Timing Analysis with TimeQuest

Software : Quartus Ⅱ 11.1. SP2
Date : 2012. June

Timing analysis tool로 TimeQuest Timing Analyzer와 Classic Timing Analyzer를 사용할 수 있었는데 Quartus Ⅱ 10.1 버전이후에는 TimeQuest Timing Analyzer만 사용할 수 있습니다. TimeQuest에서 Timing 분석을 하기 위한 방법을 소개합니다.

Flow Summary

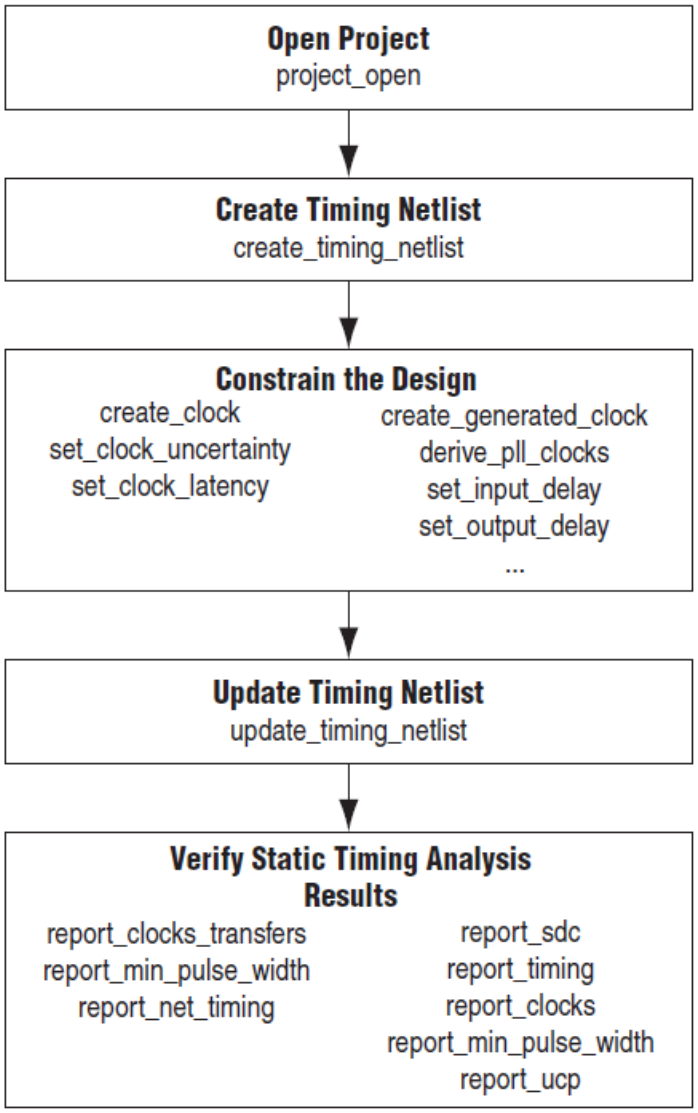
Timing 분석 단계는 아래처럼 요약할 수 있습니다.
이 문서에서는 2번 과정에 대해 집중적으로 다룰 예정입니다.
1. Compile full design at Quartus Ⅱ.
2. Analyze timing and make sdc files at TimeQuest Timing Analyzer GUI.
3. Re-compile full design at Quartus Ⅱ with sdc files.
4. Re-analyze timing at TimeQuest

1번 단계에서 기본적으로 timing analysis가 이루어지는데 이때 timing constraint가 들어가 있지 않아 정확한 결과라고 볼 수 없습니다.

TimeQuest Timing Analyzer

Timing Analysis flow

TimeQuest Timing Analyzer내에서는 아래와 같은 단계를 거쳐야 합니다. 아래에서 Constrain the Design과 Verify Static Timing Analysis results 부분을 주목하면 됩니다.

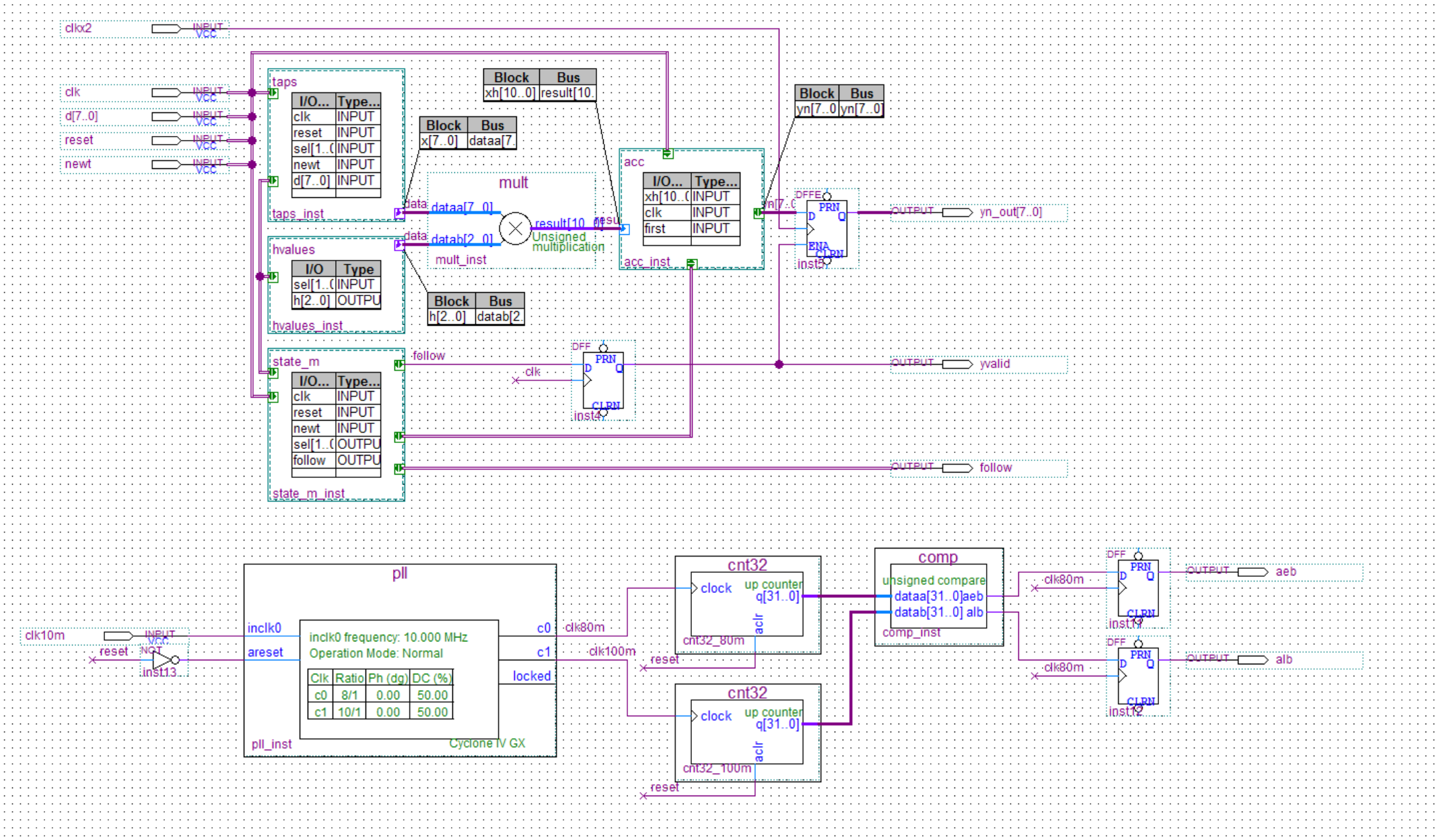


Test Design

아래 그림과 같은 디자인에서 timing을 분석할 것입니다. clock 정보는 다음과 같습니다.
clk : 50 MHz

clkx2 : 100 MHz

clk10m : 10 MHz

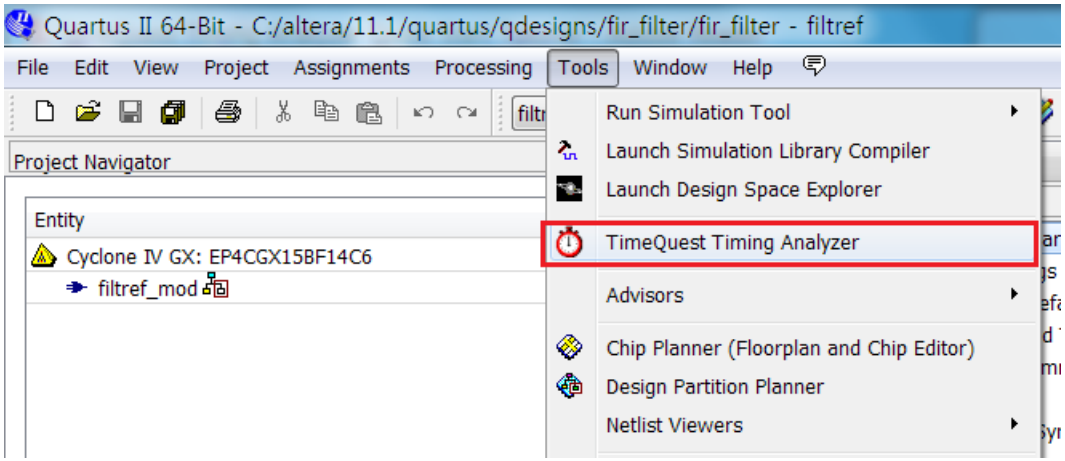


1. Compile full design at Quartus Ⅱ.

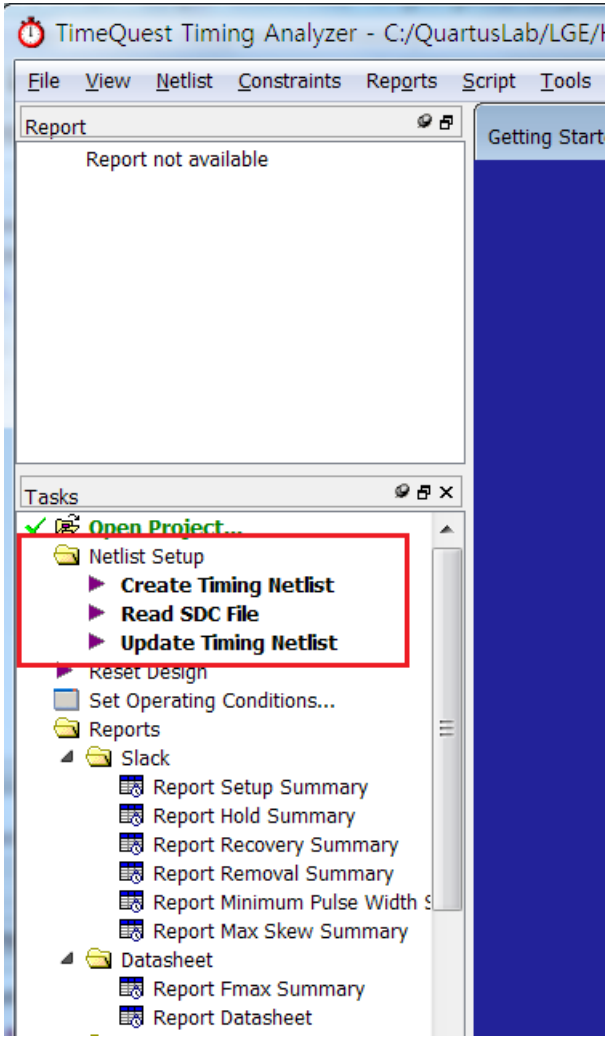
먼저 Quartus project를 full compilation을 진행합니다.

2. Analyze timing and make sdc files at TimeQuest Timing Analyzer GUI

Compilation이 끝난 후 Tools/TimeQuest Timing Analyzer를 클릭합니다.

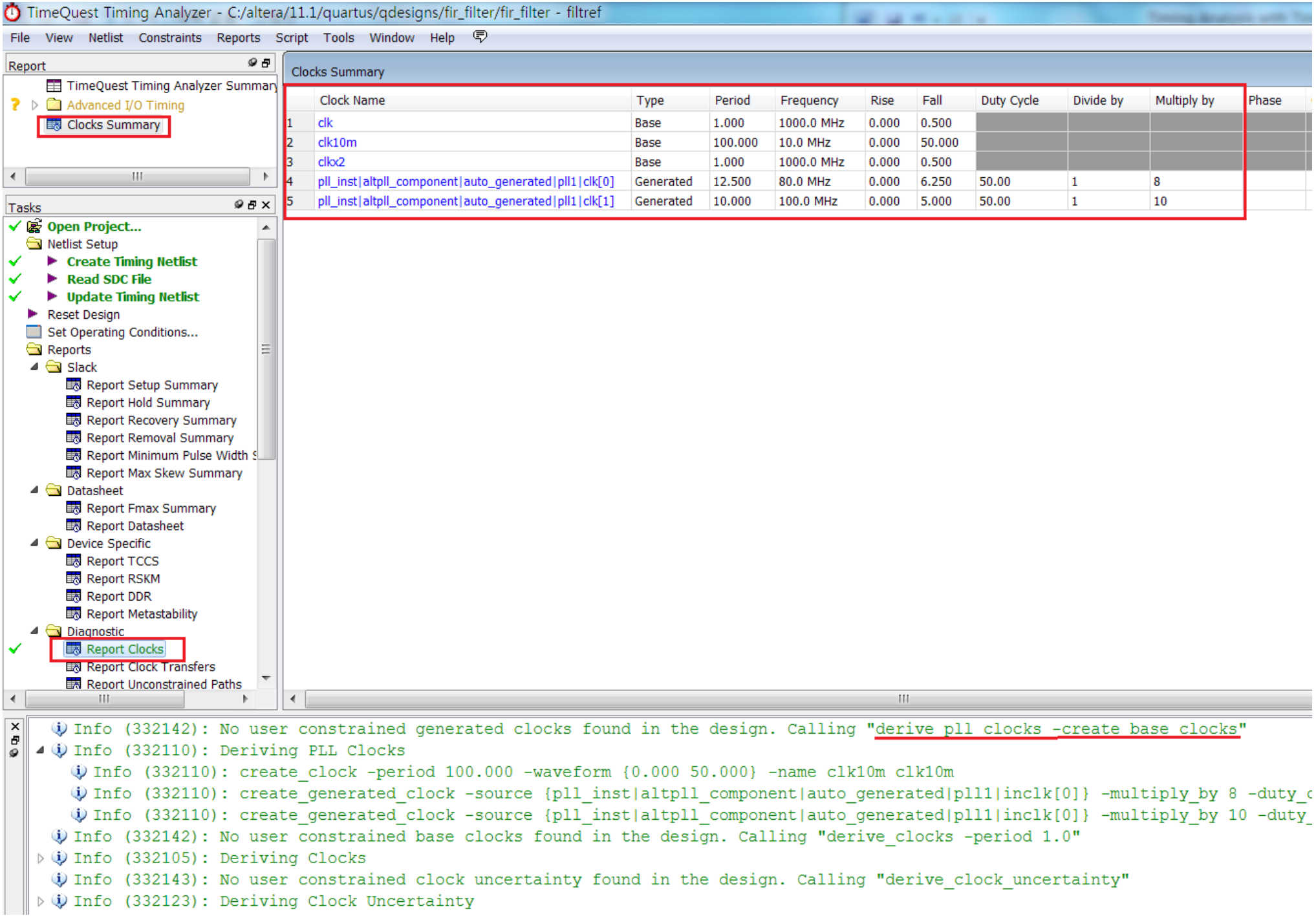


TimeQuest가 열리면 아래와 같은 화면이 나오는데 우선 netlist를 설정해야 합니다. 아래 붉은 상자 안의 과정을 수행하면 netlist를 생성합니다. 마지막 단계의 update Timing Netlist를 더블 클릭하면 앞의 두 단계를 자동으로 수행합니다. sdc file은 timing constraints가 들어있는 파일입니다. Read SDC file을 수행하면 Quartus에서 설정한 sdc file이 있는 경우 해당 file을 불러오며 sdc file이 없을 경우에는 skip합니다.

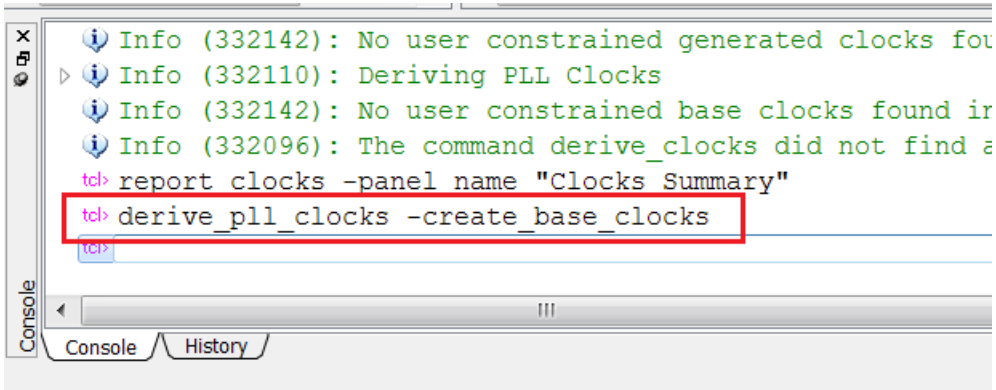


완료되면 아래 그림처럼 녹색으로 표시됩니다.

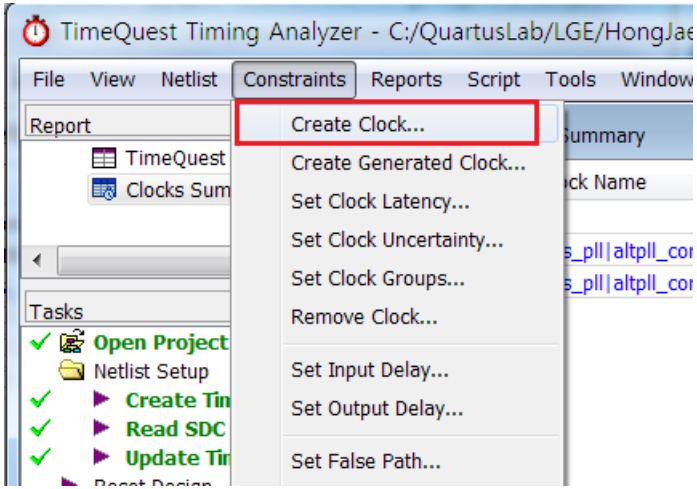
그리고 아래 붉은 상자의 Report Clocks를 더블 클릭하면 Report panel에 Clocks Summary가 생성되며 사용하고 있는 clock 정보를 보여줍니다.


TimeQuest Timing Analyzer - C:/altera/11.1/quartus/qdesigns/fir_filter/fir_filter - filtref
File View Netlist Constraints Reports Script Tools Window Help
Report
TimeQuest Timing Analyzer Summary
Advanced I/O Timing
Clocks Summary
Tasks
Open Project...
Netlist Setup
Create Timing Netlist
Read SDC File
Update Timing Netlist
Reset Design
Set Operating Conditions...
Reports
Slack
Report Setup Summary
Report Hold Summary
Report Recovery Summary
Report Removal Summary
Report Minimum Pulse Width
Report Max Skew Summary
Datasheet
Report Fmax Summary
Report Datasheet
Device Specific
Report TCCS
Report RSKM
Report DDR
Report Metastability
Diagnostic
Report Clocks
Report Clock Transfers
Report Unconstrained Paths
Clocks Summary
Clock Name Type Period Frequency Rise Fall Duty Cycle Divide by Multiply by Phase
1 clk Base 1.000 1000.0 MHz 0.000 0.500
2 clk10m Base 100.000 10.0 MHz 0.000 50.000
3 clkx2 Base 1.000 1000.0 MHz 0.000 0.500
4 pll_inst|altpll_component|auto_generated|pll1|clk[0] Generated 12.500 80.0 MHz 0.000 6.250 50.00 1 8
5 pll_inst|altpll_component|auto_generated|pll1|clk[1] Generated 10.000 100.0 MHz 0.000 5.000 50.00 1 10
Info (332142): No user constrained generated clocks found in the design. Calling "derive pll clocks -create base clocks"
Info (332110): Deriving PLL Clocks
Info (332110): create_clock -period 100.000 -waveform {0.000 50.000} -name clk10m clk10m
Info (332110): create_generated_clock -source {pll_inst|altpll_component|auto_generated|pll1|inclk[0]} -multiply_by 8 -duty_c
Info (332110): create_generated_clock -source {pll_inst|altpll_component|auto_generated|pll1|inclk[0]} -multiply_by 10 -duty_
Info (332142): No user constrained base clocks found in the design. Calling "derive_clocks -period 1.0"
Info (332105): Deriving Clocks
Info (332143): No user constrained clock uncertainty found in the design. Calling "derive_clock_uncertainty"
Info (332123): Deriving Clock Uncertainty

PLL의 입력 clock, 출력 clock은 user가 입력한 frequency가 표시되지만 그 외 clock은 1000 MHz로 report합니다.
Quartus에서 PLL을 만들 때 input clock과 output clock의 frequency, phase 정보를 주기 때문에 TimeQuest Timing analyzer는 자동으로 해당 정보를 가져갈 수 있습니다. 이 기능은 Quartus version에 따라 지원안하는 경우도 있습니다. 이럴 경우 아래 console window에 표시된 붉은 상자처럼 console창에 입력해주면 됩니다.

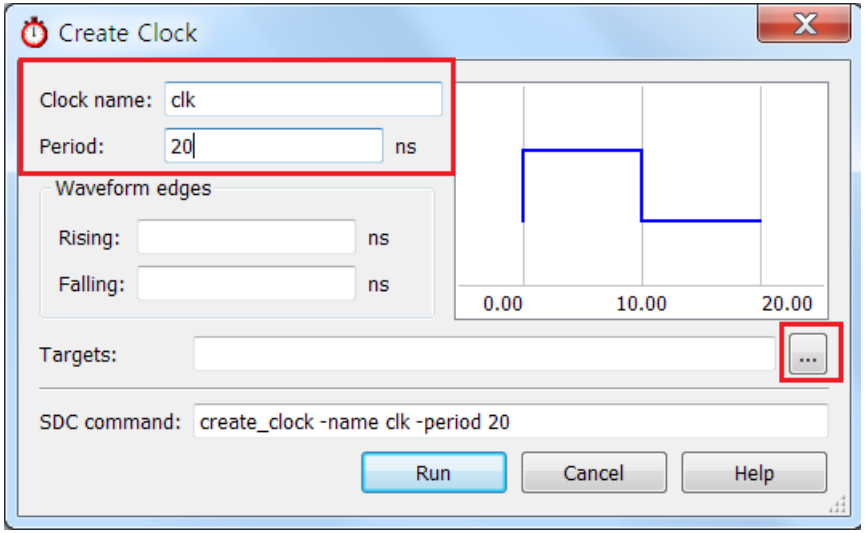


PLL을 사용하지 않는 clock이 있는 경우에 TimeQuest는 기본적으로 주기를 1 ns로 설정합니다. 그래서 report clocks에서 주파수가 1000MHz로 표시됩니다. 해당 clock에 정확한 주파수를 설정하기 위해 constraints/Create Clock을 선택합니다.

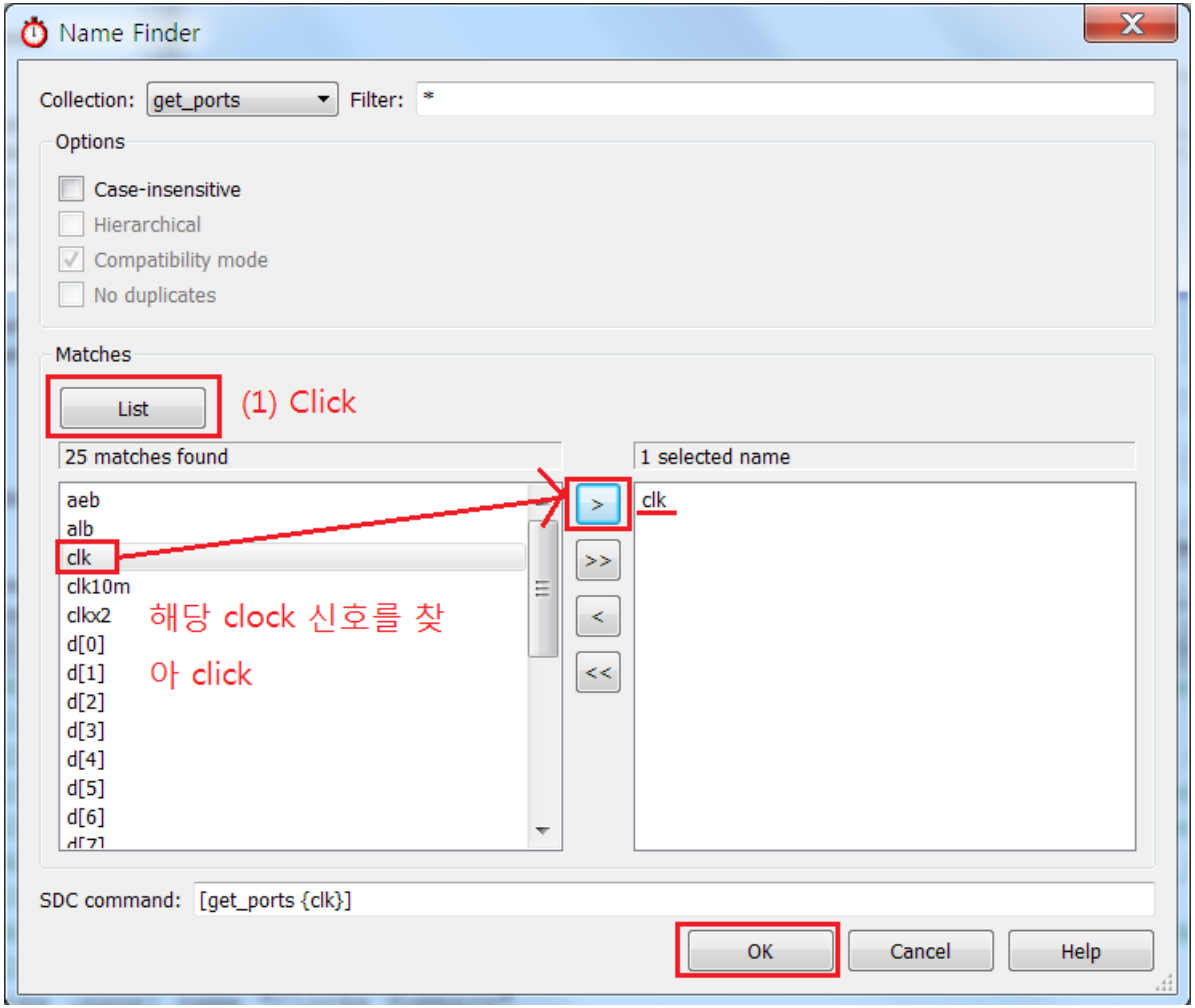


clock name에 원하는 데로 이름을 입력하고 해당 clock의 주기를 입력합니다. 만약 waveform이 falling이 먼저이면 rising과 falling에 시간을 입력합니다. 그리고 Targets옆의 

버튼을 클릭해서 clock pin을 찾아줍니다.

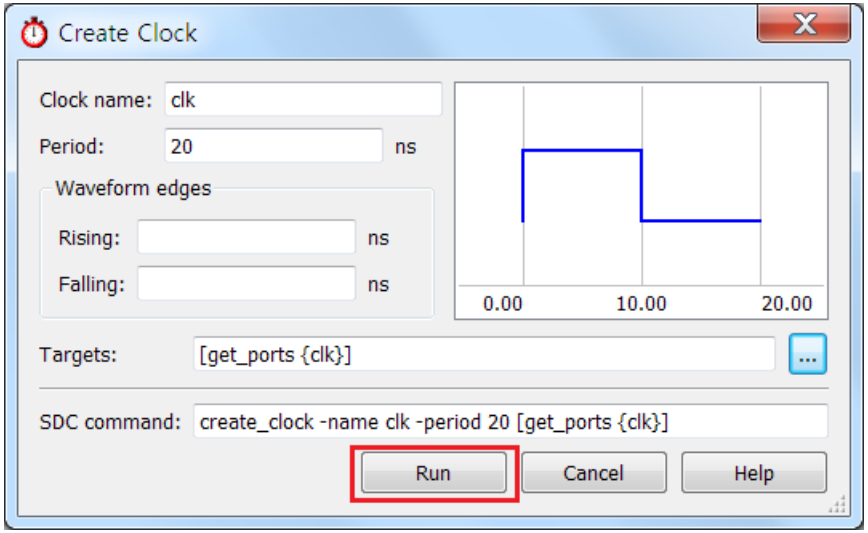


clock이 Input인 경우 Collection에서 get_ports를 선택하고 List 버튼을 클릭합니다. 그러면 아래 왼쪽에 top에 선언된 모든 입출력 port들이 보입니다. 그곳에서 해당하는 clock을 선택 후 화살표를 클릭하여 오른쪽으로 옮기고 OK를 클릭하면 됩니다.

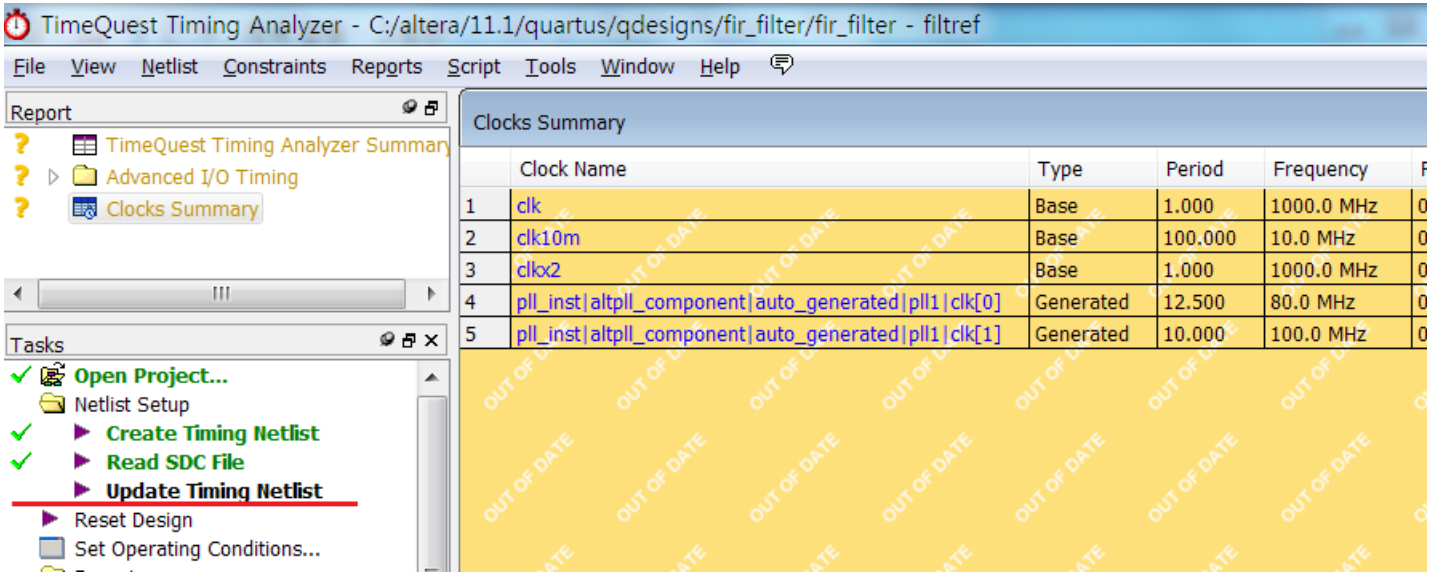


그러면 아래처럼 clk input pin이 50MHz clock이라는 constraints가 완성됩니다.

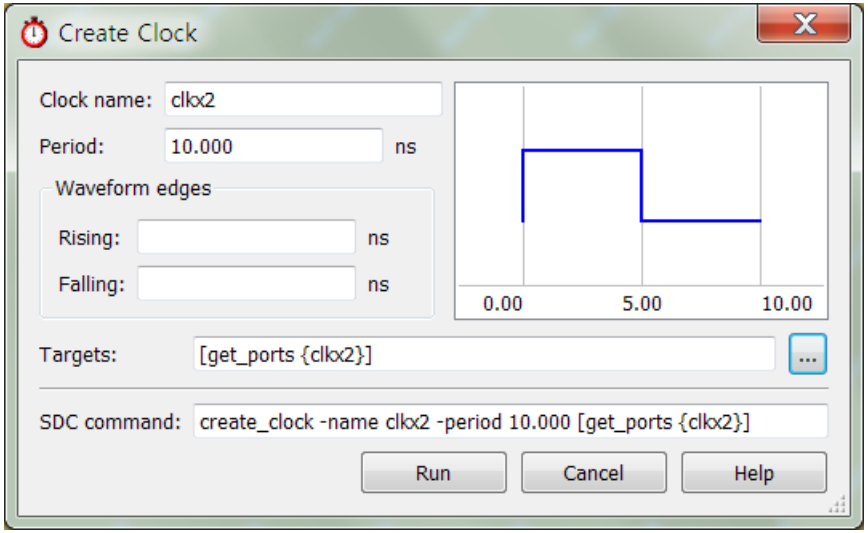
Run을 클릭하여 현재 netlist에 해당 clock 주파수 정보를 전달합니다.



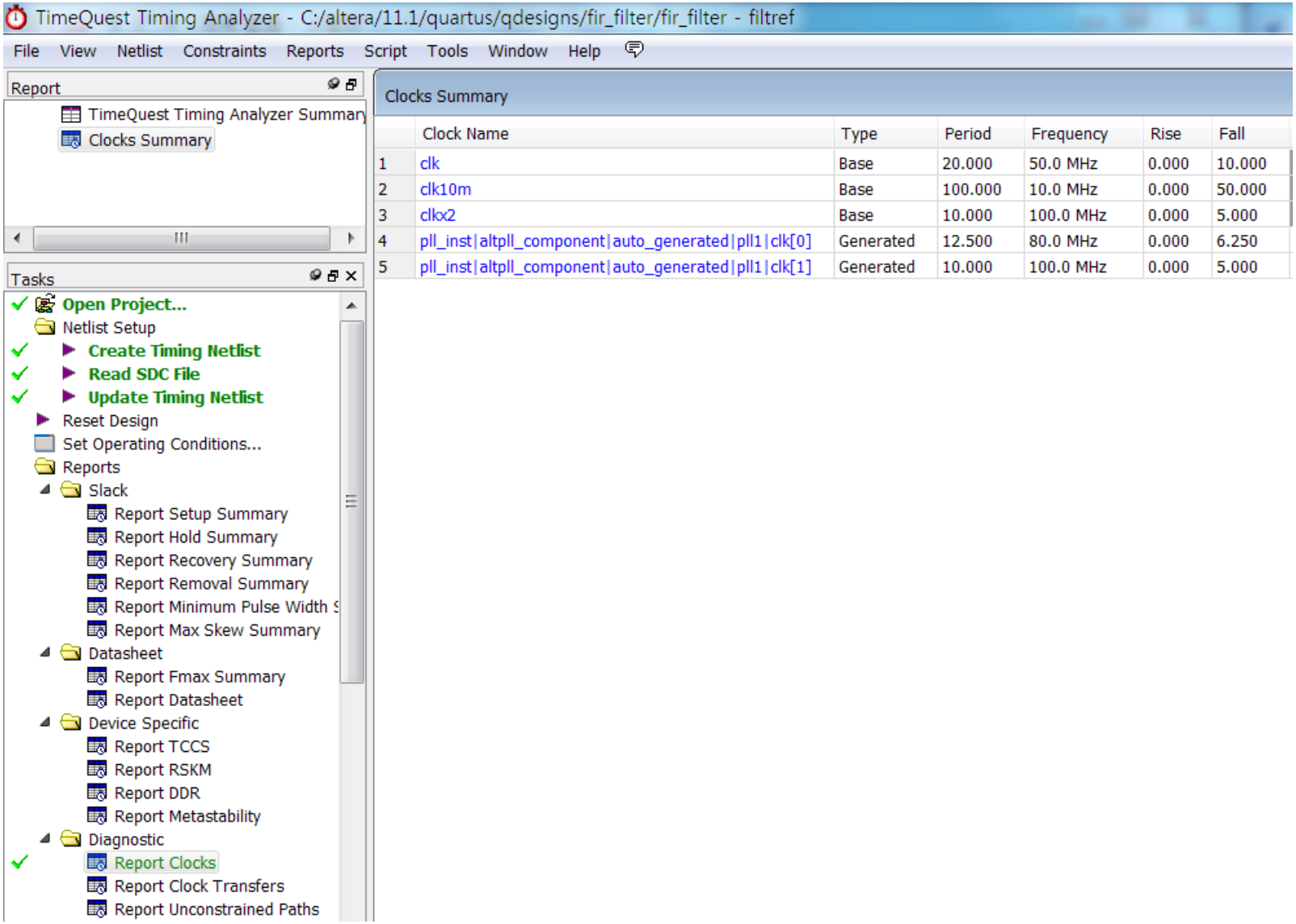
TimeQuest에서 새로운 constraints를 적용하면 자동으로 update가 되지 않기 때문에 main window의 바탕이 out of date로 노랗게 표시되며 아래 표시한 Update Timing Netlist를 다시 진행해야 합니다. 이 과정은 새로운 constraints를 적용할 때마다 반복이 됩니다.



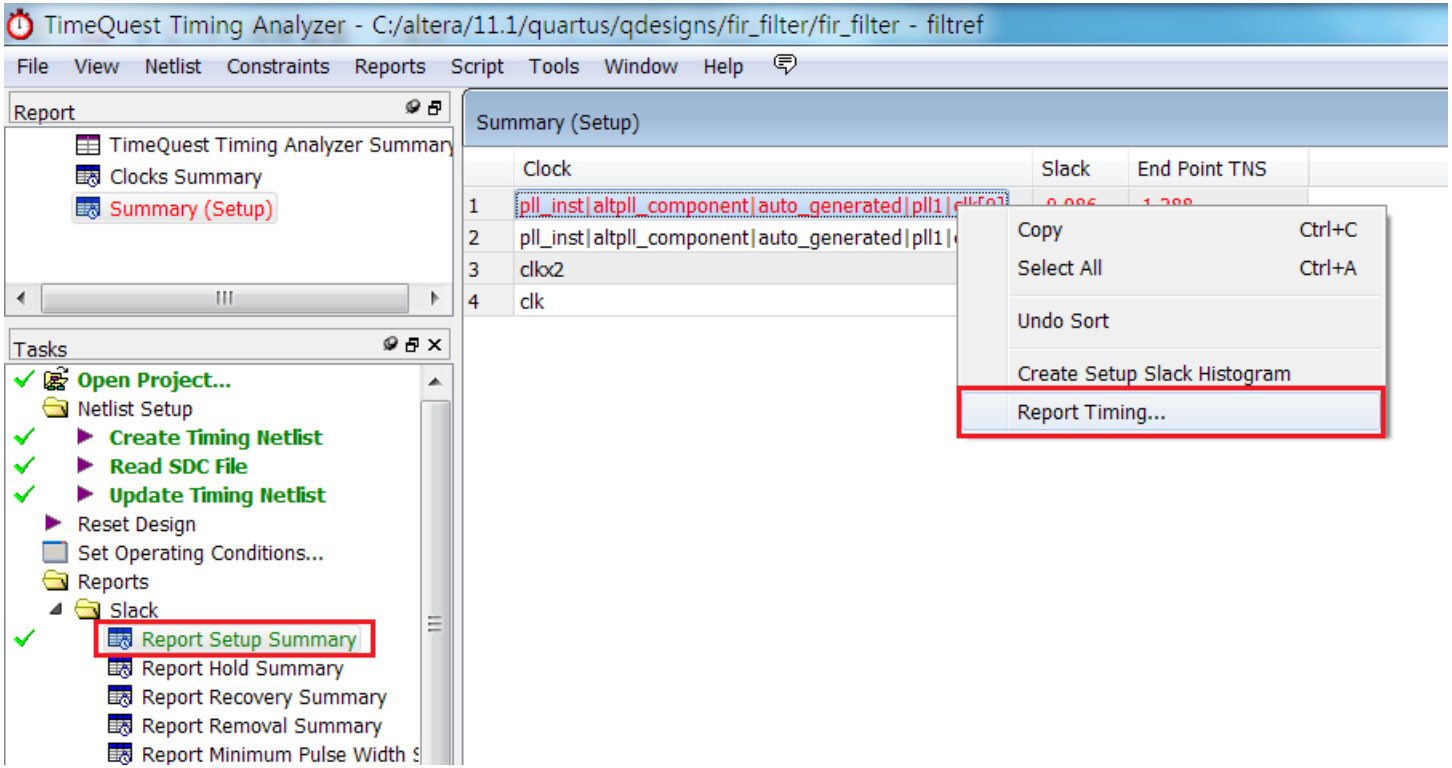
clkx2에 대해서도 동일하게 create clock을 이용하여 clock에 대한 constraint를 적용합니다.



Update Timing Netlist를 하여 clock에 대한 constraints를 update한 후 다시 report clocks를 클릭하여 모든 clock의 주파수가 design에 맞게 설정되어 있는지 확인합니다.



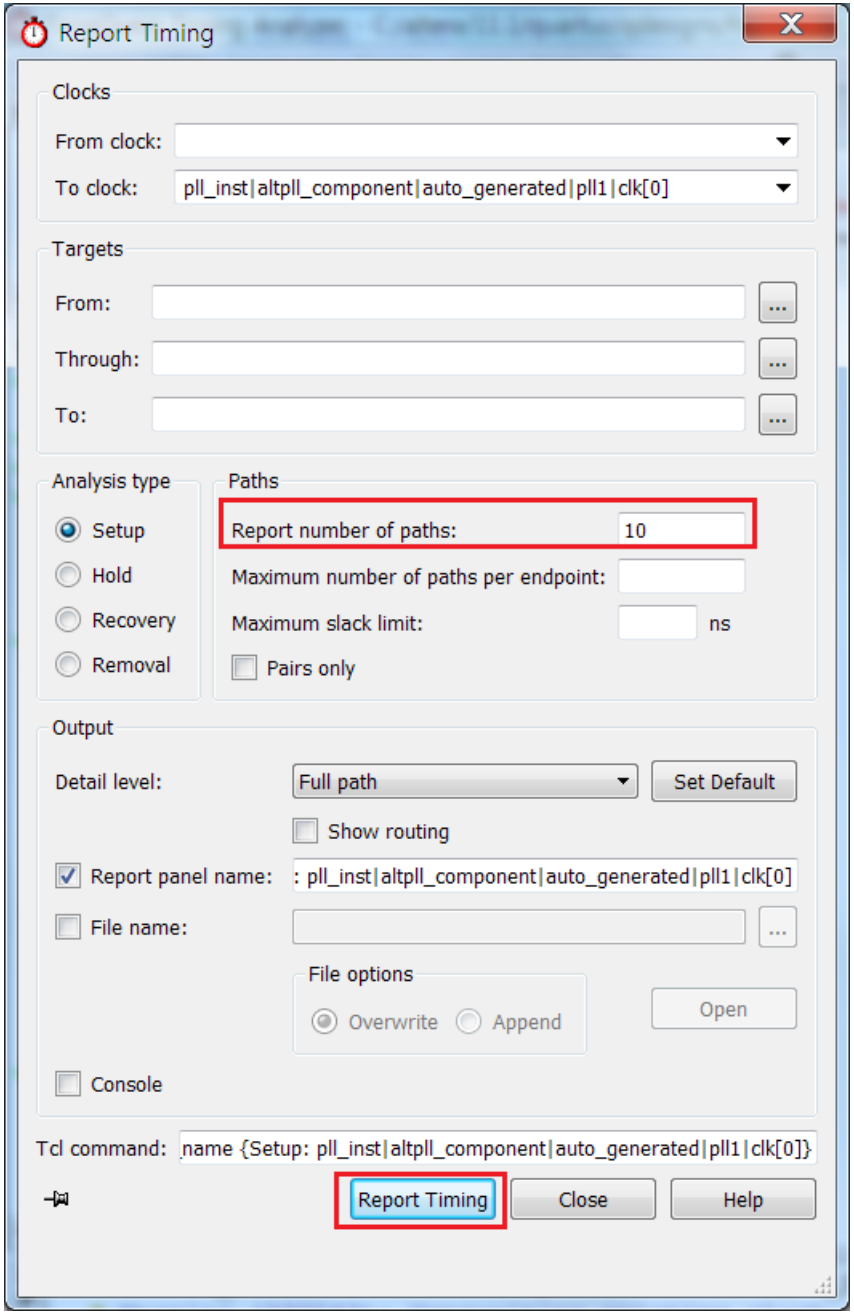
constraint가 다 적용되었으면 Reports/Slack/Report Setup Summary를 클릭하여 각 clock이 timing margin을 얼마나 확보하고 있는지 확인합니다.



위 그림에서는 PLL 출력 중 하나의 slack이 음수이며 붉은 색으로 표시되어 있는 데 이것은 Timing margin이 부족하다는 표시입니다.

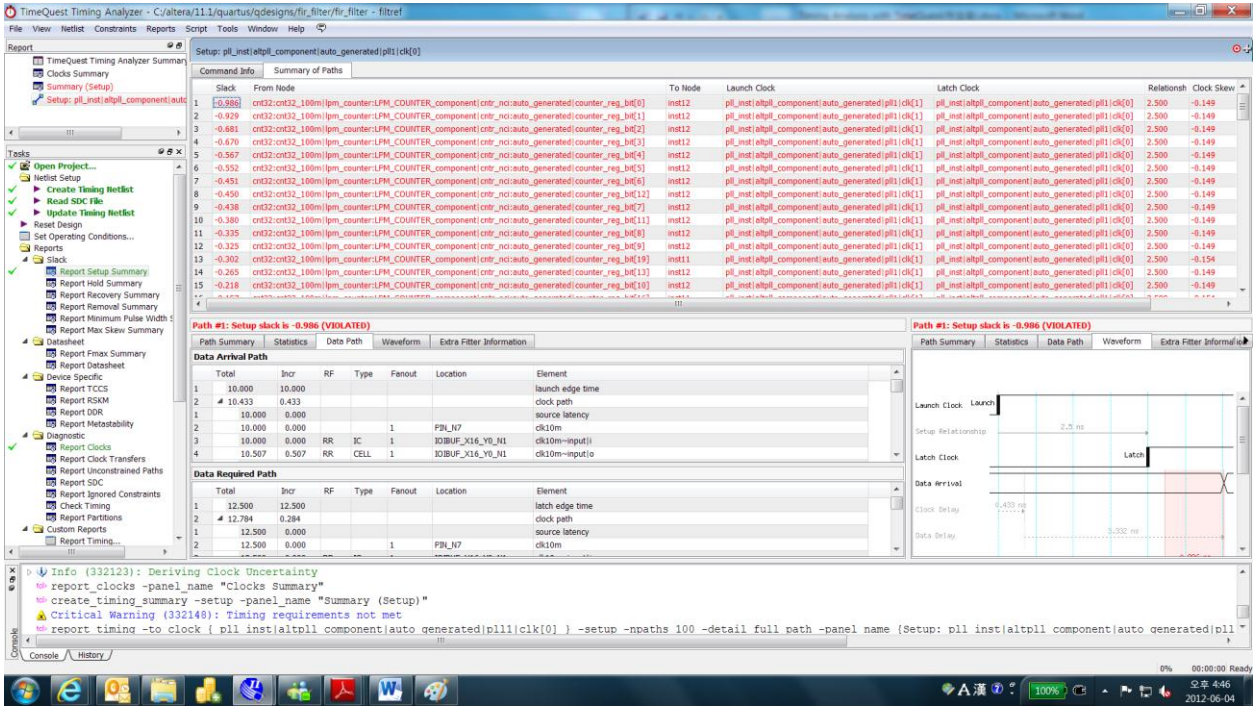
구체적으로 어디서 timing margin이 부족한 지 확인하기 위해 해당 clock을 선택하고 오른쪽 마우스를 클릭하여 Report Timing을 선택합니다.

보고 싶은 path 수는 기본적으로 10으로 설정되어 있는데 더 많은 path를 보고 싶으면 해당 숫자를 조절하면 됩니다. 나머지는 default 상태로 두고 Report Timing을 클릭합니다.



아래는 전체적인 timing 분석을 보여주는 window입니다.

Slack이 가장 안 좋은 path부터 report를 합니다.

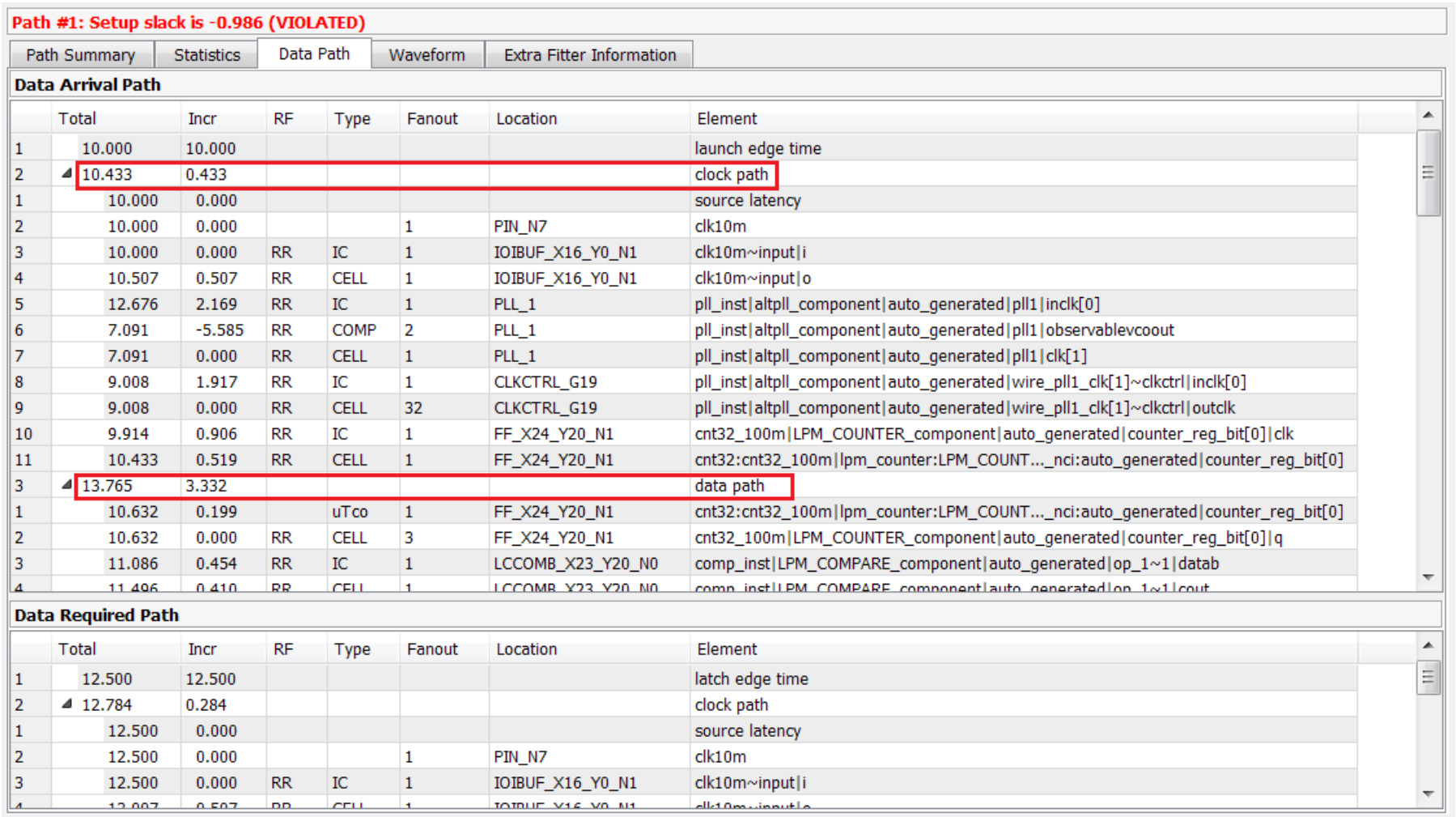


맨 윗부분에는 timing을 분석하는 path, 즉 data가 어디서 나와서 어디로 들어가는지와 source register에서 사용하는 clock과 destination register에서 사용하는 clock 정보를 보여줍니다.

Command Info | Summary of Paths

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationsh	Clock Skew	Data Delay
1	-0.986	cnt32:cnt32_100m|lpm...d|counter_reg_bit[0]	inst12	pll_inst|altpll_component|auto_generated|pll1|clk[1]	pll_inst|altpll_component|auto_generated|pll1|clk[0]	2.500	-0.149	3.332
2	-0.929	cnt32:cnt32_100m|lpm...d|counter_reg_bit[1]	inst12	pll_inst|altpll_component|auto_generated|pll1|clk[1]	pll_inst|altpll_component|auto_generated|pll1|clk[0]	2.500	-0.149	3.275
3	-0.681	cnt32:cnt32_100m|lpm...d|counter_reg_bit[2]	inst12	pll_inst|altpll_component|auto_generated|pll1|clk[1]	pll_inst|altpll_component|auto_generated|pll1|clk[0]	2.500	-0.149	3.027
4	-0.670	cnt32:cnt32_100m|lpm...d|counter_reg_bit[3]	inst12	pll_inst|altpll_component|auto_generated|pll1|clk[1]	pll_inst|altpll_component|auto_generated|pll1|clk[0]	2.500	-0.149	3.016

path를 클릭하면 아래 Data Path tab에 data arrival Path에 대한 자세한 정보를 clock path와 data path로 나누어 표시해줍니다.



Path #1: Setup slack is -0.986 (VIOLATED)

Path Summary | Statistics | Data Path | Waveform | Extra Fitter Information

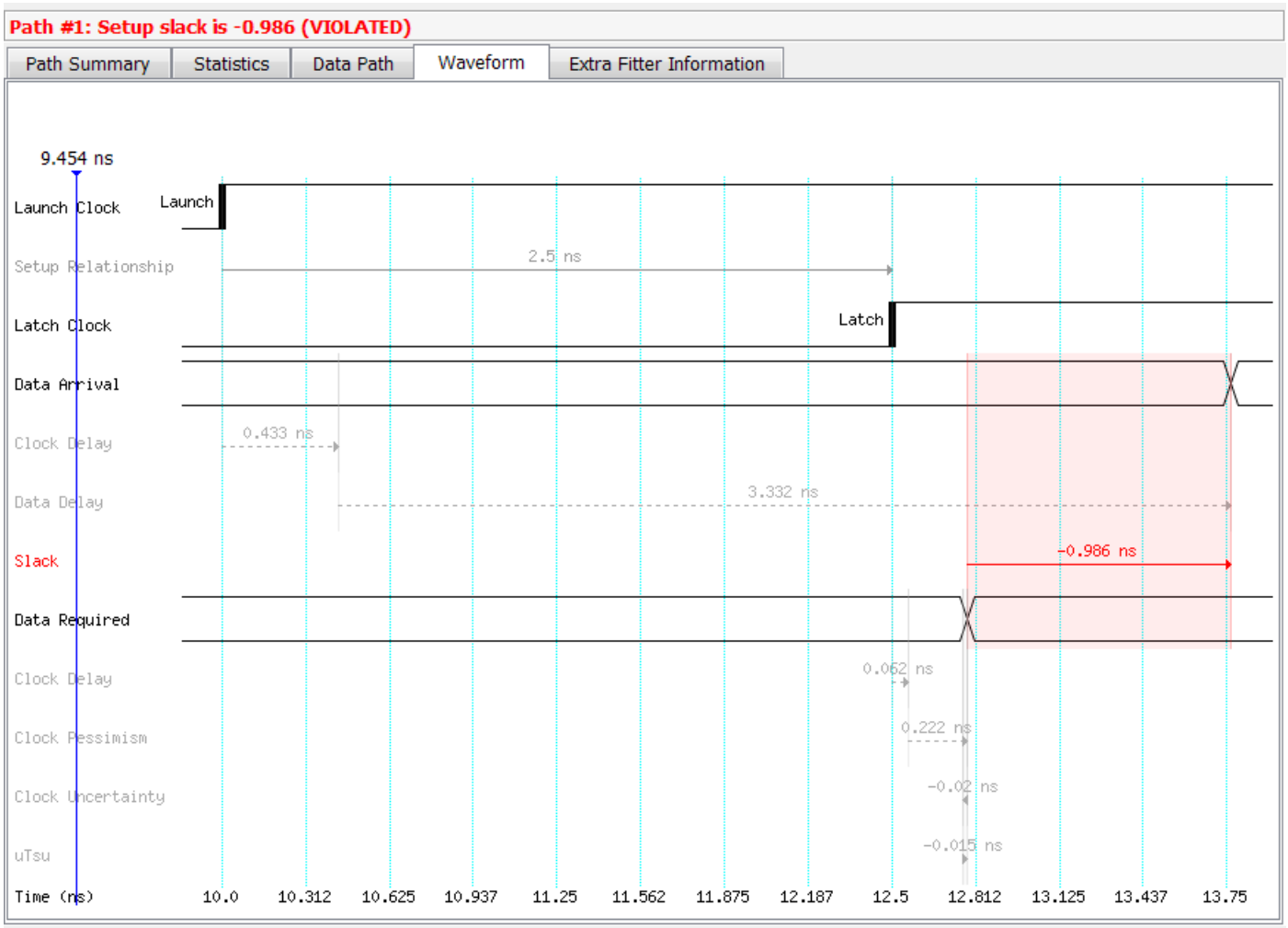
Data Arrival Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	10.000	10.000					launch edge time
2	10.433	0.433					clock path
1	10.000	0.000					source latency
2	10.000	0.000			1	PIN_N7	clk10m
3	10.000	0.000	RR	IC	1	IOIBUF_X16_Y0_N1	clk10m~input|i
4	10.507	0.507	RR	CELL	1	IOIBUF_X16_Y0_N1	clk10m~input|o
5	12.676	2.169	RR	IC	1	PLL_1	pll_inst|altpll_component|auto_generated|pll1|inclk[0]
6	7.091	-5.585	RR	COMP	2	PLL_1	pll_inst|altpll_component|auto_generated|pll1|observablevcoout
7	7.091	0.000	RR	CELL	1	PLL_1	pll_inst|altpll_component|auto_generated|pll1|clk[1]
8	9.008	1.917	RR	IC	1	CLKCTRL_G19	pll_inst|altpll_component|auto_generated|wire_pll1_clk[1]~clkctrl|inclk[0]
9	9.008	0.000	RR	CELL	32	CLKCTRL_G19	pll_inst|altpll_component|auto_generated|wire_pll1_clk[1]~clkctrl|outclk
10	9.914	0.906	RR	IC	1	FF_X24_Y20_N1	cnt32_100m|LPM_COUNTER_component|auto_generated|counter_reg_bit[0]|clk
11	10.433	0.519	RR	CELL	1	FF_X24_Y20_N1	cnt32:cnt32_100m|lpm_counter:LPM_COUNT..._nci:auto_generated|counter_reg_bit[0]
3	13.765	3.332					data path
1	10.632	0.199		uTco	1	FF_X24_Y20_N1	cnt32:cnt32_100m|lpm_counter:LPM_COUNT..._nci:auto_generated|counter_reg_bit[0]
2	10.632	0.000	RR	CELL	3	FF_X24_Y20_N1	cnt32_100m|LPM_COUNTER_component|auto_generated|counter_reg_bit[0]|q
3	11.086	0.454	RR	IC	1	LCCOMB_X23_Y20_N0	comp_inst|LPM_COMPARE_component|auto_generated|op_1~1|datab
4	11.496	0.410	RR	CELL	1	LCCOMB_X23_Y20_N0	comp_inst|LPM_COMPARE_component|auto_generated|op_1~1|cout

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	Element
1	12.500	12.500					latch edge time
2	12.784	0.284					clock path
1	12.500	0.000					source latency
2	12.500	0.000			1	PIN_N7	clk10m
3	12.500	0.000	RR	IC	1	IOIBUF_X16_Y0_N1	clk10m~input|i
4	12.997	0.497	RR	CELL	1	IOIBUF_X16_Y0_N1	clk10m~input|o

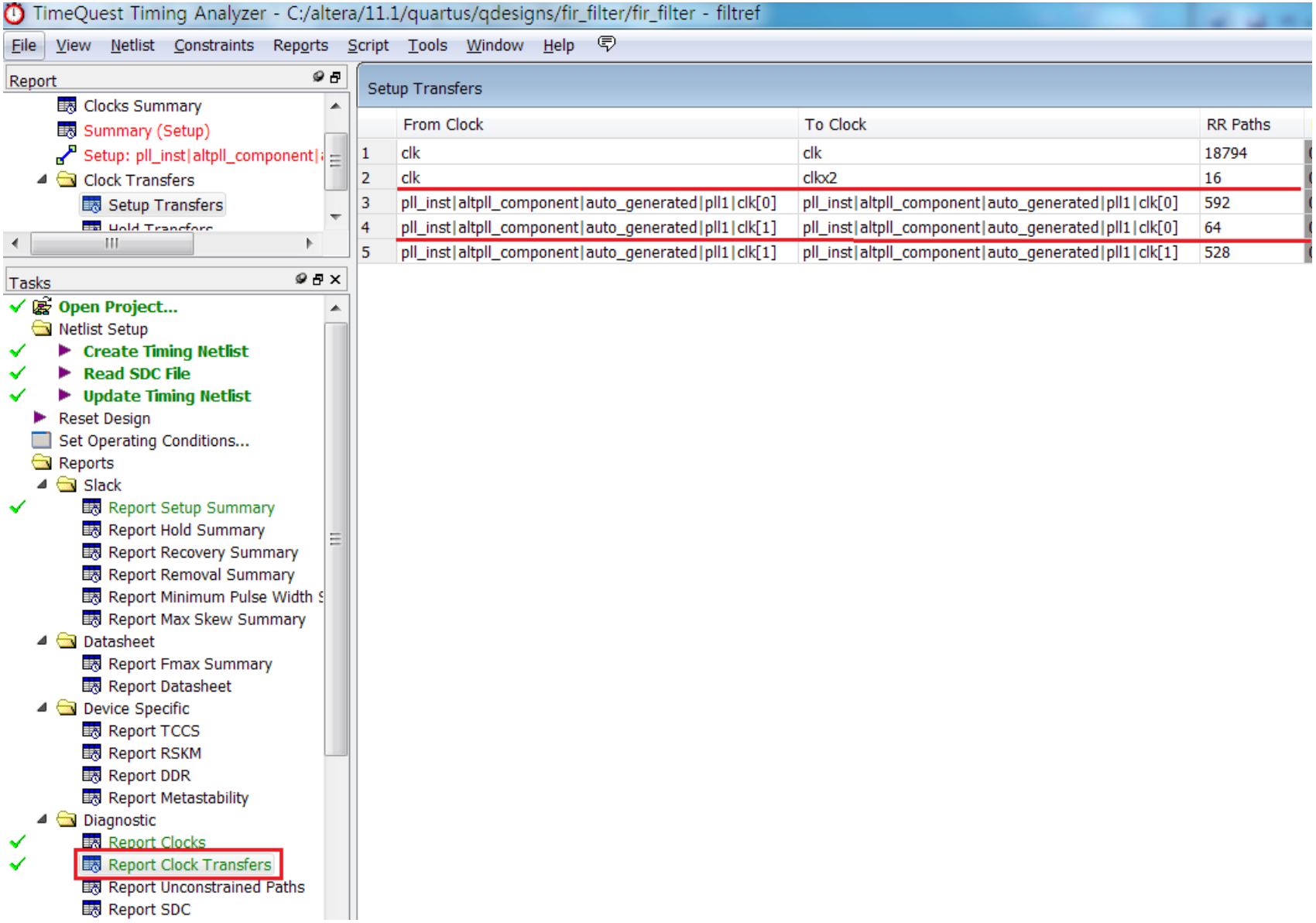
숫자뿐 아니라 waveform 형태로도 timing을 보여줍니다.



Negative slack이 발생한 path를 분석해서 timing margin이 부족한 원인을 찾아 적절하게 design을 수정하거나 적당한 option을 적용하여 negative slack을 줄이고 positive slack을 가질 수 있도록 해야 합니다.

위 같은 path에서는 source register와 destination register에서 사용하는 clock이 서로 달라 clock skew로 인해 negative slack이 발생하는 경우입니다. 이럴 경우 서로 다른 clock끼리 transfer하는 path를 분석하지 않도록 하려면 false path를 적용하면 됩니다.

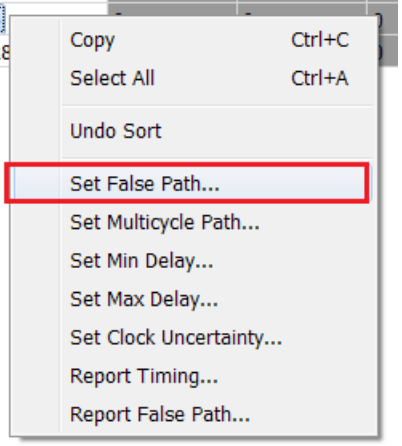
이렇게 서로 다른 clock 간에 data를 주고 받는 것이 있는 지 Diagnostic/Report Clock Transfers를 이용하여 쉽게 확인할 수 있습니다. 이 단계를 수행하면 아래 그림처럼 쉽게 확인할 수 있습니다.



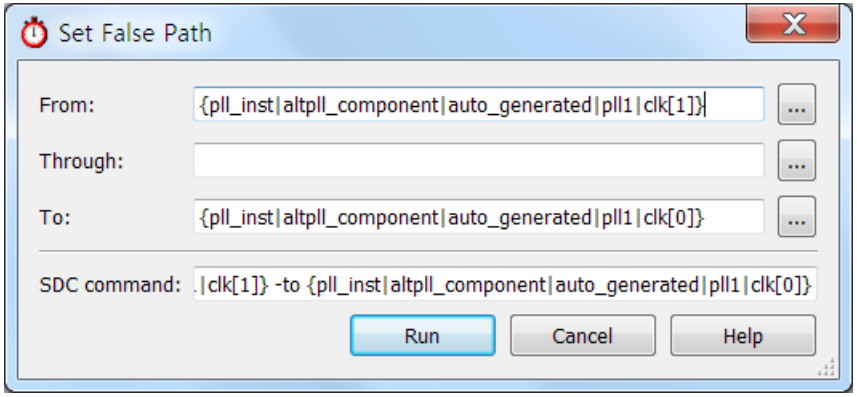
서로 다른 clock간의 path를 분석하지 않으려면 아래처럼 서로 다른 clock끼리 transfer하는 path를 선택하고 오른쪽 마우스를 눌러 Set False Path를 선택합니다.

Setup Transfers

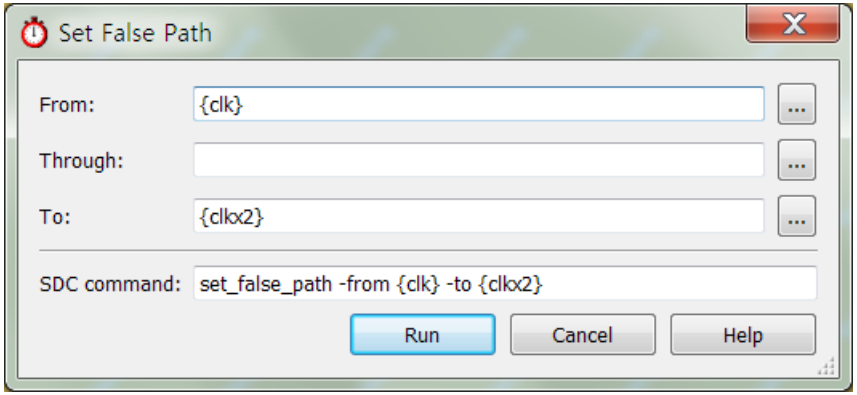
	From Clock	To Clock	RR Paths	FR Paths	RF Paths	FF
1	clk	clk	18794	0	0	0
2	clk	clkx2	16	0	0	0
3	pll_inst|altpll_component|auto_generated|pll1|clk[0]	pll_inst|altpll_component|auto_generated|pll1|clk[0]	592	0	0	0
4	pll_inst|altpll_component|auto_generated|pll1|clk[1]	pll_inst|altpll_component|auto_generated|pll1|clk[0]	64			
5	pll_inst|altpll_component|auto_generated|pll1|clk[1]	pll_inst|altpll_component|auto_generated|pll1|clk[1]	528			



아래와 같은 pop up이 뜨면 Run을 클릭하여 두 clock간의 data path는 분석하지 않겠다는 false path를 적용합니다.



마찬가지로 clk, clkx2사이도 false path를 적용합니다.

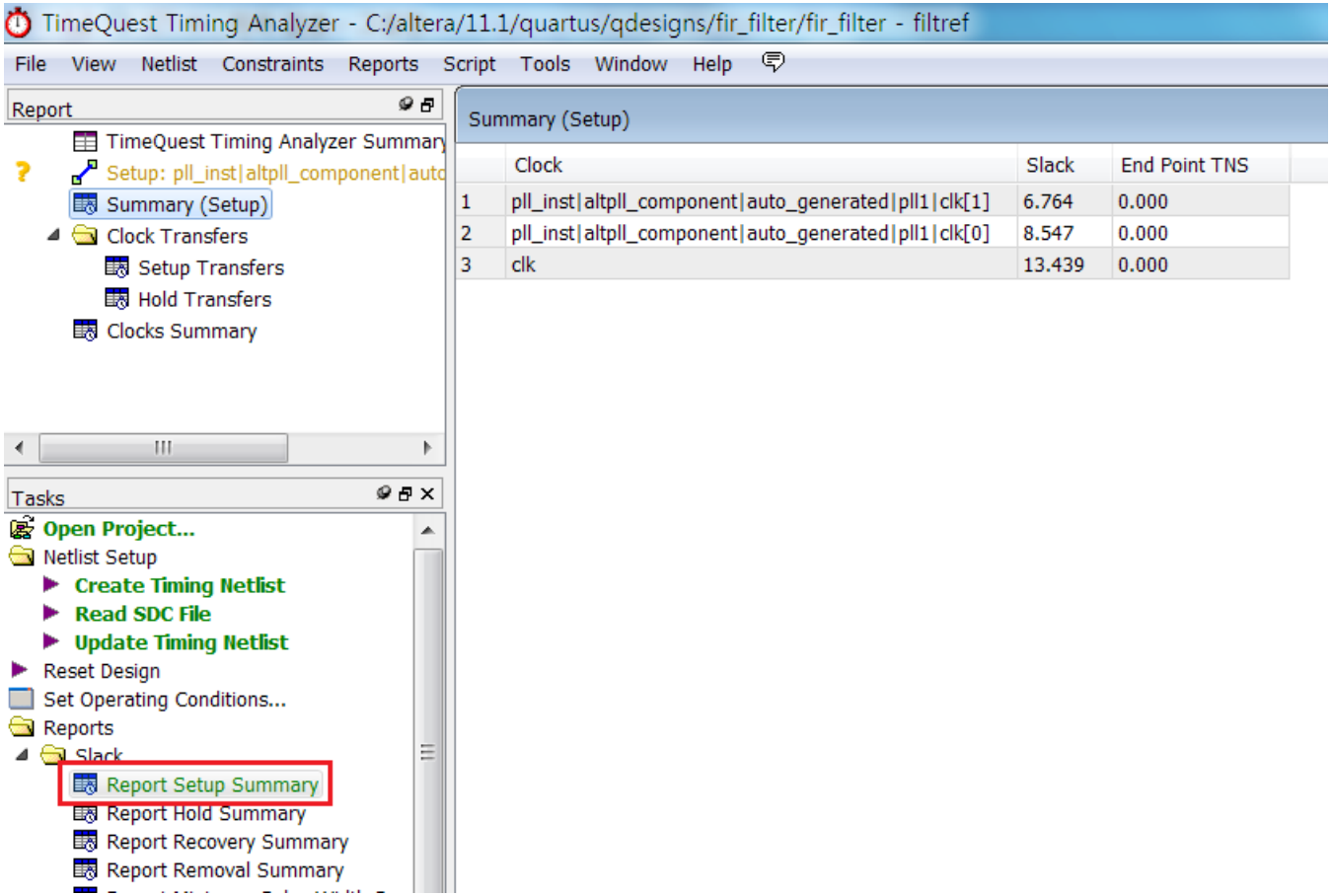


새로운 constraints가 적용되었기 때문에 다시 timing netlist를 update하고 report clock transfers를 클릭하여 아래처럼 false path가 제대로 설정되었는지 확인합니다.

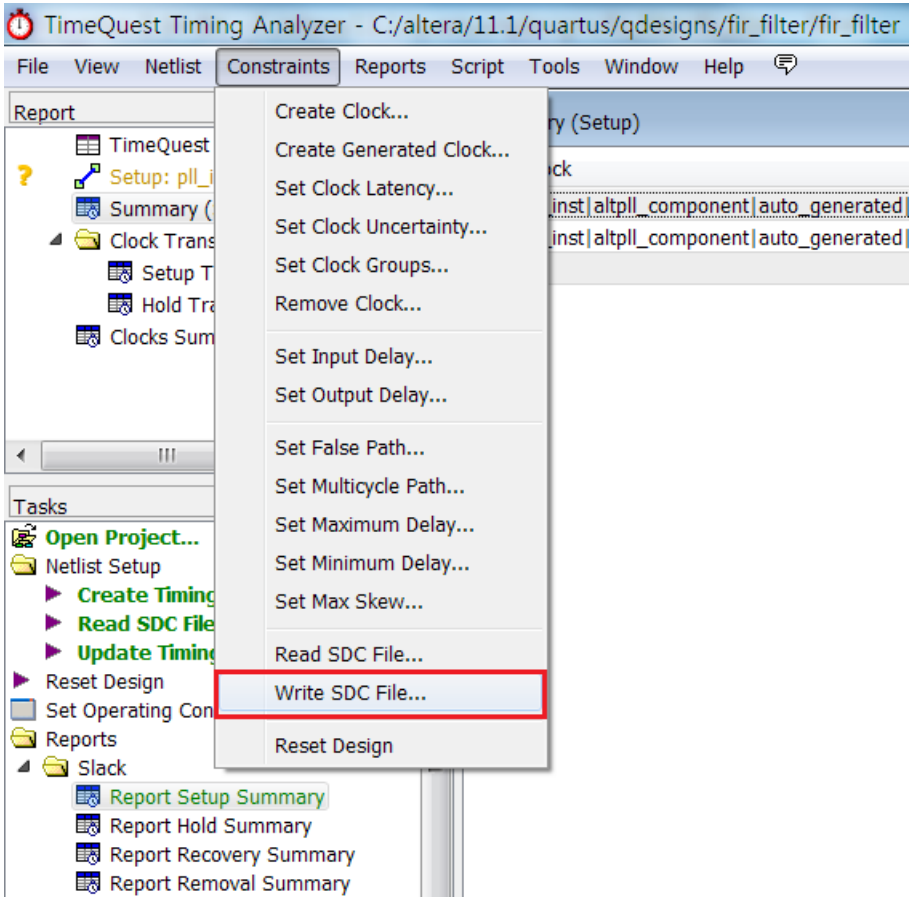
Setup Transfers

	From Clock	To Clock	RR Paths	FR Paths	RF Paths	FF Paths
1	clk	clk	18794	0	0	0
2	clk	clkx2	false path	0	0	0
3	pll_inst|altpll_component|auto_generated|pll1|clk[0]	pll_inst|altpll_component|auto_generated|pll1|clk[0]	592	0	0	0
4	pll_inst|altpll_component|auto_generated|pll1|clk[1]	pll_inst|altpll_component|auto_generated|pll1|clk[0]	false path	0	0	0
5	pll_inst|altpll_component|auto_generated|pll1|clk[1]	pll_inst|altpll_component|auto_generated|pll1|clk[1]	528	0	0	0

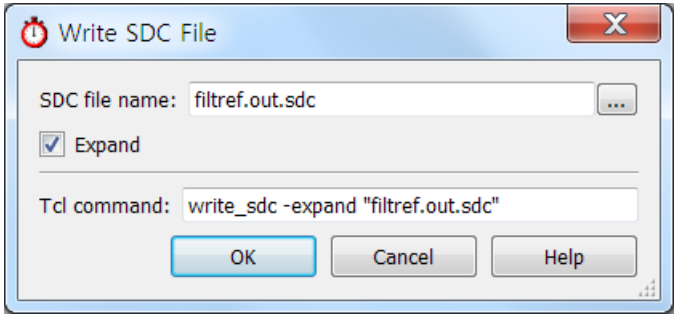
Report Setup Summary를 통해 negative slack이 발생하는 지 확인합니다.



TimeQuest에서 적용한 constraints를 sdc file로 저장하기 위해 Constraints/Write SDC File을 클릭합니다.



SDC file name과 저장할 위치를 지정하고 OK를 클릭합니다.



아래는 생성된 sdc file입니다.

```
## Generated SDC file "filtref.out.sdc"
```

```
## Copyright (C) 1991-2011 Altera Corporation
## Your use of Altera Corporation's design tools, logic functions
## and other software and tools, and its AMPP partner logic
## functions, and any output files from any of the foregoing
## (including device programming or simulation files), and any
## associated documentation or information are expressly subject
## to the terms and conditions of the Altera Program License
## Subscription Agreement, Altera MegaCore Function License
## Agreement, or other applicable license agreement, including,
## without limitation, that your use is for the sole purpose of
## programming logic devices manufactured by Altera and sold by
## Altera or its authorized distributors.  Please refer to the
## applicable agreement for further details.

## VENDOR  "Altera"
## PROGRAM "Quartus II"
## VERSION "Version 11.1 Build 259 01/25/2012 Service Pack 2.dp7 SJ Full Version"

## DATE    "Mon Jun 04 17:25:04 2012"

##
## DEVICE  "EP4CGX15BF14C6"
##

#**************************************************************
# Time Information
#**************************************************************

set_time_format -unit ns -decimal_places 3

#**************************************************************
# Create Clock
#**************************************************************

create_clock -name {clk10m} -period 100.000 -waveform { 0.000 50.000 } [get_ports {clk10m}]
create_clock -name {clk} -period 20.000 -waveform { 0.000 10.000 } [get_ports {clk}]
```

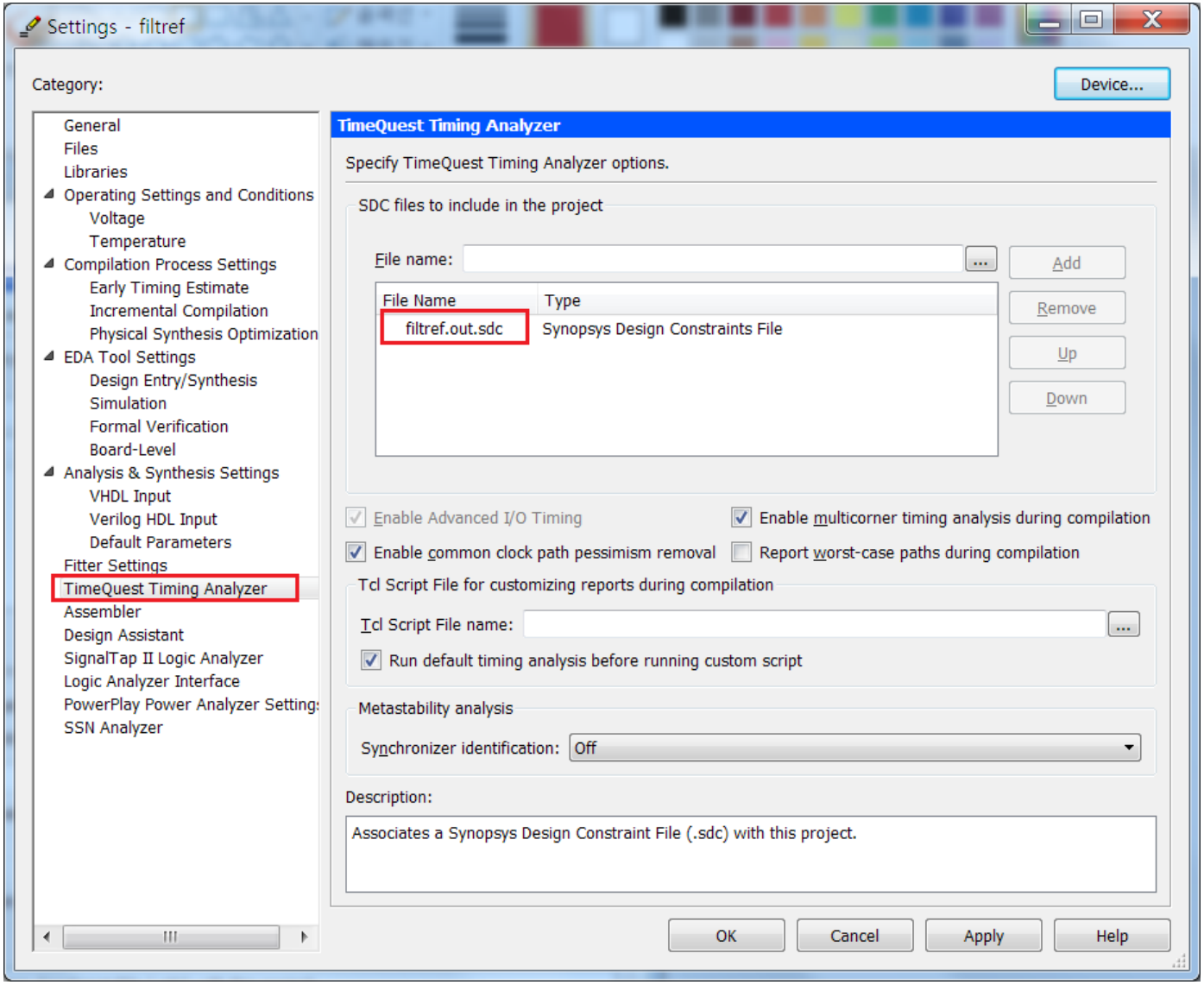
```
create_clock -name {clkx2} -period 10.000 -waveform { 0.000 5.000 } [get_ports {clkx2}]

#**************************************************************
# Create Generated Clock
#**************************************************************

create_generated_clock -name {pll_inst|altpll_component|auto_generated|pll1|clk[0]} -source [get_pins {pll_inst|altpll_component|auto_generated|pll1|inclk[0]}] -duty_cycle 50.000 -multiply_by 8 -master_clock {clk10m} [get_pins {pll_inst|altpll_component|auto_generated|pll1|clk[0]}]
create_generated_clock -name {pll_inst|altpll_component|auto_generated|pll1|clk[1]} -source [get_pins {pll_inst|altpll_component|auto_generated|pll1|inclk[0]}] -duty_cycle 50.000 -multiply_by 10 -master_clock {clk10m} [get_pins {pll_inst|altpll_component|auto_generated|pll1|clk[1]}]

#**************************************************************
# Set Clock Latency
#**************************************************************

#**************************************************************
# Set Clock Uncertainty
#**************************************************************

set_clock_uncertainty -rise_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}] -rise_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}]  0.020
set_clock_uncertainty -rise_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}] -fall_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}]  0.020
set_clock_uncertainty -fall_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}] -rise_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}]  0.020
set_clock_uncertainty -fall_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}] -fall_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}]  0.020
set_clock_uncertainty -rise_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}] -rise_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}]  0.020
set_clock_uncertainty -rise_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}] -fall_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}]  0.020
set_clock_uncertainty -rise_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}] -rise_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}]  0.020
set_clock_uncertainty -rise_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}] -fall_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}]  0.020
set_clock_uncertainty -fall_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}] -rise_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}]  0.020
set_clock_uncertainty -fall_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}] -fall_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}]  0.020
set_clock_uncertainty -fall_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}] -rise_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}]  0.020
set_clock_uncertainty -fall_from [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}] -fall_to [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}]  0.020
set_clock_uncertainty -rise_from [get_clocks {clk}] -rise_to [get_clocks {clk}]  0.020
set_clock_uncertainty -rise_from [get_clocks {clk}] -fall_to [get_clocks {clk}]  0.020
set_clock_uncertainty -rise_from [get_clocks {clk}] -rise_to [get_clocks {clkx2}]  0.040
set_clock_uncertainty -rise_from [get_clocks {clk}] -fall_to [get_clocks {clkx2}]  0.040
set_clock_uncertainty -fall_from [get_clocks {clk}] -rise_to [get_clocks {clk}]  0.020
set_clock_uncertainty -fall_from [get_clocks {clk}] -fall_to [get_clocks {clk}]  0.020
```

```
set_clock_uncertainty -fall_from [get_clocks {clk}] -rise_to [get_clocks {clkx2}]    0.040
set_clock_uncertainty -fall_from [get_clocks {clk}] -fall_to [get_clocks {clkx2}]    0.040

#**************************************************************
# Set Input Delay
#**************************************************************

#**************************************************************
# Set Output Delay
#**************************************************************

#**************************************************************
# Set Clock Groups
#**************************************************************

#**************************************************************
# Set False Path
#**************************************************************

set_false_path    -from    [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[1]}]    -to    [get_clocks {pll_inst|altpll_component|auto_generated|pll1|clk[0]}]
set_false_path    -from    [get_clocks {clk}]    -to    [get_clocks {clkx2}]

#**************************************************************
# Set Multicycle Path
#**************************************************************

#**************************************************************
# Set Maximum Delay
#**************************************************************

#**************************************************************
# Set Minimum Delay
#**************************************************************

#**************************************************************
# Set Input Transition
#**************************************************************
```

3. Re-compile full design at Quartus II with sdc files.

생성된 sdc file을 Quartus project에서 constraints file로 지정합니다.

sdc file은 여러 개를 지정할 수 있습니다. 추가되는 constraints가 있는 경우 하나의 sdc file에 update할 수도 있고 sdc file을 따로 만들어 추가해도 됩니다.



4. Re-analyze timing at TimeQuest

Quartus에서 다시 Full compilation을 합니다. compilation이 완료된 후 Timing analyzer에서 negative slack이 있는지 확인합니다.

만약 negative slack이 발생한 경우 TimeQuest를 열어 어떤 path에서 timing violation이 나는지 분석하고 negative slack을 없애기 위해 sdc를 수정하거나 Quartus에서 option을 적용하거나 design을 변경하는 등의 작업을 반복 진행하면 됩니다.