Lab Manual

UWM EE457
Static Timing Analysis

<u>**Software requirements to complete all exercises**</u>

Software Requirements: Quartus® Prime software version 16.0 or higher

(Standard or Lite)

Link to the Quartus Prime Handbook:
http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf

Exercise 1

Introduction to the TimeQuest Tool

Exercise 1

Objective:

- *Given an existing* ***.sdc*** *file, follow the TimeQuest flow to generate timing reports*
- *Learn about using the TimeQuest interface*

As you proceed through the exercises, be sure to completely read the instructions for each step and sub-step in this lab manual. Each step first summarizes what you'll be doing in that step before providing complete instructions. Use the lines next to each step (____) to keep track of your progress or to check off completed steps in the exercises.

If you have any questions or problems, please ask the instructor for assistance.

Step 1: Open and synthesize a project

To get started with using TimeQuest timing analysis, you will open a simple project and configure it to use an existing ***.sdc*** *file with the TimeQuest timing analyzer.*

____ 1. Unzip the lab project files.

____ 2. Start the Quartus Prime software, version 16.0, from the **Start** menu

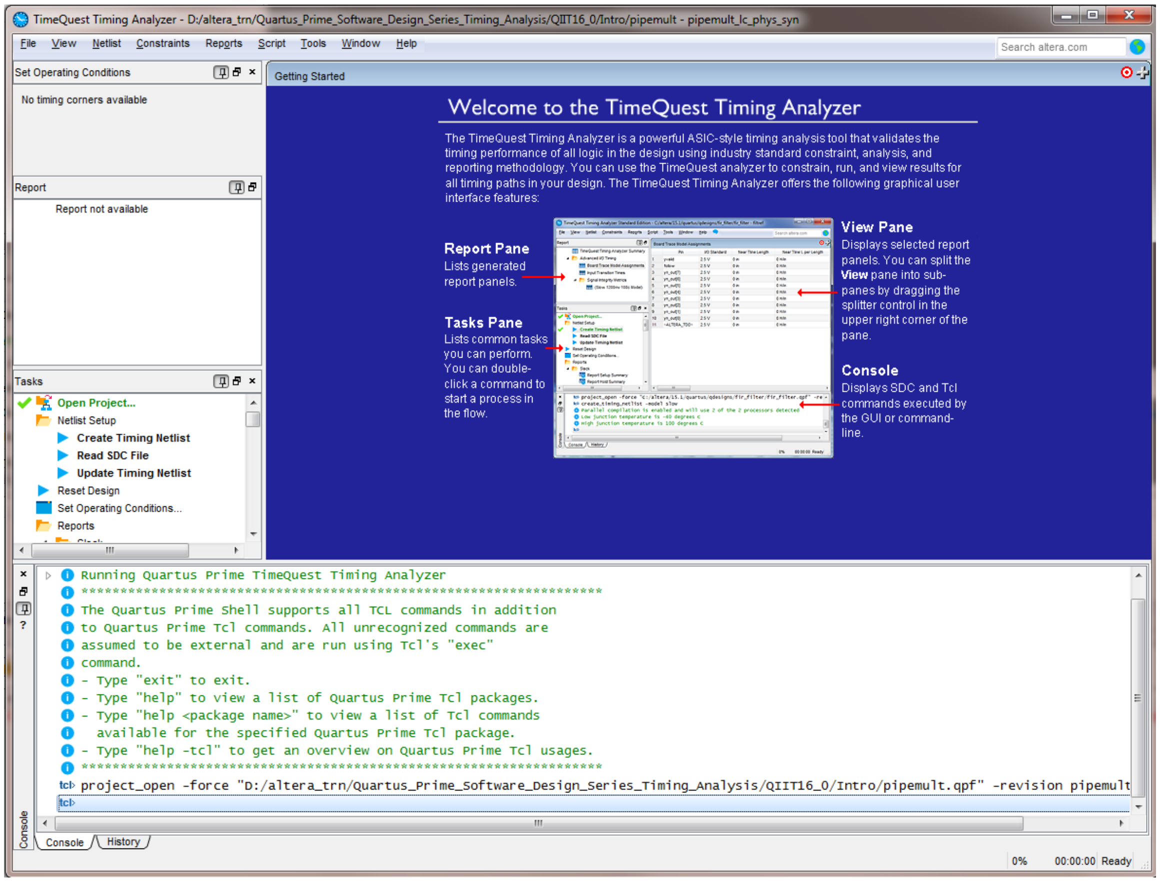
____ 3. **Open** the project **pipemult.qpf** located in the **<lab_install_directory>\Intro** directory. Remember to use the **Open Project** command from the **File** menu instead of the **Open** command (which is used for opening individual files instead of entire projects).

____ 4. Click ▶ to analyze and synthesize the design. Click **OK** when complete.

Though you could also perform a full compilation since this design is complete, you're going to follow the TimeQuest flow as if you were working on a new design that requires a long place-and-route. Performing just synthesis allows you to quickly generate a netlist in order to start constraining the design.

Step 2: Start the TimeQuest GUI and create timing netlist for analysis

____ 1. From the main Quartus Prime toolbar, click or, from the **Tools** menu, select **TimeQuest Timing Analyzer**. You can also access the TimeQuest interface through the **Compile Design** category in the **Tasks** window.


TimeQuest Timing Analyzer - D:/altera_trn/Quartus_Prime_Software_Design_Series_Timing_Analysis/QIIT16_0/Intro/pipemult - pipemult_lc_phys_syn
File View Netlist Constraints Reports Script Tools Window Help
Search altera.com
Set Operating Conditions
No timing corners available
Report
Report not available
Tasks
Open Project...
Netlist Setup
Create Timing Netlist
Read SDC File
Update Timing Netlist
Reset Design
Set Operating Conditions...
Reports
Getting Started
Console
History
0%
00:00:00
Ready

Welcome to the TimeQuest Timing Analyzer

The TimeQuest Timing Analyzer is a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in the design using industry standard constraint, analysis, and reporting methodology. You can use the TimeQuest analyzer to constrain, run, and view results for all timing paths in your design. The TimeQuest Timing Analyzer offers the following graphical user interface features:

Report Pane
Lists generated report panels.

Tasks Pane
Lists common tasks you can perform. You can double-click a command to start a process in the flow.

View Pane
Displays selected report panels. You can split the **View** pane into sub-panes by dragging the splitter control in the upper right corner of the pane.

Console
Displays SDC and Tcl commands executed by the GUI or command-line.

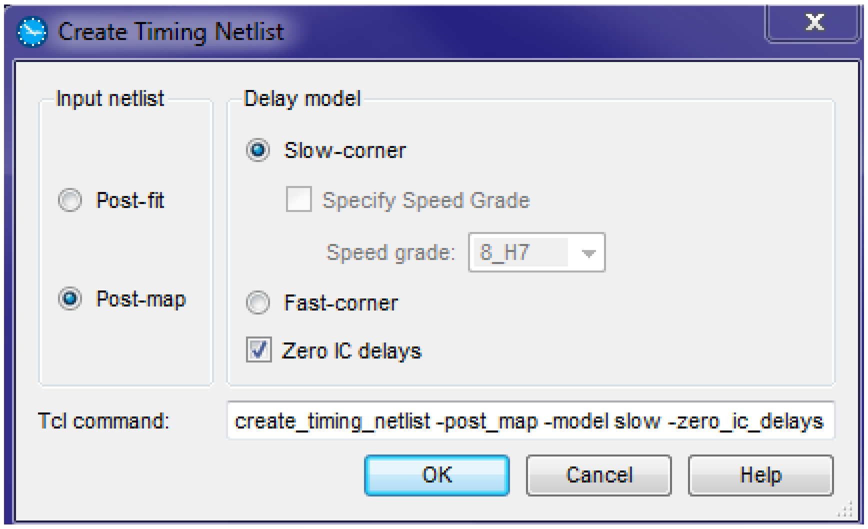
```
Running Quartus Prime TimeQuest Timing Analyzer
********************************************************************
The Quartus Prime Shell supports all TCL commands in addition
to Quartus Prime Tcl commands. All unrecognized commands are
assumed to be external and are run using Tcl's "exec"
command.
- Type "exit" to exit.
- Type "help" to view a list of Quartus Prime Tcl packages.
- Type "help <package name>" to view a list of Tcl commands
  available for the specified Quartus Prime Tcl package.
- Type "help -tcl" to get an overview on Quartus Prime Tcl usages.
********************************************************************
tcl> project_open -force "D:/altera_trn/Quartus_Prime_Software_Design_Series_Timing_Analysis/QIIT16_0/Intro/pipemult.qpf" -revision pipemult
tcl>
```

You will now go through the steps to use TimeQuest timing analysis to constrain a design and verify timing.

____ 2. Create a timing netlist.

a. From the **Netlist** menu, select **Create Timing Netlist**.

b. In the dialog box, change the **Input netlist** type to **Post-map** <u>OR</u> in the **Tcl command** field, add the `-post_map` option to the command.

c. 

Input netlist: ( ) Post-fit [✓ Post-map]

Delay model: [✓ Slow-corner]

☐ Specify Speed Grade

Speed grade: 8_H7

( ) Fast-corner

☑ Zero IC delays

Tcl command:

```
create_timing_netlist -post_map -model slow -zero_ic_delays
```

d. Click **OK**.

You could have chosen to create a timing netlist for a fast corner device in this dialog box or by using the ***Set Operating Conditions*** *command from the* ***Netlist*** *menu (after creating a netlist based on the default slow or fast corner model). This project uses a Cyclone® V device, there's a 3rd and 4th operating condition based on the temperature inversion Fast and Slow models.*

A green checkmark appears next to ***Create Timing Netlist*** *in the* ***Tasks*** *pane to indicate the command was successful. You could have double-clicked* ***Create Timing Netlist*** *in the* ***Tasks*** *pane. However, that manner of creating the netlist would use the default setting of creating a post-fit netlist, which would not work since the Fitter has not run yet.*

____ 3. Attempt to read in an SDC file by simply typing **read_sdc** at the **tcl>** prompt in the **Console** pane or by double-clicking **Read SDC** in the **Tasks** pane.

A message appears in the Console pane indicating that an SDC file for the current revision of the project has not been found. This is correct. Since you did not specify a filename, the tool automatically looked for any SDC files that were manually added to the project and then looked for an SDC file in the project directory sharing the same name as the current revision ***pipemult_lc_phys_syn****, neither of which exists. There is an SDC file in the project directory that you'll use in the next step, but its name does not match the current revision.*

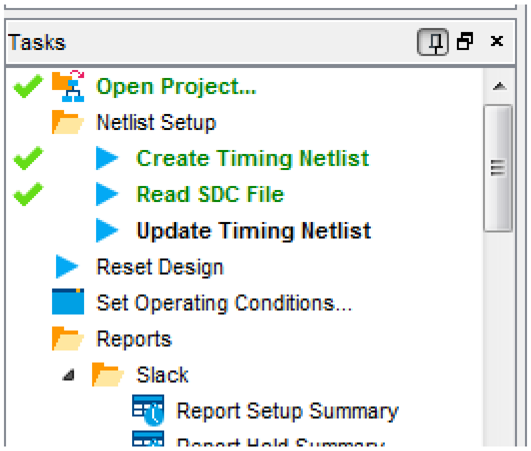
____ 4. Examine and read in an existing SDC file by going to the TimeQuest **File** menu and selecting **Open**. If necessary, change the **Files of type** to **Script files (*.tcl *.qic *.sdc)**. Select and open the **pipemult.sdc** file found in the project directory.

The SDC file opens in an SDC file editor window. Remember that the SDC file editor is the same as the Quartus Prime text editor, so you can create and edit SDC files without opening the TimeQuest interface.

Examine the constraint commands found in the ***pipemult.sdc*** *file. These commands constrain the input clock and the I/O signals in the design. They fully constrain this simple design. We'll talk much more about constraints and creating them later.*

____ 5. From the **Constraints** menu back in the TimeQuest window, select **Read SDC File** and select the **pipemult.sdc** file. The file is read in as the file to use for constraining the design.

There should now be a green checkmark next to ***Read SDC File*** *indicating you read in an SDC file successfully.*


Tasks
Open Project...
Netlist Setup
Create Timing Netlist
Read SDC File
Update Timing Netlist
Reset Design
Set Operating Conditions...
Reports
Slack
Report Setup Summary

____ 6. Update the timing netlist. In the **Tasks** pane of the TimeQuest GUI, double-click **Update Timing Netlist**.

Once the SDC file is added to the project, you can skip ***Read SDC File*** *and just update the timing netlist to automatically read in the file and update the netlist in one step. Or you can even skip both steps and start creating timing reports. The TimeQuest tool will perform all the intermediate actions required. Of course, you could also create a simple Tcl script with all the appropriate commands and run it from the TimeQuest* ***Script*** *menu.*

Step 3: Use TimeQuest reports to verify that design meets timing

Now that the netlist has been updated, you can begin generating various reports. We'll look at reports in more detail later. For now, experiment and explore the different reports you can create.

____ 1. In the **Tasks** pane, double-click **Report SDC** found in the **Diagnostic** folder.

In the ***Report*** *pane, a new folder called* ***SDC Assignments*** *appears containing four reports called* ***Create Clock****,* ***Set Clock Uncertainty****,* ***Set Input Delay****, and* ***Set Output Delay****. These reports list all the SDC constraints read into the tool from the SDC file.*

____ 2. In the **Tasks** pane, double-click **Report Clocks**.

This report verifies that clocks in the design are constrained correctly.

____ 3. In the **Tasks** pane, double-click **Report Setup Summary** found in the **Slack** folder.

This design meets setup timing in both the base and virtual clock domains.

____ 4. In the **Tasks** pane, double-click **Report Hold Summary**.

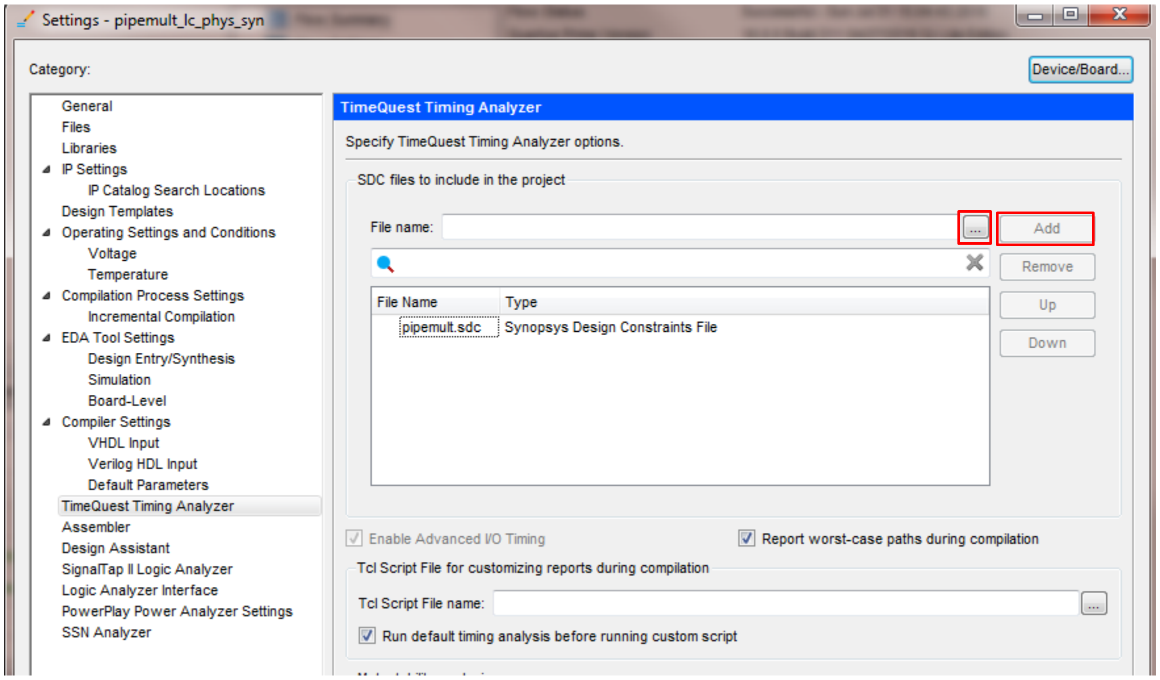
Uh oh. It looks like the design, even with the timing constraints added, just barely fails hold timing, indicated by the clock domain in red and a negative slack value. This is close enough that it may get fixed by the Fitter during placement and routing. For now, ignore the failure. Also note that these reports were based on post-map estimates; the actual post-fit timing numbers will likely be different.

Step 4: Use the SDC file to guide the Quartus Prime Fitter

____ 1. Bring the Quartus Prime software to the foreground.

____ 2. From the **Assignments** menu, select **Settings**.

____ 3. Select the **TimeQuest Timing Analyzer** category on the left if it's not already selected.



TimeQuest Timing Analyzer

Specify TimeQuest Timing Analyzer options.

SDC files to include in the project

File name:

File Name	Type
pipemult.sdc	Synopsys Design Constraints File

☐ Enable Advanced I/O Timing ☑ Report worst-case paths during compilation

Tcl Script File for customizing reports during compilation

Tcl Script File name:

☑ Run default timing analysis before running custom script

____ 4. Add the **pipemult.sdc** file to the project.

a. Use the browse button [...] to locate the file **pipemult.sdc**.

b. Click **Open** (within the new GUI that opens).

c. Click **Add**.

*When you do this for your own projects, don't forget to click **Add**!*

____ 5. If it is not already enabled, turn on **Report worst-case paths during compilation**. The settings should match the picture shown above.

____ 6. Click **OK** to close the **Settings** dialog box.

____ 7. Click ▶ or select **Start Compilation** from the **Processing** menu.

____ 8. Click **OK** when the compilation is complete.

____ 9. If the Compilation Report did not automatically appear, open it by clicking from the Quartus Prime toolbar.

____ 10. Expand the **TimeQuest Timing Analyzer** folder in the **Compilation Report**.

Are you meeting or missing timing? A quick glance of the summary reports for each of the three or four timing models will tell you. Make sure to check the fast and slow reports and for each temperature. You can check all three or four models at once with the ***Multicorner Timing Analysis Summary*** *report. Are any of the reports shown in red? If not, then timing has been validated. You do not need to check any further.*

At this point, you could generate more advanced reports and get more detail about specific paths in the design. We'll talk about how to generate these other more detailed reports in the next section.

Summary

- *Practiced basic steps for using the TimeQuest timing analyzer with the Quartus Prime*

END OF EXERCISE 1 – Zip up directory and upload to Canvas as Lab10 part a

Exercise 2

Timing Analysis: Clock Constraints

Exercise 2

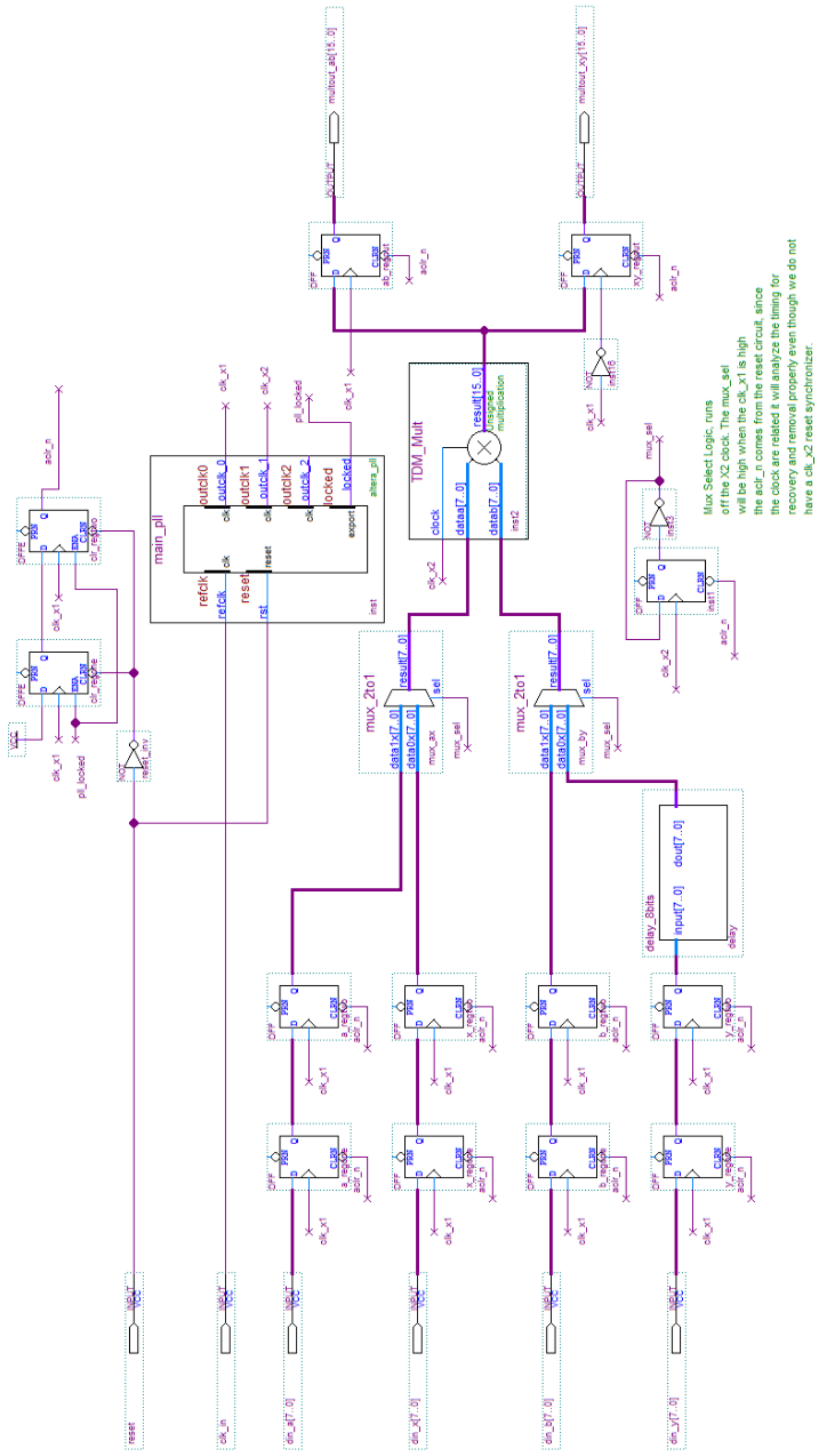
Objectives:

- *Create a new SDC file*
- *Use SDC to constrain the clocks in a design containing PLLs*

Top-Level Design:

The design used for the rest of the exercises in this training (shown on the following page) multiplies two sets of 8-bit data inputs: ***din_a*** * ***din_b*** and ***din_x*** * ***din_y***. Along with this data input, the design also receives a board clock named ***clk_in*** running at 100 MHz (clock period of 10 ns) and an asynchronous reset named ***reset***. The board clock also clocks the external devices that both drive data into the FPGA on its inputs and capture data on the outputs. To save on multiplier resource usage, the data is time-domain multiplexed (TDM) through a single multiplier running at twice the clock speed (200 MHz). The clocks for the design are generated by a PLL called ***main_pll*** with 2 output clocks. A 100 MHz PLL output called ***clk_x1*** is used to reduce clock tree delay to internal registers. A 200 MHz PLL output called ***clk_x2*** drives the multiplier at twice the input frequency. The resulting data outputs named ***multout_ab*** and ***multout_xy*** are sent off-chip to another device on the board. The Mux's that are used to control the data into the multipliers are using a divide by 2 circuit clocked by the 200Mhz clock, creating an enable signal that essentially operates at 100Mhz.

NOTE: Throughout the remaining exercises, you will be asked to create SDC commands without being directly guided by the instructions. It is up to you to figure out the correct commands using the training presentation material as well as the GUI tools and help information available in the TimeQuest timing analyzer. If you are stuck or do not understand a solution command, please ask your instructor for assistance.



Mux Select Logic, runs off the X2 clock. The mux_sel will be high when the clk_x1 is high the aclr_n comes from the reset circuit, since the clock are related it will analyze the timing for recovery and removal properly even though we do not have a clk_x2 reset synchronizer.

Step 1: Create SDC file for the design

In this step, you will open the project and locate and constrain all the design clocks.

____ 1. From the Quartus Prime software, use **Open Project…** to open the project **top.qpf** located in the **<lab_install_directory>\Timing** directory.

____ 2. Open the file **top.bdf** by double-clicking **top** in the Project Navigator or through the **Open** command in the **File** menu.

You should see the schematic from the previous page of this exercise manual.

____ 3. Synthesize the design by clicking the button or, from the **Processing** menu, go to **Start** and select **Start Analysis & Synthesis**. Click **OK** when finished if a pop up window opens.

Remember, timing constraints are used to guide the Fitter during place and route. Since we're only at the point of adding constraints to the design, there is no need to perform a full compilation because we'd have to run the Fitter again anyway. So by just synthesizing the design and generating a post-synthesis timing netlist, you can begin constraining the design much sooner.

____ 4. Open the TimeQuest timing analyzer by clicking the button or by going to the **Tools** menu and selecting **TimeQuest Timing Analyzer**.

____ 5. Create a slow model timing netlist.

a. In the TimeQuest tool, from the **Netlist** menu, select **Create Timing Netlist..**.

b. In the dialog box, change the **Input netlist** type to **Post-map** OR in the **Tcl command** field, add the `-post_map` option to the command.

*Remember, you must use the **-post_map** netlist type because you only performed synthesis (no fitting) and you cannot access the **-post_map** option from the **Create Timing Netlist** task in the **Tasks** pane.*

____ 6. From the Timequest Window use the **File** menu, select **New SDC File**.

This will open a new text editor window (it may open detached from the Quartus Prime GUI, the Attach/Detach Icon is).

____ 7. From the Text Editor window (or from the Quartus Prime Window if the editor is re-attached to Quaruts Prime GUI using) select the **File** menu, select **Save As** and save the file as **<lab_install_directory>\Timing\top.sdc**.

*Notice that an option, **Add file to current project**, is enabled in the **Save As** dialog box.*

Step 2: Create base and generated clock constraints

Next, as with many other steps in this exercise, you will be adding a constraint to the ***top.sdc*** *file. You will only be told what to constrain and given values. It will be up to you to use the proper SDC command.*

Feel free to type the SDC command directly into the ***SDC File Editor*** *or use the* ***Insert Constraint*** *submenu of the SDC File Editor's* ***Edit*** *menu, whichever you feel more comfortable with. It is recommended that you enter all constraints in the SDC File Editor instead of trying to enter constraints directly into the* ***Console*** *pane. This makes it much easier to create, edit, and manage your constraints.*

____ 1. In the **top.sdc** file, assign a **10 ns** clock to the input port **clk_in**. Use the default name **clk_in** as the name of the clock.

Access the constraint GUI dialog boxes from the the ***Insert Constraint*** *submenu of the* ***Edit*** *menu. The GUI will place the constraint at the current location of the cursor, so remember to create new lines before accessing the dialog boxes. The GUI constraint dialog boxes can be used for almost all the constraints you'll be creating. However, remember from the presentation that there are some constraint options and arguments that are not in the GUI. You can always refer back to the presentation slides to find a constraint option that you may need.*

____ 2. On the line above the **create_clock** command you've just created, insert a comment using # to indicate what the succeeding SDC command is doing. This is just good coding practice, and it may be helpful if you need to go back and review the constraints you've entered.

____ 3. In the **top.sdc** file, create a 10 ns input virtual clock, named **clk_in_vir**.

This clock drives the upstream device that drives the din inputs to the design.

____ 4. Create another 10 ns output virtual clock named **clk_out_vir**.

This clock drives the downstream device that is fed by the outputs of the design.

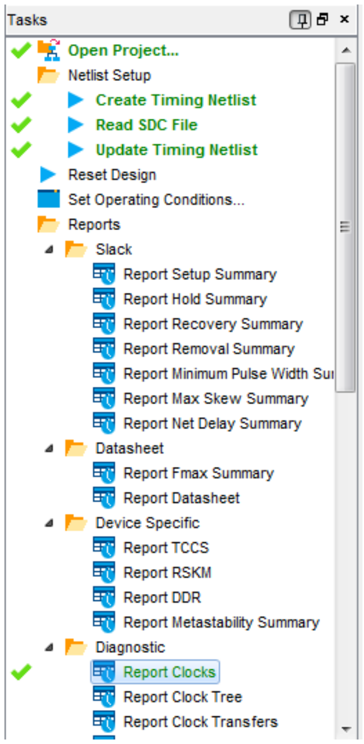
____ 5. In **top.sdc**, add the command to <u>automatically</u> create generated clocks on all of the PLL outputs based on your previously specified clock input.

A ***hint*** *for an even shorter shortcut: Instead of using the* ***create_clock*** *command (as instructed in #1 above) to create your base clock, you could use an optional argument with the command used in this step. For this lab, do not use the –create_base_clocks on the derive_pll_clocks command.*

Note: Order of constraints matter in the SDC file; for the derive_pll_clocks constraint to work properly, the create_clock constraints that creates the clock that that feeds the PLL must appear prior the derive_pll_clocks constraint.

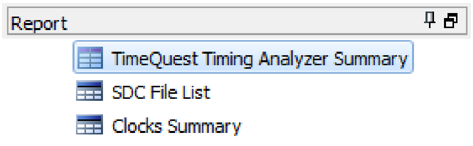
____ 6. **Save** the **top.sdc** file.

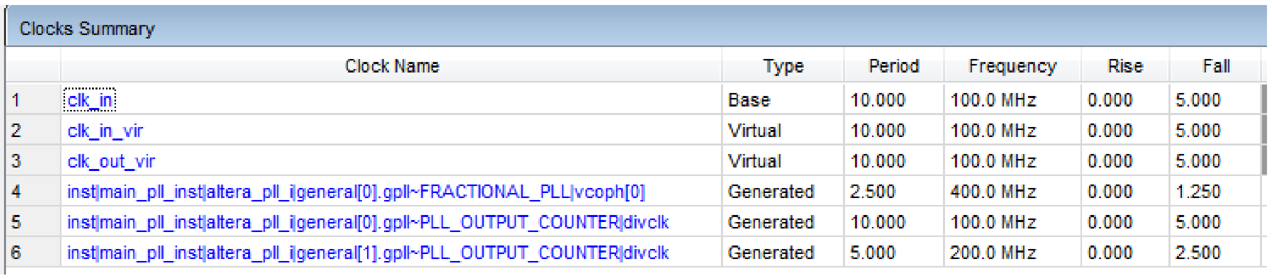
____ 7. In the TimeQuest window, generate a clock report by double-clicking **Report Clocks**.



*Typically, you would <u>read in your SDC file</u> and then <u>update the timing netlist</u>. The TimeQuest timing analyzer will automatically perform these tasks if you request a report via the **Tasks** pane and if your SDC file is the same name as your Quartus Prime revision or you have added the SDC file to your Quartus Prime project.*

If you have errors in your SDC file and get a message such as "read_sdc failed due to errors in the SDC file", scroll up in the window to see the error details.




Clocks Summary

	Clock Name	Type	Period	Frequency	Rise	Fall
1	clk_in	Base	10.000	100.0 MHz	0.000	5.000
2	clk_in_vir	Virtual	10.000	100.0 MHz	0.000	5.000
3	clk_out_vir	Virtual	10.000	100.0 MHz	0.000	5.000
4	inst|main_pll_inst|altera_pll_i|general[0].gpll~FRACTIONAL_PLL|vcoph[0]	Generated	2.500	400.0 MHz	0.000	1.250
5	inst|main_pll_inst|altera_pll_i|general[0].gpll~PLL_OUTPUT_COUNTER|divclk	Generated	10.000	100.0 MHz	0.000	5.000
6	inst|main_pll_inst|altera_pll_i|general[1].gpll~PLL_OUTPUT_COUNTER|divclk	Generated	5.000	200.0 MHz	0.000	2.500

*In the **Report** pane, new reports appear. Use the **SDC File List** report to verify which SDC file(s) have been read into the tool and if they have been read in correctly. Use the **Clocks Summary** report to verify that your clocks have been created correctly. As shown above, you should see one base clock called **clk_in**, the two virtual clocks,* *and three generated clocks, one for each output of the PLL (100Mhz and 200Mhz) and one for the Voltage Controlled Oscillator (VCO) of the PLL. Notice their rise and fall times. Write out an SDC file called **test.sdc** based on the current SDC assignments applied to the analysis.*

a. From the **Constraints** menu, select **Write SDC File**.

b. Name the file **test.sdc** and make sure the **Expand** option is turned on.

*You could also do this in the **Console** pane by typing:*

```
write_sdc -expand test.sdc
```

____ 8. Open the file **test.sdc** by going to the **File** menu and selecting **Open**.

____ 9. Adjust the **Files of type**, if needed, to **Script Files**, and select **test.sdc**.

*Notice in **test.sdc** that the individual generated clock commands have been expanded out, replacing the **derive_pll_clocks** command.*

____ 10. Reopen **top.sdc** if you closed it.

____ 11. Copy the 3 `create_generated_clock` commands from **test.sdc** into **top.sdc**. If you'd like, split the long commands into multiple lines using backslashes (\).

____ 12. In **top.sdc**, <u>**comment out**</u> the `derive_pll_clocks` command using "#".

*Note that commenting out the SDC extension is not necessary. You could simply keep the **derive_pll_clocks** command in your SDC file. The advantage of keeping the command is that whenever the PLL settings are changed, the generated clock commands are updated automatically. However, you can't rename the generated clocks to something more meaningful, which is what you will do next. Also, **derive_pll_clocks** is not a standard SDC command and will typically not be recognized by other tools that support SDC.*

____ 13. Use the table below to change 2 of the PLL generated output clock names to more easily recognized names. For each generated clock command in **top.sdc**, change the name of the clock (the ***–name*** argument <u>ONLY</u>) from the name in the left column of the table to the name in the right column, make sure you do not delete the curley brackets {}. Do not modify the generated clock for the PLL's VCO clock, i.e. you are only editing 2 of the 3 generated clocks constraints

Change `-name` argument from:	**To:**
inst|main_pll_inst|altera_pll_i| general[0].gpll~PLL_OUTPUT_COUNTER|divclk	clk_x1
inst|main_pll_inst|altera_pll_i| general[1].gpll~PLL_OUTPUT_COUNTER|divclk	clk_x2

This helps with the readability of later SDC commands and timing reports.

____ 14. In **top.sdc**, add the command to <u>automatically</u> derive clock uncertainties. Save **top.sdc**, and close **test.sdc**.

Step 3: Analyze the clock constrained design

____ 1. Reset the design. To do this, do <u>ONE</u> of the following:

a. From the **Constraints** menu, select **Reset Design**.

b. Double-click **Reset Design** in the **Tasks** pane.

c. In the **Console** pane, type `reset_design`.

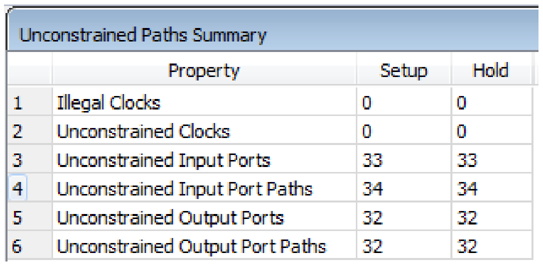
Resetting the design tells the timing analyzer to flush all timing constraints from the netlist, thus allowing you to "start over" with new constraints on the same netlist. It is analogous to deleting the timing netlist and creating a new netlist of the same type.

You should now see that all of your reports have been deleted.

____ 2. Run Report Clocks again.

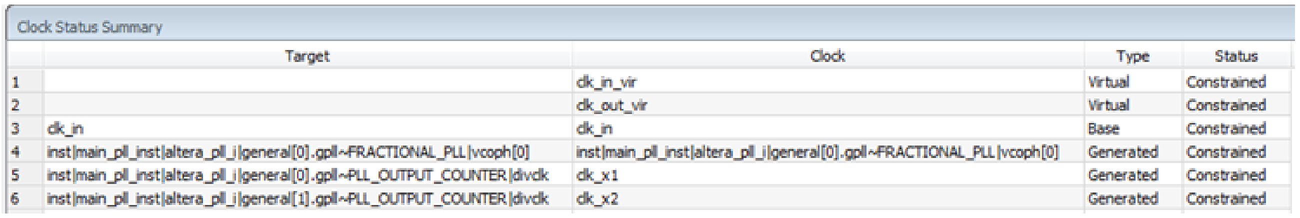
Your clocks should match all the clock names you entered in the SDC file.

____ 3. Check to see that all clocks are constrained. What report should you run?



Unconstrained Paths Summary

	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	0	0
3	Unconstrained Input Ports	33	33
4	Unconstrained Input Port Paths	34	34
5	Unconstrained Output Ports	32	32
6	Unconstrained Output Port Paths	32	32



Clock Status Summary

	Target	Clock	Type	Status
1		clk_in_vir	Virtual	Constrained
2		clk_out_vir	Virtual	Constrained
3	clk_in	clk_in	Base	Constrained
4	inst|main_pll_inst|altera_pll_i|general[0].gpll~FRACTIONAL_PLL|vcoph[0]	inst|main_pll_inst|altera_pll_i|general[0].gpll~FRACTIONAL_PLL|vcoph[0]	Generated	Constrained
5	inst|main_pll_inst|altera_pll_i|general[0].gpll~PLL_OUTPUT_COUNTER|divclk	clk_x1	Generated	Constrained
6	inst|main_pll_inst|altera_pll_i|general[1].gpll~PLL_OUTPUT_COUNTER|divclk	clk_x2	Generated	Constrained

You should find no unconstrained clocks or illegal clocks.

Summary

- *Created a new SDC file*
- *Constrained the input, virtual, and PLL clocks in the design*

END OF EXERCISE 2 – Zip up your directory and upload to Canvas as lab10b

Exercise 3

Timing Analysis: Synchronous I/O Constraints

Exercise 3

Objective:

- *Constrain the synchronous I/O paths in the design*

Step 1: Constrain synchronous input paths using SDC

Now you will define the delays to external devices on the design's I/O ports in order to make sure the design will be able to meet I/O timing based on the clocks created in the previous lab. You'll start by constraining the inputs.

____ 1. Open **top.sdc** if it's not already open.

*The upstream devices (sending data to **din_a**, **din_b**, **din_x**, and **din_y**) and the board have timing numbers as shown in the following table. Assume the board is being laid out with a star topology clocking scheme. Thus, all devices using the master clock (**clk_in**) should be clocked at (relatively) the same time.*

	Minimum	***Maximum***
Clock-to-output delay of upstream device (ns)	**0.7**	**2.1**
Estimated PCB data trace delay (between devices) (ns)	**0.35**	**0.7**
Board clock skew (ns)	**-0.35**	**0.35**

____ 2. Use this table along with the schematic from Exercise 2 to fully constrain all input data ports with respect to **clk_in_vir**, using these system parameters. This can be done with only 2 commands (`-max` and `-min`) if you use wildcards. To refresh your memory, the equations for doing this are shown below:

Input delay (max) = board delay (max) – clock skew (min) + ext. t_{co} (max)

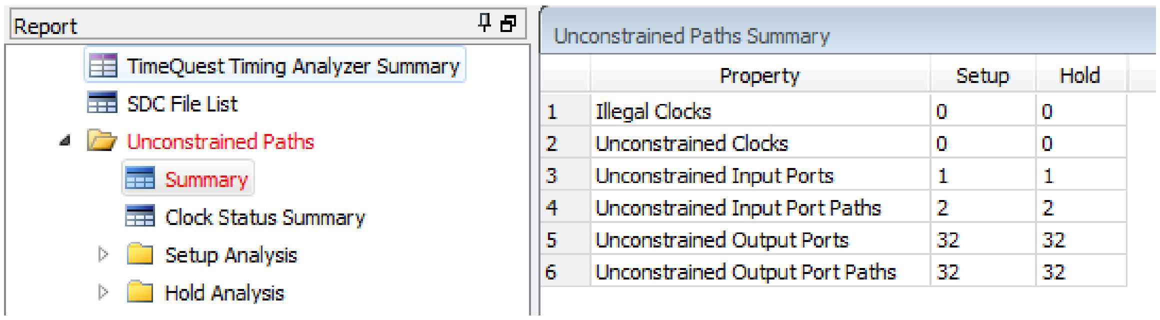
Input delay (min) = board delay (min) – clock skew (max) + ext. t_{co} (min)

*You can either calculate the values yourself and put them in the constraints directly or use the Tcl **expr** command to have the values calculated by the tool.*

____ 3. Save top.sdc.

____ 4. Reset the design and check your newly entered constraints. Use the **Tasks** pane to run reports to check for ignored constraints and for unconstrained paths.

*Use the **Ignored Constraints** report to make sure you didn't type anything incorrectly. Use the **Unconstrained Paths** report to see what hasn't been constrained yet. The goal with constraining is to get the numbers in this report down to zero.*



Unconstrained Paths Summary

	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	0	0
3	Unconstrained Input Ports	1	1
4	Unconstrained Input Port Paths	2	2
5	Unconstrained Output Ports	32	32
6	Unconstrained Output Port Paths	32	32

*At this point, you should have only 1 remaining unconstrained input port and 2 unconstrained input port paths along with 32 unconstrained output ports and paths. If you look at the **Setup Analysis** and **Hold Analysis** reports (under **Unconstrained Paths**), you will see that the unconstrained input port and paths are caused by the reset. You will fix these in exercise 4.*

Step 2: Constrain synchronous output paths using SDC

Now you need to define the FPGA's output timing relative to the external device it drives.

*The downstream devices receive data from **multout_ab** on the **rising** edge of the clock and **multout_xy** on the **falling** edge of the clock. These devices and the board have timing numbers as shown in the following table.*

External device parameters (ns)	**T_h = 1.0**	**T_{su} = 1.0**

	Minimum	***Maximum***
Estimated PCB data trace delay (between devices) (ns)	**0.35**	**0.7**
Board clock skew (ns)	**-0.35**	**0.35**

____ 1. Use this table along with the schematic to fully constrain all <u>output data ports</u> with respect to **clk_out_vir**, using these system parameters. This can be done with 4 commands (2 for **multout_ab** and 2 for **multout_xy**) if you use wildcards. To refresh your memory, the equations for doing this are shown below:

Output delay (max) = board delay (max) – clock skew (<u>min</u>) + t_{su}

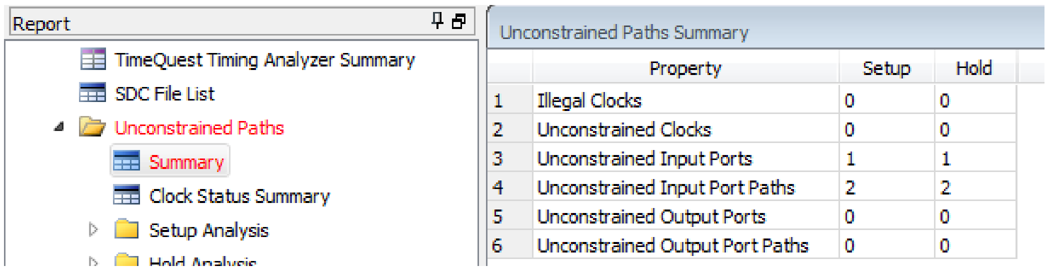
Output delay (min) = board delay (min) – clock skew (<u>max</u>) - t_h

*Remember that if data is latched at the external device by the <u>falling edge of the clock</u> on the output, you need to use the **–clock_fall** argument.*

____ 2. Save **top.sdc**.

____ 3. Reset the design and check your newly entered constraints. Use the Tasks pane to run reports to check for ignored constraints and for unconstrained paths.

*Use the **Ignored Constraints** report to make sure you didn't type anything incorrectly. Use the **Unconstrained Paths** report to see what hasn't been constrained.*



Unconstrained Paths Summary

	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	0	0
3	Unconstrained Input Ports	1	1
4	Unconstrained Input Port Paths	2	2
5	Unconstrained Output Ports	0	0
6	Unconstrained Output Port Paths	0	0

At this point, you should have 1 remaining unconstrained input ports and 2 unconstrained input port paths. The outputs should all be completely constrained. If your Unconstrained Paths Summary report does not match the screenshot above, go back to your SDC file to figure out what's wrong and fix your constraints.

You could now take this SDC file into the Quartus Prime software and use it to compile your design. However, as you can see, there are still some remaining paths to constrain before that would be truly useful. Remember, the idea with constraining is that you know how the design should work. Thus, you need to pass all timing information on to the compiler so it can adjust the fit of your design based on that information. Without complete timing information, the compiled design may not meet your timing requirements.

Summary

- *Constrained the synchronous I/O interfaces of the design*

END OF EXERCISE 3 – Zip up directory and upload to Canvas as lab10c

Exercise 4

Timing Analysis: Timing Exceptions & Analysis

Exercise 4

Objectives:

- *Constrain asynchronous input signals*
- *Eliminate timing violations by using timing exceptions*

Step 1: Constrain the asynchronous path

In this step, you will constrain the asynchronous input path of the design.

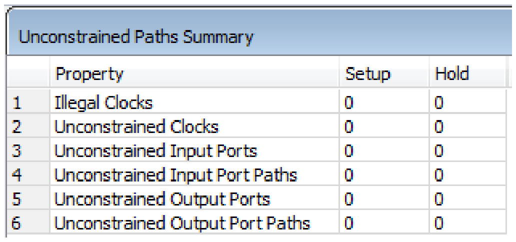
Look at the reset circuit in the design schematic. You can see that the design has an active low reset driven by an external source. Let's assume the input path is truly asynchronous (no timing on the external path), so the solution to synchronize the reset internally is the correct one. The circuit shown is a common method of synchronizing a reset to an internal clock domain. Since this a truly asynchronous input and no external timing is known, a false path exception is needed to constrain the path.

Note the PLL has a "locked" output indicating tha t the PLL is locked, this is tied to the enable input of the first register of the reset synchronizer. The circuit is prevented from coming out of reset until the PLL has achieved locked and the clocks are stable.

____ 1. In **top.sdc**, use an SDC command to constrain all paths **from** the asynchronous reset input.

____ 2. Save **top.sdc**.

____ 3. Reset the design and check your ignored constraints and unconstrained paths reports.


Unconstrained Paths Summary

	Property	Setup	Hold
1	Illegal Clocks	0	0
2	Unconstrained Clocks	0	0
3	Unconstrained Input Ports	0	0
4	Unconstrained Input Port Paths	0	0
5	Unconstrained Output Ports	0	0
6	Unconstrained Output Port Paths	0	0

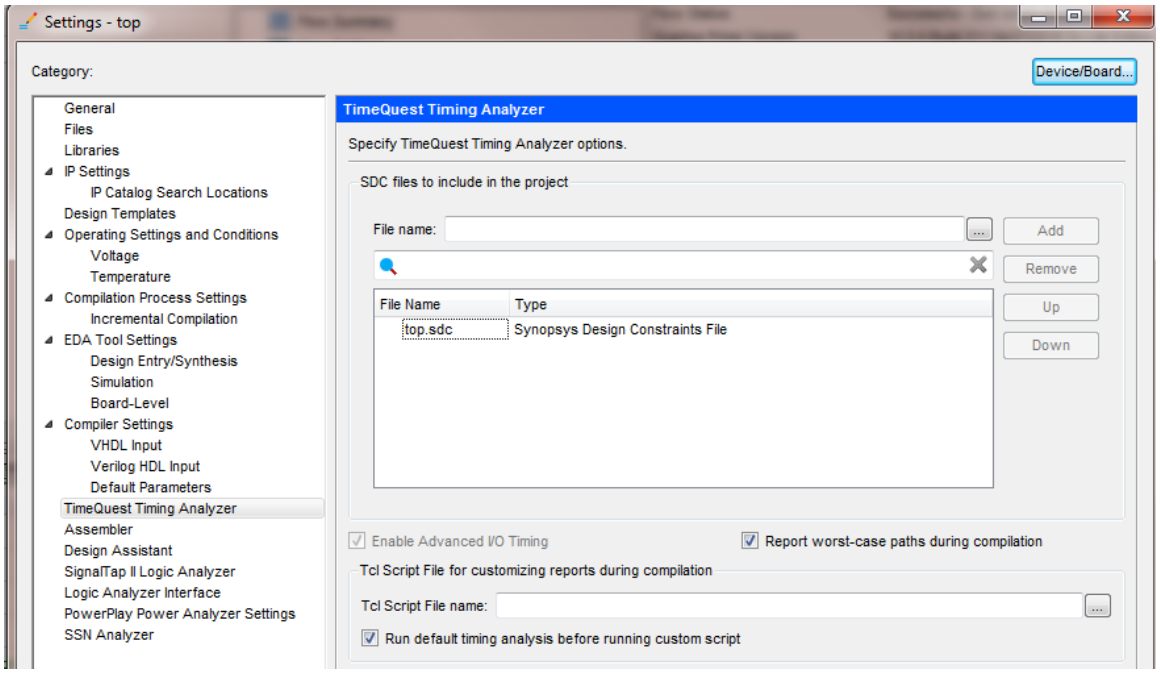
Your design should now be fully constrained.

Step 2: Run compilation using SDC file

Now you'll run the Quartus Prime Fitter using the constraints stored in the SDC file.

____ 1. Bring the Quartus Prime software to the foreground.

____ 2. From the **Assignments** menu, select **Settings**. Select the **TimeQuest Timing Analyzer** category on the left if it's not already selected.



Specify TimeQuest Timing Analyzer options.

SDC files to include in the project

File name:

File Name	Type
top.sdc	Synopsys Design Constraints File

Enable Advanced I/O Timing

☑ Report worst-case paths during compilation

Tcl Script File for customizing reports during compilation

Tcl Script File name:

☑ Run default timing analysis before running custom script

____ 3. Add the **top.sdc** file to the project if it has not been added already. The file should have been added here when it was saved initially. If it wasn't:

a. Use the browse button [...] to locate the file **top.sdc**.

b. Click **Open**.

c. Click **Add** to add the file to the list.

____ 4. If it's not already enabled, turn on **Report worst-case paths**. Click **OK**.

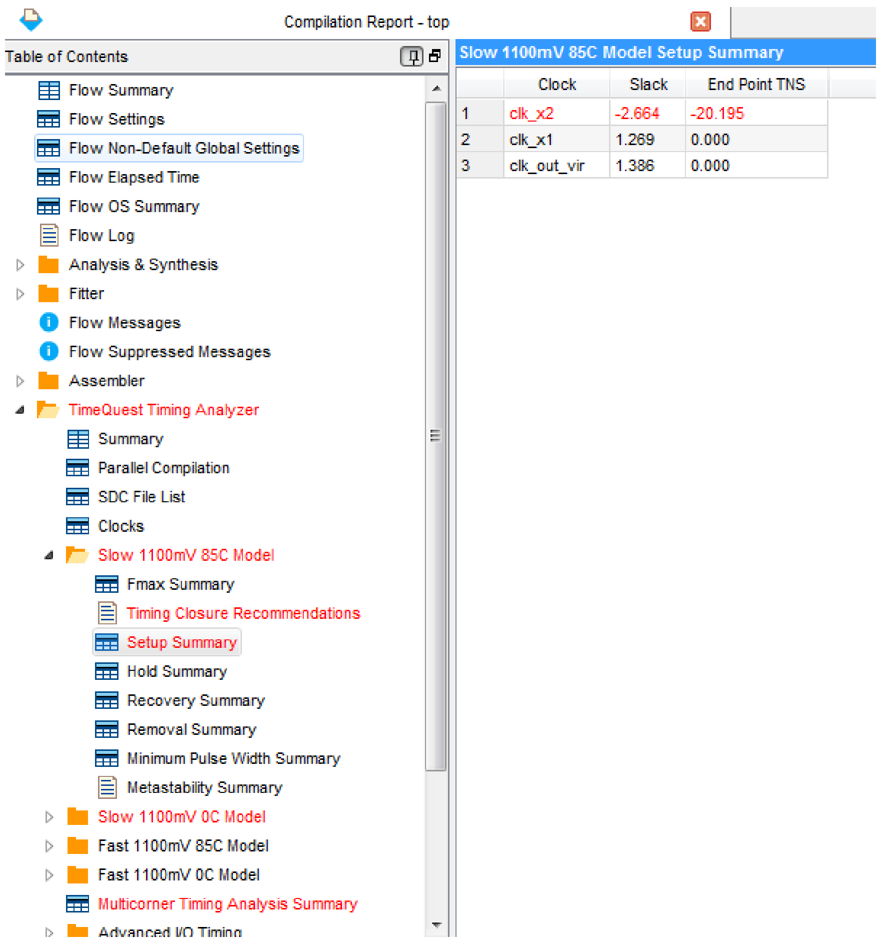
____ 5. Click the ▶ button or select **Start Compilation** from the **Processing** menu. Click **OK** when complete of a GUI open up asking to click OK.

____ 6. If it's not already visible, open the compilation report by clicking and expand the **TimeQuest Timing Analyzer** folder.

____ 7. Look at the **Multicorner Timing Analysis Summary** to see timing information for all analyses (setup, hold, recovery, and removal) for the two slow corner models and the fast corner model.

You should see the various models showing up in red or balck in the ***Slow Model*** *folders, the* ***Fast Model*** *folders, and the* ***Multicorner Timing Analysis Summary.*** *Reports appear in red or black indicating that the design is failing timing or meeting timing, If the report is in black then the design is meeting timing and if the report is in red the design is not meeting timing. Typically, the slow model would fail setup timing because slow signals do not reach their destinations early enough to meet setup requirements. The fast model for a design might fail hold timing because fast signals may change their values before meeting hold timing requirements. Remember that if we can meet setup timing in the slow model and hold timing in the fast model, the design will meet timing across all PVT.*

____ 8. Expand the **Slow Model 1100mV 85C Model** folder and select the **Setup Summary** table to see which clock is failing.



Slow 1100mV 85C Model Setup Summary

	Clock	Slack	End Point TNS
1	clk_x2	-2.664	-20.195
2	clk_x1	1.269	0.000
3	clk_out_vir	1.386	0.000

*The **Setup Summary** table indicates that clock **clk_x2** is failing timing by greater than **2 ns** (your value might be slightly different) in the setup analysis, but that's about all the information the **Setup Summary** report in the Compilation Report gives us. There is no indication in any of the reports here what path(s) is/are failing.*

Step 3: Analyze clk_x2 clock domain using TimeQuest reports

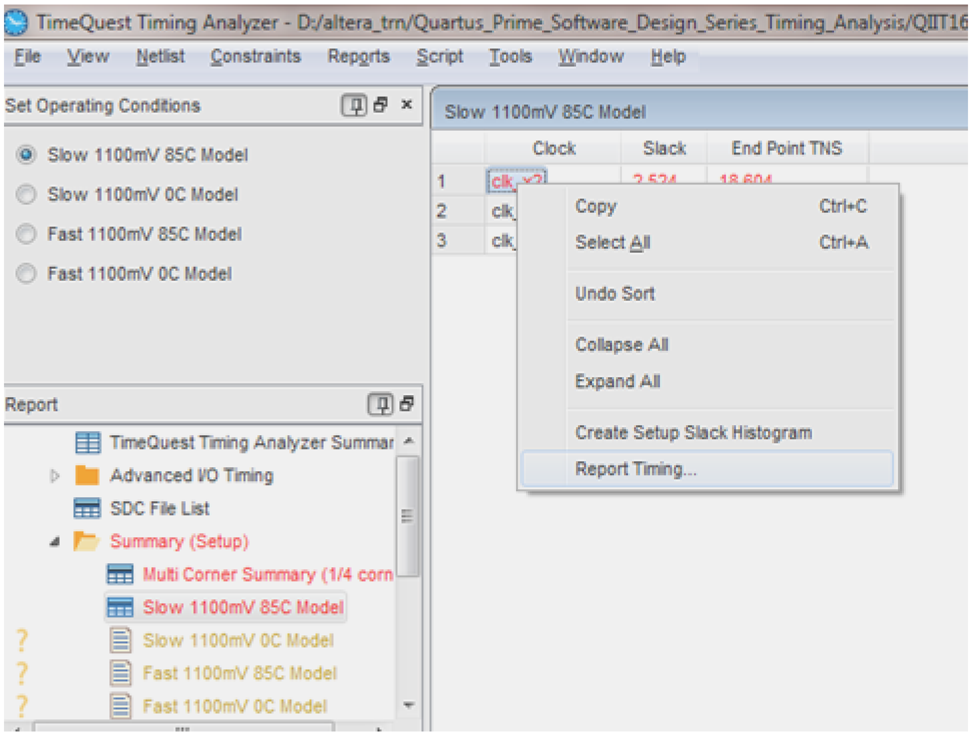
____ 1. Open the TimeQuest Timing Analzer tool again if it is not already open.

____ 2. Double-click **Report Setup Summary** in the **Tasks** pane. Remember this automatically runs **Create Timing Netlist** (now with the **Post-fit** netlist), **Read SDC File**, and **Update Timing Netlist**.

*You will see a table similar to the **Setup Summary** table in the **Compilation Report** indicating the **clk_x2** clock domain is failing.*

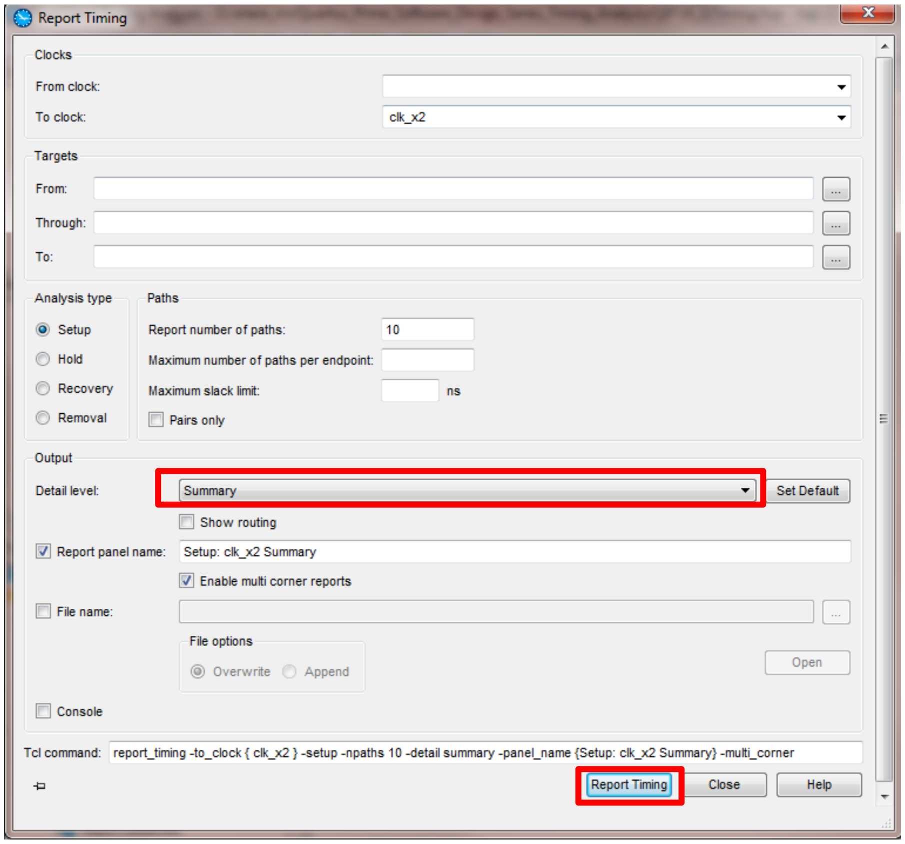
*One way to run a detailed analysis of the **clk_x2** clock domain is to use **Report Timing** (**Tasks** pane, **Reports** menu, or `report_timing` command). You will do this in an even easier way by means of the TimeQuest GUI.*

____ 3. In the **Summary Setup** table, right-click **clk_x2** and select **Report Timing**.



	Clock	Slack	End Point TNS
1	clk_x2	-2.524	-18.604
2	clk		
3	clk		

____ 4. In the **Report Timing** dialog box, select **Summary** in the **Output** ⇒ **Detail Level** drop-down menu. Click the **Report Timing** button.



Report Timing

Clocks
From clock:
To clock: clk_x2

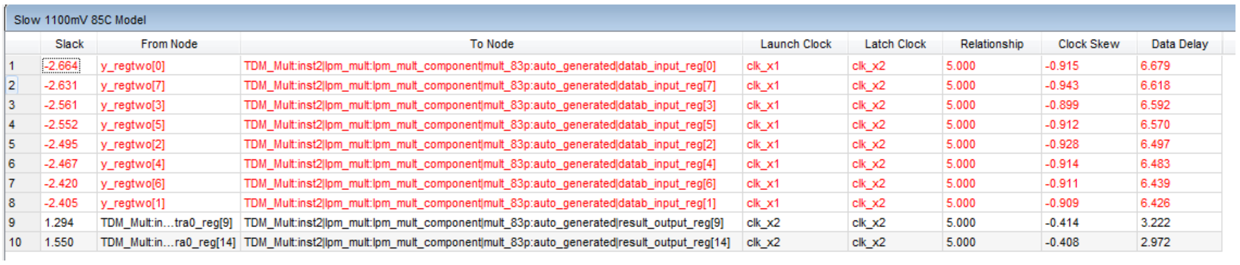
Targets
From:
Through:
To:

Analysis type: [✓ Setup] Hold Recovery Removal

Paths
Report number of paths: 10
Maximum number of paths per endpoint:
Maximum slack limit: ns
☐ Pairs only

Output
Detail level: Summary
☐ Show routing
☑ Report panel name: Setup: clk_x2 Summary
☑ Enable multi corner reports
☐ File name:
File options: [✓ Overwrite] Append
☐ Console

Tcl command: report_timing -to_clock { clk_x2 } -setup -npaths 10 -detail summary -panel_name {Setup: clk_x2 Summary} -multi_corner



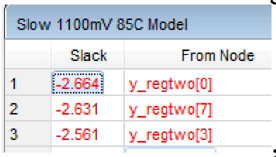
Slow 1100mV 85C Model

	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-2.664	y_regtwo[0]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|datab_input_reg[0]	clk_x1	clk_x2	5.000	-0.915	6.679
2	-2.631	y_regtwo[7]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|datab_input_reg[7]	clk_x1	clk_x2	5.000	-0.943	6.618
3	-2.561	y_regtwo[3]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|datab_input_reg[3]	clk_x1	clk_x2	5.000	-0.899	6.592
4	-2.552	y_regtwo[5]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|datab_input_reg[5]	clk_x1	clk_x2	5.000	-0.912	6.570
5	-2.495	y_regtwo[2]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|datab_input_reg[2]	clk_x1	clk_x2	5.000	-0.928	6.497
6	-2.467	y_regtwo[4]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|datab_input_reg[4]	clk_x1	clk_x2	5.000	-0.914	6.483
7	-2.420	y_regtwo[6]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|datab_input_reg[6]	clk_x1	clk_x2	5.000	-0.911	6.439
8	-2.405	y_regtwo[1]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|datab_input_reg[1]	clk_x1	clk_x2	5.000	-0.909	6.426
9	1.294	TDM_Mult:in...tra0_reg[9]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|result_output_reg[9]	clk_x2	clk_x2	5.000	-0.414	3.222
10	1.550	TDM_Mult:in...ra0_reg[14]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|result_output_reg[14]	clk_x2	clk_x2	5.000	-0.408	2.972

____ 5. Notice the pattern of the failing registers

From node________________________

To node __________________________

____ 6. Examine in detail the worst slack timing path for **clk_x2**. Select the worst slack in the list (first row) 

Slow 1100mV 85C Model

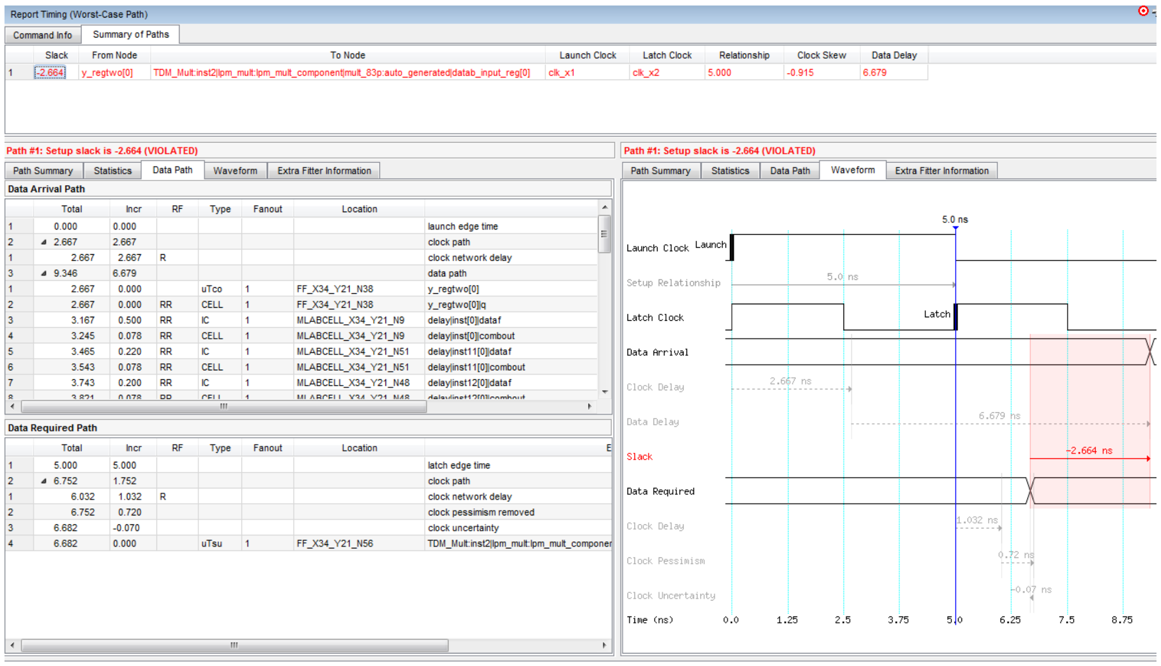
	Slack	From Node
1	-2.664	y_regtwo[0]
2	-2.631	y_regtwo[7]
3	-2.561	y_regtwo[3]

, right-click, and select **Report Worst-Case Path**.

*A new report appears called **Report Timing (Worst-Cast Path)**. The top of this new report shows the same information as in the summary report.*

____ 7. Click through the tabs in the lower panels of the reports to see different information about the selected failing path.

*The **Path Summary** tab slightly expands upon the summary report by displaying the data arrival and required times calculated by the timing analyzer. The **Statistics** tab displays statistical information breaking down the amount of time the signal spends traveling through path interconnect (IC) and logic cells. The **Data Path** tab provides detailed information about the actual device logic and interconnect the signal passes through as part of the data arrival and data required paths. The **Waveform** tab displays waveforms that show exactly how the timing analyzer arrived at its slack calculation. Click and drag in the waveform to add cursors that "snap" to events such as the launch and latch edges. Finally, the **Extra Fitter Information** tab displays any placement constraints that may exist on any of the elements in the path as well as a Chip Planner thumbnail highlighting the path.*



	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-2.664	y_regtwo[0]	TDM_Mult:inst2|lpm_mult:lpm_mult_component|mult_83p:auto_generated|datab_input_reg[0]	clk_x1	clk_x2	5.000	-0.915	6.679

Path #1: Setup slack is -2.664 (VIOLATED)

Data Arrival Path

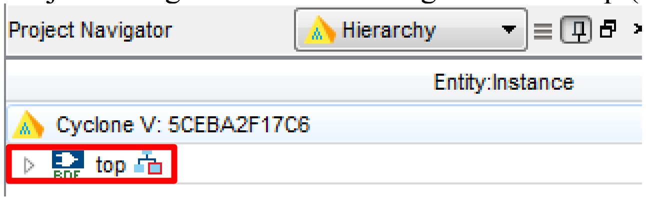
	Total	Incr	RF	Type	Fanout	Location	
1	0.000	0.000					launch edge time
2	2.667	2.667					clock path
1	2.667	2.667	R				clock network delay
3	9.346	6.679					data path
1	2.667	0.000		uTco	1	FF_X34_Y21_N38	y_regtwo[0]
2	2.667	0.000	RR	CELL	1	FF_X34_Y21_N38	y_regtwo[0]|q
3	3.167	0.500	RR	IC	1	MLABCELL_X34_Y21_N9	delay|inst[0]|dataf
4	3.245	0.078	RR	CELL	1	MLABCELL_X34_Y21_N9	delay|inst[0]|combout
5	3.465	0.220	RR	IC	1	MLABCELL_X34_Y21_N51	delay|inst11[0]|dataf
6	3.543	0.078	RR	CELL	1	MLABCELL_X34_Y21_N51	delay|inst11[0]|combout
7	3.743	0.200	RR	IC	1	MLABCELL_X34_Y21_N48	delay|inst12[0]|dataf
8	3.821	0.078	RR	CELL	1	MLABCELL_X34_Y21_N48	delay|inst12[0]|combout

Data Required Path

	Total	Incr	RF	Type	Fanout	Location	E
1	5.000	5.000					latch edge time
2	6.752	1.752					clock path
1	6.032	1.032	R				clock network delay
2	6.752	0.720					clock pessimism removed
3	6.682	-0.070					clock uncertainty
4	6.682	0.000		uTsu	1	FF_X34_Y21_N56	TDM_Mult:inst2|lpm_mult:lpm_mult_componer

You should see the timing report shown here, which provides all the detailed timing information about the requested path.

____ 8. Analyze the design to determine how to fix the violations. Bring the **top.bdf** schematic to the foreground. This can be done by going to Quartus Prime and in the Project Navigator double clicking on the the top (BDF) file



*Notice there is a delay block called **delay_8bits** in the output path of register **y_regtwo**, which is the second bank of registers fed by the input bus **din_y**. This delay block has been inserted on purpose to represent some logical delay in the design. This delay block is causing the timing violations for **clk_x2** driving the multiplier. Also notice that the **x_regtwo** and **y_regtwo** outputs are connected to the zero input of the multiplexers (**mux_ax** & **mux_by**), whose select lines are controlled by mux select logic. The mux select logic uses the **clk_x2** to clock the divide by 2 circuit essentially* *replicating the **clk_x1** clock.. This means that the multiplexers only selects the output of registers **x_regtwo** and **y_regtwo** during the negative cycle of **clk_x1**.*

You could use the clk_x1 as the mux_sel directly but that can cause issues as the clock gets combined with logic. Good design practice does not use clocks for anything except clockin register. Using a divide by 2 circuit using the faster clock is common, care must be taken that the mux_sel signal has the proper polarity, i.e. when th mux_sel is high or when the mux_sel is low that is is muxing the proper mux input to the output.

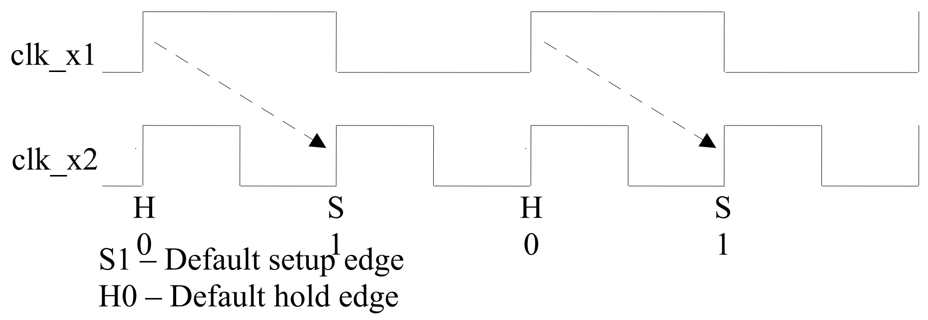
*The TimeQuest timing analyzer automatically assumes data from all the registers must reach the multiplier within one cycle of the **clk_x2** clock (the first 5 ns or when **clk_x1** is high). But this is only true for the outputs of **a_regtwo** and **b_regtwo**. The **x_regtwo** and **y_regtwo** data is computed on the second cycle of **clk_x2**, so this data has 10 ns to reach the multiplier. So, in this case, a **multicyle** exception can fix the analysis while accurately describing the way the design is supposed to work.*

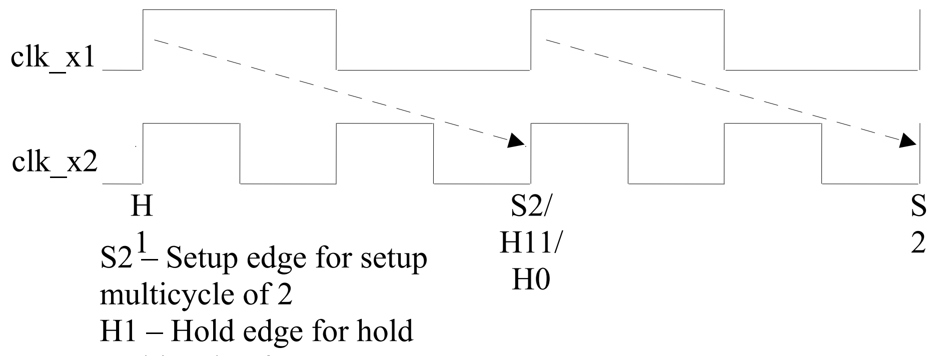
Step 4: Use multi-cycle constraints to correct failing internal paths

____ 1. In **top.sdc**, add an SDC command to make the paths **from** the **x_regtwo*** and **y_regtwo*** registers have a setup multicycle path of **2** (no **–to** argument is required). The multicycle assignment should be based on the **destination clock edges**.

Hint: This can be done with one or two lines in the SDC file. You may use the get_cells or get_registers calls to find your collection for the from nodes.

____ 2. Add a **hold multicycle** of **1** for these same paths.

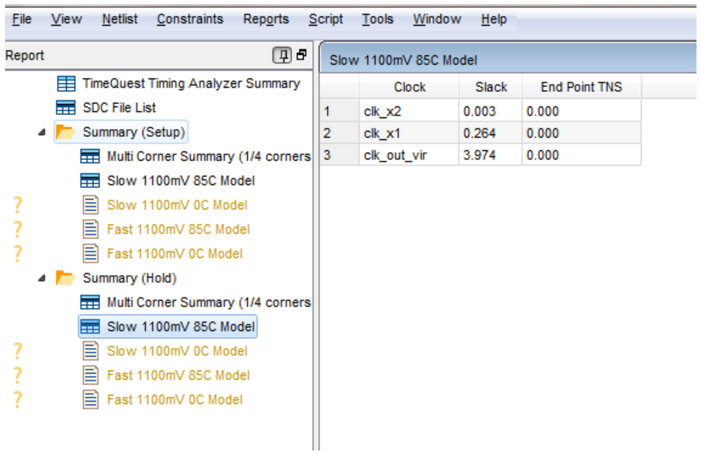



clk_x1
clk_x2
H1
S2/ H1/ H0
S2
S2 – Setup edge for setup multicycle of 2
H1 – Hold edge for hold multicycle of 1

These setup and hold multicycle constraints tell the timing analyzer that data from the ***x_regtwo*** *and* ***y_regtwo*** *registers have the full two cycles of the* ***clk_x2*** *clock to reach the multiplier (opening the window).*

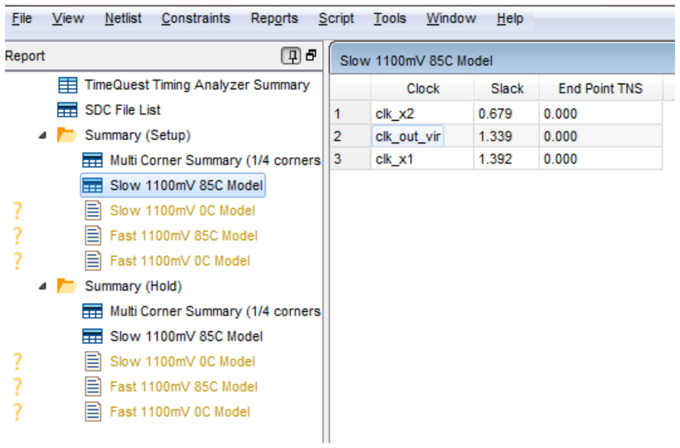
____ 3. Save **top.sdc**.

____ 4. Reset the design in TimeQuest (no need to recompile yet), and run **Report Setup Summary** and **Report Hold Summary**.



Slow 1100mV 85C Model

	Clock	Slack	End Point TNS
1	clk_x2	0.003	0.000
2	clk_x1	0.264	0.000
3	clk_out_vir	3.974	0.000



Slow 1100mV 85C Model

	Clock	Slack	End Point TNS
1	clk_x2	0.679	0.000
2	clk_out_vir	1.339	0.000
3	clk_x1	1.392	0.000

All clock domains should now be shown in black to indicate they are meeting timing.

____ 5. Manually delete the timing netlist (don't use **Reset design**) and generate a **fast** model netlist. Verify that hold timing is met in the fast corner. If your design is failing hold timing, don't worry, we will recompile later to solve that.

Remember that we're more concerned about hold timing than setup timing in the fast corner. The multicycle constraints should have fixed all setup timing failures, but you should make it a habit in your own designs to not forget about hold timing by regenerating timing reports using the fast model netlist.

Step 5: Analyze asynchronous timing in the design

____ 1. From the **Tasks** pane, generate the recovery and removal summary reports.

*You should see only positive slack on all asynchronous signals in the design. The summary reports indicate that all asynchronous signals are within the **clk_x1** clock domain. Let's examine the asynchronous paths in more detail.*

____ 2. Create a report of asynchronous paths for the **clk_x1** clock domain.

a. In the **Summary (Recovery)** report, select and right-click **clk_x1**, and select **Report Timing**.

b. Set the report **Detail level** to **Path Only**,

c. Set the number of paths to analyze to **100**.

d. Click **Report Timing**.

____ 3. Scroll to the bottom of the summary report and note the number of paths reported. What is the common source for all these paths? Check the schematic to find this instance in the design.

*Notice that this is the second register of the reset synchronization circuit that generates the **aclr_n** signal. Since this signal connects to the asynchronous reset pins of 96 registers in the design, 96 paths are listed in the report. The input **reset** signal is also asynchronous, but since we set **reset** to be a false path, it does not get analyzed or included in the report.*

*Note that both the launch and latch clocks for 96 paths is the **clk_x1**. Remember that the recovery analysis is analogous to the setup analysis of synchronous signals and removal analysis is analogous to hold. Since this asynchronous signal is synchronized into the **clk_x1** clock domain, **clk_x1** is the clock indicated in the recovery and removal summary reports.*

*Also, since the PLL is creating the **clk_x1** and **clk_x2** and they are interger multiples of each other, we can utilize the reset synchronizer to reset our **clk_x2** mux_sel logic. You can re-run the Report timing for Recovey analysis selecting the **clk_x2** and there should be 1 path to report as there is only 1 register in the **clk_x2** domain being reset by the reset synchronizing circuit. Notice the Launch clock is **clk_x1** in this case*

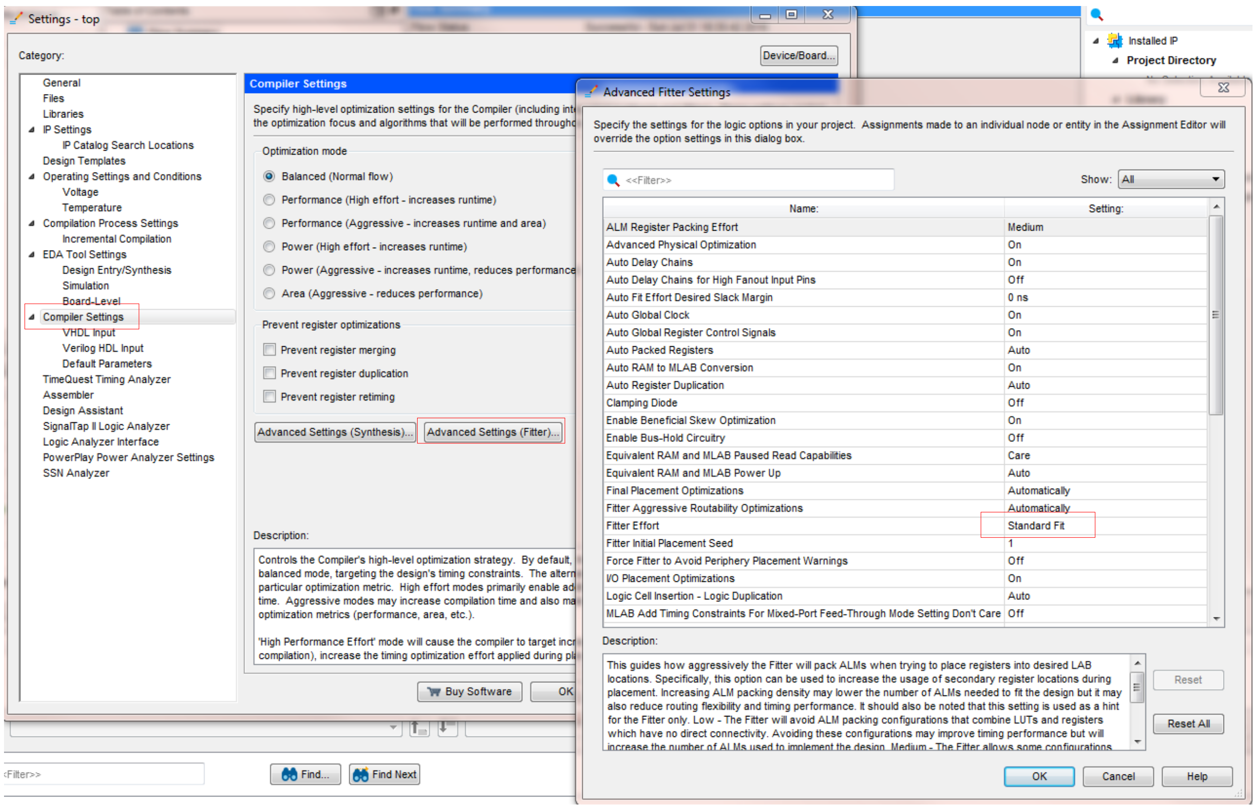
____ 4. Close the TimeQuest Timing Analyzer.

____ 5. Recompile the design in the Quartus Prime software.

____ 6. When the recompilation is complete, examine the timing reports in the Compilation Report and/or the timing analyzer to make sure the design now fully meets timing in all corners for setup, hold, recover, and removal.

If the design still does not meet timing, use the skills you've learned today to analyze any failing paths, go over your SDC constraints, and figure out what's wrong. If you need help, ask the instructor for further assistance.

For hold violations, you can adjust the fitter effort level to Standard Fit vs Auto Fit. This will work harder during the fit an can help with meeting hold time violation. This window can be found form the Quartus Prime Assignment->Settings pull down menu and selecting Compiler Settings (see below)



Specify high-level optimization settings for the Compiler (including int
the optimization focus and algorithms that will be performed througho

Optimization mode

[✓ Balanced (Normal flow)]
Performance (High effort - increases runtime)
Performance (Aggressive - increases runtime and area)
Power (High effort - increases runtime)
Power (Aggressive - increases runtime, reduces performance
Area (Aggressive - reduces performance)

Prevent register optimizations

☐ Prevent register merging
☐ Prevent register duplication
☐ Prevent register retiming

Description:

Controls the Compiler's high-level optimization strategy. By default,
balanced mode, targeting the design's timing constraints. The altern
particular optimization metric. High effort modes primarily enable ad
time. Aggressive modes may increase compilation time and also ma
optimization metrics (performance, area, etc.).

'High Performance Effort' mode will cause the compiler to target incr
compilation), increase the timing optimization effort applied during pl

Specify the settings for the logic options in your project. Assignments made to an individual node or entity in the Assignment Editor will override the option settings in this dialog box.

Name:	Setting:
ALM Register Packing Effort	Medium
Advanced Physical Optimization	On
Auto Delay Chains	On
Auto Delay Chains for High Fanout Input Pins	Off
Auto Fit Effort Desired Slack Margin	0 ns
Auto Global Clock	On
Auto Global Register Control Signals	On
Auto Packed Registers	Auto
Auto RAM to MLAB Conversion	On
Auto Register Duplication	Auto
Clamping Diode	Off
Enable Beneficial Skew Optimization	On
Enable Bus-Hold Circuitry	Off
Equivalent RAM and MLAB Paused Read Capabilities	Care
Equivalent RAM and MLAB Power Up	Auto
Final Placement Optimizations	Automatically
Fitter Aggressive Routability Optimizations	Automatically
Fitter Effort	Standard Fit
Fitter Initial Placement Seed	1
Force Fitter to Avoid Periphery Placement Warnings	Off
I/O Placement Optimizations	On
Logic Cell Insertion - Logic Duplication	Auto
MLAB Add Timing Constraints For Mixed-Port Feed-Through Mode Setting Don't Care	Off

Description:

This guides how aggressively the Fitter will pack ALMs when trying to place registers into desired LAB locations. Specifically, this option can be used to increase the usage of secondary register locations during placement. Increasing ALM packing density may lower the number of ALMs needed to fit the design but it may also reduce routing flexibility and timing performance. It should also be noted that this setting is used as a hint for the Fitter only. Low - The Fitter will avoid ALM packing configurations that combine LUTs and registers which have no direct connectivity. Avoiding these configurations may improve timing performance but will increase the number of ALMs used to implement the design. Medium - The Fitter allows some configurations

Summary

- *Constrained an asynchronous control input*
- *Added timing exceptions (i.e. false path and multicycle) to match design functionality*

END OF EXERCISE 4 - zip up directory and load to Canvas as lab10d