timing Analysis . . . 2
Timing Analysis for Generated Clocks . . . 3
Inter-Clock Domain Analysis with Two Asynchronous Clocks . . . 7
Inter-Clock Domain Analysis for Generated Clocks . . . 12
Analyzing Source Synchronization . . . 15
Analyzing Design with Jitter/Clock Uncertainty in SmartTime . . . 15
Analyzing a Multicycle Path with Single Clock Domain . . . 18
Analyzing a Multicycle Path with Inter-Clock Domain . . . 21
Analyzing Clock Gating . . . 26
Four Corner Analysis . . . 30
Timing Analysis for Min-WORST or Max-BEST Scenario . . . 32
Appendix A: Applying a Clock Constraint . . . 35
Appendix B: Applying a Generated Clock Constraint . . . 37
Appendix C: Enabling Inter-Clock Domains Analysis . . . 40
Appendix D: Applying a Multicycle Clock Constraint . . . 41

Introduction

Complex and sophisticated clocking schemes and exceptions are currently used in low power and high-reliability Microsemi FPGA devices. Increasing complexity results in the need for more timing analysis capabilities that will be required for sign-off and validation. The SmartTime FPGA timing analysis tool, available in the Microsemi Libero® Integrated Design Environment (IDE) software suite, allows you to do the basic timing analysis for simple clocking schemes as well as the required analysis of complex clocking schemes. This application note describes advanced timing analysis with detailed steps using the Microsemi SmartTime FPGA timing analysis tool.

This document gives a quick overview of timing analysis using the SmartTime tool and then provides an example of advanced timing analysis as listed below:

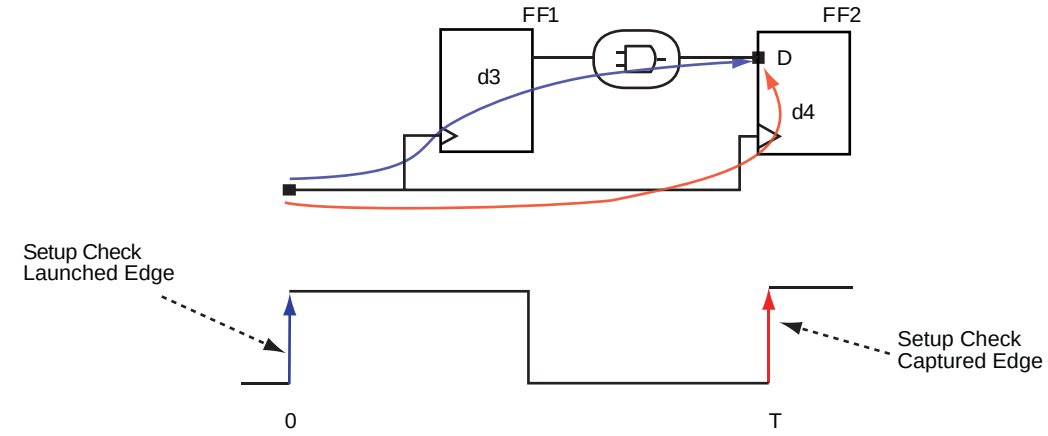
1. Timing analysis for a generated clock
2. Inter-clock domain analysis with two asynchronous clocks
3. Inter-clock domain analysis for generated clocks
4. Analyzing source synchronization
5. Analyzing a design with jitter/clock uncertainty in SmartTime
6. Analyzing a multicycle path with a single clock domain
7. Analyzing a multicycle path with inter-clock domain
8. Analyzing clock gating
9. Four corner analysis

Overview of SmartTime Timing Analysis

SmartTime is the gate-level static timing analysis (STA) tool for SmartFusion® customizable system-on-chip (cSoC), RTAX™-S/SL, Fusion®, IGLOO®, ProASIC®3, Axcelerator®, eX, and SXA families. The SmartTime graphical user interface (GUI) provides the SmartTime Timing Analyzer for static timing analysis and SmartTime Constraints Editor for applying SDC constraints in the design.

The SmartTime Timing Analyzer has two timing analysis views: Maximum Delay Analysis and Minimum Delay Analysis. The maximum delay analysis view checks the setup timing and the minimum delay analysis checks the hold timing. SmartTime constraints editor enables you to create, view, and edit the timing constraints of the selected scenario for use with SmartTime timing analysis.

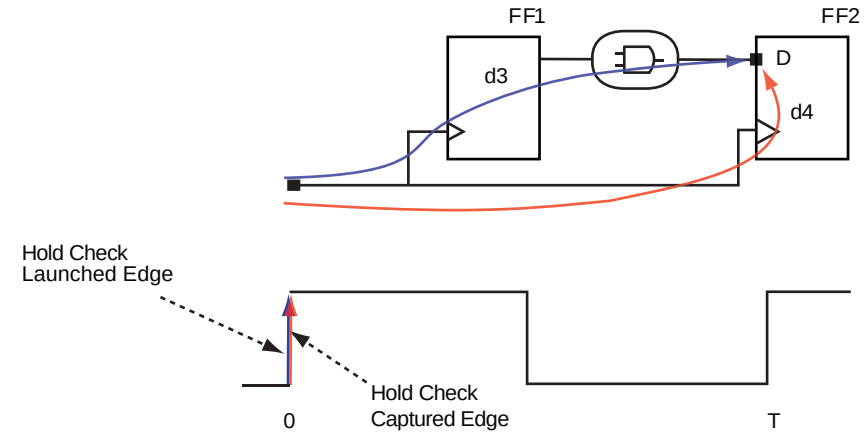
The setup check in SmartTime involves comparing the latest data arrival time (longest data path delay) with the earliest required time (shortest clock path delay). The hold check in SmartTime involves comparing the earliest data arrival time (shortest data path delay) with the latest required time (longest clock path delay). Both setup and hold checks calculate the timing delay with respect to launched edge and captured edge, as shown in Figure 1 and Figure 2 on page 3. This is the base for all timing analysis and also used for all advanced timing analysis. Refer to the *SmartTime* Tutorial to understand basic timing analysis using the SmartTime tool.



Setup Check Calculation

- Arrival Time = Launched Edge (0) + Max. Clock to FF1 + Max. Data Path
- Required Time = Captured Edge (T) + Min. Clock to FF2 – Setup of FF2

***Figure 1* • Setup Check Calculation**



Hold Check Calculation

- Arrival Time = Launched Edge (0) + Min. Clock to FF1 + Min. Data Path
- Required Time = Captured Edge (0) + Max. Clock to FF2 + Hold of FF2

***Figure 2* • Hold Check Calculation**

The following section describes various methods of advanced timing analysis.

Timing Analysis for Generated Clocks

Many designs have clocks that are generated internally via phase-locked loop (PLL), clock divider, or other allowed methods. The SmartTime tool allows you to generate the clock constraints for the internally generated clocks and verifies their timing behavior. You need to apply a clock constraint on the main clock. For the clock generated via PLL, SmartTime creates the constraints for the generated clocks and applies it automatically during timing analysis. For the clock generated via clock divider, you need to manually apply the generated clock constraint.

Consider the design example shown in Figure 3. CLKA is the main clock, running at 50 MHz. PLL_50_20_0/Core:GLA and DFN1_0:Q are generated via PLL and clock divider. The following section describes the timing analysis for these two generated clocks.

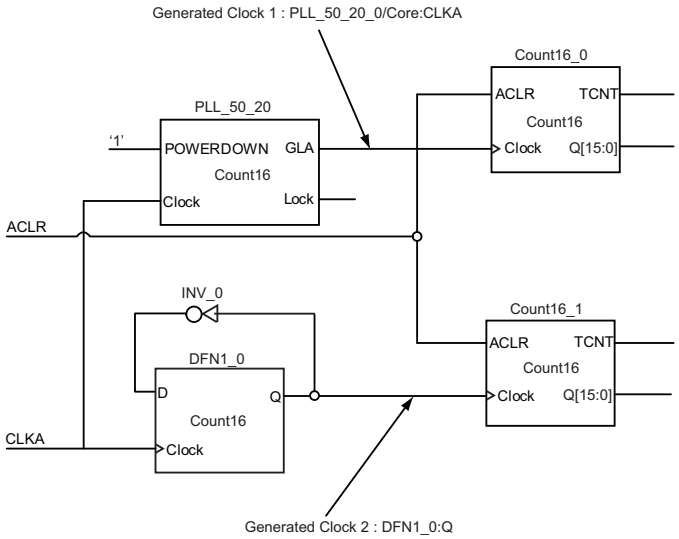
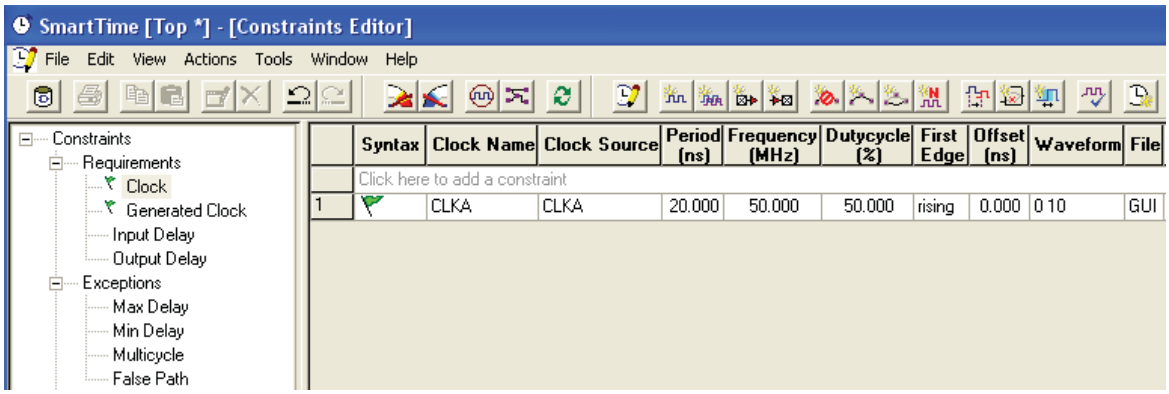


***Figure 3* • Design Example for Generated Clock**

Analyzing Generated Clock Domain Timing with SmartTime

1. Specify the reference clock frequency and other attributes. Refer to "Appendix A: Applying a Clock Constraint" on page 35 for creating a clock constraint using the GUI.




```
create_clock -name { CLKA } -period 20.000 -waveform { 0.000 10.000 } { CLKA }
```

***Figure 4* • Clock Constraints Using Constraints Editor and SDC**

On applying the reference clock constraint, the generated clock constraint for the PLL will be created by the SmartTime tool automatically. SmartTime reads the netlist that has the PLL divider setting and then automatically populates the divider ratio. However, you still need to identify other generated clocks and apply generated clock constraints.

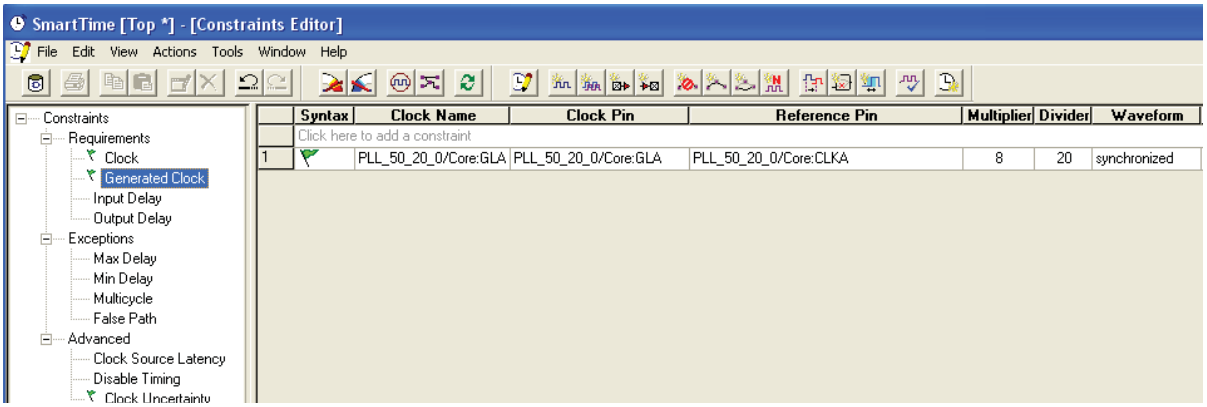
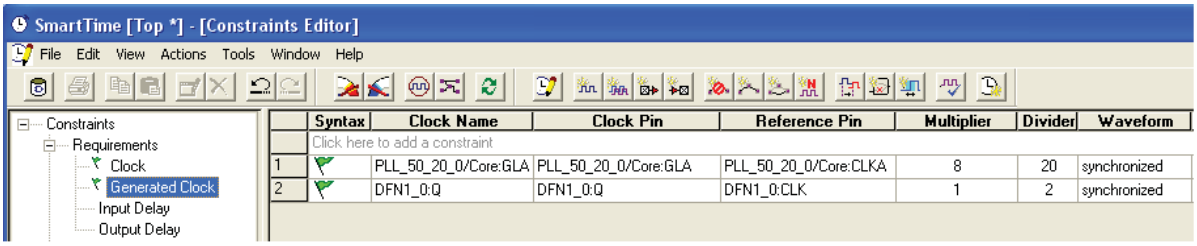


***Figure 5* • Automatically Generated PLL Clock Constraint in Constraints Editor**

2. Identify the generated clock and apply the generated clock constraint. Refer to "Appendix B: Applying a Generated Clock Constraint" on page 37 for creating a generated clock constraint using the GUI.




```
create_generated_clock -name{PLL_50_20_0/Core:GLA}-divide_by 20 -multiply_by 8 -
source/{PLL_50_20_0/Core:CLKA}{PLL_50_20_0/Core:GLA}
create_generated_clock -name{DFN1_0:Q}-divide_by 2 -source{DFN1_0:CLK}{DFN1_0:Q}
```

Figure 6 • **Generated Clock Constraint**

The maximum delay analysis view displays the timing analysis for the reference clock, CLKA, and the two generated clocks, PLL_50_20_0/Core:GLA and DFN1_0:CLK.

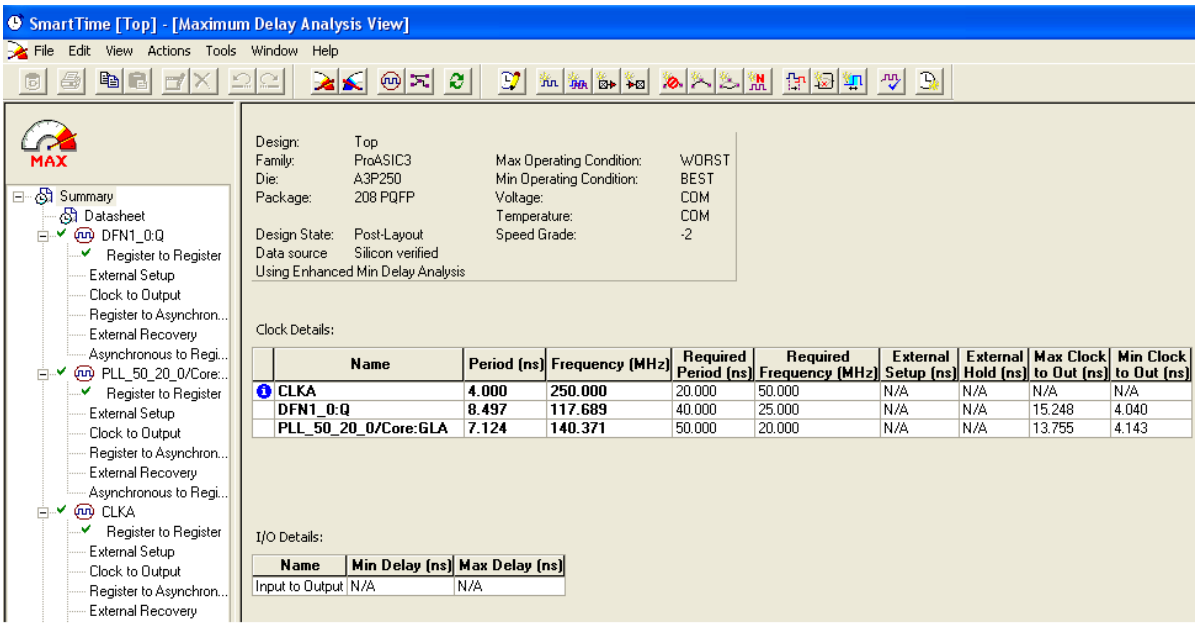


Figure 7 • **Maximum Delay Analysis Showing All Clocks**

The timing analysis for the internally generated clocks is shown in Figure 8.

Summary for path
From: Count16_1/DFN1C1_NU_0:CLK
To: Count16_1/DFN1E1C1_NU_12/U1:D

	Data Required Time (ns)	Data Arrival Time (ns)	Slack (ns)
1	43.354	11.851	31.503

Path details

Pin Name	Type	Net Name	Cell Name	Op	Delay (ns)	Total (ns)	Fanout	Edge
Data arrival time calculation								
DFN1_0:Q					0.000	0.000		
DFN1_0:Q	Clock source			+	0.000	0.000		r
	Clock generation			+	1.963	1.963		
DFN1_0_RNIKE25:A	net	DFN1_0_Q_i		+	0.759	2.722		r
DFN1_0_RNIKE25:Y	cell		ADLIB:CLKINT	+	0.558	3.280	17	r
Count16_1/DFN1C1_NU_0:CLK	net	DFN1_0_Q		+	0.502	3.782		r
Count16_1/DFN1C1_NU_0:Q	cell		ADLIB:DFN1C1	+	0.434	4.216	4	r
Count16_1/U_AND3_0_1_2:A	net	Count16_1/DFN1C1_NU_0		+	1.142	5.358		r
Count16_1/U_AND3_0_1_2:Y	cell		ADLIB:AND3	+	0.392	5.750	7	r
Count16_1/U_U_AND3_0_to_8:A	net	Count16_1/NU_0_1_2		+	2.749	8.499		r
Count16_1/U_U_AND3_0_to_8:Y	cell		ADLIB:AND3	+	0.392	8.891	8	r
Count16_1/DFN1E1C1_NU_12/U0:S	net	Count16_1/NU_0_to_8		+	2.441	11.332		r
Count16_1/DFN1E1C1_NU_12/U0:Y	cell		ADLIB:MX2	+	0.278	11.610	1	f
Count16_1/DFN1E1C1_NU_12/U1:D	net	Count16_1/DFN1E1C1_NU_12/Y		+	0.241	11.851		f
data arrival time						11.851		
Data required time calculation								
DFN1_0:Q	Clock Constraint				40.000	40.000		
DFN1_0:Q	Clock source			+	0.000	40.000		r
	Clock generation			+	1.963	41.963		
DFN1_0_RNIKE25:A	net	DFN1_0_Q_i		+	0.759	42.722		r
DFN1_0_RNIKE25:Y	cell		ADLIB:CLKINT	+	0.558	43.280	17	r
Count16_1/DFN1E1C1_NU_12/U1:CLK	net	DFN1_0_Q		+	0.502	43.782		r
Count16_1/DFN1E1C1_NU_12/U1:D	Library setup time		ADLIB:DFN1C1	-	0.428	43.354		
data required time						43.354		

***Figure 8* • Setup Check for the Generated Clock**

Note: SmartTime automatically calculates the clock generation delay. Figure 9 on page 7 shows the calculation of the delay from the CLKA port to the output pin of the clock divider.

	CLKA					0.000	0.000		
	CLKA	Clock source			+	0.000	0.000		r
	CLKA_pad/U0/U0:PAD	net	CLKA		+	0.000	0.000		r
	CLKA_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	0.758	0.758	1	r
	CLKA_pad/U0/U1:A	net	CLKA_pad/U0/NET1		+	0.000	0.758		r
	CLKA_pad/U0/U1:Y	cell		ADLIB:CLKIO	+	0.260	1.018	2	r
	DFN1_0:CLK	net	CLKA_c		+	0.511	1.529		r
	DFN1_0:Q	cell		ADLIB:DFN1	+	0.434	1.963	1	r

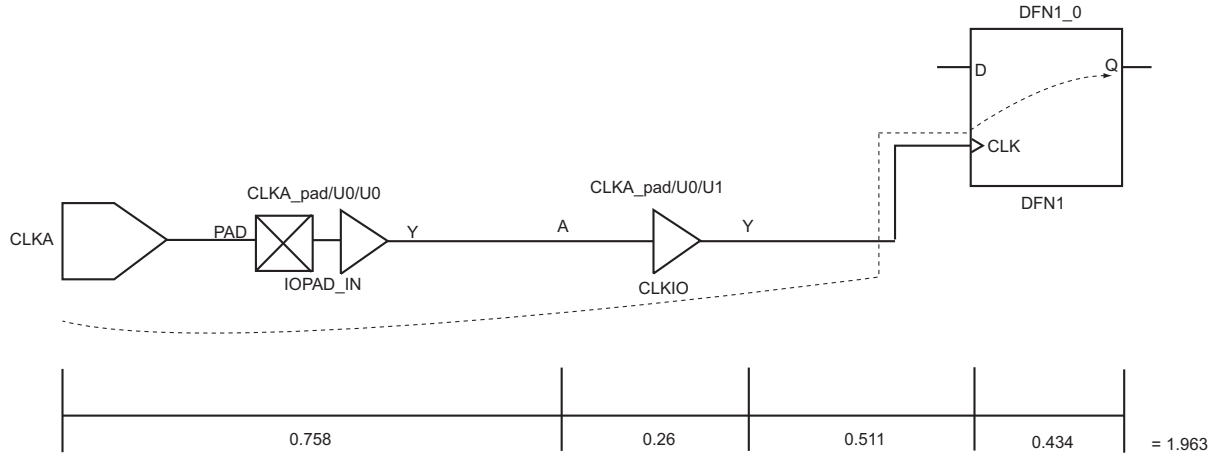


***Figure 9* • Delay Calculation for Clock Generation**

Inter-Clock Domain Analysis with Two Asynchronous Clocks

SmartTime enables inter-clock domain timing checks for designs containing functional paths that exist across two clock domains (the register launching the data and the register capturing the data are clocked by two asynchronous clock sources). Accurate specifications of both clocks are required to allow a valid inter-clock domain timing check.

Note: The default SmartTime setting does not show inter-clock domain analysis. You need to change the setting (see "Appendix C: Enabling Inter-Clock Domains Analysis" on page 40) to enable the inter-clock domain analysis. Depending on the design, some of the inter-clock domain paths are valid timing paths and some are false paths. It is the designer's responsibility to identify these paths and apply the timing exception as needed.

For an inter-clock domain path, SmartTime analyzes the relationship between all the active clock edges over a common period equal to the least common multiple of the two clock periods. For a setup check, the tightest relation of launch to capture is considered to ensure that the data arrives before the capture edge.

The hold check verifies that a setup relationship is not overwritten by a following data launch. The clock edge used for setup and hold analysis is shown in Figure 10.

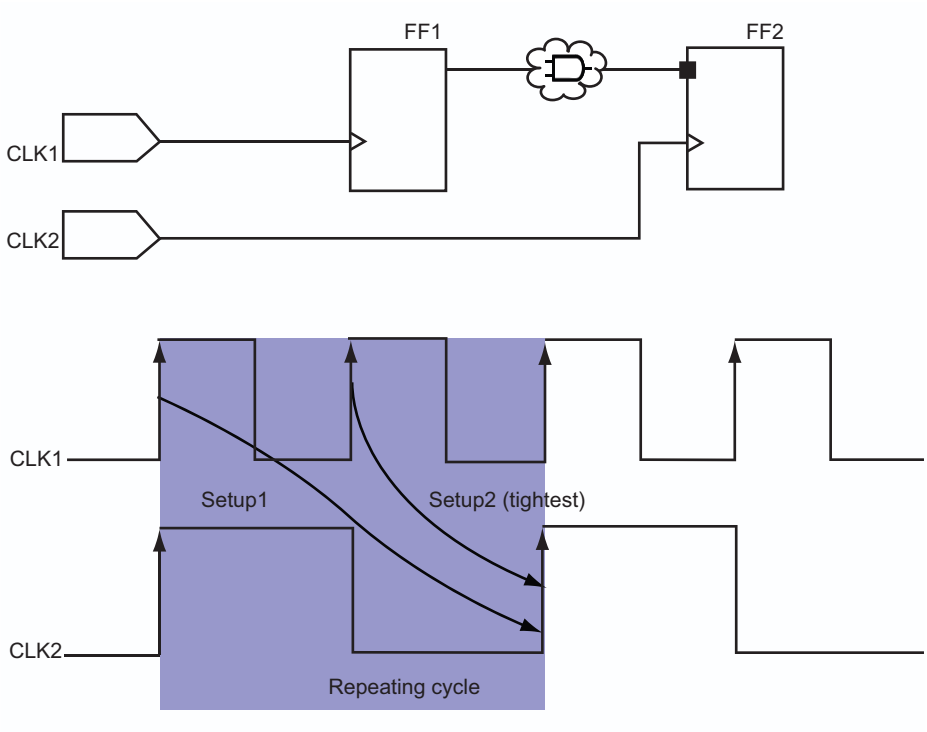


***Figure 10* • Clock Relationship for Inter-Domain Clocks**

Consider the inter-domain design example shown in Figure 11. Note the path from the CLK1 domain to the CLK2 domain, which is a valid inter-clock domain path. Assume that CLK1 is 100 MHz and CLK2 is 75 MHz and both have zero offset. The "Analyzing Inter-Clock Domain Timing with SmartTime" section on page 9 shows how to analyze this cross-clock domain path.

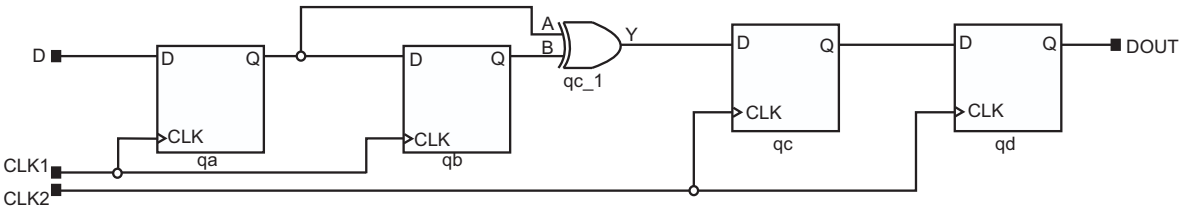
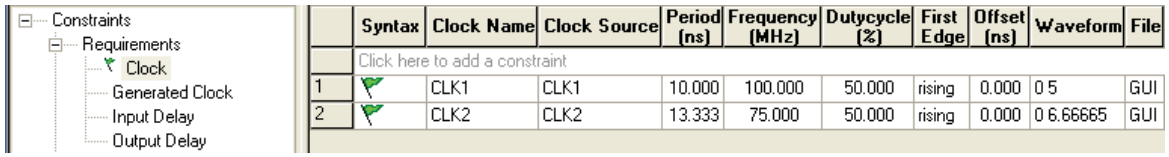


***Figure 11* • Inter-Clock Domain Example**

Analyzing Inter-Clock Domain Timing with SmartTime

1. Specify the clock frequency and other attributes for both reference clocks. Refer to "Appendix A: Applying a Clock Constraint" on page 35 for creating a generated clock constraint using the GUI.




```
create_clock -name{CLK1}-period 10.000-waveform{0.000 5.000}{CLK1}
create_clock -name{CLK2}-period 13.333-waveform{0.000 6.667}{CLK2}
```

***Figure 12* • Clock Constraint Using Constraints Editor and SDC**

2. Enable inter-clock domain analysis. Refer to "Appendix C: Enabling Inter-Clock Domains Analysis" on page 40.

The maximum delay analysis view displays the timing analysis for CLK1 to CLK2 under CLK2 domain analysis, as shown in Figure 13.

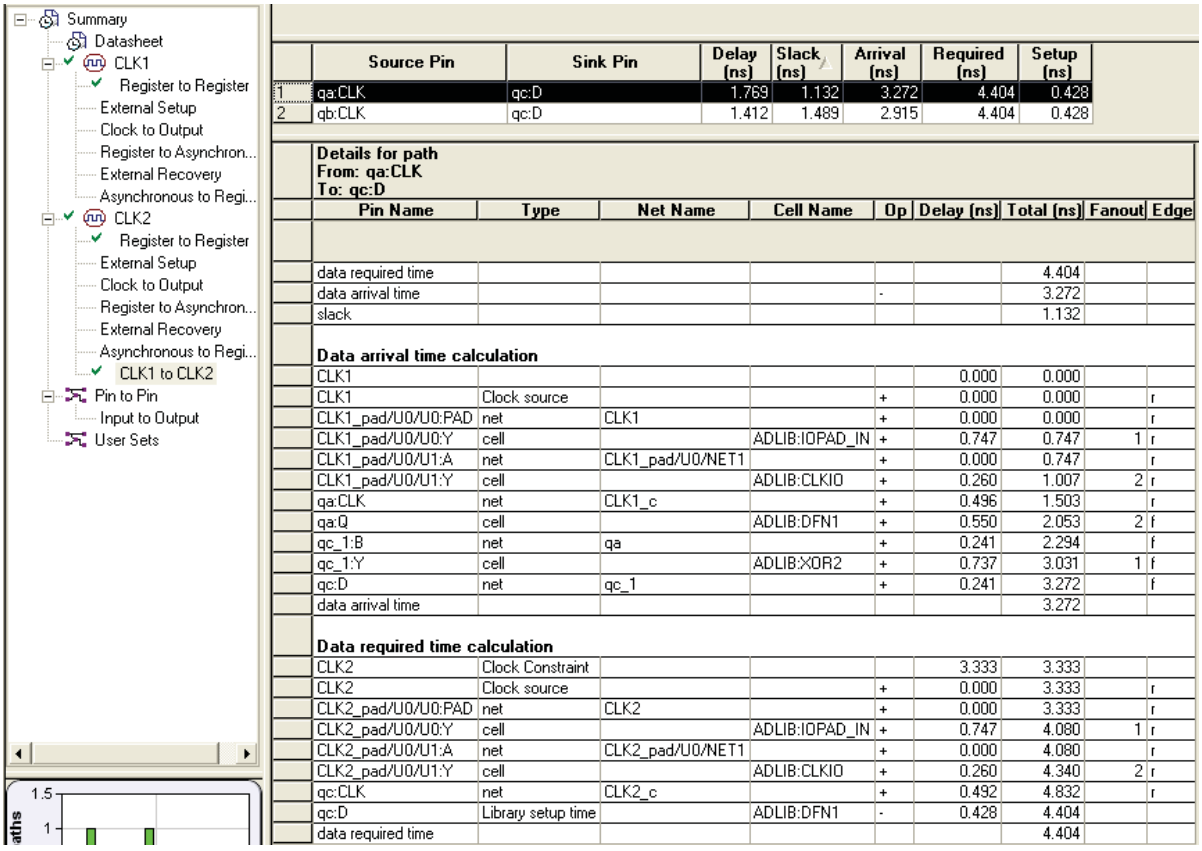


***Figure 13* • Inter-Clock Domain Timing Analysis in SmartTime Maximum Delay Analysis View**

The clock edges and data path (longest data path is from qa to qc register) used in the setup calculation are shown in Figure 14.

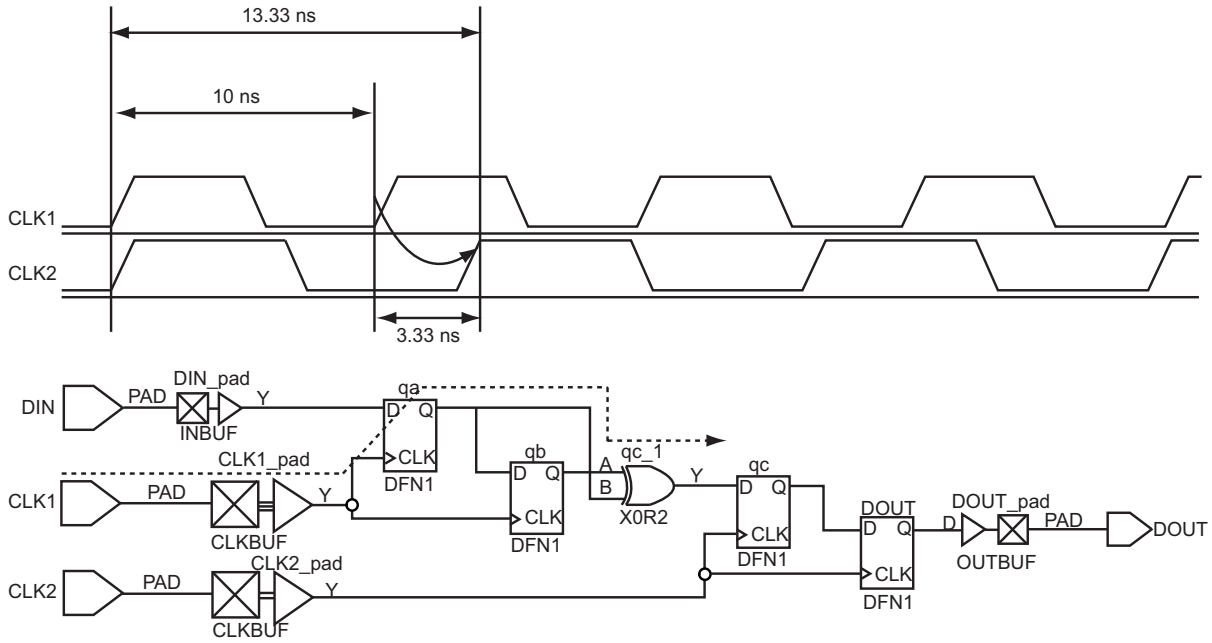
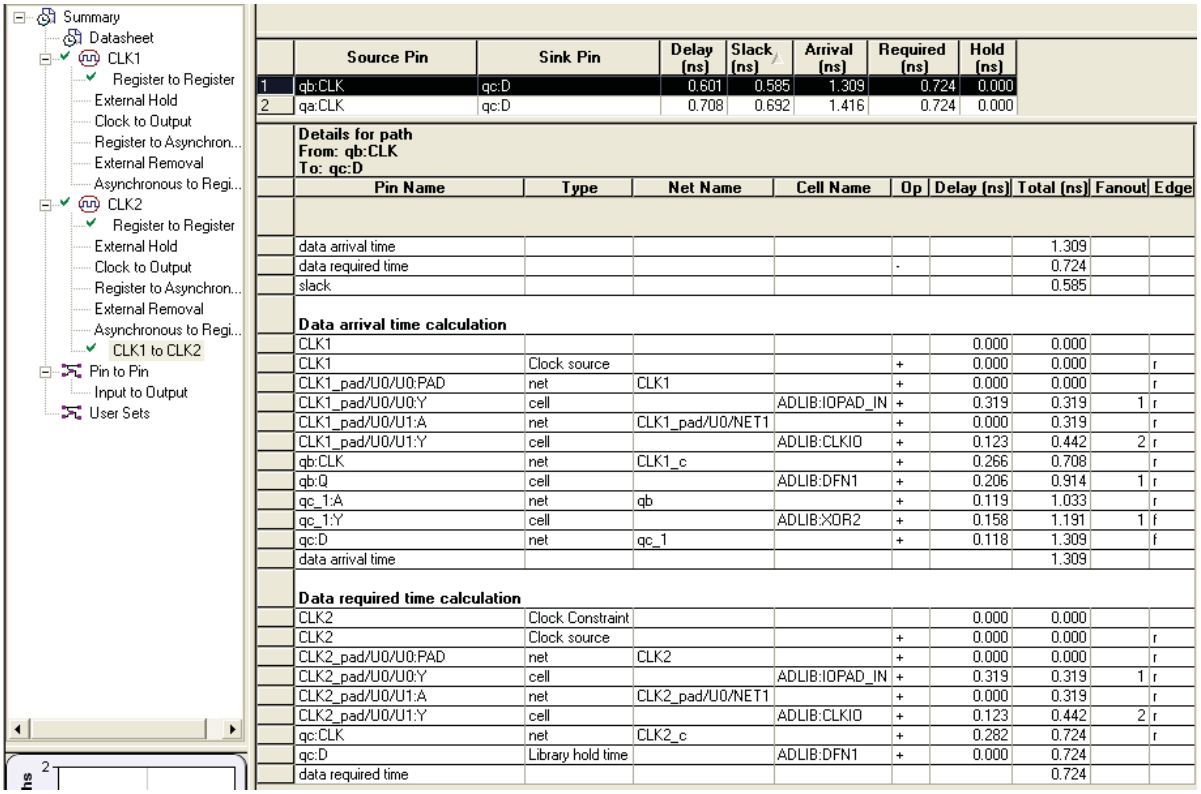


***Figure 14* • Clock Edges and Data Path Used in Intra-Clock Domain Setup Calculation**

Similarly, the minimum delay analysis view displays the hold analysis from CLK1 to CLK2.

The hold check from the CLK1 to CLK2 domain is shown in Figure 15.



	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Hold (ns)
1	qb:CLK	qc:D	0.601	0.585	1.309	0.724	0.000
2	qa:CLK	qc:D	0.708	0.692	1.416	0.724	0.000

Details for path
From: qb:CLK
To: qc:D

Pin Name	Type	Net Name	Cell Name	Op	Delay (ns)	Total (ns)	Fanout	Edge
data arrival time						1.309		
data required time				-		0.724		
slack						0.585		
Data arrival time calculation								
CLK1					0.000	0.000		
CLK1	Clock source			+	0.000	0.000		r
CLK1_pad/U0/U0:PAD	net	CLK1		+	0.000	0.000		r
CLK1_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	0.319	0.319	1	r
CLK1_pad/U0/U1:A	net	CLK1_pad/U0/NET1		+	0.000	0.319		r
CLK1_pad/U0/U1:Y	cell		ADLIB:CLKIO	+	0.123	0.442	2	r
qb:CLK	net	CLK1_c		+	0.266	0.708		r
qb:Q	cell		ADLIB:DFN1	+	0.206	0.914	1	r
qc_1:A	net	qb		+	0.119	1.033		r
qc_1:Y	cell		ADLIB:XOR2	+	0.158	1.191	1	f
qc:D	net	qc_1		+	0.118	1.309		f
data arrival time						1.309		
Data required time calculation								
CLK2	Clock Constraint				0.000	0.000		
CLK2	Clock source			+	0.000	0.000		r
CLK2_pad/U0/U0:PAD	net	CLK2		+	0.000	0.000		r
CLK2_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	0.319	0.319	1	r
CLK2_pad/U0/U1:A	net	CLK2_pad/U0/NET1		+	0.000	0.319		r
CLK2_pad/U0/U1:Y	cell		ADLIB:CLKIO	+	0.123	0.442	2	r
qc:CLK	net	CLK2_c		+	0.282	0.724		r
qc:D	Library hold time		ADLIB:DFN1	+	0.000	0.724		
data required time						0.724		

***Figure 15* • Inter-Clock Domain Timing Analysis in Minimum Delay Analysis View**

The clock edges and data path (shortest data path is from qb to qc register) used in the hold calculation are shown in Figure 16.

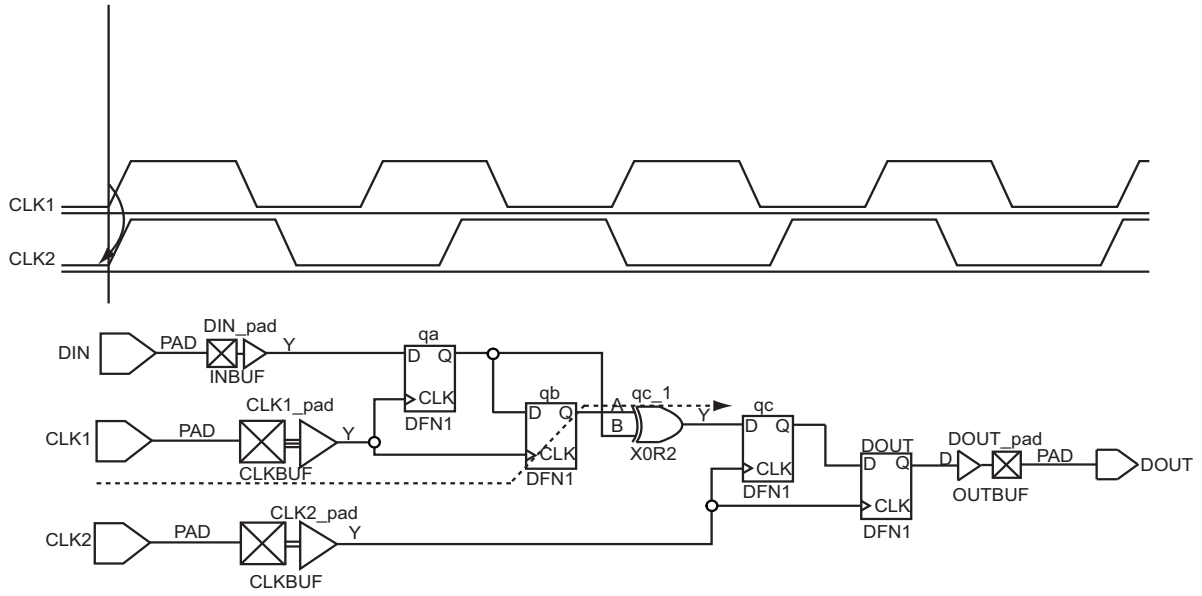


***Figure 16* • Data Path for Hold Check**

Inter-Clock Domain Analysis for Generated Clocks

Designs with internally generated clocks can also have a cross-clock domain path. SmartTime enables you to specify the generated clock constraint for the internally generated clocks and then apply cross-clock domain analysis. As mentioned in the previous section, it is the designer's responsibility to identify a path as valid or false and apply timing exceptions as required.

Consider the design example shown in Figure 17, where CLKA is a reference clock and DFN1_0:CLK is generated via clock divider. There is a path where data is launched from CLKA to be captured in the DFN1_0:CLK domain. The "Constraints Using GUI" section shows how to analyze the cross-clock domain between the main clock and internally generated clock.

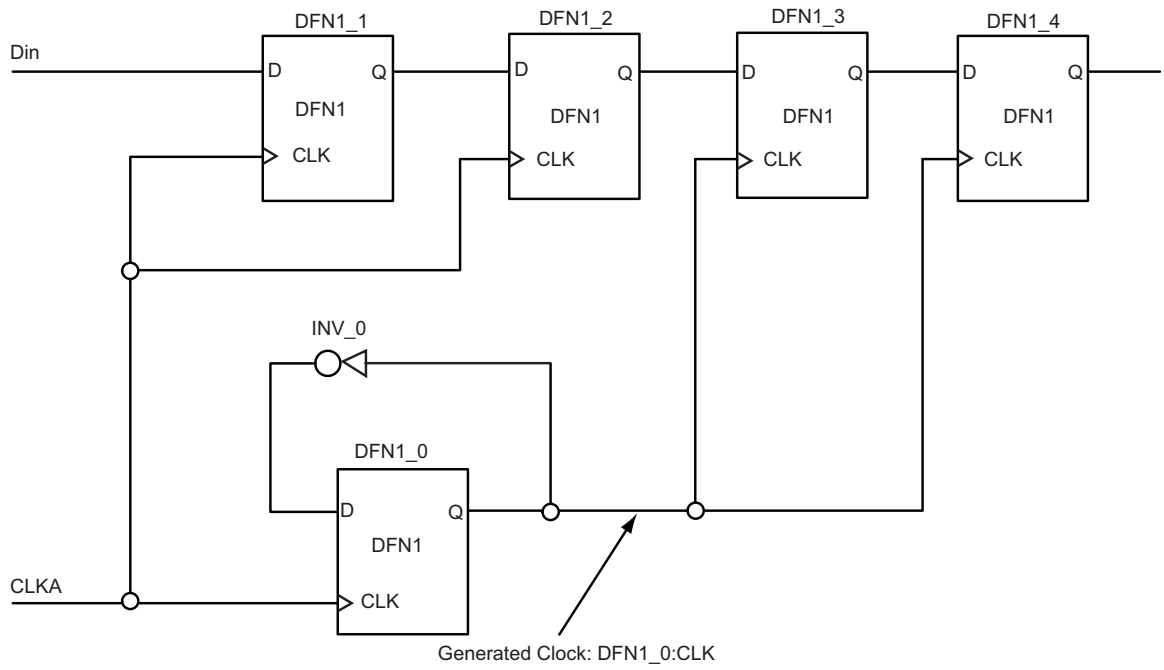
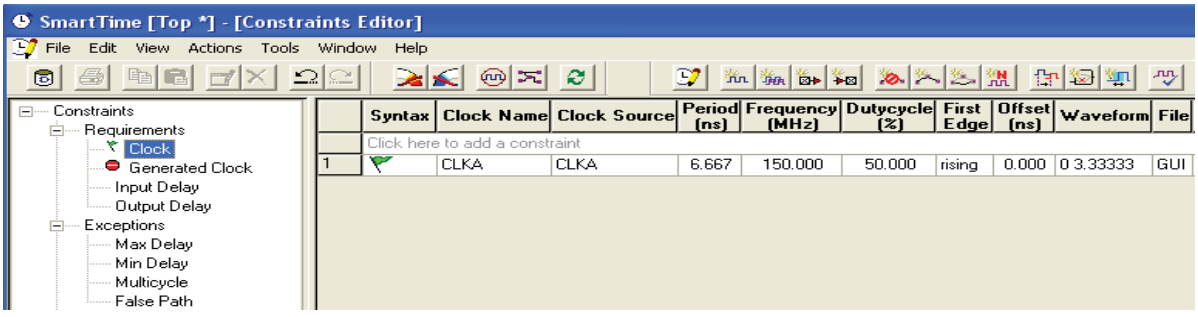


***Figure 17* • Design Example for Inter-Clock Domain Analysis Using Generated Clocks**

Constraints Using GUI

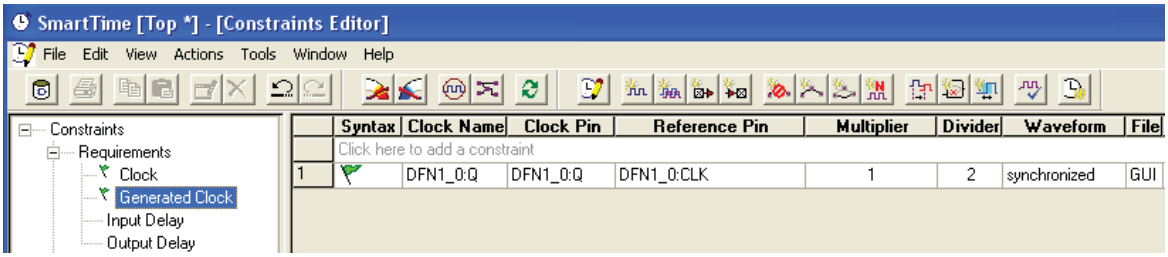
1. Specify the reference clock frequency and other attributes. Refer to "Appendix A: Applying a Clock Constraint" on page 35 for creating a generated clock constraint using the GUI.




```
create_clock -name{CLKA}-period 6.667 -waveform{0.000 3.333}{CLKA}
```

***Figure 18* • Clock Constraint Using Constraints Editor and SDC**

2. Identify the generated clock and then apply the generated clock constraint. Refer to "Appendix B: Applying a Generated Clock Constraint" on page 37 for creating a generated clock constraint using the GUI.

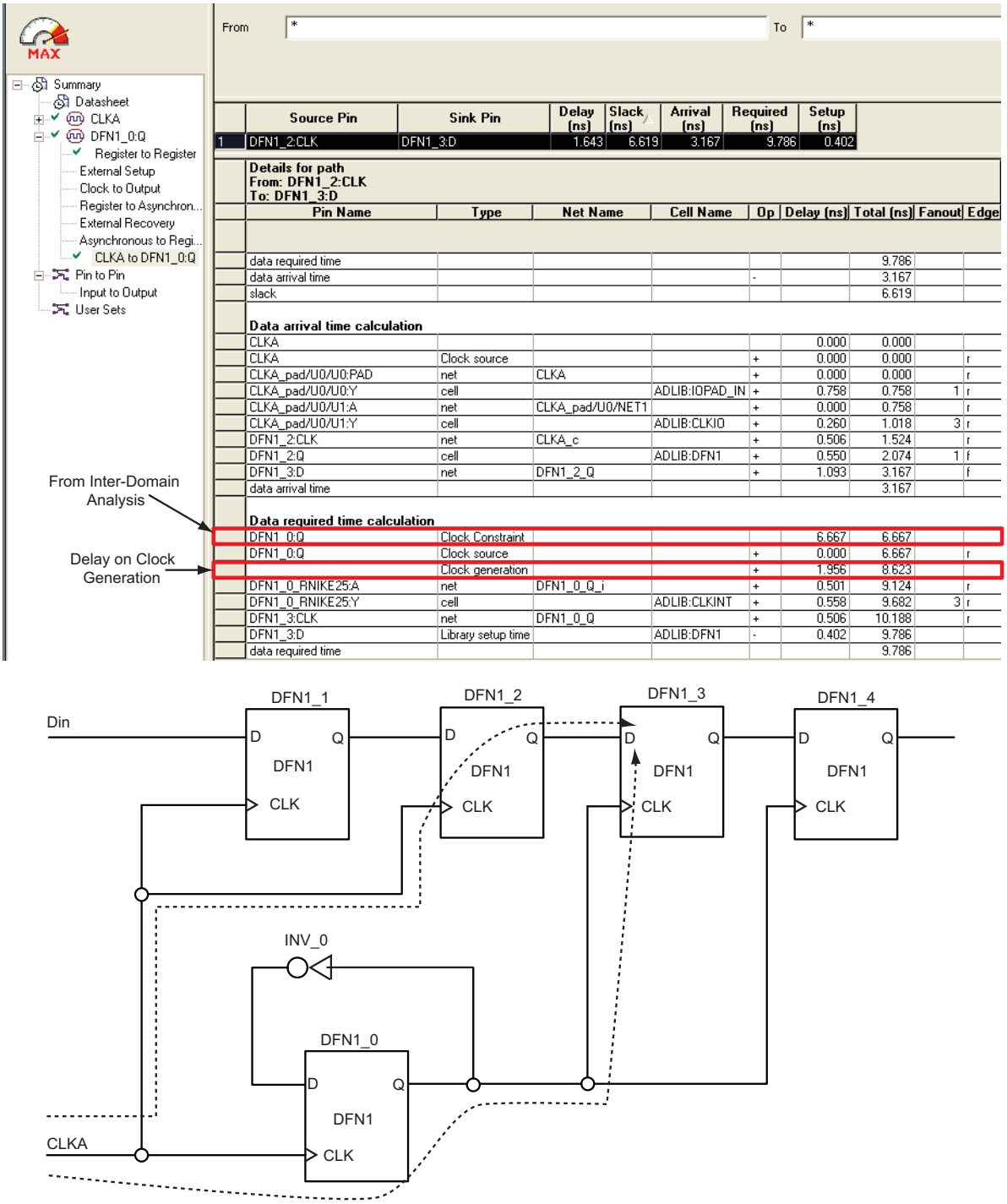



```
create_generated_clock -name{DFN1_0:Q}-divide_by2 -source{DFN1_0:CLK}
{DFN1_0:Q}
```

Figure 19 • **Generated Clock Constraint**

3. Enable inter-clock domain analysis. Refer to "Appendix C: Enabling Inter-Clock Domains Analysis" on page 40.

The maximum delay analysis view displays the timing analysis from CLKA to DFN1_0:Q under DFN1_0:Q domain analysis, as shown in Figure 20 on page 14. SmartTime calculates clock generation delays and clock constraints using clock edges between the clocks automatically.



	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)
1	DFN1_2:CLK	DFN1_3:D	1.643	6.619	3.167	9.786	0.402

Details for path
From: DFN1_2:CLK
To: DFN1_3:D

Pin Name	Type	Net Name	Cell Name	Op	Delay (ns)	Total (ns)	Fanout	Edge
data required time						9.786		
data arrival time				-		3.167		
slack						6.619		
Data arrival time calculation								
CLKA					0.000	0.000		
CLKA	Clock source			+	0.000	0.000		r
CLKA_pad/U0/U0:PAD	net	CLKA		+	0.000	0.000		r
CLKA_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	0.758	0.758	1	r
CLKA_pad/U0/U1:A	net	CLKA_pad/U0/NET1		+	0.000	0.758		r
CLKA_pad/U0/U1:Y	cell		ADLIB:CLKIO	+	0.260	1.018	3	r
DFN1_2:CLK	net	CLKA_c		+	0.506	1.524		r
DFN1_2:Q	cell		ADLIB:DFN1	+	0.550	2.074	1	f
DFN1_3:D	net	DFN1_2_Q		+	1.093	3.167		f
data arrival time						3.167		
Data required time calculation								
DFN1_0:Q	Clock Constraint				6.667	6.667		
DFN1_0:Q	Clock source			+	0.000	6.667		r
	Clock generation			+	1.956	8.623		
DFN1_0_RNIKE25:A	net	DFN1_0_Q_i		+	0.501	9.124		r
DFN1_0_RNIKE25:Y	cell		ADLIB:CLKINT	+	0.558	9.682	3	r
DFN1_3:CLK	net	DFN1_0_Q		+	0.506	10.188		r
DFN1_3:D	Library setup time		ADLIB:DFN1	-	0.402	9.786		
data required time						9.786		

Figure 20 • **Setup Check for Inter-Clock Domain Clock Using Generated Clock**

Analyzing Source Synchronization

This section describes the techniques for constraining and analyzing source-synchronization. Source-synchronous clocking refers to the technique of sourcing a clock along with the data. The timing of unidirectional data signals is referred to a clock sourced by the same device that generates the signals. Constraining source-synchronous interfaces can be complex. In addition to using the reference clock constraint, you need to constrain the source synchronous outputs by specifying the output delay relative to the reference clock.

Refer to the *Source-Synchronous Clock Designs: Timing Constraints and Analysis* application note to understand source-synchronous clock design timing constraints and analysis in detail.

Analyzing Design with Jitter/Clock Uncertainty in SmartTime

SmartTime uses the relationship between launched clock edge and captured clock edge during inter-clock domain timing analysis. However, the non-idealities of the clock generation and clock distribution system, also called jitter, manifest themselves as uncertainties of the clock edge arrivals. The clock-to-clock uncertainty constraint in SmartTime enables you to specify these uncertainties between different clocks. Clock-to-clock uncertainty defines the timing uncertainty between two clock waveforms or maximum clock skew.

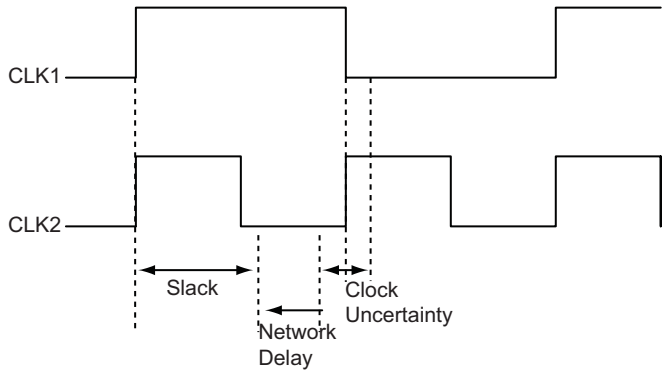


***Figure 21* • Clock-to-Clock Uncertainty**

A design example with two external clocks, CLK1 and CLK2, is shown in Figure 22. Assume that these two clocks have a tracking jitter of 2 ns. During timing analysis, this tracking jitter can be added as clock_uncertainty.

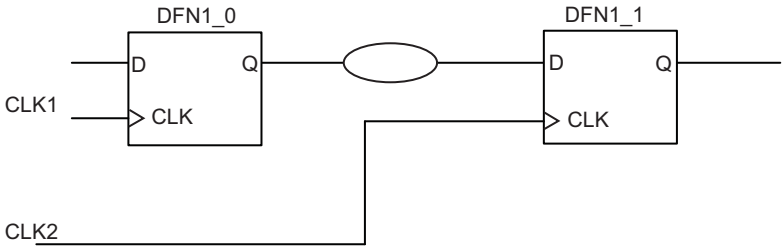
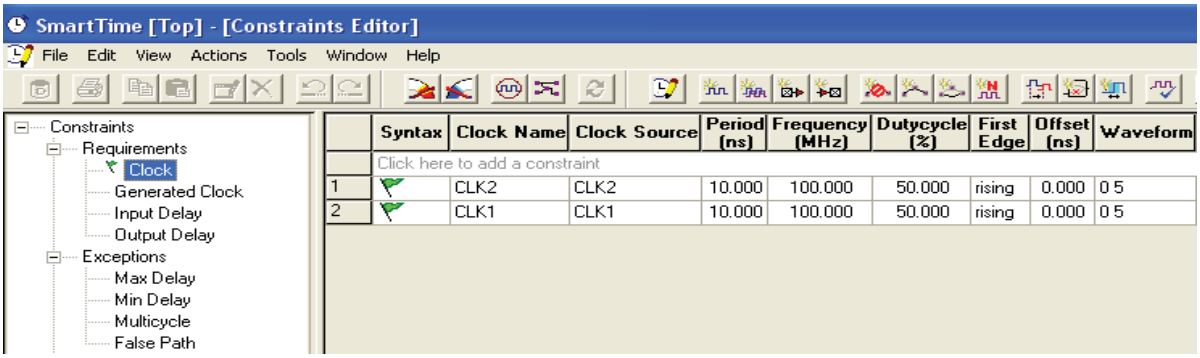


***Figure 22* • Example of Inter-Clock Uncertainty for Rise-Rise Setup Check**

Analyzing a Design with Clock Uncertainty

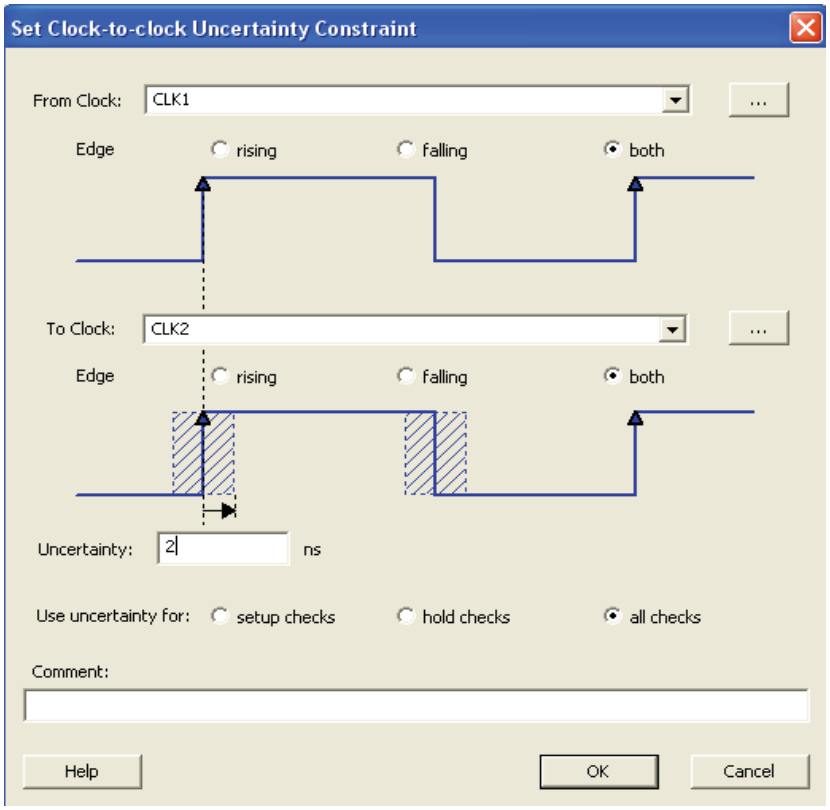
1. Specify the clock frequency and other attributes for both reference clocks. Refer to "Appendix A: Applying a Clock Constraint" on page 35 for creating a generated clock constraint using the GUI.




```
create_clock -name{CLK2}-period 10.000-waveform{0.000 5.000}{CLK2}
create_clock -name{CLK1}-period 10.000-waveform{0.000 5.000}{CLK1}
```

***Figure 23* • Clock Constraint Using Constraints Editor and SDC**

2. Add the clock-to-clock uncertainty constraint by clicking the button on the toolbar and applying 2 ns of clock uncertainty between CLK1 and CLK2.

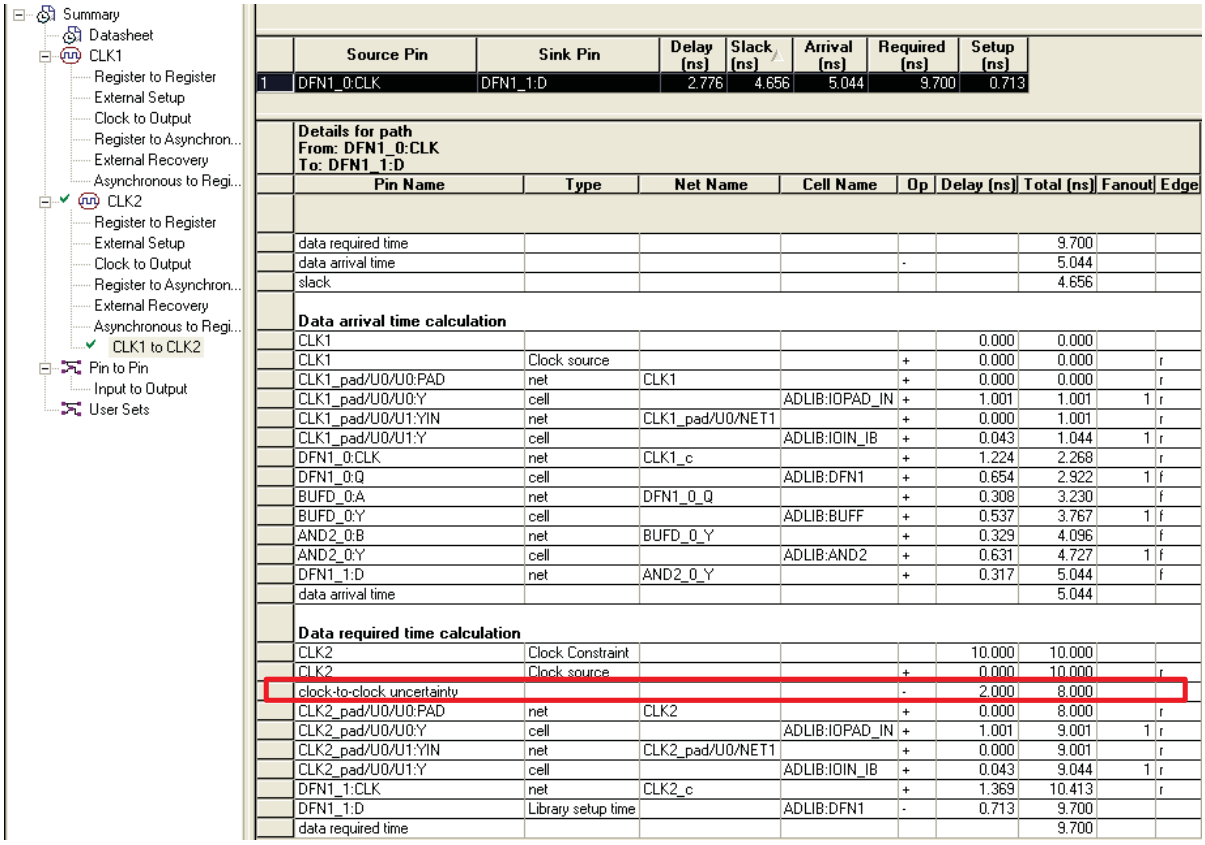



```
set_clock_uncertainty1-from{CLK1}-to{CLK2}
```

***Figure 24* • Applying Clock-to-Clock Uncertainty Constraint**

3. Enable inter-clock domain analysis. Refer to "Appendix C: Enabling Inter-Clock Domains Analysis" on page 40.

SmartTime timing analysis view uses the clock-to-clock uncertainty constraint for timing checks. Figure 25 shows how the clock-to-clock uncertainty constraint is used in a setup check.



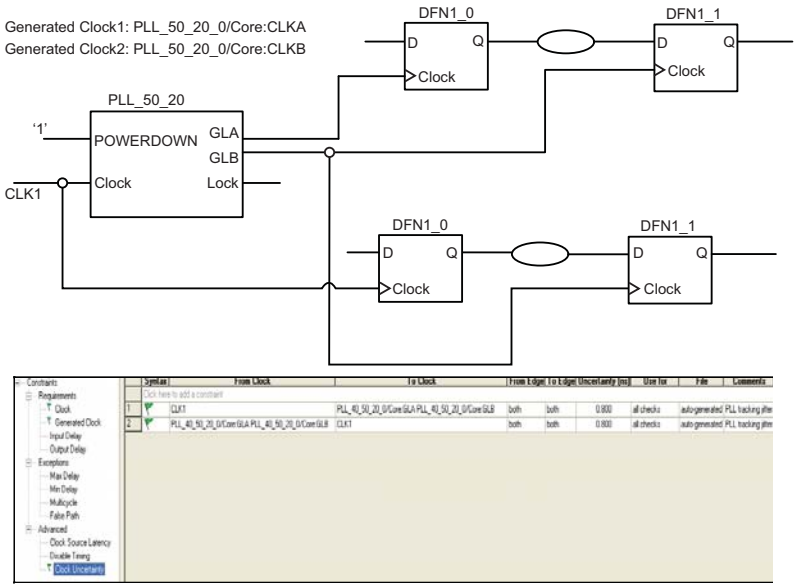
	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)
1	DFN1_0:CLK	DFN1_1:D	2.776	4.656	5.044	9.700	0.713

Details for path
From: DFN1_0:CLK
To: DFN1_1:D

Pin Name	Type	Net Name	Cell Name	Op	Delay (ns)	Total (ns)	Fanout	Edge
data required time						9.700		
data arrival time				-		5.044		
slack						4.656		
Data arrival time calculation								
CLK1					0.000	0.000		
CLK1	Clock source			+	0.000	0.000		r
CLK1_pad/U0/U0:PAD	net	CLK1		+	0.000	0.000		r
CLK1_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	1.001	1.001	1	r
CLK1_pad/U0/U1:YIN	net	CLK1_pad/U0/NET1		+	0.000	1.001		r
CLK1_pad/U0/U1:Y	cell		ADLIB:IOIN_IB	+	0.043	1.044	1	r
DFN1_0:CLK	net	CLK1_c		+	1.224	2.268		r
DFN1_0:Q	cell		ADLIB:DFN1	+	0.654	2.922	1	f
BUFD_0:A	net	DFN1_0_Q		+	0.308	3.230		f
BUFD_0:Y	cell		ADLIB:BUFF	+	0.537	3.767	1	f
AND2_0:B	net	BUFD_0_Y		+	0.329	4.096		f
AND2_0:Y	cell		ADLIB:AND2	+	0.631	4.727	1	f
DFN1_1:D	net	AND2_0_Y		+	0.317	5.044		f
data arrival time						5.044		
Data required time calculation								
CLK2	Clock Constraint				10.000	10.000		
CLK2	Clock source			+	0.000	10.000		r
clock-to-clock uncertainty				-	2.000	8.000		
CLK2_pad/U0/U0:PAD	net	CLK2		+	0.000	8.000		r
CLK2_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	1.001	9.001	1	r
CLK2_pad/U0/U1:YIN	net	CLK2_pad/U0/NET1		+	0.000	9.001		r
CLK2_pad/U0/U1:Y	cell		ADLIB:IOIN_IB	+	0.043	9.044	1	r
DFN1_1:CLK	net	CLK2_c		+	1.369	10.413		r
DFN1_1:D	Library setup time		ADLIB:DFN1	-	0.713	9.700		
data required time						9.700		

***Figure 25* • Setup and Hold Check**

Note: If you use a PLL in the design, SmartTime automatically adds the clock uncertainty between the PLL reference clock and the PLL output clock. However, SmartTime will not add clock uncertainty between the output clocks (the output clocks are generated from the same VCO clock). Figure 26 on page 18 shows a design example with data paths from the PLL reference clock and PLL output clock and also on the PLL output clocks. When adding the reference clock constraint, the generated clock and clock uncertainty constraint are added by the tool automatically.




```
create_clock -name{CLK1}-period 25.000-waveform{0.000 12.500}{CLK1}

create_generated_clock -name{PLL_40_50_20_0/Core:GLA}-divide_by40 -
multiply_by20 -source{PLL_40_50_20_0/Core:CLKA}{PLL_40_50_20_0/Core:GLA}

create_generated_clock -name{PLL_40_50_20_0/Core:GLB}-divide_by16 -
multiply_by20 -source{PLL_40_50_20_0/Core:CLKA}{PLL_40_50_20_0/Core:GLB}
```

***Figure 26* • Design Example and Constraint for Clock-to-Clock Uncertainty Using a PLL Design**

Analyzing a Multicycle Path with Single Clock Domain

Multicycle paths are data paths that may need more than one clock cycle to latch data at the captured register. The multicycle path constraint enables you to move the captured clock edge forward or the launched clock edge backward. When a multicycle constraint is applied to setup, it modifies the setup relationship by moving the captured (destination) clock edge to the right.

Similarly, when a multicycle constraint is applied to hold, it modifies the hold relationship, changing the launched (source) clock edge to the left. Applying the multicycle path constraint requires design knowledge.

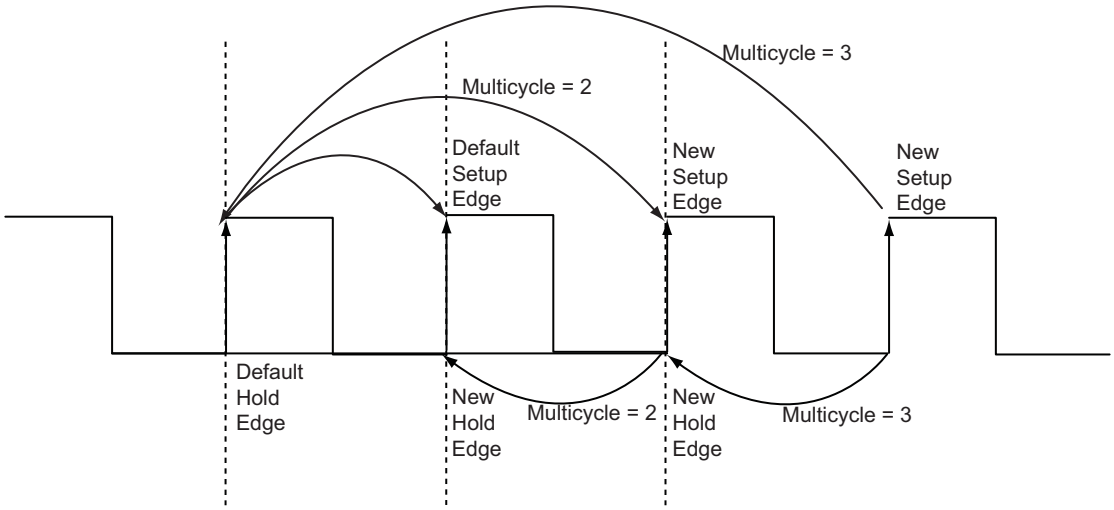


***Figure 27* • Setup and Hold Check on Multicycle Path**

Figure 28 shows a design example where you assume that the path from DFN1_1 to DFN1_2 is a multicycle path. The "Analyzing a Multicycle Path with SmartTime" section on page 20 shows how to apply and analyze the multicycle constraint on this design.

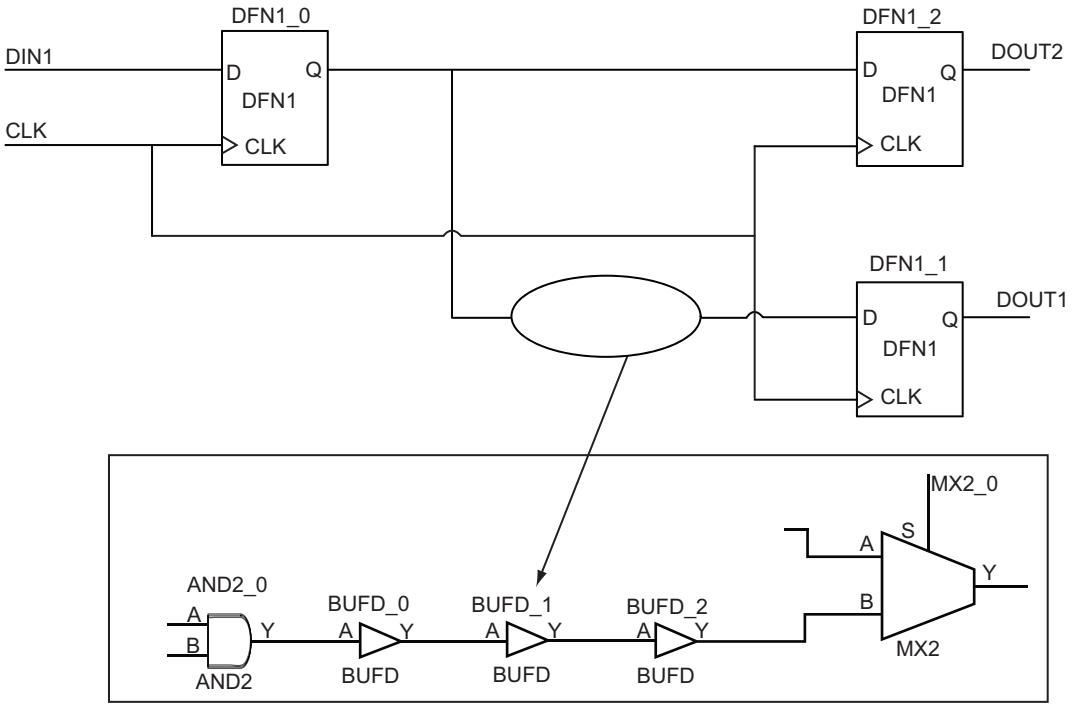
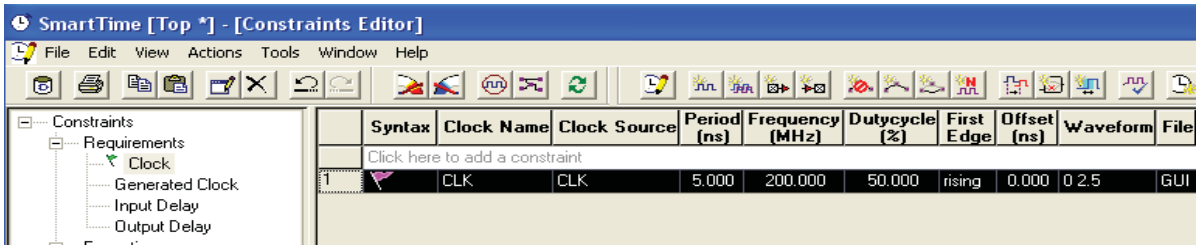


***Figure 28* • Design with Multicycle Path**

Analyzing a Multicycle Path with SmartTime

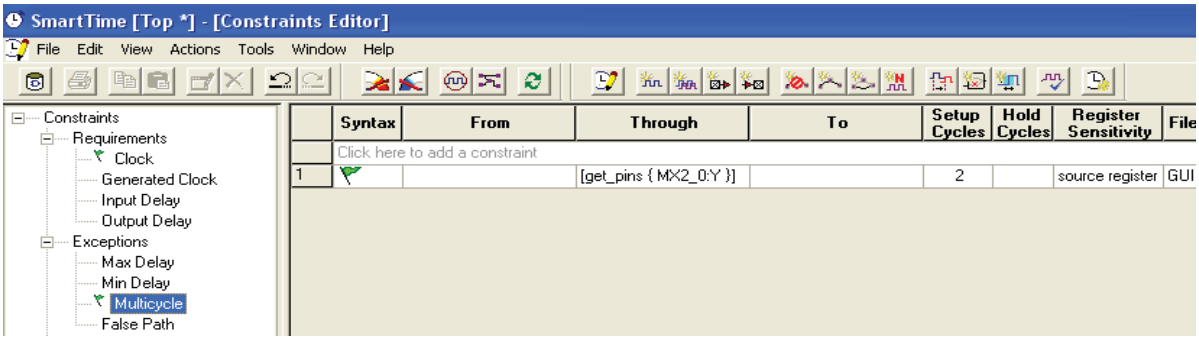
1. Specify the clock frequency and other attributes for the reference clock. Refer to "Appendix A: Applying a Clock Constraint" on page 35 for creating a generated clock constraint using the GUI.




```
create_clock -name{CLK}-period 5.000-waveform{0.000 2.500}{CLK}
```

***Figure 29* • Clock Constraint Using Constraints Editor and SDC**

2. Identify a through pin for Multicycle and apply a multicycle constraint. Refer to "Appendix D: Applying a Multicycle Clock Constraint" on page 41 for creating a multicycle clock constraint using the GUI.




```
set_multicycle_path-setup2 -through [get_pins{MX2_0:Y}]
```

***Figure 30* • Multicycle Path Constraints in Constraint Editor**

SmartTime timing analysis view uses the above multicycle constraint for a timing check. Figure 31 shows how the multicycle constraint is used in a setup check.

Details for path
From: DFN1_0:CLK
To: DFN1_1:D

Pin Name	Type	Net Name	Cell Name	Op	Delay (ns)	Total (ns)	Fanout	Edge
data arrival time				-		7.304		
slack						4.194		
Data arrival time calculation								
CLK					0.000	0.000		
CLK	Clock source			+	0.000	0.000		r
CLK_pad/U0/U0:PAD	net	CLK		+	0.000	0.000		r
CLK_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	1.016	1.016	1	r
CLK_pad/U0/U1:A	net	CLK_pad/U0/NET1		+	0.000	1.016		r
CLK_pad/U0/U1:Y	cell		ADLIB:CLKIO	+	0.348	1.364	3	r
DFN1_0:CLK	net	CLK_c		+	0.678	2.042		r
DFN1_0:Q	cell		ADLIB:DFN1	+	0.737	2.779	2	f
AND2_0:A	net	DFN1_0_Q		+	0.308	3.087		f
AND2_0:Y	cell		ADLIB:AND2	+	0.386	3.473	1	f
BUFD_0:A	net	AND2_0_Y		+	0.308	3.781		f
BUFD_0:Y	cell		ADLIB:BUFF	+	0.537	4.318	1	f
BUFD_1:A	net	BUFD_0_Y		+	0.308	4.626		f
BUFD_1:Y	cell		ADLIB:BUFF	+	0.537	5.163	1	f
BUFD_2:A	net	BUFD_1_Y		+	0.308	5.471		f
BUFD_2:Y	cell		ADLIB:BUFF	+	0.537	6.008	1	f
MX2_0:B	net	BUFD_2_Y		+	0.323	6.331		f
MX2_0:Y	cell		ADLIB:OR2A	+	0.650	6.981	1	f
DFN1_1:D	net	MX2_0_Y		+	0.323	7.304		f
data arrival time						7.304		
Data required time calculation								
CLK	Multicycle Constraint				10.000	10.000		
CLK	Clock source			+	0.000	10.000		r
CLK_pad/U0/U0:PAD	net	CLK		+	0.000	10.000		r
CLK_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	1.016	11.016	1	r
CLK_pad/U0/U1:A	net	CLK_pad/U0/NET1		+	0.000	11.016		r
CLK_pad/U0/U1:Y	cell		ADLIB:CLKIO	+	0.348	11.364	3	r
DFN1_1:CLK	net	CLK_c		+	0.673	12.037		r
DFN1_1:D	Library setup time		ADLIB:DFN1	-	0.539	11.498		
data required time						11.498		

***Figure 31* • Setup for Multicycle Path**

Analyzing a Multicycle Path with Inter-Clock Domain

The analysis of a multicycle path in a cross-clock domain is complex. If the captured clock is generated from the launched (source) clock and also runs slower than the launched clock, then moving the launched clock one cycle forward is not equal to moving the end clock one cycle backward. The different options give totally different timing windows. So you need to be careful when applying multicycle setup and multicycle hold for this condition.

A design example with a reference clock (CLK) and generated clock (DFN1_3:CLK) is shown in Figure 32 on page 22. Assume that the path through the AND gate is a multicycle path and the designer wants to apply a multicycle constraint for this path.

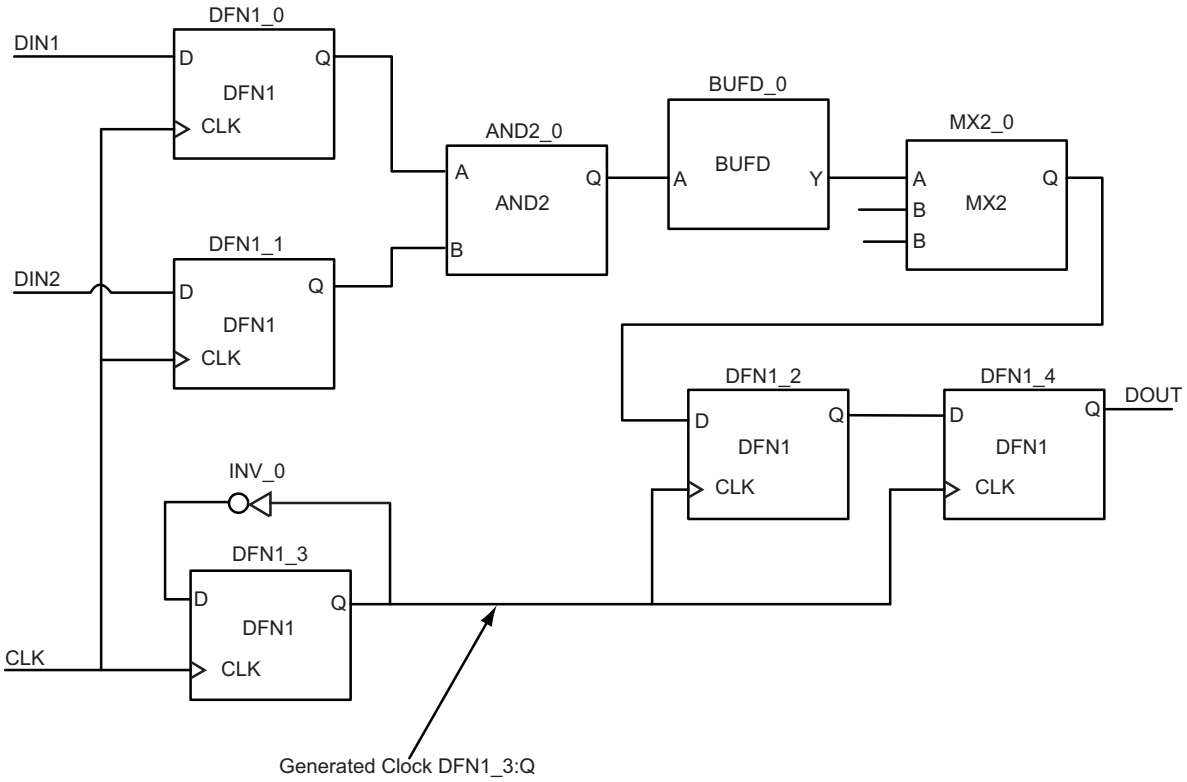


***Figure 32* • Design for Multicycle Path with Inter-Clock Domain**

The setup and hold analysis under various conditions are shown in Figure 33. Due to the offset between CLK and DFN1_3:Q, SmartTime by default uses setup check 1 (SC1) for the setup check. However, you should use setup check 2 (SC2) for setup checks. For hold check, SmartTime uses hold check 2 (HC2) by default. If you use the wrong edge, hold check 1 (HC1), you may see a timing violation.

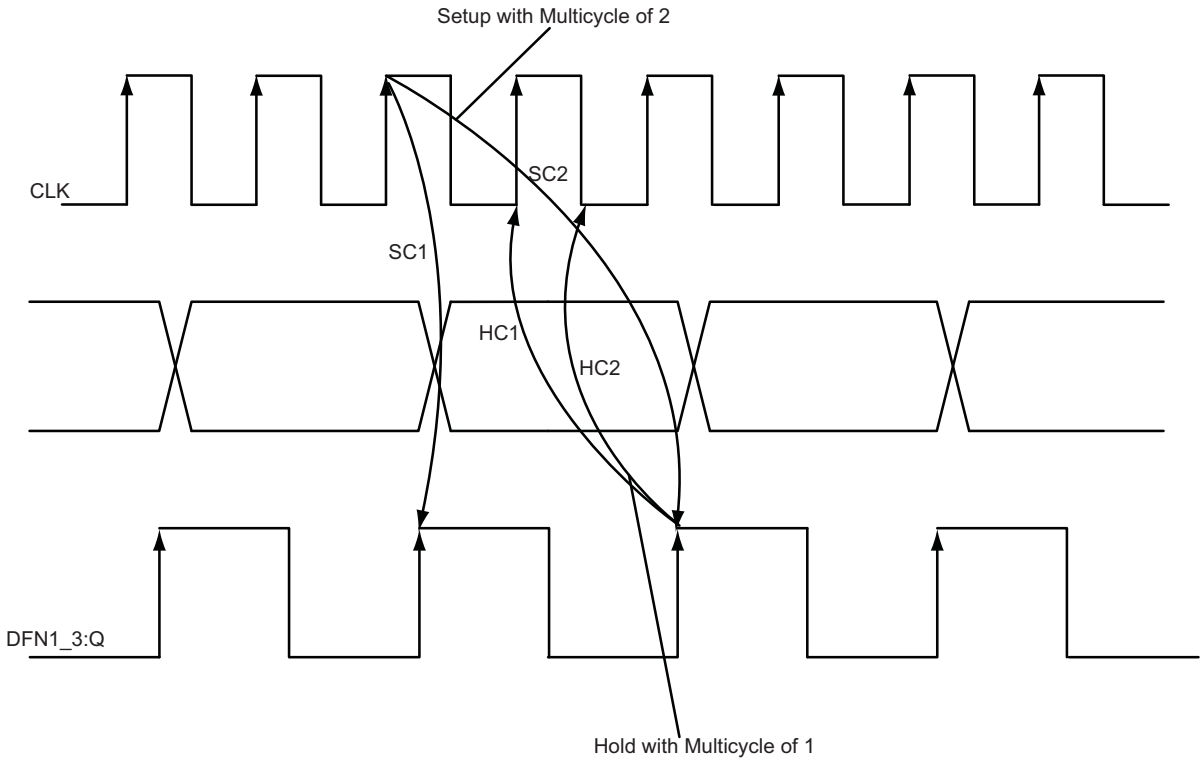
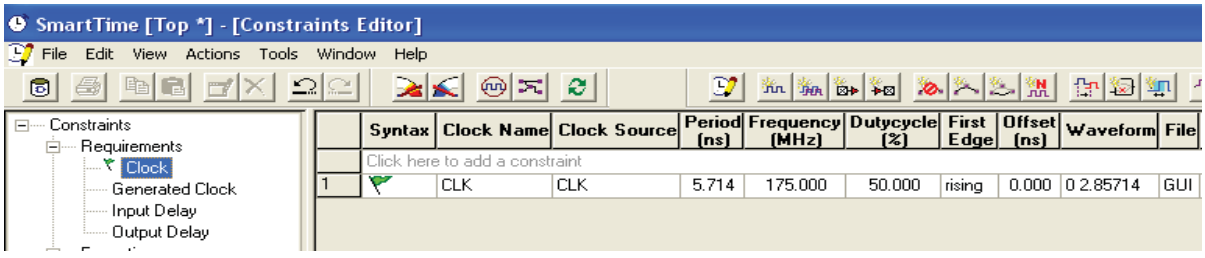


***Figure 33* • Launched and Captured Edges During Multicycle Analysis**

Analyzing a Multicycle Path on a Generated Clock with SmartTime

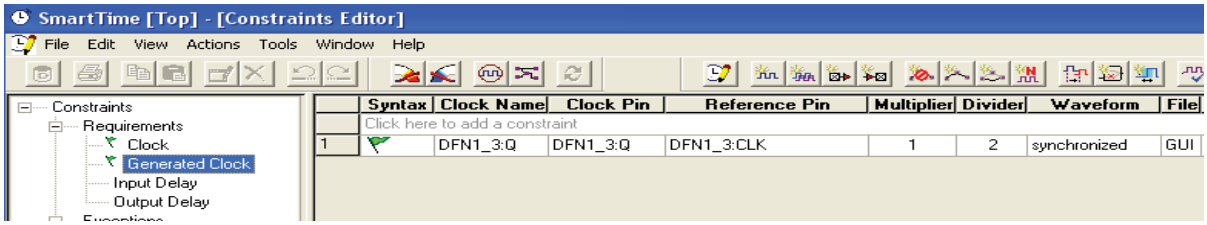
1. Specify the reference clock frequency and other attributes. Refer to "Appendix A: Applying a Clock Constraint" on page 35 for creating a generated clock constraint using the GUI.




```
create_clock -name{CLK}-period 5.714-waveform{0.000 2.857}{CLK}
```

***Figure 34* • Clock Constraint Using Constraints Editor and SDC**

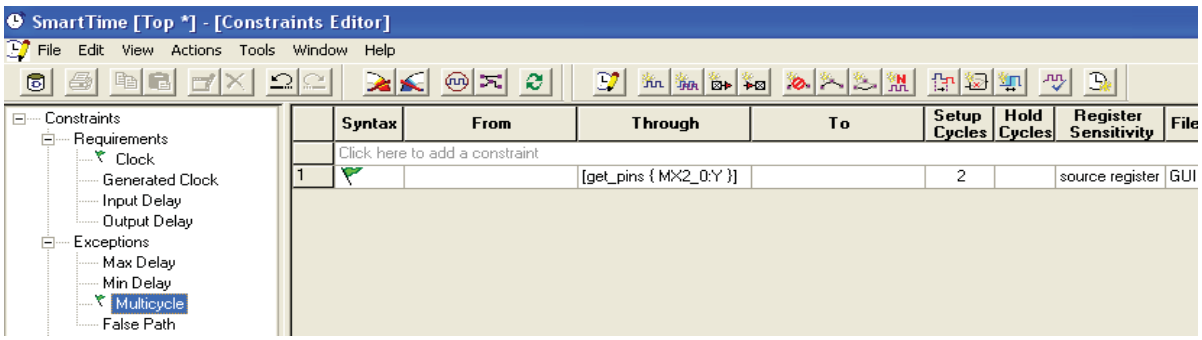
2. Identify the generated clock constraint and apply the generated clock constraint. Refer to "Appendix B: Applying a Generated Clock Constraint" on page 37 for creating a generated clock constraint using the GUI.




```
create_generated_clock -name{DFN1_3:Q}-divide_by2 -source{DFN1_3:CLK}
{DFN1_3:Q}
```

Figure 35 • **Generated Clock Constraint**

3. Enable inter-clock domain analysis. Refer to "Appendix C: Enabling Inter-Clock Domains Analysis" on page 40.
4. Identify a through pin for multicycle and apply a multicycle constraint. Refer to "Appendix D: Applying a Multicycle Clock Constraint" on page 41 for creating a generated clock constraint using the GUI. Ensure that the **Setup Check only** option is selected, since multicycle constraint is applied to setup check only in this design example. For hold check, the default edge is used.




```
set_multicycle_path-setup2-through[get_pins{MX2_0:Y}]
```

Figure 36 • **Multicycle Path Constraints in Constraint Editor**

SmartTime timing analysis view uses this multicycle path constraint for setup checks. Figure 37 shows a setup check and a hold check.

Details for path
From: DFN1_1:CLK
To: DFN1_2:D

Pin Name	Type	Net Name	Cell Name	Op	Delay (ns)	Total (ns)	Fanout	Edge
data required time						23.418		
data arrival time				-		11.894		
slack						11.524		
Data arrival time calculation								
CLK					0.000	0.000		
CLK	Clock source			+	0.000	0.000		r
CLK_pad/U0/U0:PAD	net	CLK		+	0.000	0.000		r
CLK_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	1.560	1.560	1	r
CLK_pad/U0/U1:A	net	CLK_pad/U0/NET1		+	0.000	1.560		r
CLK_pad/U0/U1:Y	cell		ADLIB:CLKIO	+	1.016	2.576	3	r
DFN1_1:CLK	net	CLK_c		+	0.784	3.360		r
DFN1_1:Q	cell		ADLIB:DFN1	+	1.399	4.759	1	f
AND2_0:B	net	DFN1_1_Q		+	0.931	5.690		f
AND2_0:Y	cell		ADLIB:AND2	+	1.517	7.207	1	f
BUFD_0:A	net	AND2_0_Y		+	1.797	9.004		f
BUFD_0:Y	cell		ADLIB:BUFF	+	1.177	10.181	1	f
MX2_0:A	net	BUFD_0_Y		+	0.286	10.467		f
MX2_0:Y	cell		ADLIB:MX2	+	1.141	11.608	1	f
DFN1_2:D	net	MX2_0_Y		+	0.286	11.894		f
data arrival time						11.894		
Data required time calculation								
DFN1_3:Q	Multicyle Constraint				17.142	17.142		
DFN1_3:Q	Clock source			+	0.000	17.142		r
	Clock generation			+	4.300	21.442		
DFN1_3_RNINE25:A	net	DFN1_3_Q_i		+	0.365	21.807		r
DFN1_3_RNINE25:Y	cell		ADLIB:CLKINT	+	1.982	23.789	3	r
DFN1_2:CLK	net	DFN1_3_Q		+	0.974	24.763		r
DFN1_2:D	Library setup time		ADLIB:DFN1	-	1.345	23.418		
data required time						23.418		

Details for path
From: DFN1_0:CLK
To: DFN1_2:D

Pin Name	Type	Net Name	Cell Name	Op	Delay (ns)	Total (ns)	Fanout	Edge
data arrival time						7.103		
data required time				-		5.076		
slack						2.027		
Data arrival time calculation								
CLK					0.000	0.000		
CLK	Clock source			+	0.000	0.000		r
CLK_pad/U0/U0:PAD	net	CLK		+	0.000	0.000		r
CLK_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	1.039	1.039	1	r
CLK_pad/U0/U1:A	net	CLK_pad/U0/NET1		+	0.000	1.039		r
CLK_pad/U0/U1:Y	cell		ADLIB:CLKIO	+	0.676	1.715	3	r
DFN1_0:CLK	net	CLK_c		+	0.556	2.271		r
DFN1_0:Q	cell		ADLIB:DFN1	+	0.597	2.868	1	r
AND2_0:A	net	DFN1_0_Q		+	0.245	3.113		r
AND2_0:Y	cell		ADLIB:AND2	+	0.504	3.617	1	r
BUFD_0:A	net	AND2_0_Y		+	1.416	5.033		r
BUFD_0:Y	cell		ADLIB:BUFF	+	0.605	5.638	1	r
MX2_0:A	net	BUFD_0_Y		+	0.267	5.905		r
MX2_0:Y	cell		ADLIB:MX2	+	0.933	6.838	1	r
DFN1_2:D	net	MX2_0_Y		+	0.265	7.103		r
data arrival time						7.103		
Data required time calculation								
DFN1_3:Q	Clock Constraint				0.000	0.000		
DFN1_3:Q	Clock source			+	0.000	0.000		r
	Clock generation			+	2.868	2.868		
DFN1_3_RNINE25:A	net	DFN1_3_Q_i		+	0.259	3.127		r
DFN1_3_RNINE25:Y	cell		ADLIB:CLKINT	+	1.259	4.386	3	r
DFN1_2:CLK	net	DFN1_3_Q		+	0.690	5.076		r
DFN1_2:D	Library hold time		ADLIB:DFN1	+	0.000	5.076		
data required time						5.076		

Figure 37 • **Setup and Hold Checks for Multicycle Path**

Figure 38 shows that if you use the wrong value for hold, it displays the incorrect hold violation.

Details for path
From: DFN1_0:CLK
To: DFN1_2:D

Pin Name	Type	Net Name	Cell Name	Op	Delay (ns)	Total (ns)	Fanout	Edge
data arrival time						7.103		
data required time				-		10.790		
slack						-3.687		
Data arrival time calculation								
CLK					0.000	0.000		
CLK	Clock source			+	0.000	0.000		r
CLK_pad/U0/U0:PAD	net	CLK		+	0.000	0.000		r
CLK_pad/U0/U0:Y	cell		ADLIB:IOPAD_IN	+	1.039	1.039	1	r
CLK_pad/U0/U1:A	net	CLK_pad/U0/NET1		+	0.000	1.039		r
CLK_pad/U0/U1:Y	cell		ADLIB:CLKIO	+	0.676	1.715	3	r
DFN1_0:CLK	net	CLK_c		+	0.556	2.271		r
DFN1_0:Q	cell		ADLIB:DFN1	+	0.597	2.868	1	r
AND2_0:A	net	DFN1_0_Q		+	0.245	3.113		r
AND2_0:Y	cell		ADLIB:AND2	+	0.504	3.617	1	r
BUFD_0:A	net	AND2_0_Y		+	1.416	5.033		r
BUFD_0:Y	cell		ADLIB:BUFF	+	0.605	5.638	1	r
MX2_0:A	net	BUFD_0_Y		+	0.267	5.905		r
MX2_0:Y	cell		ADLIB:MX2	+	0.933	6.838	1	r
DFN1_2:D	net	MX2_0_Y		+	0.265	7.103		r
data arrival time						7.103		
Data required time calculation								
DFN1_3:Q	Multicyle Constraint				5.714	5.714		
DFN1_3:Q	Clock source			+	0.000	5.714		r
	Clock generation			+	2.868	8.582		
DFN1_3_RNINE25:A	net	DFN1_3_Q_i		+	0.259	8.841		r
DFN1_3_RNINE25:Y	cell		ADLIB:CLKINT	+	1.259	10.100	3	r
DFN1_2:CLK	net	DFN1_3_Q		+	0.690	10.790		r
DFN1_2:D	Library hold time		ADLIB:DFN1	+	0.000	10.790		
data required time						10.790		

Figure 38 • **Wrong Clock Edge Used in Hold Check**

Analyzing Clock Gating

The gated clock signal occurs when a clock path contains logic other than inverters or buffers. The default setting in SmartTime timing analysis view enables setup and hold analysis for the reference clock. However, it does not do timing analysis on the gating cells between the gating signal and clock. It is possible for the gated signal to have transitions while clock pulses are passing through the gating cells and this can lead to both clipped and spurious clock pulses. This section provides detailed information on doing this timing analysis manually.

Figure 39 on page 27 shows the most generalized circuit for the gated clock and the timing waveform. The GATED_CLK signal propagates through AND2_0 to the downstream flip-flops only when CLK_EN is high. In order to be glitch free, the output from DFN0_0 should arrive at input B of AND2_0 after the falling edge of CLK arrives at input A of AND2_0 and before the next rising CLK arrives at input A of AND2_0. The setup check analysis should use the following timing calculation:

- Launched edge: The data path starts at CLK, goes through DFN0_0 (D->Q), and then ends at the AND2_0:B pin. Both rising edge and falling edge timing must be calculated and the larger result will be used.
- Captured edge: The clock path starts at CLK and ends at the AND2_0:A pin. The timing must be checked for rising edge only.

The hold check analysis should use the following timing:

- Launched edge: The data path starts at CLK, goes through DFN0_0 (D->Q), and ends at the AND2_0:B pin. The timing must be for the falling edge of the clock to the DFN1_0 FF and then for either rising or falling edge to the AND2_0 gate, whichever is shorter.
- Captured edge: The clock path starts at CLK and ends at the AND2_0:A pin. The timing must be for delays of the negative edge of the clock.

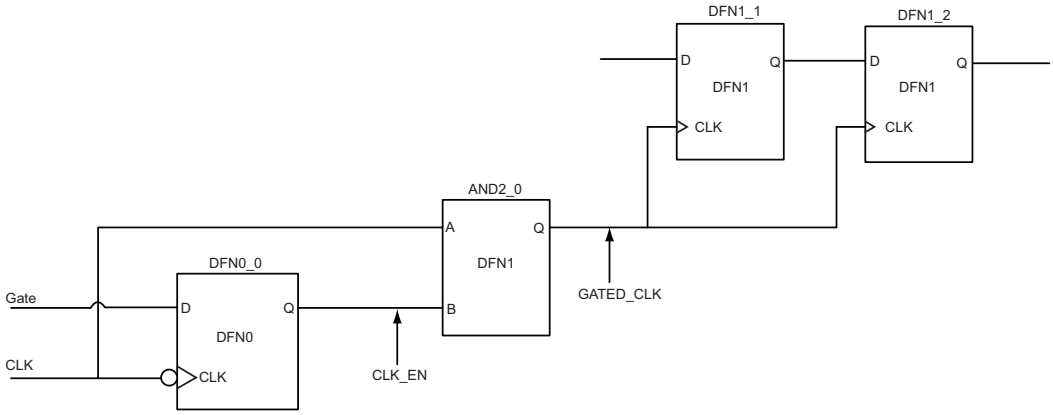


***Figure 39* • Gated Clock Example**

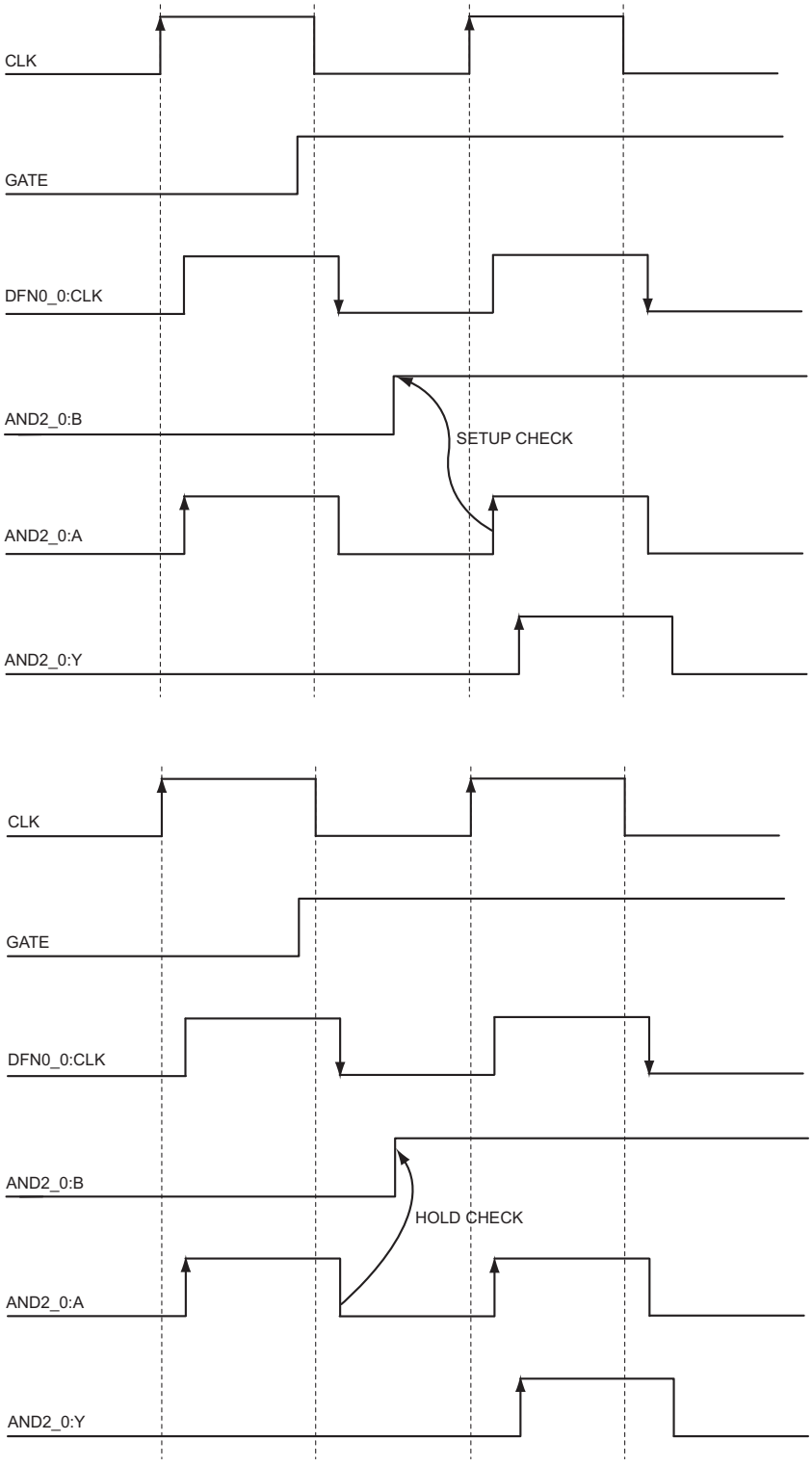
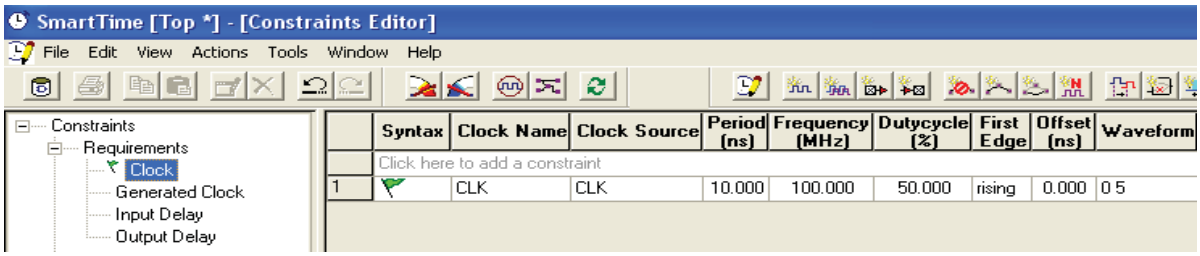


Figure 40 • **Timing Waveform**

Analyzing a Gated Clock

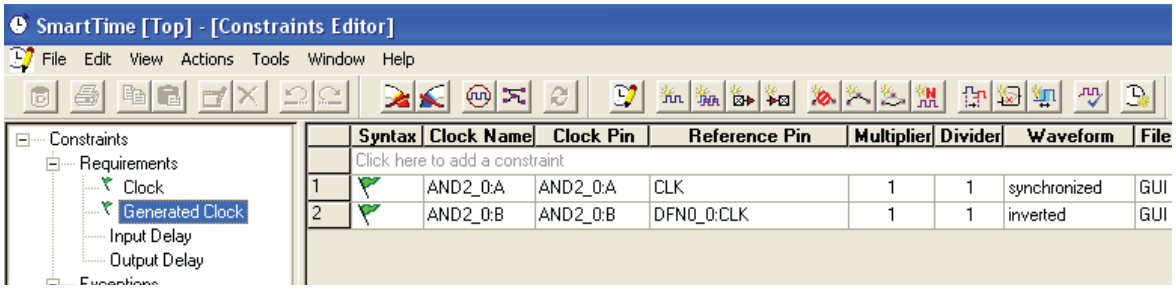
1. Specify the reference clock frequency and other attributes. Refer to "Appendix A: Applying a Clock Constraint" on page 35 for creating a generated clock constraint using the GUI.




```
create_clock -name{CLK}-period 10.000-waveform{0.000 5.000}{CLK}
```

***Figure 41* • Clock Constraint Using Constraints Editor and SDC**

2. Identify AND2_0:A and AND2_0:B as generated clocks and apply the generated clock constraint. Refer to "Appendix B: Applying a Generated Clock Constraint" on page 37 for creating a generated clock constraint using the GUI.

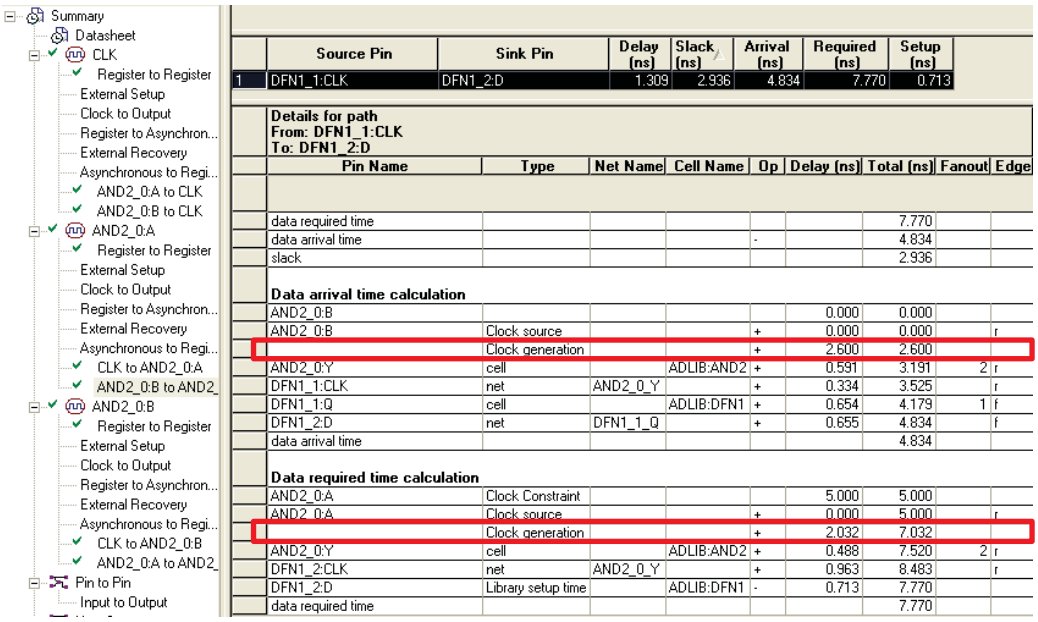



```
create_generated_clock -name{AND2_0:A}-divide_by1 -source{CLK}{AND2_0:A}
create_generated_clock -name{AND2_0:B}-divide_by1 -invert -
source{DFN0_0:CLK}{AND2_0:B}
```

***Figure 42* • Generated Clock Constraint**

3. Enable inter-clock domain analysis. Refer to "Appendix C: Enabling Inter-Clock Domains Analysis" on page 40.

SmartTime maximum delay analysis view shows the reference clock and AND2_0:A to AND2_0:B clock domain. You need to get the delay values from the expanded data path and do setup and hold calculation on the gated cell. Figure 43 shows one of the expanded paths and the setup and hold calculation on the gated cell.



	Source Pin	Sink Pin	Delay (ns)	Slack (ns)	Arrival (ns)	Required (ns)	Setup (ns)
1	DFN1_1:CLK	DFN1_2:D	1.309	2.936	4.834	7.770	0.713

Details for path
From: DFN1_1:CLK
To: DFN1_2:D

Pin Name	Type	Net Name	Cell Name	Op	Delay (ns)	Total (ns)	Fanout	Edge
data required time						7.770		
data arrival time				-		4.834		
slack						2.936		
Data arrival time calculation								
AND2_0:B					0.000	0.000		
AND2_0:B	Clock source			+	0.000	0.000		r
	Clock generation			+	2.600	2.600		
AND2_0:Y	cell		ADLIB:AND2	+	0.591	3.191	2	r
DFN1_1:CLK	net	AND2_0_Y		+	0.334	3.525		r
DFN1_1:Q	cell		ADLIB:DFN1	+	0.654	4.179	1	f
DFN1_2:D	net	DFN1_1_Q		+	0.655	4.834		f
data arrival time						4.834		
Data required time calculation								
AND2_0:A	Clock Constraint				5.000	5.000		
AND2_0:A	Clock source			+	0.000	5.000		r
	Clock generation			+	2.032	7.032		
AND2_0:Y	cell		ADLIB:AND2	+	0.488	7.520	2	r
DFN1_2:CLK	net	AND2_0_Y		+	0.963	8.483		r
DFN1_2:D	Library setup time		ADLIB:DFN1	-	0.713	7.770		
data required time						7.770		

***Figure 43* • Inter-Clock Domain AND2_0:A to AND2_0:B Path**

Setup check = Capture edge - launch edge = (10 + 2.032) - (5 + 2.6) = 4.432 ns.

Hold check = Capture edge - launch edge = (5 + 2.6) - (5 + 2.032) = 0.56 ns.

Note: You need to do similar calculations using the delay numbers under the minimum delay analysis view.

Four Corner Analysis

The delay of a path or gate depends on factors such as voltage, temperature, process, and loading. Figure 44 on page 31 shows timing delay under various conditions. In SmartTime, the default maximum delay analysis checks the setup timing under worst case scenario and the minimum delay analysis checks the hold timing under best case scenario. However, for some designs these scenarios do not always cover all the corner case scenarios.

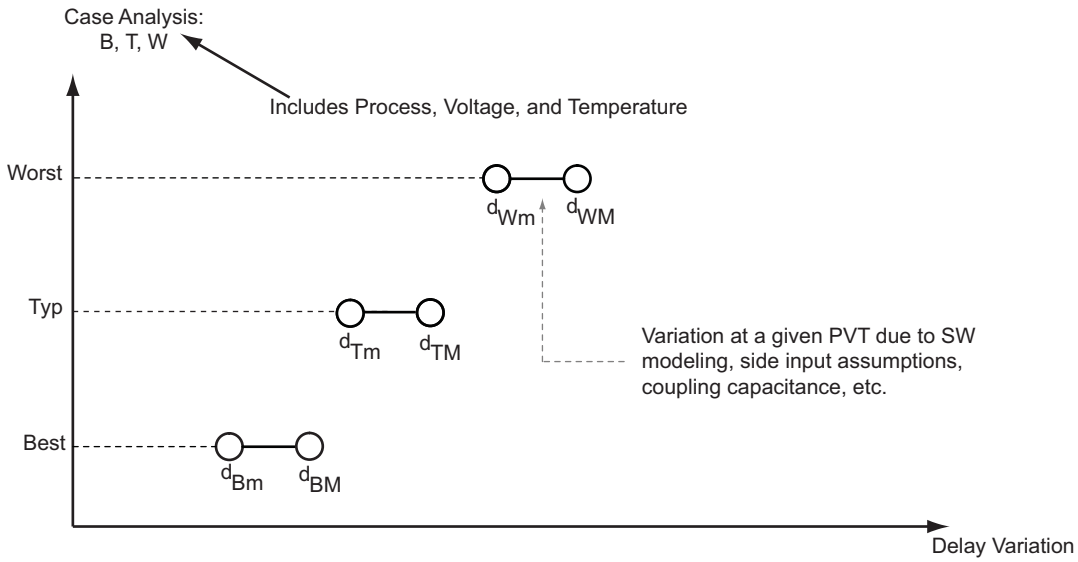


Figure 44 • **Timing Delay under Various Conditions**

The combination of temperature, voltage and process allow timing variation across various corners. This is why designers want to qualify their design across many conditions. Although multiple corner cases exist where a design can be analyzed, a designer normally uses the following four corners for timing analysis: Min-BEST, Min-WORST, Max-BEST, and Max-WORST. The most extreme timing numbers are found at these corners.

SmartTime performs analysis for Max-WORST and Min-BEST scenarios by default. In general, this is correct for most of the designs. However, if you have very tight slack, the analysis for the other two cases should be done by changing the SmartTime default setting. The "Timing Analysis for Min-WORST or Max-BEST Scenario" section on page 32 shows how to perform analysis for all four corner cases using SmartTime.

Timing Analysis for Min-WORST or Max-BEST Scenario

1. Open the SmartTime Options dialog box (Figure 45) by selecting **Tools** > **Options** from the SmartTime menu bar. You can see that the maximum delay analysis is based on BEST and the minimum delay analysis is based on WORST condition.

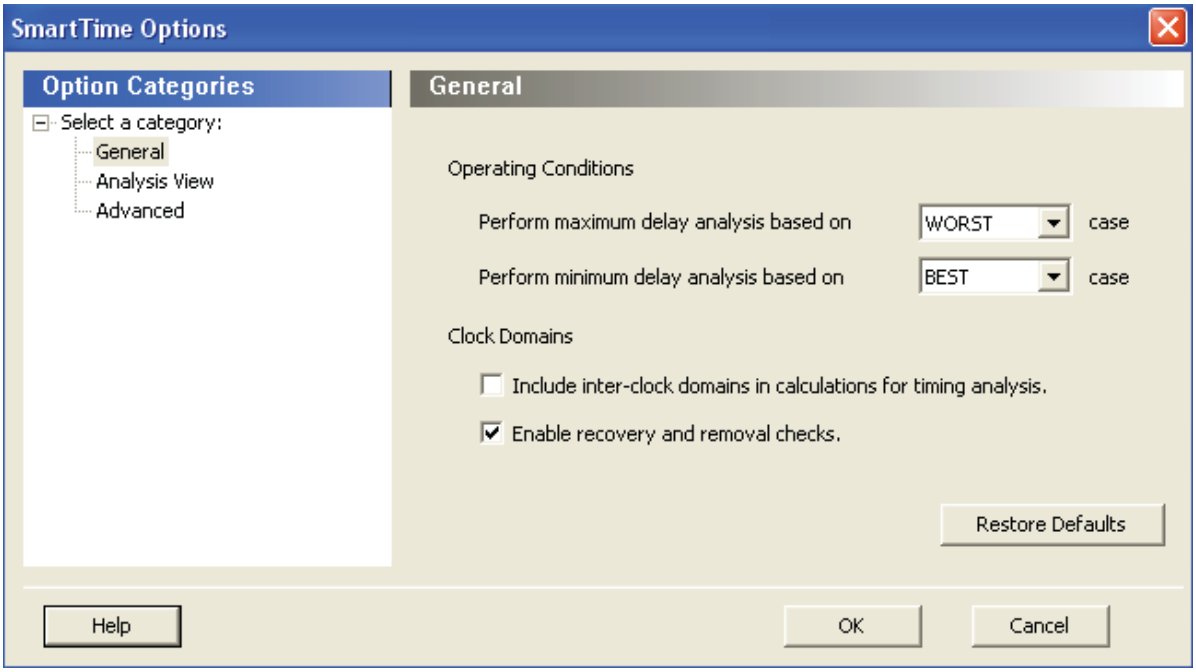


***Figure 45* • SmartTime Options Dialog Box**

2. Under Operating Conditions, change **Perform maximum delay analysis based on** to **BEST** and **Perform minimum delay analysis based on** to **WORST**.

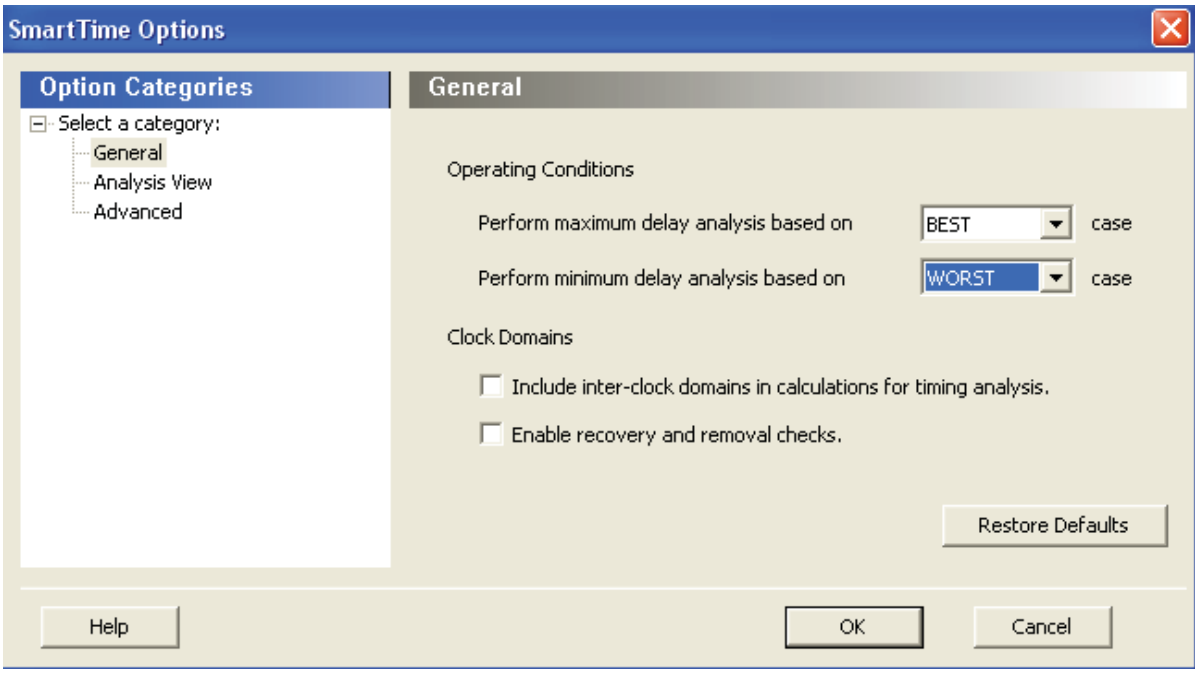


***Figure 46* • SmartTime Options Dialog Box for Max-BEST and Min-WORST Analysis**

With this setup, maximum delay analysis view shows setup check under Min-WORST condition and minimum delay analysis view shows hold check for Max-BEST condition.

Consider the design example shown in Figure 47. The clock network has some buffers which add skew on the clock network.

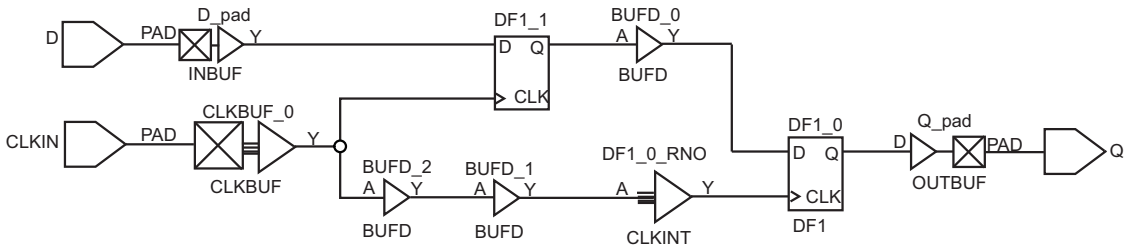


***Figure 47* • Design Example for Min-BEST and Min-WORST Analysis**

The default minimum delay analysis view shows a slack of –1.32 ns for the register-to-register path. However, changing the minimum delay analysis view to WORST shows slack of –1.877 ns. You can see that the Min-BEST condition does not always have the worst case scenario for hold check.

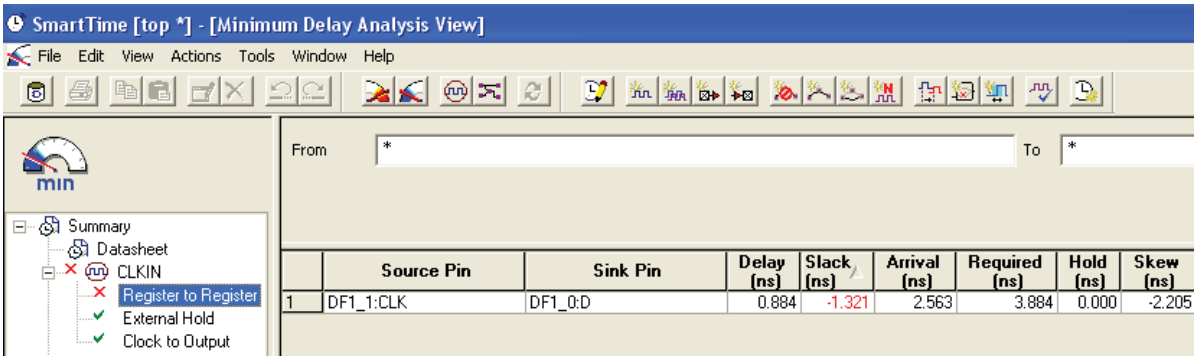


***Figure 48* • Register-to-Register Path for Min-BEST Condition**

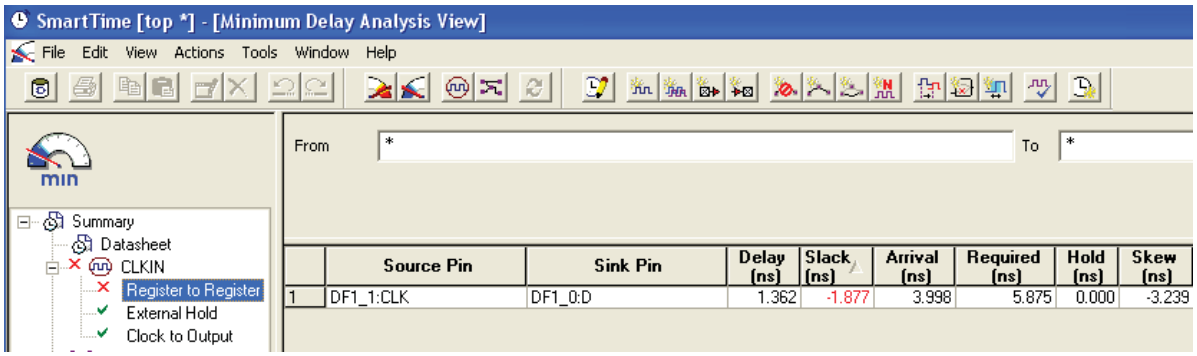


***Figure 49* • Register-to-Register Path for Min-WORST Condition**

In summary, the SmartTime timing analyzer default setting only checks for Max-WORST and Min-BEST conditions. You need to change the settings to check for Min-WORST or Max-BEST condition if you have tight margin in your design.

Appendix A: Applying a Clock Constraint

1. Open the SmartTime constraints editor by clicking the **Constraints Editor** button in the Designer GUI. The clock constraint is displayed in the SmartTime Constraints Editor, as shown in Figure 50.

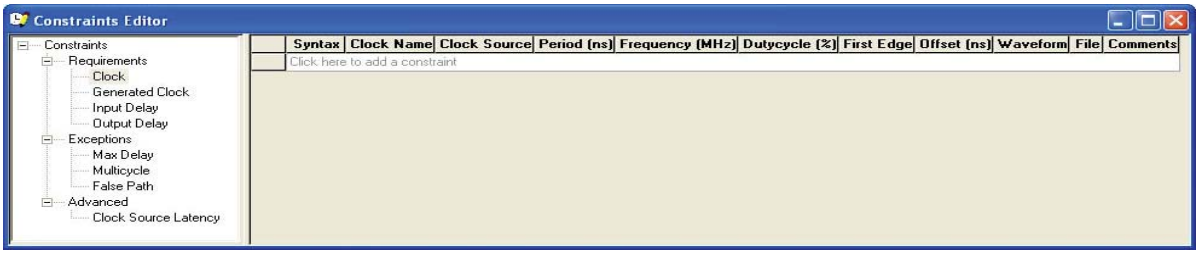


***Figure 50* • SmartTime Constraints Editor**

2. Add a clock constraint by clicking the new clock constraint button in the SmartTime toolbar, or by selecting **Actions** > **Constraint** > **Clock** from the SmartTime Menu bar. The Create Clock Constraint dialog box is displayed.

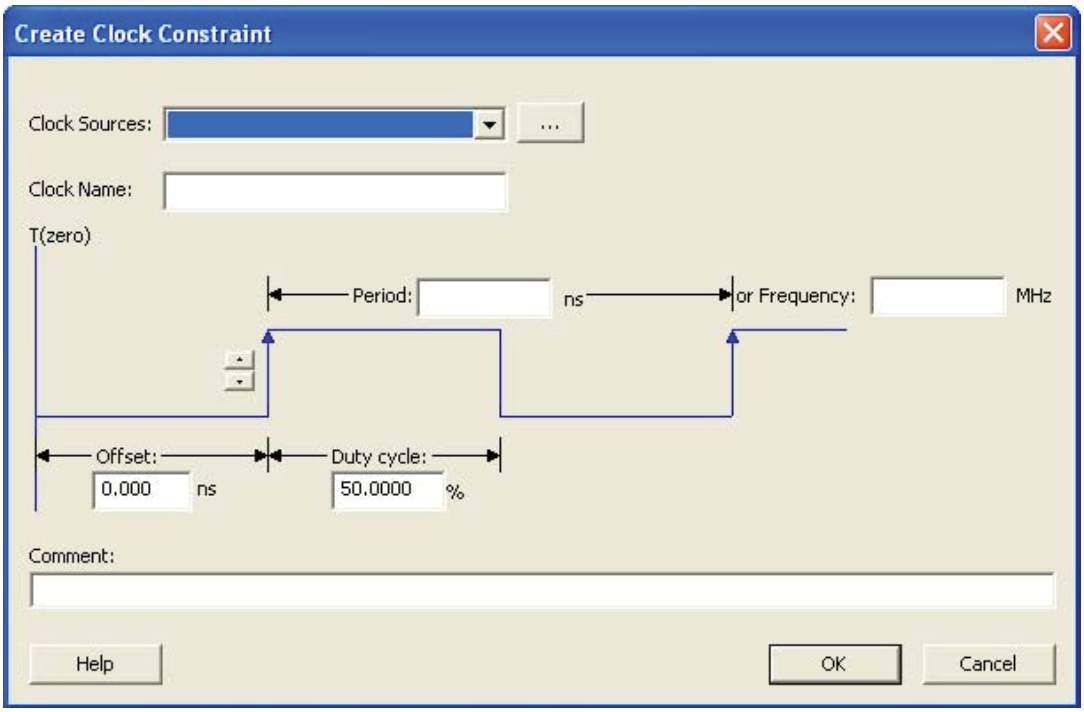


***Figure 51* • Create Clock Constraint Dialog Box**

3. Select the clock source pin from the **Clock Sources** drop-down list or by clicking the browse button. Select the pin CLK as the clock source. Click **OK** to close the Clock Source Pin dialog box.

4. Enter 150 as the **Frequency** in the Create Clock Constraint box and accept all other default values. Click **OK** to create the clock constraint.

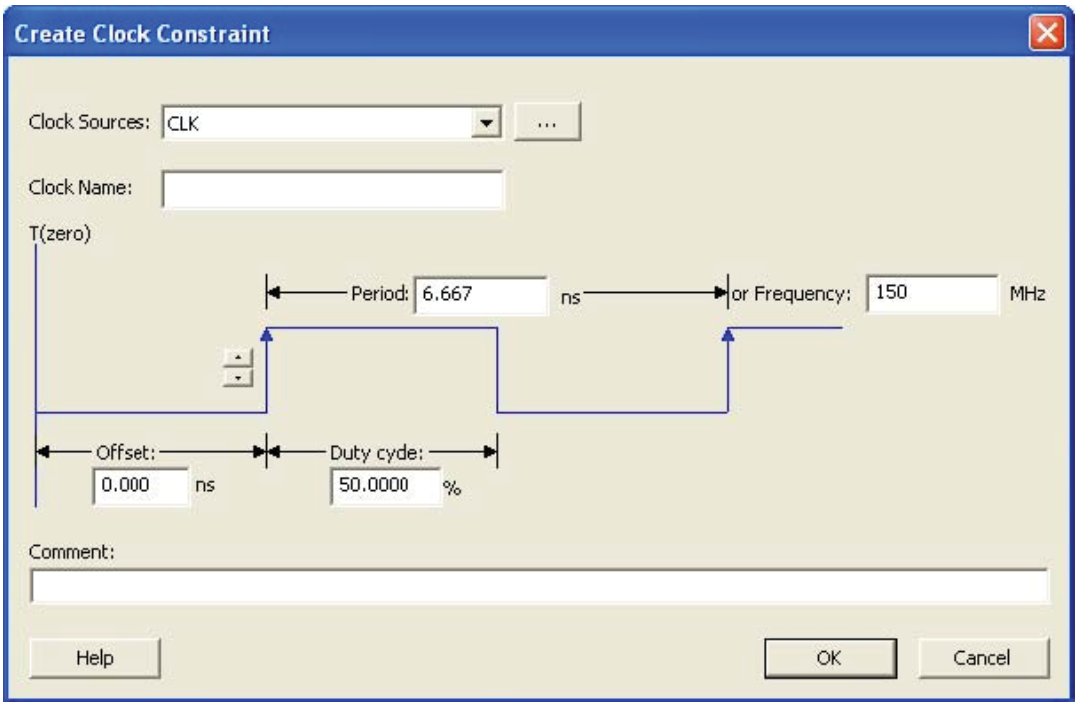


***Figure 52* • Entering a Clock Constraint in the Create Clock Constraint Dialog Box**

The clock constraint is visible in the SmartTime Constraints Editor.

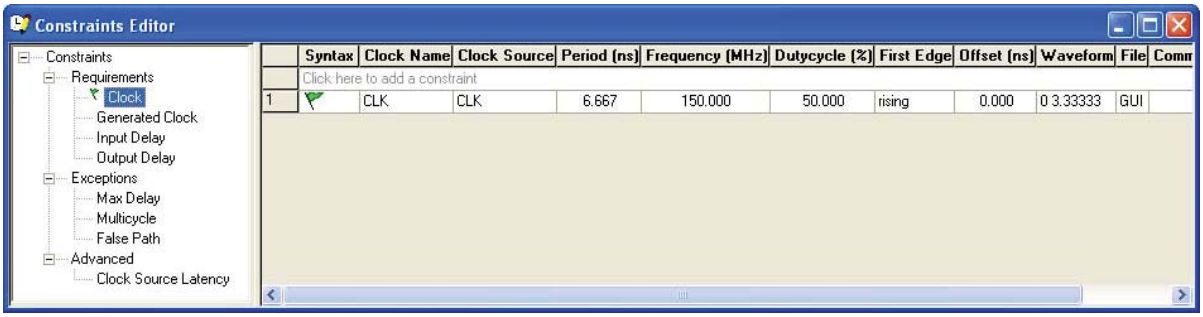


***Figure 53* • SmartTime Constraints Editor with Clock Constraint**

Appendix B: Applying a Generated Clock Constraint

1. Open the SmartTime constraints editor by clicking the **Constraints Editor** button in the Designer GUI. The clock constraint is visible in the SmartTime Constraints Editor, as shown in Figure 54.

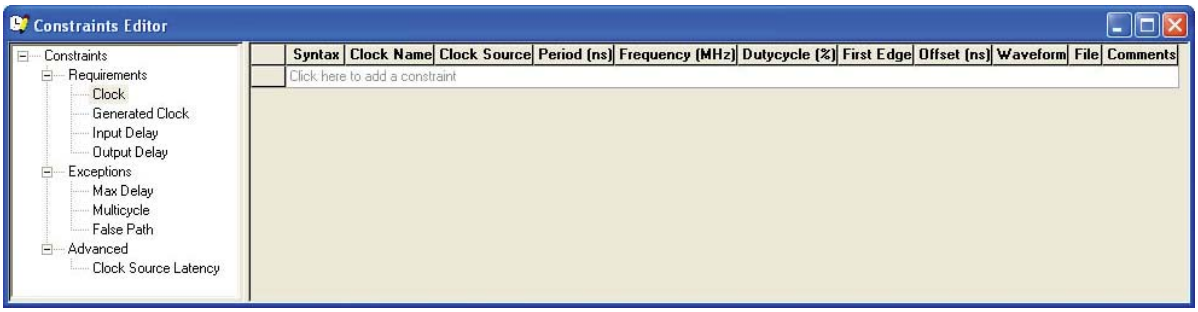


***Figure 54* • SmartTime Constraints Editor**

2. Right-click **Generated Clock** in the Constraints Editor window. The Create Generated Clock Constraint dialog box is displayed.

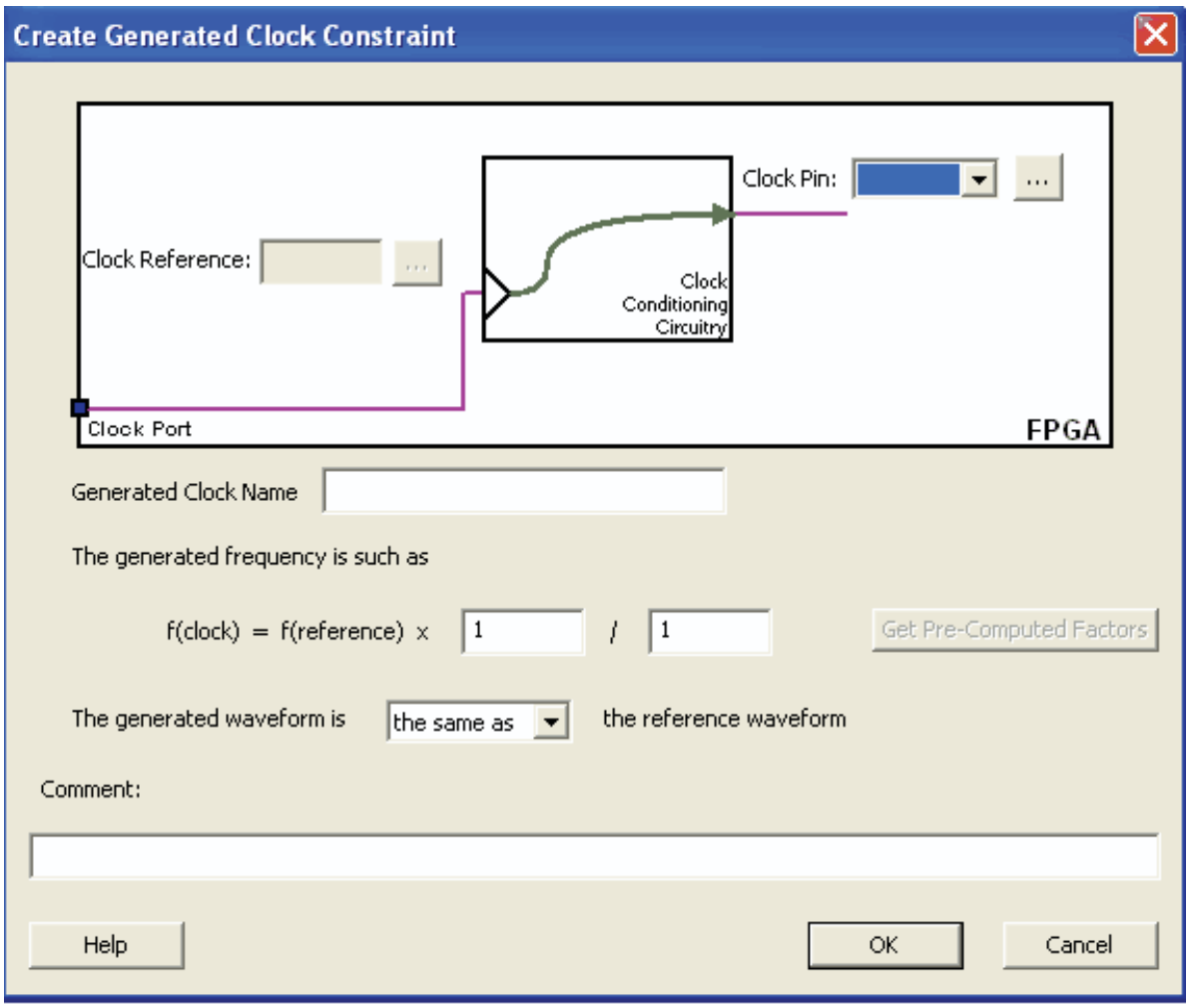


***Figure 55* • Generated Clock in the Constraint Window**

3. Browse to select a **Clock Pin**. The Select Generated Clock Source dialog box displays with the list of available generated clock source pins, as shown in Figure 56.

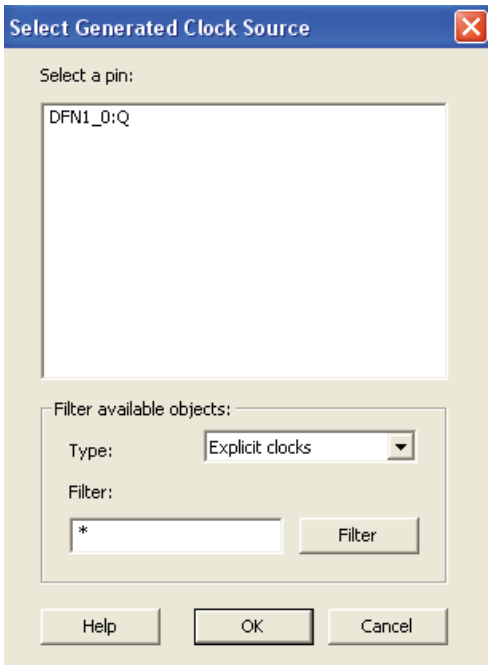


***Figure 56* • Select Generated Clock Source Dialog Box**

4. Select the DFN1_0:Q pin and click **OK** to save the clock constraint details. In some cases, the generated clock pins are not defined as Explicit clocks. You need to change the filter type and add the generated clock source pin.

5. Browse to select a Reference Pin. The Select Generated Clock Reference dialog box displays the list of available clock reference pins, as shown in Figure 57.

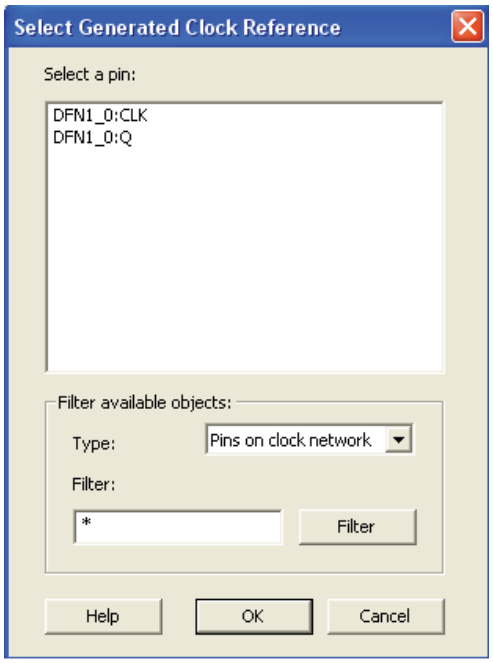


Figure 57 • **Select Generated Clock Reference Dialog Box**

6. Select the DFN1_0:CLK pin and click **OK** to save the clock constraint details. Note that DFN1_0:CLK is actually CLKA.
7. Enter the division factor of 2, since DFN1_0:Q is a "divided by 2" clock of DFN1_0: CLK.
8. Enter the first edge of the generated waveform as "the same as" with respect to the reference waveform.
9. Click **OK**. The new constraint appears in the Constraints List.

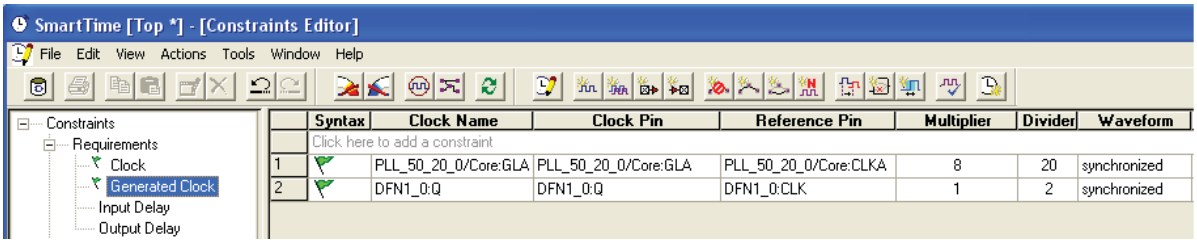


Figure 58 • **Constraints List**

Appendix C: Enabling Inter-Clock Domains Analysis

1. Select **Tools** > **Options** from the SmartTime menu bar.
2. Select the **Include Inter-clock domains in calculations for timing analysis** check box in the SmartTime Options dialog box to select inter-clock domain analysis, as shown in Figure 59. Click **OK**.

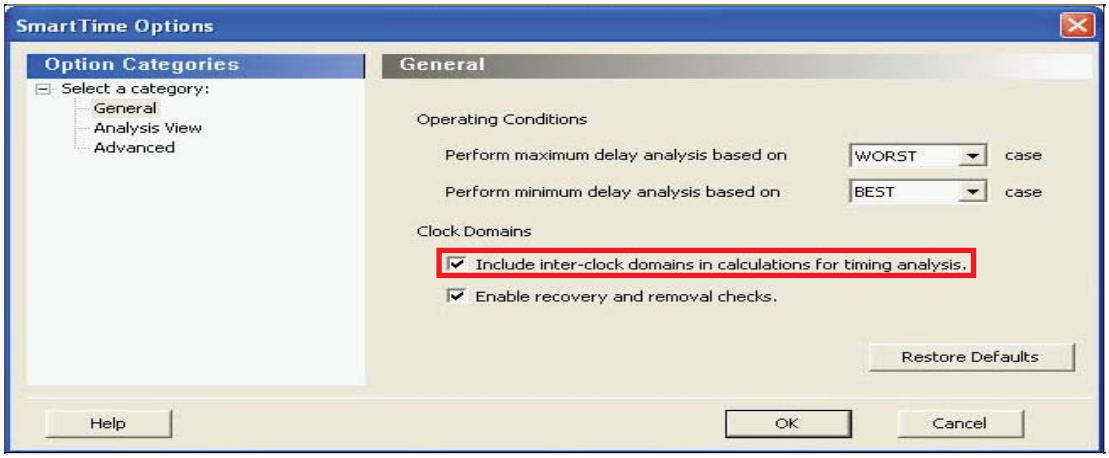


***Figure 59* • Enabling Inter-Clock Domain Analysis**

Appendix D: Applying a Multicycle Clock Constraint

1. Right-click **Multicycle** under exception in the SmartTime constraints editor. The Set Multicycle Constraint dialog box is displayed.

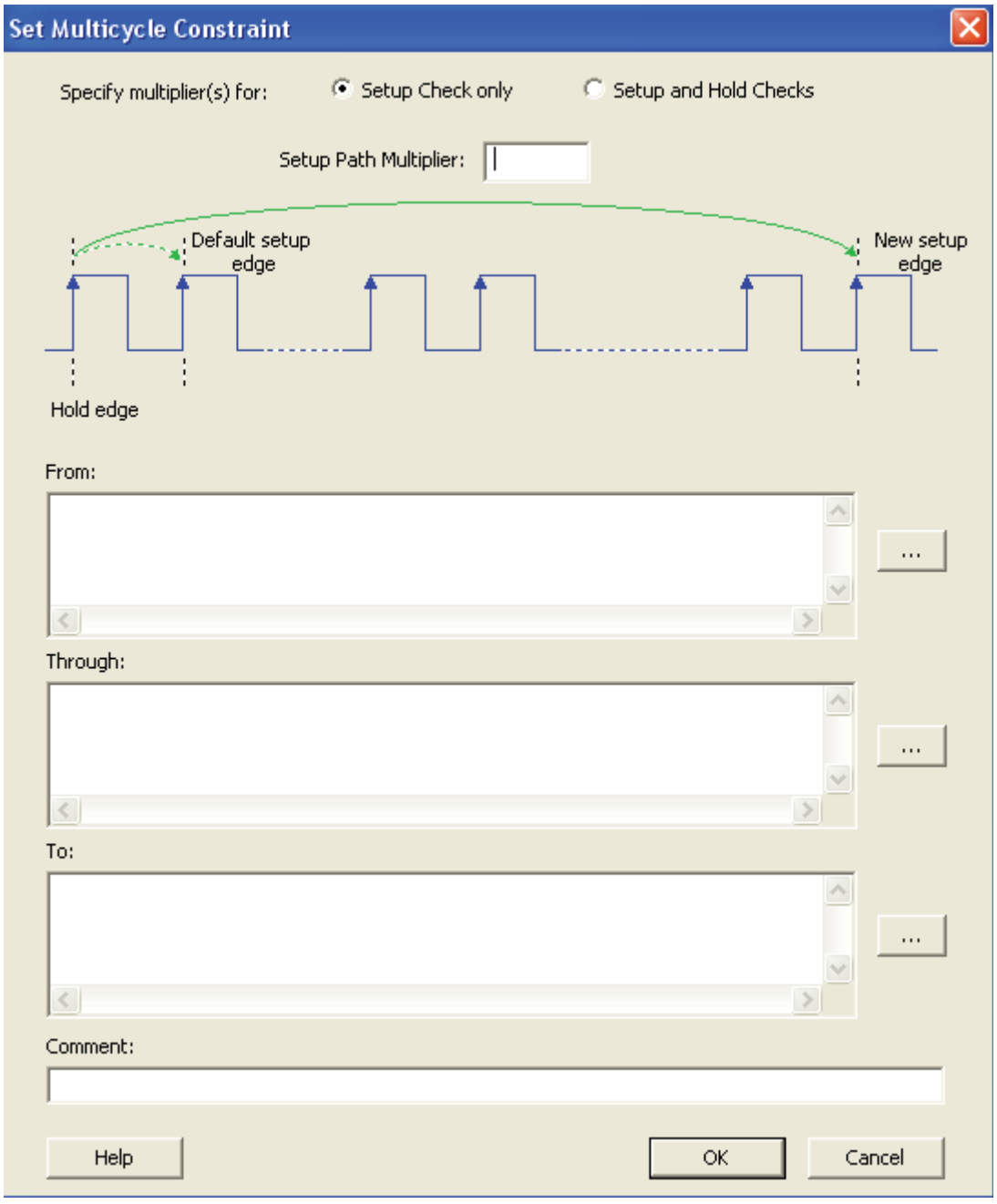


***Figure 60* • Set Multicycle Constraint Dialog Box**

2. Select the **Setup Check only** radio button as the multicycle constraint. It is applied to setup only for this design. For hold check, the default edge is used.
3. Enter **Setup Path Multiplier** (2, for example).

4. Click the browse button at the **Through** text box. The Select Through Pins for Multicycle Constraint dialog box is displayed with the list of available pins in the design, as shown in Figure 61.

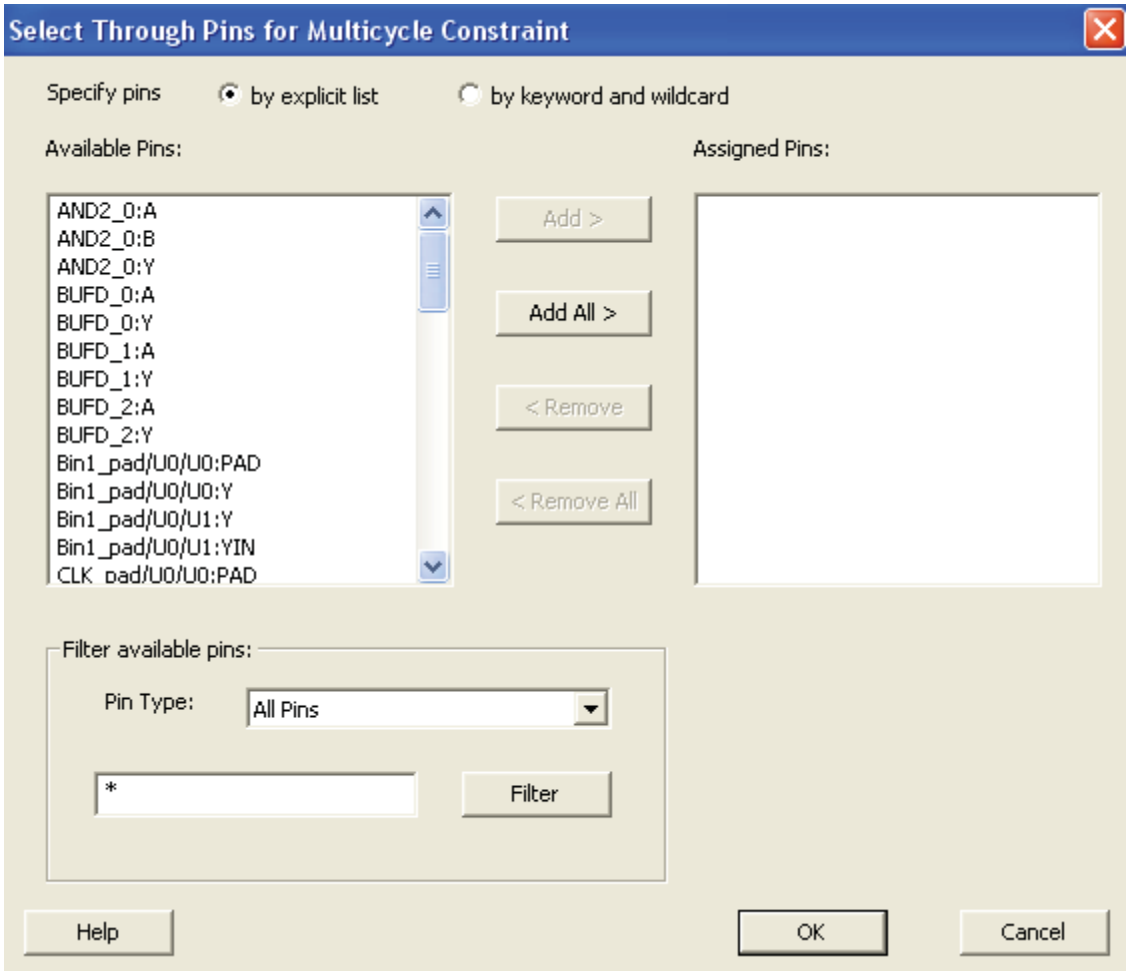


***Figure 61* • Adding Through Pins for Multicycle Path**

5. Select a through pin (MX2_0:Y, for example) and then click **OK** to save this dialog box setting. On applying the multicycle path constraint, it will be shown in the constraint Editor.

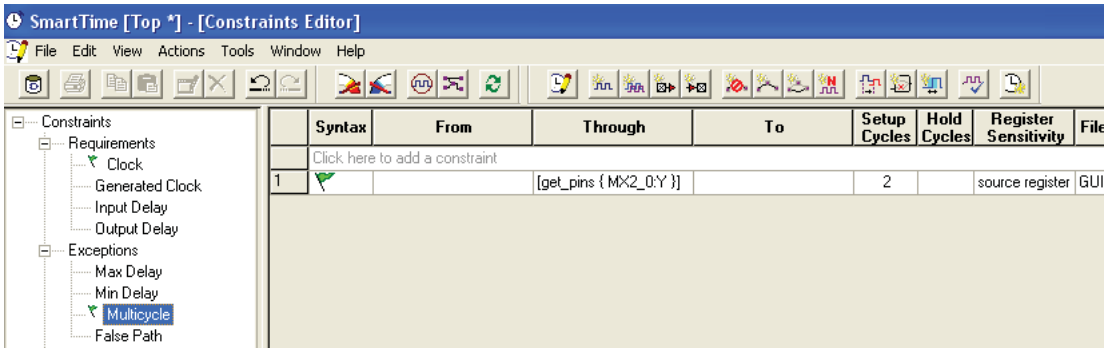


***Figure 62* • Multicycle Path Constraints in Constraint Editor**

Microsemi Corporate Headquarters
One Enterprise Drive, Aliso Viejo CA 92656
Within the USA: (800) 713-4113
Outside the USA: (949) 221-7100
Fax: (949) 756-0308 · www.microsemi.com

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

51900247-0/11.11