



# Intel® Quartus® Prime Pro Software Timing Analysis

## Exercise Manual

Copyright © 2022 Intel Corporation.

This document is intended for personal use only. Unauthorized distribution, modification, public performance, public display, or copying of this material via any medium is strictly prohibited.

**Software Requirements:**

Intel® Quartus® Prime Pro Edition software v22.1

Intel Stratix® 10 device support

**Hardware Requirements:**

None

# Introduction to the Timing Analyzer

## Objectives:

- *Perform timing analysis using the Intel® Quartus® Prime Pro software Timing Analyzer design flow to generate timing reports*
- *Run some timing reports to analyze design*

## Top-Level Design:

*The pipemult design used in this exercise is a variation of the design used in the introductory Intel Quartus Prime Pro course. The design utilizes 2 32-bit input data buses which pass through a multiplier with the 64-bit output fed to a 64-word deep RAM. The 64-bit output of the RAM is sent to an external downstream device using an LVDS SERDES Intel FPGA IP. The LVDS IP serializes each of the 8 bytes of the 64-bit word into 8 serial streams for transmission. The incoming data and the FPGA core run at 200 MHz. The LVDS serial streams run at 1.6 Gbps.*

*In between each of the blocks in pipemult are pipeline registers for improving design performance.*

*Note: Intel Stratix® 10 FPGAs have additional reset requirements in the form of the Reset Release IP. This block is not implemented in this design for the purpose of simplicity.*

## Step A: Open and compile a project

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

- \_\_\_ 1. In the VMWare Learning Platform (VLP) in a Windows\* OS Explorer window, navigate to the **C:\fpga\_trn\ Quartus\_Prime\_Pro\_Timing\_Analysis** directory. This is your **<lab\_install\_directory>**.
- \_\_\_ 2. **Right-click** the **IQPTAW\_Lab\_22\_1\_v1.zip** file, go to the **7-Zip** submenu, and select **Extract Here** to create the **IQPTAW\_Lab\_22\_1\_v1** directory.

*This path to the subfolder **IQPTAW\_Lab\_22\_1\_v1** will be referred to as the **<lab install directory>**.*

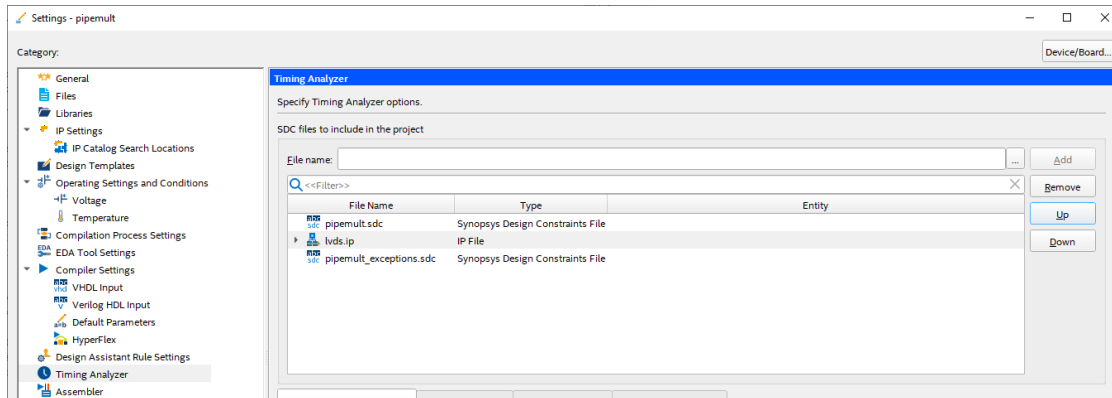
- \_\_\_ 3. Start the **Intel® Quartus® Prime Pro Edition software version 22.1**.


*In the VLP, the Intel Quartus Prime software is installed on an external drive. Please use the following steps to ensure you launch the correct version for this class. (There is also a shortcut in the Windows taskbar.)*

- a. In Windows\* OS Explorer, locate the external drive named **FPGA\_Training**.
- b. On that drive, go to the folder **\intelFPGA\_pro\22.1\quartus\bin64**.
- c. Double-click the file **quartus.exe**.
- \_\_\_ 4. You should confirm by looking at the status bar at the top or by clicking on the **About Quartus Prime** dialog box (**Help** menu) that you indeed have the **22.1 Prime Pro Edition** of the software opened, not the Standard Edition.
- \_\_\_ 5. In the **<lab install directory>**, right-click **Intro.zip**, go to the **7-Zip** submenu, and select **Extract Here** to create the **Intro** project directory.
- \_\_\_ 6. Open the project **pipemult.qpf** located in the **Intro** directory. Be sure to use **File -> Open Project** and not **File -> Open**.


*Next, let's add our SDC file to the project.*

- \_\_\_ 7. From the **Assignments** menu, select **Settings**.
- \_\_\_ 8. Select the category **Timing Analyzer** on the left if it's not already selected.



- \_\_\_ 9. Add the **pipemult.sdc** and **pipemult\_exceptions.sdc** files to the project.
- Use the browse button  and locate the files **pipemult.sdc** and **pipemult\_exceptions.sdc** in the project directory.
  - Highlight both files and click Open.
  - Set the correct file order. Use the Up and Down buttons on the right side of the window to make sure that **pipemult.sdc** is read first (i.e. is first in the list).


*The LVDS IP will automatically create clocks for the design if they are not defined already. Since we want to name some of the clocks, we want these clocks to be declared before the IP SDC is read. Since timing exceptions may refer to clocks created in either project or IP SDC files, we typically want them read last. But for this design, the exceptions refer to specific timing paths, so being last is not necessary.*

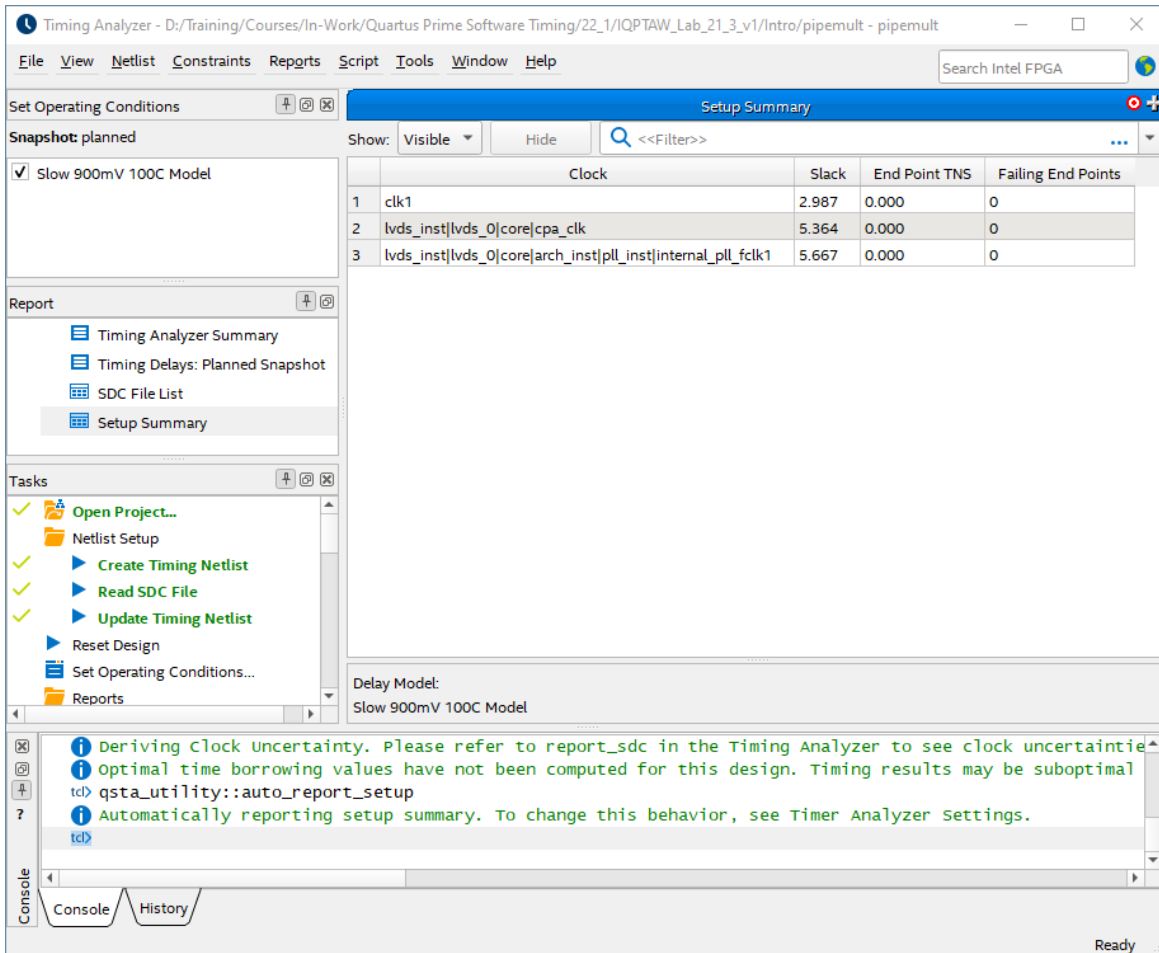
- \_\_\_ 10. Click **OK** to close the Settings dialog box.
- \_\_\_ 11. Create a Plan netlist or database. In the **Compilation Dashboard**, locate and click on the **Plan** button , under **Fitter → Fitter (Implement)**.

*Though you could also perform a full compilation since this design is complete, you're going to follow the Timing Analyzer flow as if you were working on a new design that requires a long place-and-route. A planned netlist provides enough information to begin constraining the design and is the earliest we can stop the compiler for this purpose.*

*Your netlist generation should be finished in ~ 2 min.*

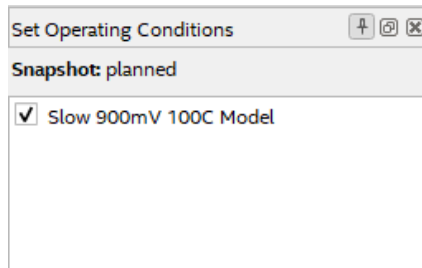
## Step B: Start the Timing Analyzer GUI to check constraints and perform initial timing checks

1. In the Intel Quartus Prime Pro software GUI toolbar, click  or, from the **Tools** menu, select **Timing Analyzer**.



The Timing Analyzer opens ready for you to begin creating reports.

At this point, you can use the Timing Analyzer to check design constraints and get some initial performance numbers before you do a full compilation.



Notice in the Set Operating Conditions pane, that the Snapshot confirms that the **Planned** netlist is being used. For that reason, only 1 timing model is available.

2. In the **Report** pane, click on the **SDC File List** report.

SDC File Path	Instance	Promoted	Status	Read at
1 pipemult_exceptions.sdc		No	OK	Tue Jul 26 10:01:30 2022
2 pipemult.sdc		No	OK	Tue Jul 26 10:01:30 2022
3 lvds/altera_iopll_1931/synth/lvds_altera_iopll_1931_d7jzioa.sdc		No	OK	Tue Jul 26 10:01:30 2022
4 lvds/altera_lvds_core14_191/synth/lvds_altera_lvds_core14_191_aca5ihq.sdc		No	OK	Tue Jul 26 10:01:30 2022

3. In the **Tasks** pane, double-click **Report SDC** found in the **Constraint Diagnostics** folder.

In the **Report** pane, a new folder called **SDC Assignments** appears containing reports called

- **Create Clock**
- **Create Generated Clock**
- **Set Clock Uncertainty**
- **Set Input Delay**
- **Set False Path**
- **Set Multicycle Path**
- **Disable Min Pulse Width**

These reports list all the SDC constraints automatically created or read into the tool from the SDC file(s). Use this report to see the expected constraint types are there and that their intended values are set.

4. In the **Tasks** pane, double-click **Report Clocks** in the **Clocks** folder.

This report focuses on clock domains, clock domain settings and clock relationships. Use it to confirm that your clocks are constrained correctly. You may have clocks that are generated by IP cores, so this report allows you to see those and verify them against your understanding of how the IP core works.



Clocks							
Show:	Visible	Hide	<<Filter>>				
	Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle
1	clk1	Base	6.666	150.0 MHz	0.000	3.333	
2	clk1_in_virt	Virtual	6.666	150.0 MHz	0.000	3.333	
3	lvds_inst lvds_0 core arch_inst pll_inst internal_pll_fclk1	Generated	0.833	1200.0 MHz	0.416	0.832	50.00
4	lvds_inst lvds_0 core arch_inst pll_inst internal_pll_loaden0	Generated	6.666	150.0 MHz	5.832	6.665	12.50
5	lvds_inst lvds_0 core arch_inst pll_inst internal_pll_loaden1	Generated	6.666	150.0 MHz	5.832	6.665	12.50
6	lvds_inst lvds_0 core arch_inst pll_inst internal_pll_m_cnt_clk	Generated	6.666	150.0 MHz	0.000	3.333	50.00
7	lvds_inst lvds_0 core arch_inst pll_inst internal_pll_vcoph	Generated	0.833	1200.0 MHz	0.000	0.416	50.00
8	lvds_inst lvds_0 core cpa_clk	Generated	6.666	150.0 MHz	6.332	9.665	50.00
9	pipemult_out_clk	Base	1.666	600.0 MHz	0.000	0.833	
10	pipemult_out_clk_n	Base	1.666	600.0 MHz	0.833	1.666	

Here you can see 10 different clocks have been created. Most of these are generated by the PLL within the LVDS IP core.

Since you have not seen anything alarming in your verification of the SDC constraints, you will next look at some timing reports to get a sense of how much work the Fitter has to do to get the design to meet timing. While this may not make much sense on a complete FPGA design, when you are working on a submodule of the design, this type of pre-optimization analysis may highlight parts of the module that you may want fix before the module is integrated into the larger design.

5. In the **Report** pane, select the **Setup Summary** report.


Set Operating Conditions	Setup Summary		
Snapshot: planned	Show:	Visible	Hide
<input checked="" type="checkbox"/> Slow 900mV 100C Model	<<Filter>>		
Report			
SDC File List			
Setup Summary			
SDC Assignments			
Clocks			
	Clock	Slack	End Point TNS
1	clk1	2.987	0.000
2	lvds_inst lvds_0 core cpa_clk	5.364	0.000
3	lvds_inst lvds_0 core arch_inst pll_inst internal_pll_fclk1	5.667	0.000

This design meets setup timing in both the base and virtual clock domains. A failure here indicates that you could have potential for timing failures in finished compilation. Depending on the cause and your capability to make changes, you may want to restructure some code before running full compilation and relying on the Fitter to fix the timing.

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

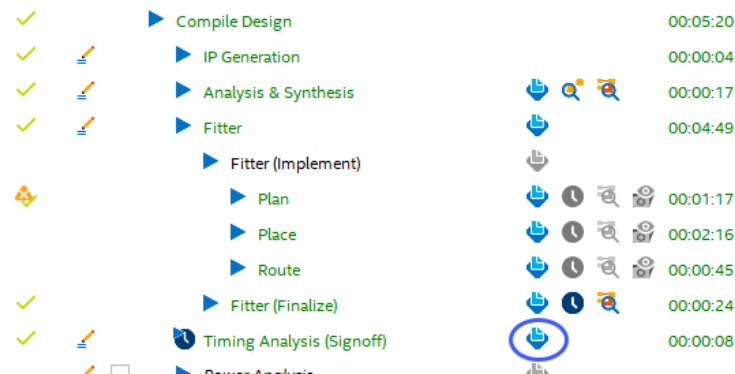
*This design passes hold timing in the LVDS IP generated clock domain. Hold timing is targeted by Fitter placement and routing optimizations later in the Fitter process. As a result, any hold timing failures may get fixed by the Fitter without any additional work from you. Note that these reports were based on Planned netlist estimates; the actual post-fit timing numbers will likely be different.*

### Step C: Use the SDC file to guide the Intel® Quartus® Prime software Fitter and view the results

1. Return to the **Intel Quartus Prime Pro software GUI**.
2. Complete the full compilation. In the **Compilation Dashboard**, click on the **Compile Design** button .

*Since you did not make any changes to the design or the SDC file, you could have continued with the next stage of the Fitter. But for the sake of this lab, we are going to finish with a full compilation.*

*The compilation should complete in ~ 5 minutes. The Timing Analyzer opens automatically ready for more detailed analysis, if needed.*




3. Return to the Intel Quartus Prime Pro software GUI again. Open the **Compilation Report** by clicking on the  button next **Timing Analysis (Signoff)** in the **Compilation Dashboard**.
4. Expand the **Timing Analyzer folder** in the **Compilation Report**.

Table of Contents	
1	Flow Suppressed Messages
▼	Timing Analyzer
	Summary
	Timing Delays: Final Snapshot
	Parallel Compilation
	SDC File List
	Clocks
	Timing Closure Summary (BETA)
	Fmax Summary
	Setup Summary
	Hold Summary
	Recovery Summary
	Removal Summary
	Minimum Pulse Width Summary
▶	Metastability Summary
▶	Advanced I/O Timing
▶	Clock Transfers
▶	Report TCCS
▶	Worst-Case Timing Paths
▶	Report RSKM
▶	Unconstrained Paths
	Multicorner Timing Analysis Summary
▶	Design Assistant (Signoff)
i	Messages

*Is your timing passing or failing? A quick glance shows the reports that are in read deal with timing closure and design assistant but not necessarily timing related from the SDC files. It is helpful to read the reports in red to get an understanding as to how the design can be further improved even if the design is meeting timing. The summary reports here still provide useful information. For example, you may wish to see the margins by which your design is passing. This could tell you how difficult it might be if you need to make any large changes to the design.*

*You can also see timing reports generated for specific hardware IP, such as Report TCCS and Report RSKM created for the LVDS IP.*

*At this point, you could generate more advanced reports and get more detail about specific paths in the design. You will get a chance to do this later in the class.*

## END OF EXERCISE

# Timing Analysis: Clock Constraints

**Objectives:**

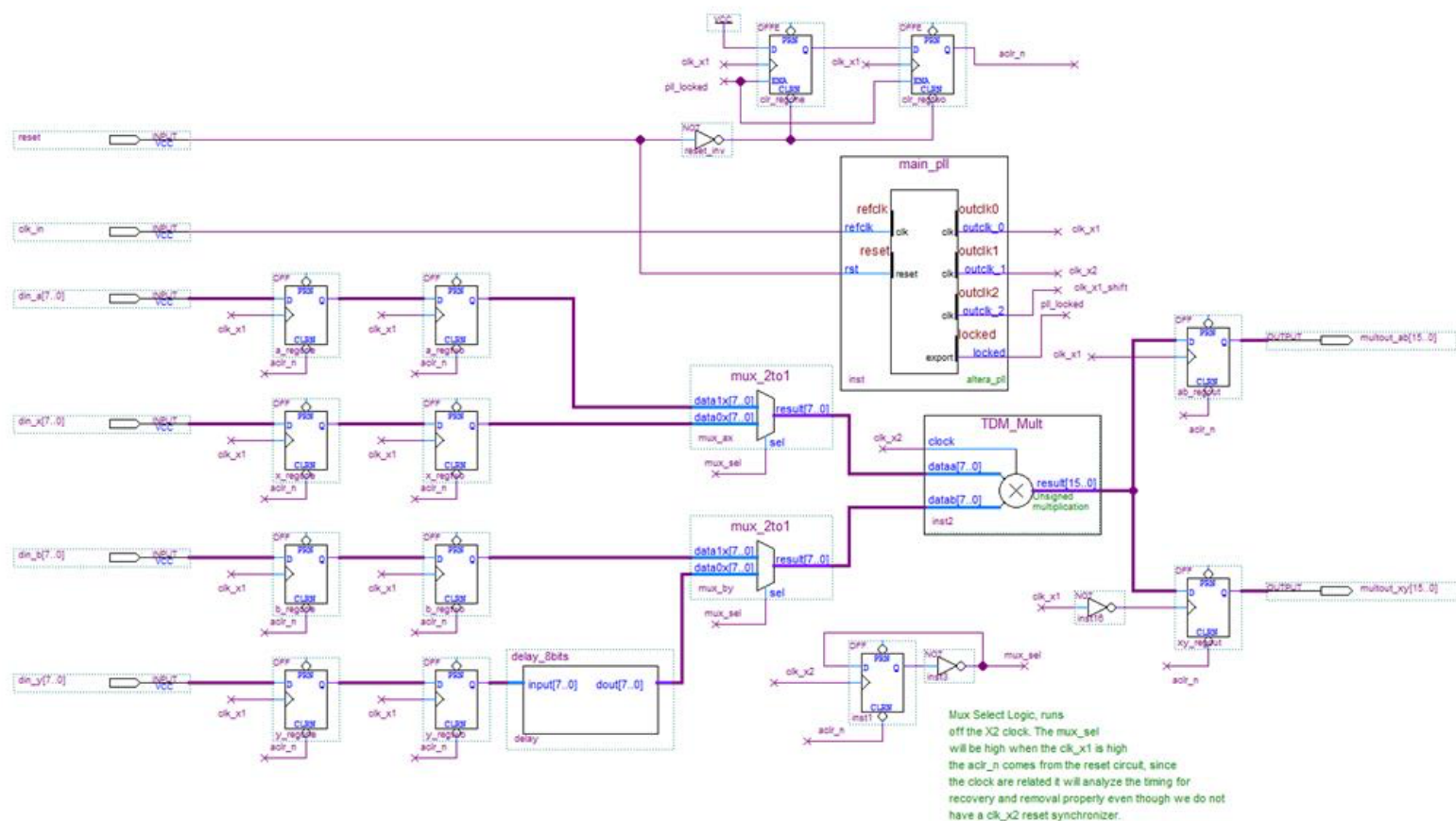
- Review PLL generated clock constraints
- Create a new project SDC file

**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 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 3 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. There's also a shifted 100 MHz output used to drive the select line of muxes. The resulting data outputs named **multout\_ab** and **multout\_xy** are sent off-chip to another device on the board.

Throughout the remaining exercises, you will be asked to create SDC commands using information from the lecture. If you are stuck or do not understand a solution command, please do not hesitate to ask your instructor for assistance. In addition, solutions are available in the unzipped lab files in the subdirectory to the lab installation directory named **Solution**.

Note: Intel Stratix® 10 FPGAs have additional reset requirements in the form of the Reset Release IP. This block is not implemented in this design for the purpose of simplicity.

**Circuit Diagram:**

## Step A: Open and name PLL output clocks

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

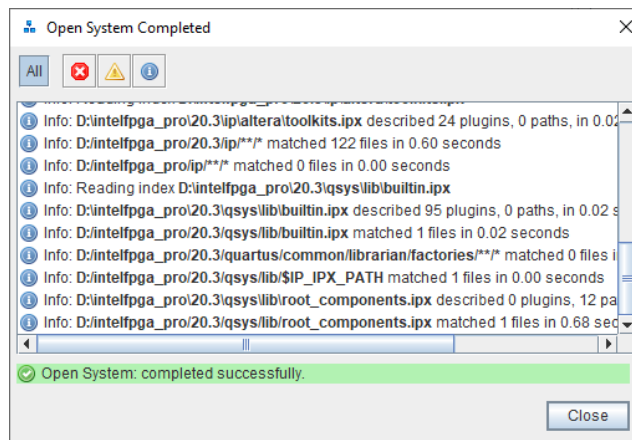
- \_\_\_\_ 1. In the <lab install directory>, right-click **Timing.zip**, go to the **7-Zip** submenu, and select **Extract Here** to create the **Timing** project directory.
- \_\_\_\_ 2. From the Intel® Quartus® Prime Pro Design software, select **File** → **Open Project...** to open the project **top.qpf** located in the **Timing** directory.
- \_\_\_\_ 3. **Open** the file **top.bdf** by double-clicking **top** in the **Project Navigator** (or by using the **File** → **Open** menu option).

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

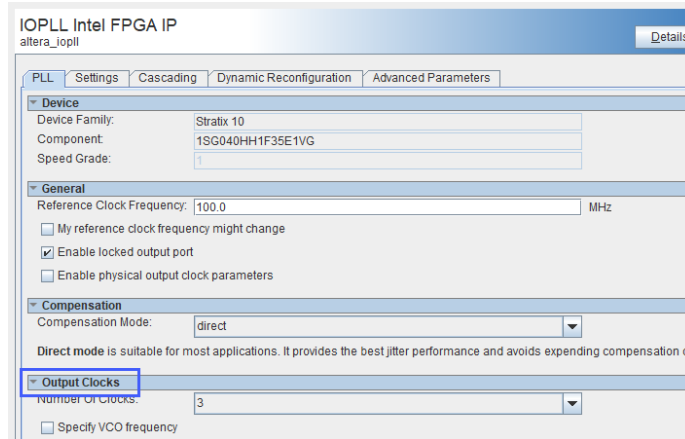
*Before you begin building your SDC file, you can start by defining the clock names in the IO PLL Intel FPGA IP core. These clock names will be used to define the names in your clock constraints.*

- \_\_\_\_ 4. Open the **main\_pll.ip** IP parameterization file (**File** menu → **Open**).

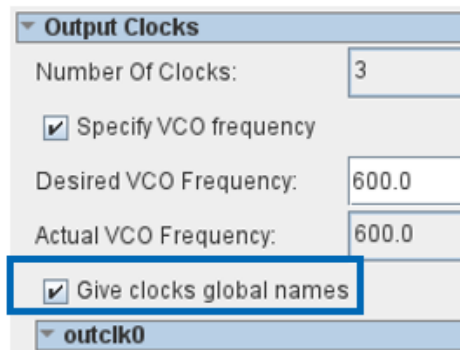
*The IOPLL IP Parameter Editor Pro window opens for the IP and begins initialization.*



- \_\_\_\_ 5. Click **Close** when the **Open System Completed** window successfully completes.



- \_\_\_ 6. On the **main PLL tab**, scroll down to the **Output Clocks** section.



Here you can see that 3 output clocks have been enabled for this PLL.

- \_\_\_ 7. **Enable** the option to **Give clocks global names**.


Further down in the Output Clocks section, you will find the settings for the 3 output clocks. The first field for each is the Clock Name: field.

Output Clock (Clock Name: Field)	New Clock Name
outclk0	clk_x1
outclk1	clk_x2
outclk2	sel_mux

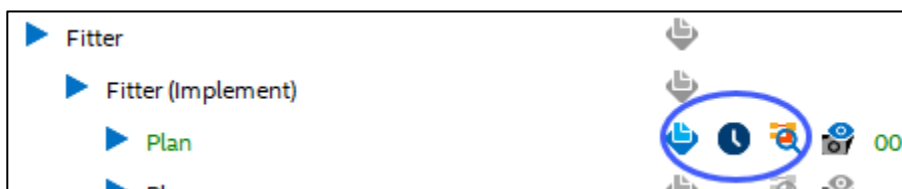
- \_\_\_ 8. Use the table above to rename each of the output clocks by typing the new clock name into the **Clock Name: field**.
- \_\_\_ 9. Generate updated files for the IOPLL IP by clicking on the **Generate HDL** button. Click **Generate** again when the **Generation** window appears and **Yes** to **save your unsaved changes** to the IP.
- \_\_\_ 10. Click **Close** when the Generate window changes to **Generation Completed**.
- \_\_\_ 11. Close the IOPLL IP Parameter Editor window.




## Step B: Compile the project and review clocks

1. Run the compilation stopping at the end of the **Fitter** → **Plan** state by clicking on the **Fitter** → **Plan**  button in the Compilation Dashboard.

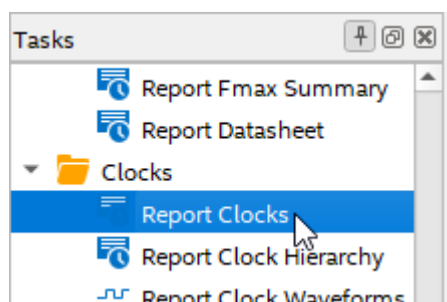
*Remember, timing constraints are used to guide the Fitter during place and route. Since we're only at the point of adding constraints to the design, there is no need to perform a full compilation because we'd have to run the Fitter again anyway. So, by just creating a planned database of the design and generating a post-planned timing netlist, you can iterate on changes much more quickly.*



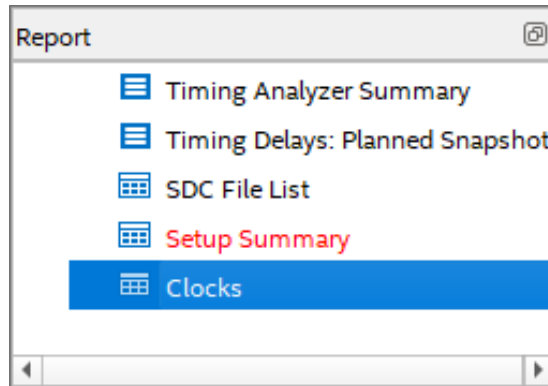
2. Open the **Timing Analyzer** by clicking the  button in the **Compilation Dashboard** next to the Plan execution button (circled above).


*The Timing Analyzer opens. Automatically, the planned snapshot (netlist) is used to generate a timing netlist and that timing netlist is loaded into the Timing Analyzer. In older versions of the Intel Quartus Prime Pro software, this process was done manually.*

*You may notice that one of the clocks in the design is failing timing. Since you are only at the point of creating the SDC file, you can safely ignore.*



3. In the **Tasks** pane of the Timing Analyzer window, generate a clock report by double-clicking **Report Clocks** task.



Clocks																	
Show:	Visible	Hide	 <<Filter>>														
	Clock Name	Type	Period	Frequency	Rise	Fall	Duty Cycle	Divide by	Multiply by	Phase	Offset	Edge List	Edge Shift	Inverted	Master	Source	Targets
1	clk_x1	Generated	10.000	100.0 MHz	0.000	5.000	50.00	6	6					false	inst5[...refclk	clk_in	{ inst...k[0] }
2	clk_x2	Generated	5.000	200.0 MHz	0.000	2.500	50.00	3	6					false	inst5[...refclk	clk_in	{ inst...k[1] }
3	inst5[iopll_0_m_cnt_clk	Generated	60.000	16.67 MHz	0.000	30.000	50.00	6	1					false	inst5[...refclk	clk_in	{ inst...reg }
4	inst5[iopll_0_refclk	Base	10.000	100.0 MHz	0.000	5.000											{ clk_in }

In the Report pane, use the Clocks report to review the automatically generated clock constraints derived from the PLL settings. As shown above, you should see one base clock and three generated clocks. Notice their rise and fall times used to determine timing relationships between them.

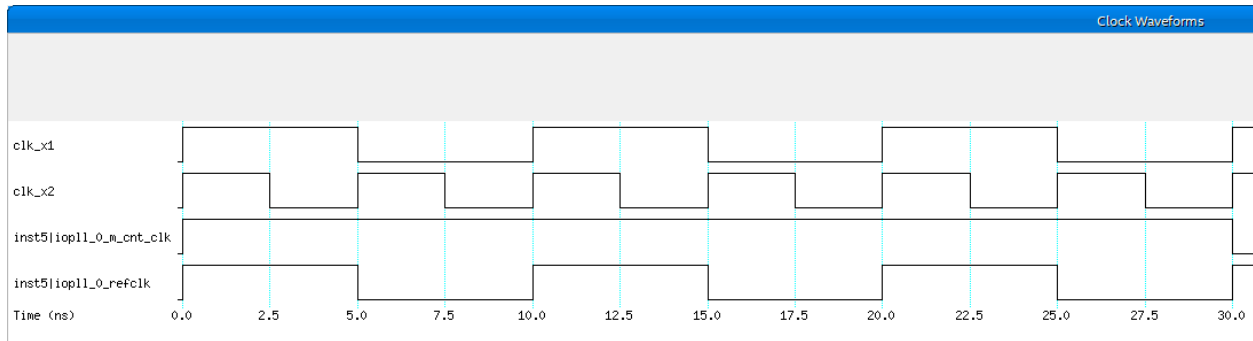
Let's view the clocks in another way.


4. In the **Tasks** pane of the Timing Analyzer window again, generate a clock report by double-clicking the **Report Clock Hierarchy** task in the **Clocks** folder.

Q <<Filter>>				
	Clock Name	Type	Period	Frequency
1	▼ inst5[iopll_0_refclk	Base	10.000	100.0 MHz
1	clk_x1	Generated	10.000	100.0 MHz
2	clk_x2	Generated	5.000	200.0 MHz
3	inst5[iopll_0_m_cnt_clk	Generated	60.000	16.67 MHz

In the Report pane, use the Clock Hierarchy Summary report to review the automatically generated clock constraints again. Note the clock information is the same as shown in the Clocks report but displayed with collapsible groups to show the relationship between the clocks. This view can be essential in a design with many clocks and derived clocks.

5. In the **Tasks** pane of the Timing Analyzer window, generate a clock report by double-clicking the **Report Clock Waveforms** task in the **Clocks** folder.



In the Report pane, use the Clock Waveforms  report to review the clock constraints again. Now, you can see visually the clocks and their relationships. Use this view to further ensure your understanding of the clocks in your design.

6. In the **Tasks** pane of the Timing Analyzer window, double-click on the **Report Unconstrained Paths** in the **Constraint Diagnostics** folder.

Unconstrained Paths Summary			
Show: <span>Visible</span> <span>Hide</span> <span>Search &lt;&lt;Filter&gt;&gt; ...</span>		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

Here you should see that that no illegal or unconstrained clocks have been found. But there are plenty of unconstrained I/O and I/O path. You will address those in the upcoming lab.

### Step C: Begin project SDC file

Next, you will create the SDC file to hold your timing constraints for the design.

1. From the **Timing Analyzer** window use the **File** menu, select **New SDC File**.

This will open a new text editor window. (Note: it may open attached to or detached from the Intel Quartus Prime Pro software GUI. The Attach/Detach Icon



is in the Text Editor toolbar).

2. With the Text Editor window in the foreground (or selected in the Intel Quartus Prime Pro software main window) select the **File** menu and **Save As**. Save the file as `<lab_install_directory>\Timing\top.sdc`.

You can leave the option **Add file to current project** enabled in the Save As dialog box.

*The next few steps are optional but are being used here to demonstrate the write SDC command. For documentation purposes and readability, it might be useful to have automatically generated clocks noted in your design SDC. So, you are going to use the write SDC command to write out the constraints and note them in your design SDC file.*

- \_\_\_ 3. In the **Tasks** pane of the Timing Analyzer, double click on the **Write SDC File** task. Accept the default name **top.out.sdc** and click **OK**.
- \_\_\_ 4. Open the newly created **top.out.sdc**.
- \_\_\_ 5. Locate and copy the **create\_clock** and **3 create\_generated\_clock** commands from **top.out.sdc** into **top.sdc**. If you'd like, split the long commands into multiple lines using backslashes (\).
- \_\_\_ 6. In **top.sdc**, **comment out** the commands you just copied over. You can place a '#' at the beginning of each line or highlight the entire block of code, right-click and select Comment Selection.

*Here you can also include note to indicate that these commands are automatically generated.*

- \_\_\_ 7. Close top.out.sdc.

*Again, these last few steps are not needed, but may be helpful when reviewing your SDC file later.*

## END OF EXERCISE

# Timing Analysis: Synchronous I/O Constraints

**Objectives:**

- Constrain the synchronous I/O paths in the design

**Step A: Add virtual clocks for I/O constraints**

Now you will define the delays to external devices on the design's I/O ports 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 defining virtual clocks.

1. Create a 10 ns input virtual clock, named clk\_in\_vir.

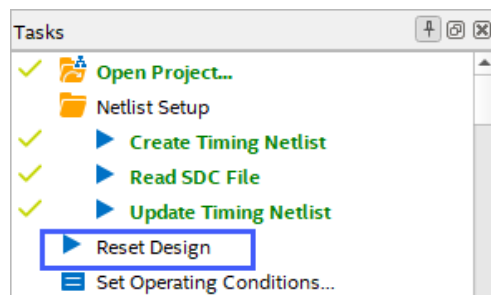
*This clock represents the clock that will drive the upstream device to send data that is captured by the din inputs to the design.*

2. Create another 10 ns output virtual clock named clk\_out\_vir.

*This clock represents the clock that will drive the downstream device to capture data that is sent by the outputs of the design.*

3. Save top.sdc.

*Since you have changed your SDC file, you should reset the design and re-run your reports to include the changed constraints.*

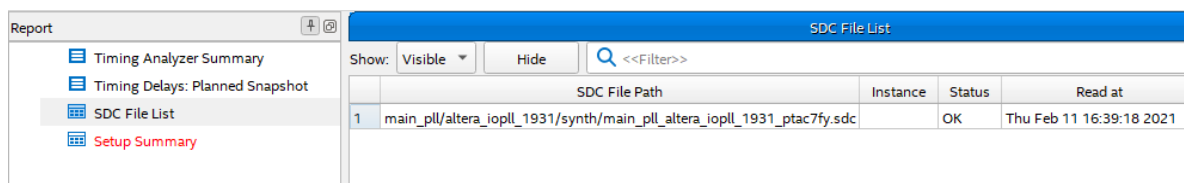


4. In the **Tasks** pane, double-click on **Reset Design**.

5. Run **Report Clocks** to check your work.

*Wait! Is this Clocks report correct? Why are your new virtual clocks missing?!! Let's figure out why as this a good example of debugging SDC file issues.*

6. In the **Reports** pane, click on the **SDC File List** report.



Here you can see that your **top.sdc** file is not included in your SDC file list.

Whenever you do not see expected constraints shown in a report, a good check is to confirm the Timing Analyzer has read ALL the files it should. You may not know all the SDC files listed here as some may be from IP cores (as in this case with the PLL), but any user you create and add to your project should be listed. If they are not, then you must figure out why. Many users not seeing their constraints listed immediately begin trying to debug the constraint. But you have narrowed down that it is a file issue and not a constraint issue. That alone can save you some time.

In this case, you added **top.sdc** to the project AFTER you opened the Timing Analyzer. So, when the Timing Analyzer read your project SDC files it was still using the file list from when it was launched.

- \_\_\_\_ 7. Force the Timing Analyzer to read **top.sdc** by doing one of the following:
  - Closing and re-opening the Timing Analyzer (which may not be practical on a large design where it takes the Timing Analyzer some time to open)
  - Manually performing a **Read SDC (Tasks** pane or **Constraints** menu) on **top.sdc** each time after you reset the design (until you re-launch the Timing Analyzer)
- \_\_\_\_ 8. Now you can double-click on the following reports to check your work
  - **Report Ignored Constraints** – to make sure you did not type anything incorrectly
  - **Report Clocks** – to see all defined and auto generated clocks together
  - **Report Clock Waveforms** – to see the alignment of your virtual, base and generated clocks
  - **Report Unconstrained Paths** – to see what has not been constrained yet
  - **Report SDC** – to verify your constraints and their values have been accepted
- \_\_\_\_ 9. If you find any errors, you should fix them, reset the design and re-run the reports until the constraints are appear correct to you.

	Clock Name	Type	Period	Frequency	Rise	Fall	Duty C
1	clk_in_vir	Virtual	10.000	100.0 MHz	0.000	5.000	
2	clk_out_vir	Virtual	10.000	100.0 MHz	0.000	5.000	
3	clk_x1	Gen...ted	10.000	100.0 MHz	0.000	5.000	50.00
4	clk_x2	Gen...ted	5.000	200.0 MHz	0.000	2.500	50.00
5	inst5 iopl0_m_cnt_clk	Gen...ted	160.000	6.25 MHz	0.000	80.000	50.00
6	inst5 iopl0_refclk	Base	10.000	100.0 MHz	0.000	5.000	

Here is a screen capture of the Clocks report with all clocks listed.

## Step B: Constrain synchronous input paths using SDC

Now you will define the delays to external devices on the design's I/O ports 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	0.6
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:

$$\text{Input delay (max)} = \text{board delay (max)} - \text{clock skew (min)} + \text{ext. tco (max)}$$

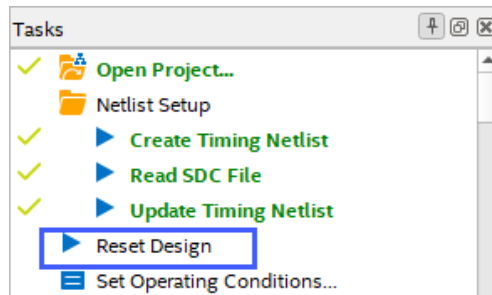
$$\text{Input delay (min)} = \text{board delay (min)} - \text{clock skew (max)} + \text{ext. tco (min)}$$



*Recommendation: use the Tcl **set** and **expr** commands to use variables and have the values calculated by the tool. Of course, you can manually calculate the values yourself and enter them into the constraints directly.*

\_\_\_ 3. Save top.sdc.

*Since you have changed your SDC file, you should reset the design and re-run your reports to include the changed constraints.*



\_\_\_ 4. In the **Tasks** pane, double-click on **Reset Design**.

\_\_\_ 5. Double-click on the following reports to check your work:

- **Report Ignored Constraints** – to make sure you did not type anything incorrectly
- **Report Unconstrained Paths** – to see what has not been constrained yet
- **Report SDC** – to verify your constraints and their values have been accepted

\_\_\_ 6. If you find any errors, you should fix them, reset the design and re-run the reports until the constraints appear correct to you.

Unconstrained Paths Summary			
Show:	Visible ▾	Hide	Q <<Filter>> ...
	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.*

## Step C: 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.

Estimated PCB data trace delay (between devices) (ns)	$t_h = 1.0$	$t_{su} = 1.0$
--	-------------	----------------

	Minimum	Maximum
Estimated PCB data trace delay (between devices) (ns)	0.35	0.6
Board clock skew (ns)	-0.35	0.35

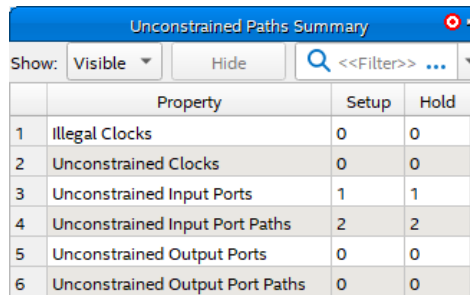
1. Use this table along with the schematic to **fully constrain all output data ports** with respect to **clk\_out\_vir**, using these system parameters. This can be done with **4 commands (2 for multout\_ab and 2 for multout\_xy)** if you use wildcards. To refresh your memory, the equations for doing this are shown below:

<b>Output delay (max) = board delay (max) – clock skew (min) + tsu)</b>
<b>Output delay (min) = board delay (min) – clock skew (max) - th</b>

Remember that if data is latched at the external device by the falling edge of the clock on the output, you need to use the **-clock\_fall** argument.

2. Save top.sdc.  
Since you have changed your SDC file, you should reset the design and re-run your reports to include the changed constraints.
3. In the Tasks pane, double-click on **Reset Design**.

- \_\_\_\_ 4. Double-click on the following reports to check your work:
- **Report Ignored Constraints** – to make sure you did not type anything incorrectly
  - **Report Unconstrained Paths** – to see what has not been constrained yet
  - **Report SDC** – to verify your constraints and their values have been accepted
- \_\_\_\_ 5. If you find any errors, you should fix them, reset the design and re-run the reports until the constraints appear correct to you.



	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. 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 Intel® Quartus® Design 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.*

## END OF EXERCISE

# Timing Analysis: Timing Exceptions & Analysis

**Objectives:**

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

**Step A: 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 ddesign 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 is truly an 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 that 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.


Show: Visible ▾ Hide 🔍 <<Filter>>			
	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 B: Run compilation using SDC file

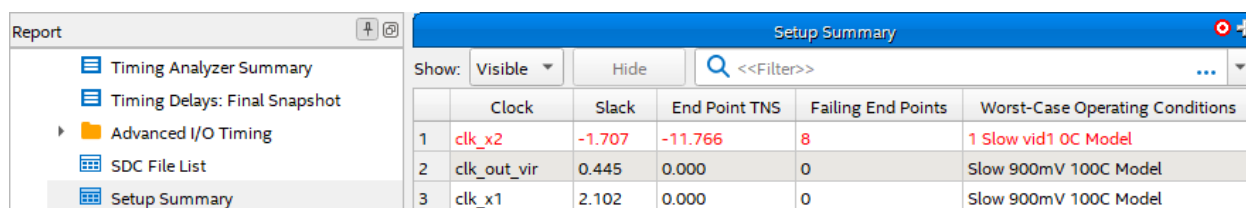
Now you'll run a full compilation in the Intel® Quartus® Prime Pro software using the constraints stored in the SDC file.

The SDC file was automatically added to the project when you created it, so you just need to launch the compile.

1. In the Intel Quartus Prime Pro software, go to the **Compilation Dashboard** and click on **Compile Design**  button at the top.

Compile time is ~5 minutes.

When the compilation is completed, the Timing Analyzer automatically opens with the timing netlist created, the project SDC file read and the timing netlist updated. This means it is ready for you to just begin running reports.

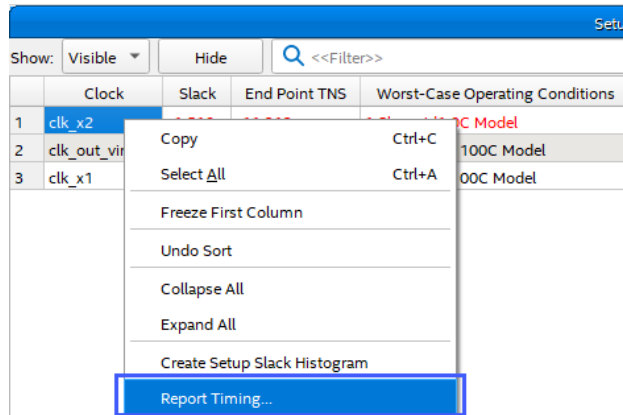


	Clock	Slack	End Point TNS	Failing End Points	Worst-Case Operating Conditions
1	clk_x2	-1.707	-11.766	8	1 Slow vid1 0C Model
2	clk_out_vir	0.445	0.000	0	Slow 900mV 100C Model
3	clk_x1	2.102	0.000	0	Slow 900mV 100C Model

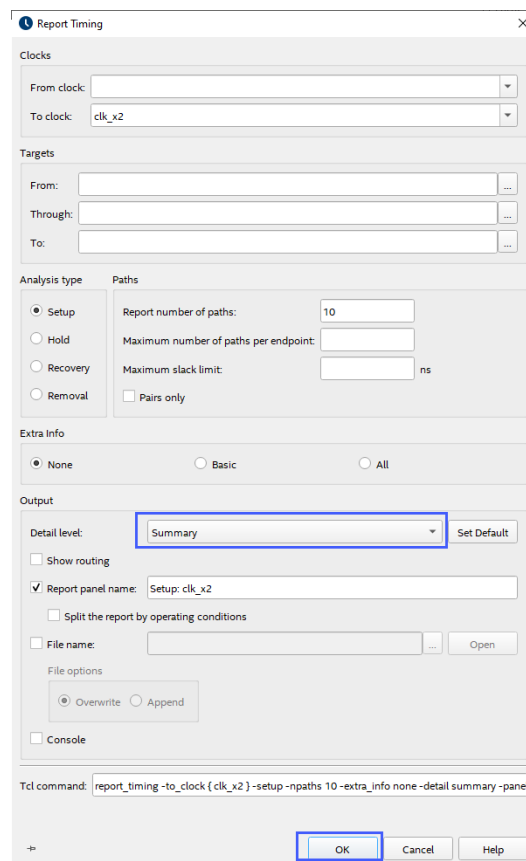
Already you can see that you have a **Setup timing failure in the clk\_x2 domain** when the **0°C slow model** is used. You can see that clock clk\_x2 is failing timing by greater than 1 ns (your value might be slightly different), but that's about all the information the Setup Summary report in the Compilation Report gives us. There is no indication in any of the reports here what path(s) is/are failing.

## Step C: Analyze clk\_x2 clock domain using Timing Analyzer reports

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 Timing Analyzer GUI.



1. In the Summary Setup table, right-click clk\_x2 and select Report Timing.



2. In the **Report Timing** dialog box, select **Summary** in the **Output → Detail Level** drop-down menu. Click **OK**.

Setup: clk_x2								
Show:	Visible	Hide	<<Filter>>					
	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew	Data Delay
1	-1.707	y_regtwo[2]	inst[lpm_mult_0 lpm_mult_component auto_generated dataa_input_reg[2]	clk_x1	clk_x2	5.000	-0.128	6.549
2	-1.620	y_regtwo[4]	inst[lpm_mult_0 lpm_mult_component auto_generated dataa_input_reg[4]	clk_x1	clk_x2	5.000	-0.128	6.462
3	-1.485	y_regtwo[6]	inst[lpm_mult_0 lpm_mult_component auto_generated dataa_input_reg[6]	clk_x1	clk_x2	5.000	-0.110	6.345
4	-1.474	y_regtwo[3]	inst[lpm_mult_0 lpm_mult_component auto_generated dataa_input_reg[3]	clk_x1	clk_x2	5.000	-0.125	6.319
5	-1.413	y_regtwo[5]	inst[lpm_mult_0 lpm_mult_component auto_generated dataa_input_reg[5]	clk_x1	clk_x2	5.000	-0.113	6.270
6	-1.377	y_regtwo[7]	inst[lpm_mult_0 lpm_mult_component auto_generated dataa_input_reg[7]	clk_x1	clk_x2	5.000	-0.107	6.240
7	-1.368	y_regtwo[0]	inst[lpm_mult_0 lpm_mult_component auto_generated dataa_input_reg[0]	clk_x1	clk_x2	5.000	-0.123	6.215
8	-1.351	y_regtwo[1]	inst[lpm_mult_0 lpm_mult_component auto_generated dataa_input_reg[1]	clk_x1	clk_x2	5.000	-0.118	6.203
9	1.977	inst1	inst[lpm_mult_0 lpm_mult_component auto_generated dataa_input_reg[1]	clk_x2	clk_x2	5.000	0.042	3.035
10	2.318	inst1	inst[lpm_mult_0 lpm_mult_component auto_generated dataa_input_reg[2]	clk_x2	clk_x2	5.000	0.064	2.716

3. Notice the pattern in the names of the failing registers.

From node names	
To node names	

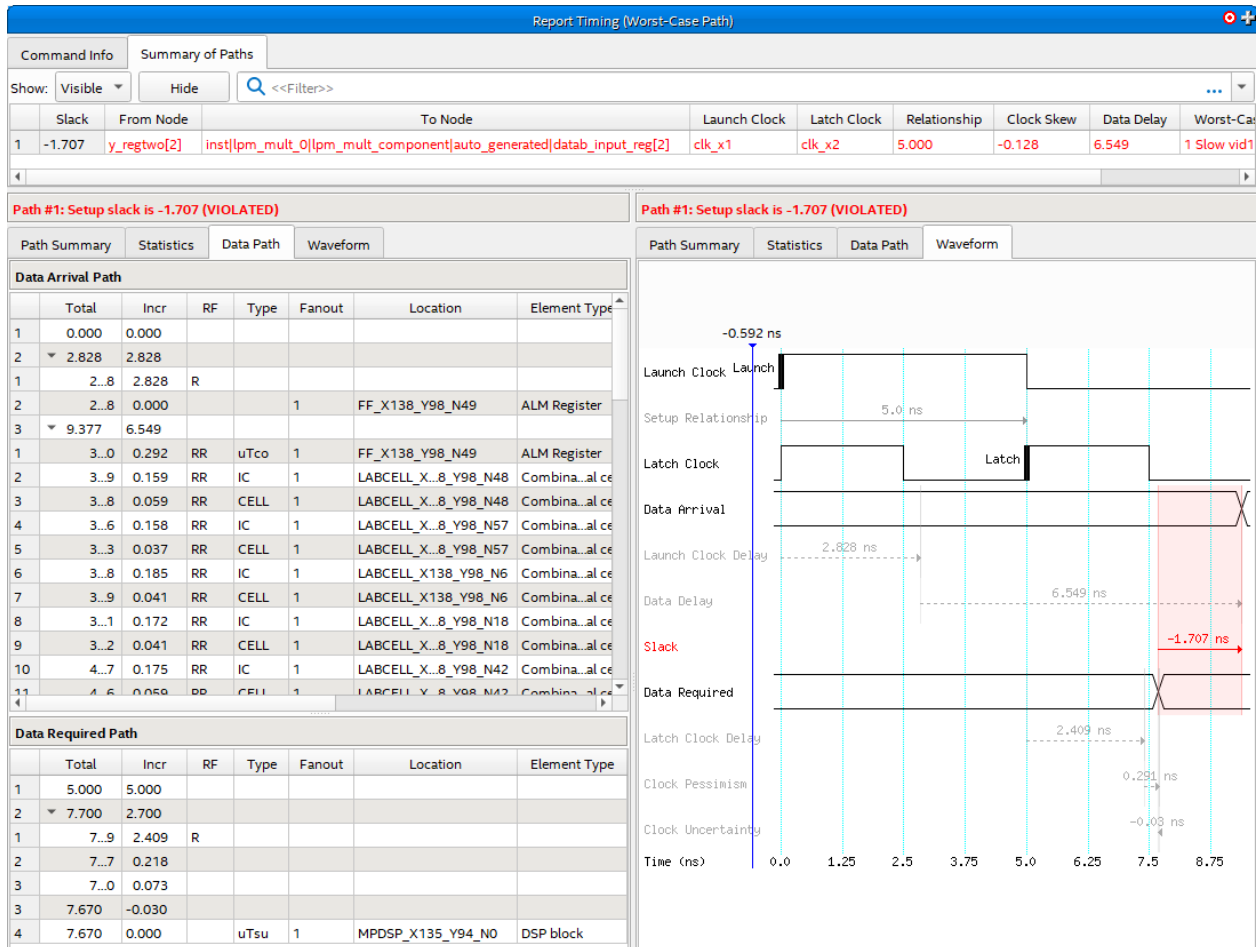
4. Examine in detail the **worst slack timing path** for **clk\_x2**. **Select the worst slack** in the list (first row), right-click, and select **Report Worst-Case Path**.

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

5. 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.*





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

6. Analyze the design to determine how to fix the violations. Bring the top.bdf schematic to the foreground or review the schematic at the beginning of the Clocks exercise.

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 select 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 the `mux_sel` is high or when the `mux_sel` is low that is muxing the proper mux input to the output.

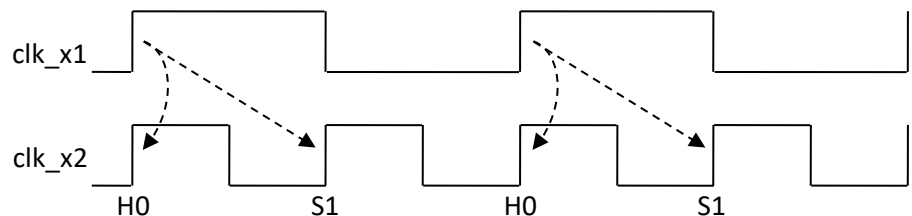
The Timing Analyzer automatically **assumes** data from all the registers must reach the multiplier **within one cycle of the `clk_x2` clock** (the first 5 ns or when `clk_x1` is high). But this is only true for the outputs of **`a_regtwo`** and **`b_regtwo`**. The **`x_regtwo`** and **`y_regtwo`** data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

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

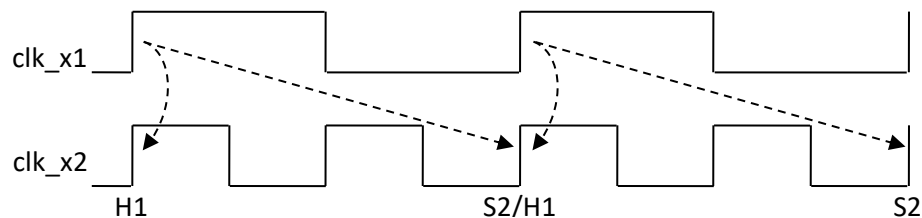
1. In `top.sdc`, add an SDC command to make the **paths from the `x_regtwo*` and `y_regtwo*` registers** have a **setup multicycle path of 2** (no `-to` argument is required). The multicycle assignment should be based on the destination clock edges.

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

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



S1 – Default setup edge  
H0 – Default hold edge



S2 – Setup edge for setup multicycle of 2  
H1 – Hold edge for hold multicycle of 1

*These setup and hold multicyle 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 and check your ignored constraints and unconstrained paths reports.
- \_\_\_ 5. Double-click on the **Report All Summaries** task (under **Macros**).

**Setup Summary**

	Clock	Slack	End Point TNS	Failing End Points	Worst-Case Operating Conditions
1	clk_out_vir	1.075	0.000	0	Slow 900mV 100C Model
2	clk_x1	1.220	0.000	0	Slow 900mV 100C Model
3	clk_x2	2.219	0.000	0	Slow 900mV 100C Model

**Hold Summary**

	Clock	Slack	End Point TNS	Failing End Points	Worst-Case Operating Conditions
1	clk_x2	0.000	0.000	0	1 Slow vid1 100C Model
2	clk_x1	0.140	0.000	0	Slow 900mV 0C Model
3	clk_out_vir	2.248	0.000	0	Fast 900mV 0C Model

*All clock domains and reports (the Setup and Hold Summaries are shown above) should now be shown in black to indicate they are meeting timing.*

*If the design still had not met timing, you would then recompile in the Intel Quartus Prime software to allow the compiler a chance to produce a better solution with the updated constraints. But, since all tests are passing, your timing analysis is finished!*

## END OF EXERCISE

## **Legal Disclaimers/Acknowledgements**

- Intel technologies may require enabled hardware, software or service activation
- No product or component can be absolutely secure
- Your costs and results may vary
- Altera, APEX, Arria, Avalon, Cyclone, eASIC, easicopy, Enpirion, Hyperflex, Intel, the Intel logo, Intel Agilex, Intel Atom, Intel Core, Intel Inside, the Intel Inside logo, Intel Optane, Intel Xeon Phi, MAX, Nios, OpenVINO, the OpenVINO logo, Pentium, Quartus, Stratix, the Stratix logo, and Xeon are trademarks of Intel Corporation or its subsidiaries. OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos
- \*Other names and brands may be claimed as the property of others