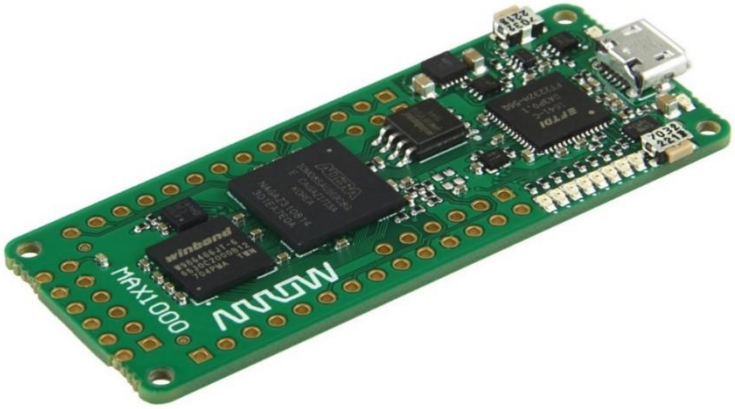
# MAX1000

# Timing Closure Lab




**Software and hardware requirements to complete all exercises**

**Software Requirements:** Quartus® Prime Lite or Standard Edition version 18.0 or 18.1

**Hardware Requirements:** ARROW MAX1000 Board

Revision 1.0

# 1. Introduction

This tutorial provides comprehensive information to help you understand how to set and use your FPGA design environment to ensure that your design meets the timing requirements. The TimeQuest timing analyzer is a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in a design using industry standard constraint, analysis, and reporting methodology.

**Lab Notes:** Many of the names that the lab asks you to choose for files, components, and other objects in this exercise must be spelled exactly as directed. This nomenclature is necessary because the pre-written software application includes variables that use the names of the hardware peripherals. Naming the components differently can cause errors. There are also other similar dependencies within the project that require you to enter the correct names.

# 2. Getting Started

The first objective is to ensure that you have all the necessary hardware items and software installed so that the lab can be completed successfully. Below is a list of items required to complete this lab:

- MAX1000 Board **(**10M08SAU169C8G)
- USB Cable
- Lab files: Timing_Closure_lab_template: Template files are required to complete the lab. Includes: top.vhd, fibonacci.vhd, timing_closure_lab_pinout.csv
- Quartus Prime 18.0 Lite was used for this lab. Previous/newer versions should work (If no Quartus Prime is installed, refer to MAX1000 User Guide for instructions)
- Installed Arrow USB Drivers (If not, refer to MAX1000 User Guide for instructions)
- Personal computer or laptop running 64-bit Linux / Windows 7 or later with at least an Intel i3 core (or equivalent), 4GB RAM and 12 GB of free hard disk space
- Basic knowledge of VHDL
- A desire to learn!

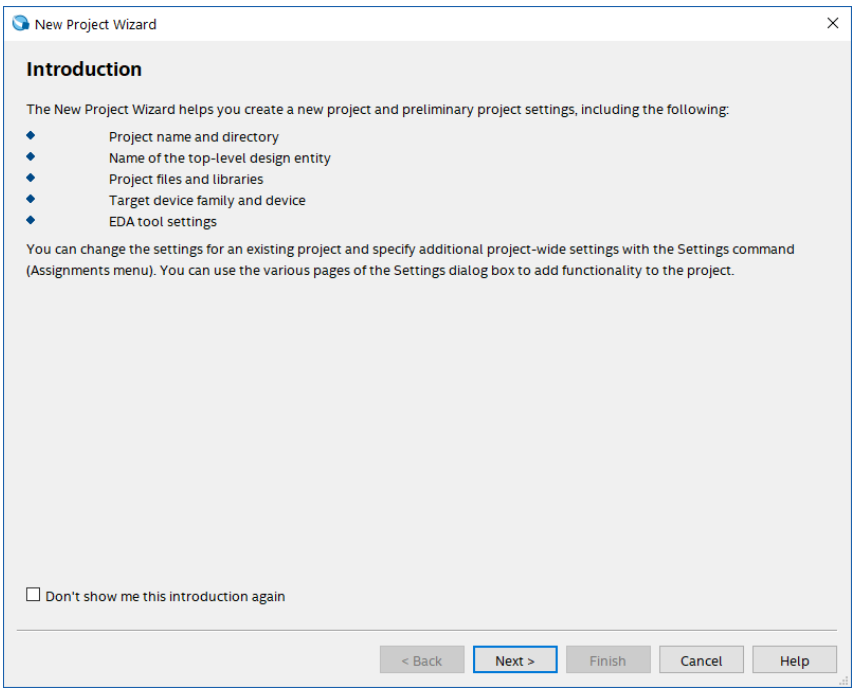
# 3. Project with MAX1000

## 3.1 Quartus Prime project

### 3.1.1 Create a new Quartus Prime project

3.1.1.1 If not already open, from the Start menu or the Desktop, open the Quartus Prime 18.0 Lite software.

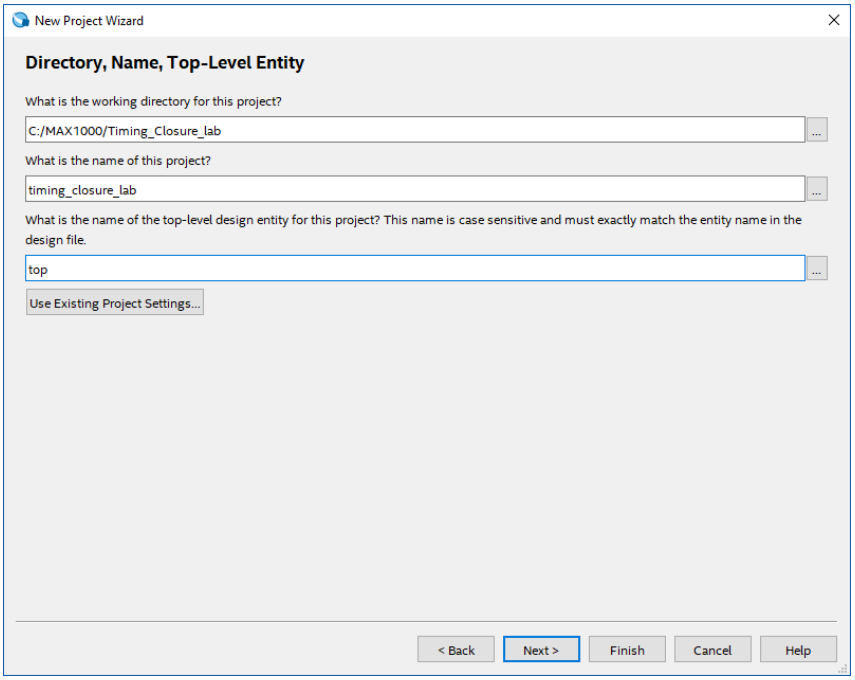
3.1.1.2 Create a new project using the New Project Wizard: **File → New Project Wizard**.




3.1.1.3 Click **Next**.

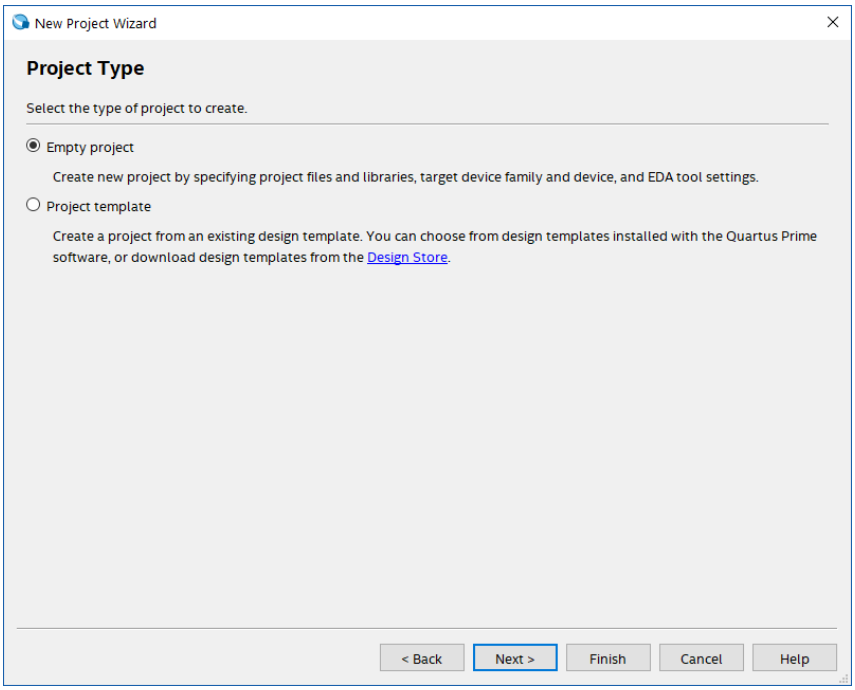
3.1.1.4 Configure the New Project Wizard directory, name and top-level entity information:

- Enter a directory in which you will store your Quartus project files for this design, for example, **C:/MAX1000/Timing_Closure_lab**
- Specify the name of the project: **timing_closure_lab**
- Specify the name of the top-level entity: **top**



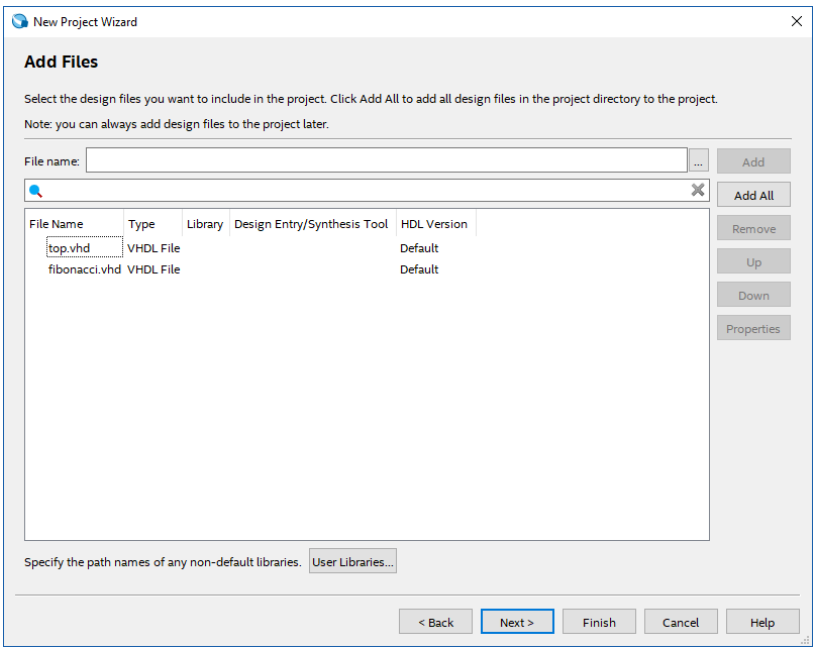

3.1.1.5 Click **Next**.

3.1.1.6 On the Project Type page, select **"Empty project"** and click **Next**.



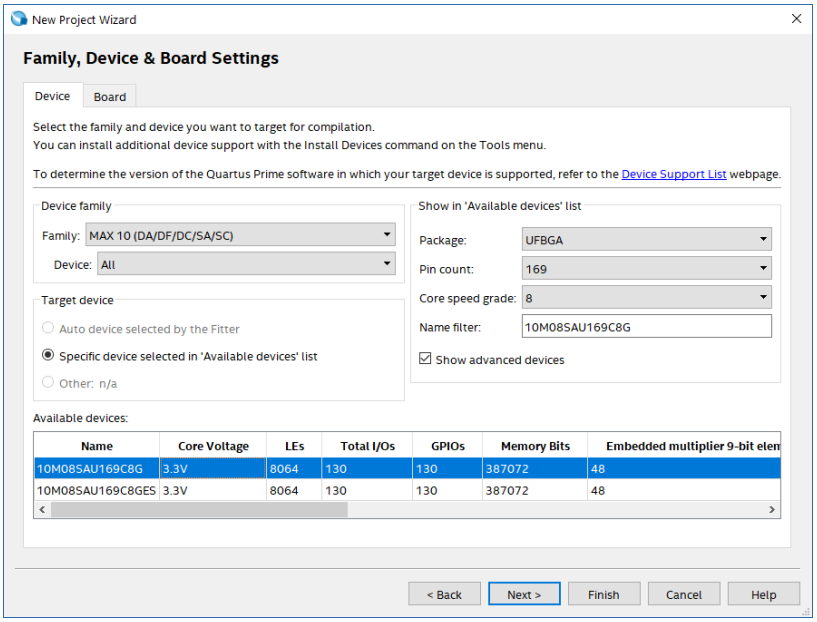
3.1.1.7 On the Add Files page, add source files to the project by clicking on the button and browse into the lab files folder where you will locate the provided design files and add:

- **top.vhd**
- **fibonacci.vhd**




3.1.1.8 Click **Next**.

3.1.1.9 Specify Family and Device Settings. Use pull-down menus to select MAX10 family or enter the part number in the Name Filter text box. The part number is **10M08SAU169C8G**.




3.1.1.10 Click **Finish**.

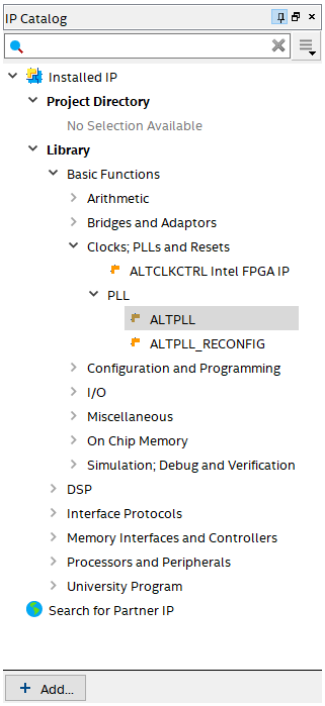
## 3.2 Design entry

**Overview:** In this module you will complete the design with missing components, elements.

### 3.2.1 Add PLL to the Quartus project

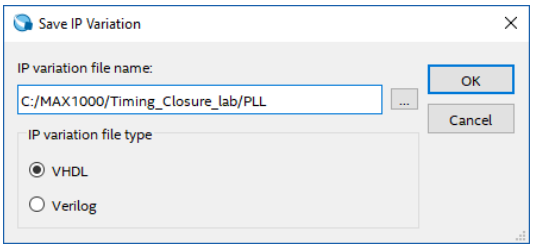
3.2.1.1 From the IP Catalog panel on the left side, expand the menus for the **Basic Functions → Clocks; PLLs and Resets → PLL** and double click on **ALTPLL**.

If the IP catalog is not visible, then right click on the toolbar and select IP catalog.



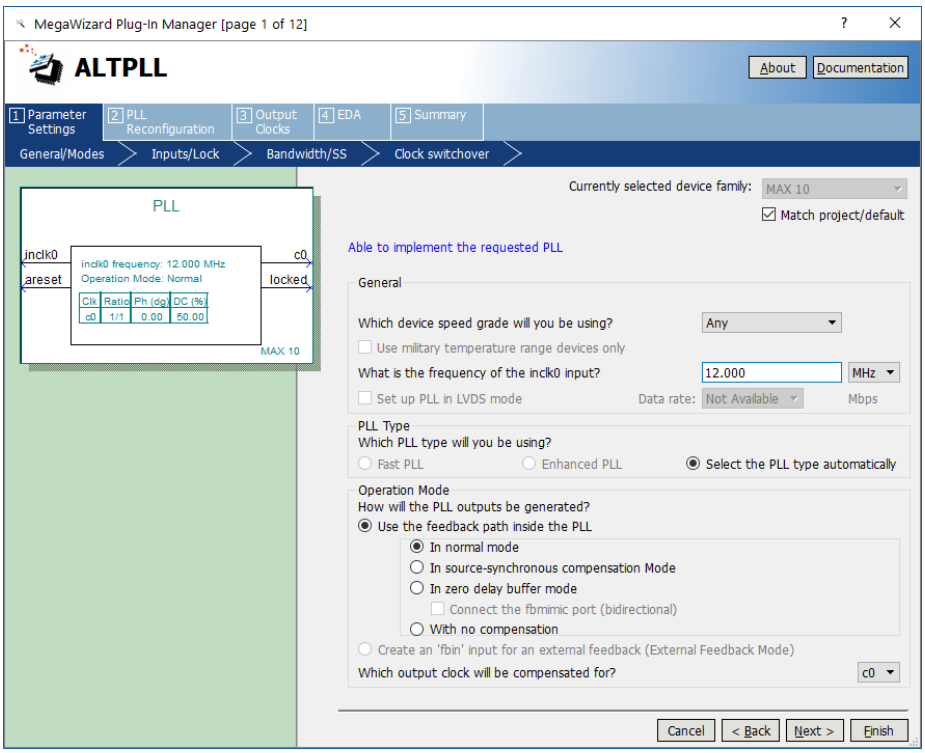

3.2.1.2 On the Save IP Variation window, enter the following information.

- IP variation file name: **<project_directory>/PLL**
- IP variation file type: **VHDL**



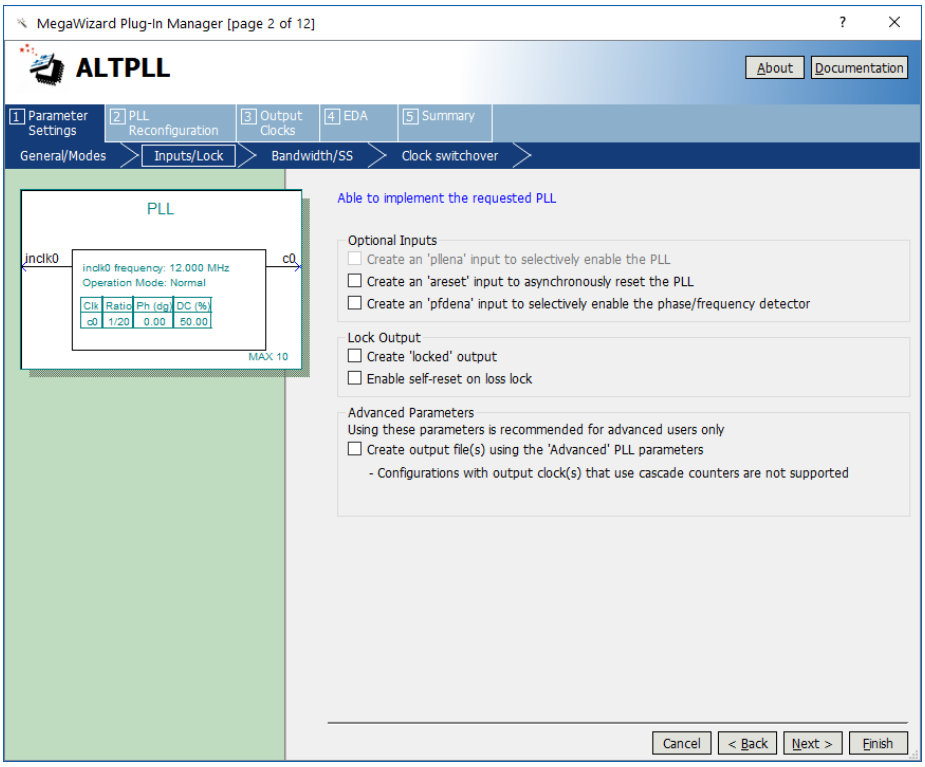

3.2.1.3 Click **OK**.

3.2.1.4 Under General/Modes tab (page 1 of 12) of PLL MegaWizard change the frequency of clock input to **12 MHz.** This source is provided by the internal oscillator in the MAX10 FPGA.



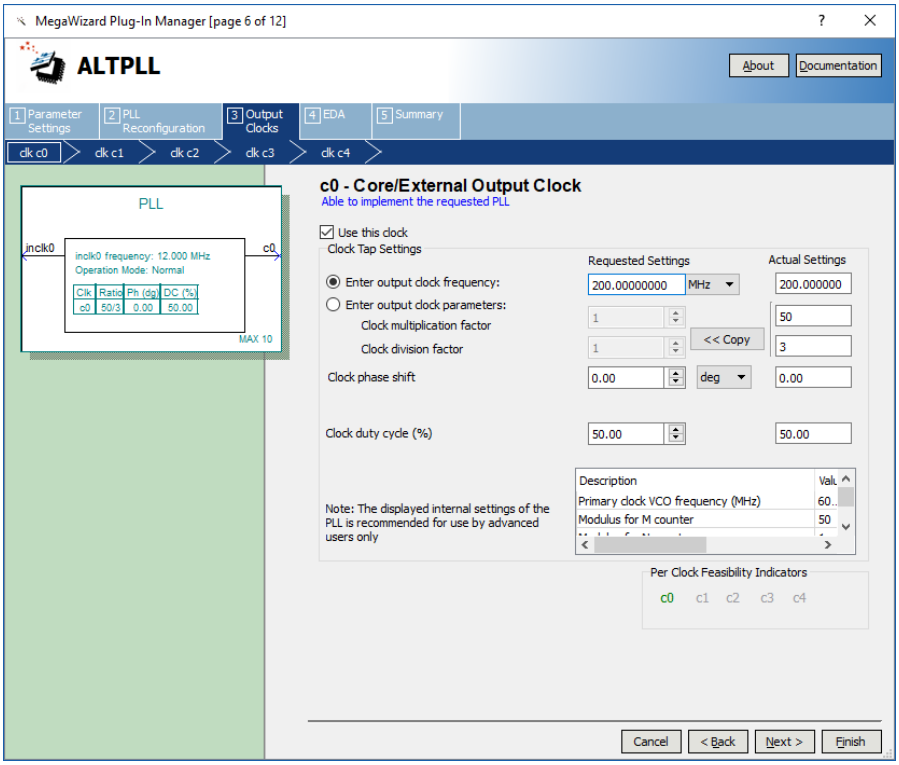

3.2.1.5 Click **Next**.

3.2.1.6 Under Input/Lock tab (page 2 of 12) uncheck 'areset' input and locked output option.



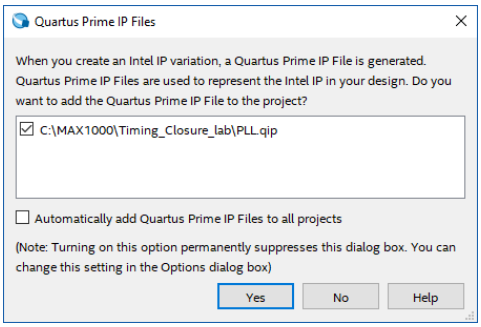
3.2.1.7 Click **Next** until you reach the **Output Clocks** tab (page 6 of 12).

3.2.1.8 Under the clk c0 tab (page 6 of 12) select "Enter output clock frequency" and enter **200 MHz**.




3.2.1.9 Click **Finish.** This will take you to the Summary tab (page 12 of 12). Click **Finish** again to close ALTPLL MegaWizard Manager.

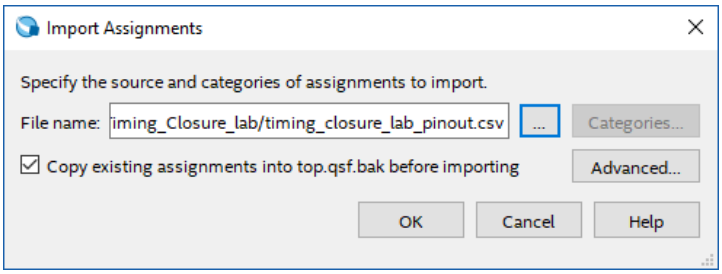
3.2.1.10 In the pop-up Quartus Prime IP Files accept all defaults and click **Yes**.



### 3.2.2 Import pin assignments

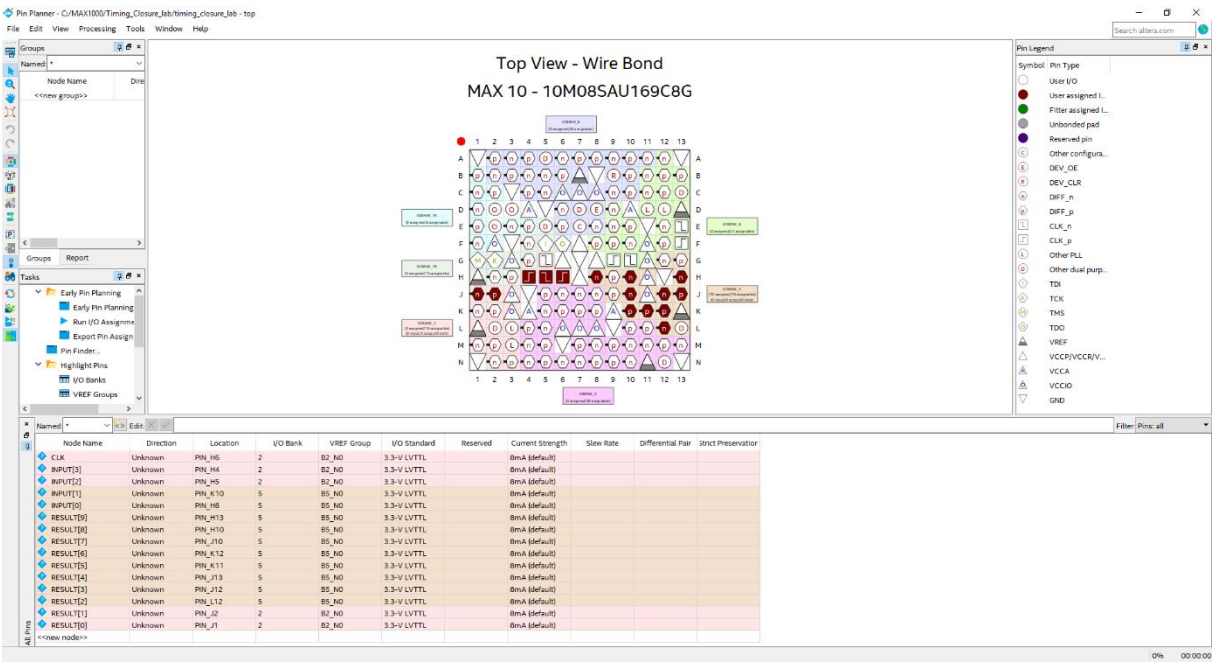
3.2.2.1 Select **Assignments → Import Assignments…**

3.2.2.2 Add source file by clicking on the ... button and browse into the lab file folder where you will locate the provided design files and add **timing_closure_lab_pinout.csv**.




3.2.2.3 Press **OK**.

3.2.2.4 Open **Pin Planner** by clicking on button on the toolbars, or **Assignments → Pin Planner** in order to check the import. In the Pin Planner you should see the following:



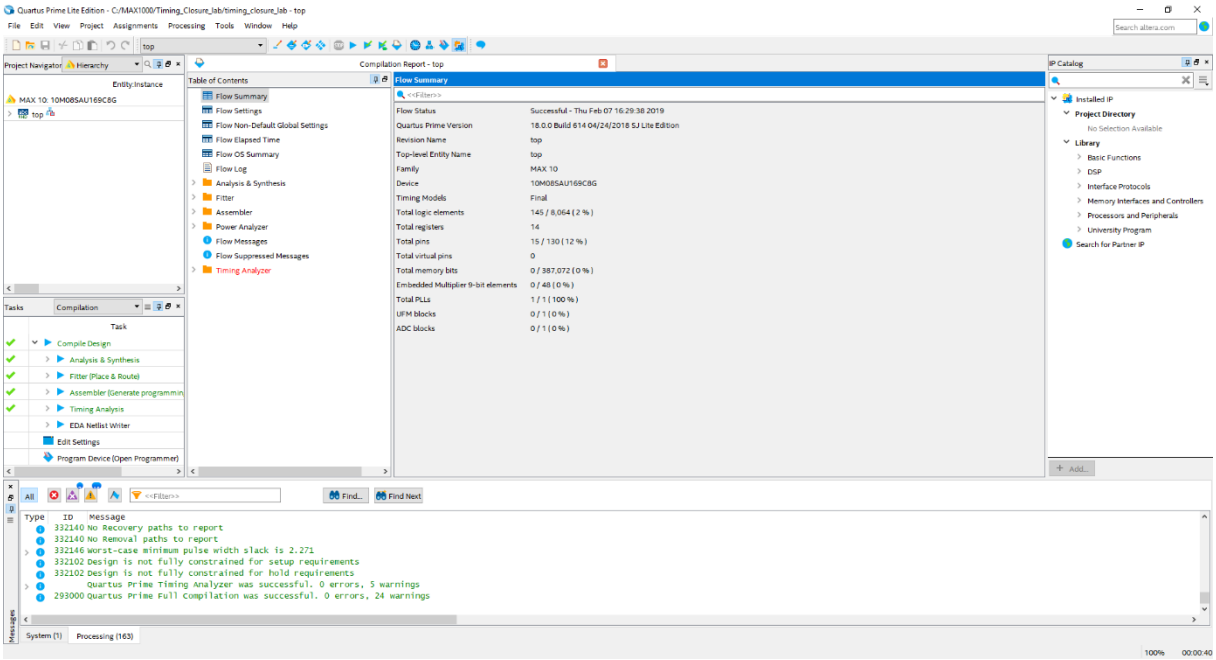

3.2.2.5 **Close** the Pin Planner.

## 3.3 Timing check

### 3.3.1 Compiling the design

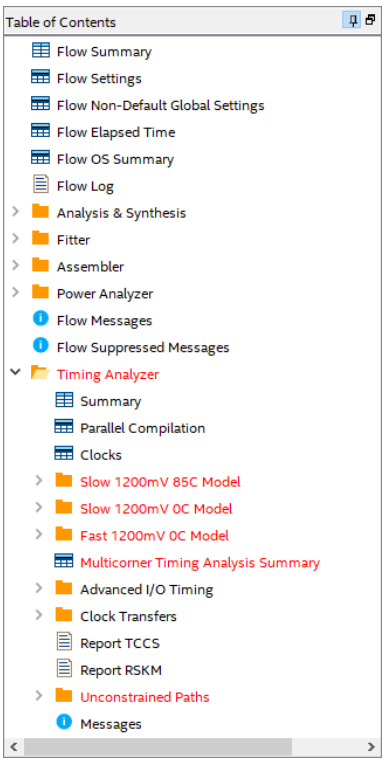
3.3.1.1 Start Compilation by clicking on button on the toolbars, or **Processing → Start Compilation**.

There should be no errors. If there are errors, they should be fixed before re-compilating. The 100% in the lower right corner or a green checkmark next to the Compile Design in the Compilation task window indicates that the compilation was successful.



### 3.3.2 Timing Analyzer report

3.3.2.1 In the Compilation Report window, expand Timing Analyzer folder. The red highlight shows the problematic areas, which are the unconstrained path and the operating frequency.




3.3.2.2 Expand **Slow 1200mV 85C Model** folder and click on **Setup Summary**. Here you can see the slack of clock-controlled signal.

The Slow 1200mV 85C timing model provides worst-case conditions for the FPGA operating. This condition is ideal for performing a setup check in static timing analysis.

Slow 1200mV 85C Model Setup Summary

<<Filter>>

| | Clock | Slack | End Point TNS |
|---|---|---|---|
| 1 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | -16.540 | -142.030 |

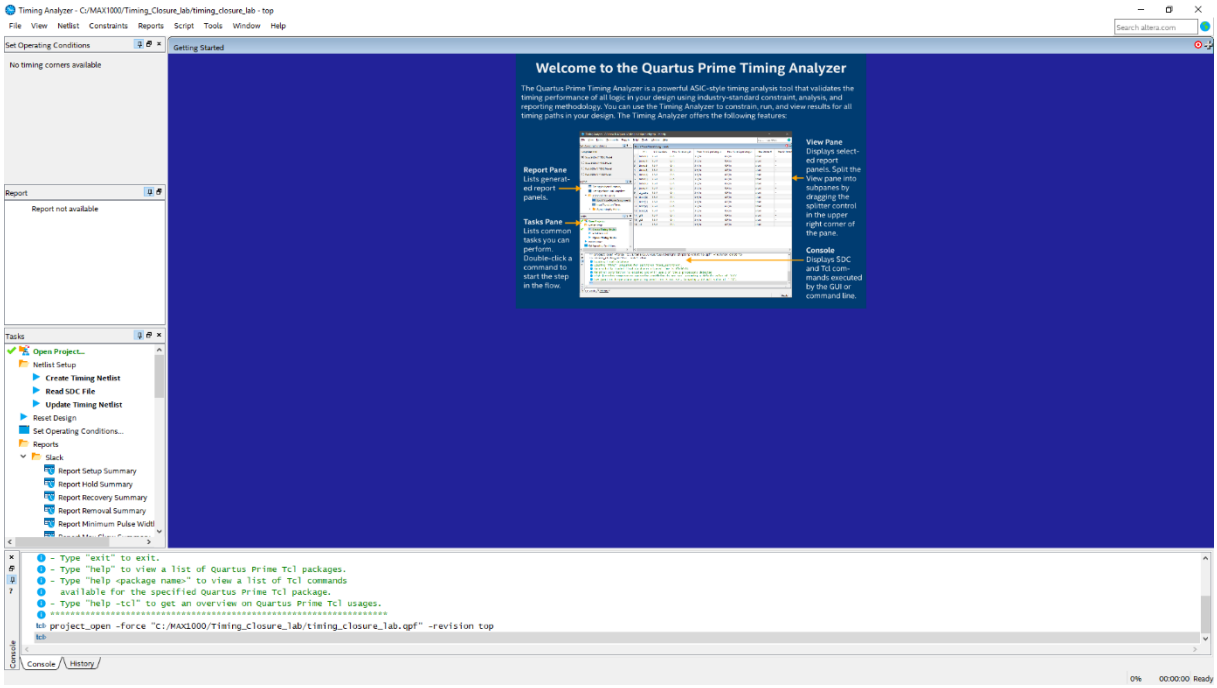
3.3.2.3 Click on **Fmax Summary** to see the maximum frequency for the current design. Our design should run on 200MHz, so we will fix it in in the following steps.

Slow 1200mV 85C Model Fmax Summary

<<Filter>>

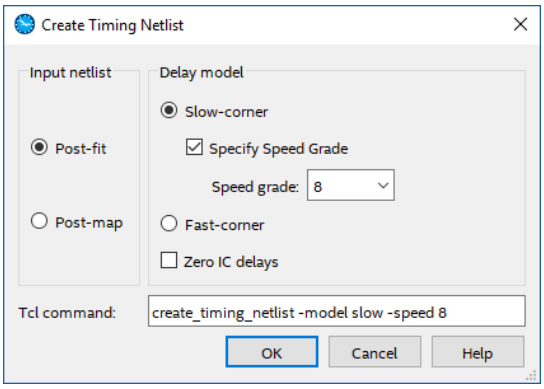
| | Fmax | Restricted Fmax | Clock Name | Note |
|---|---|---|---|---|
| 1 | 46.43 MHz | 46.43 MHz | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | |

3.3.2.4 Open **Timing Analyzer** by clicking on button on the toolbars, or **Tools → Timing Analyzer**.

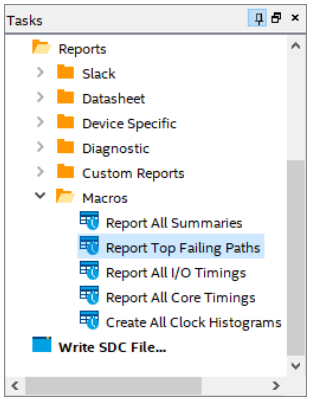



3.3.2.5 Select **Netlist → Create Timing Netlist…** from the menu.

3.3.2.6 Make sure, that the **Post-fit** is chosen for Input netlist and check Specify Speed Grade. Set Speed Grade to **8**.



3.3.2.7 In the Task window, browse for **Reports\Macros** and double click on **Report Top Failing Path**.

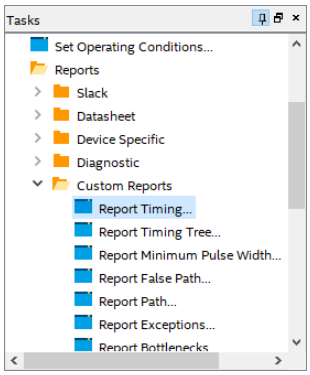



3.3.2.8 In this report you can see the 200 worst-performing paths.

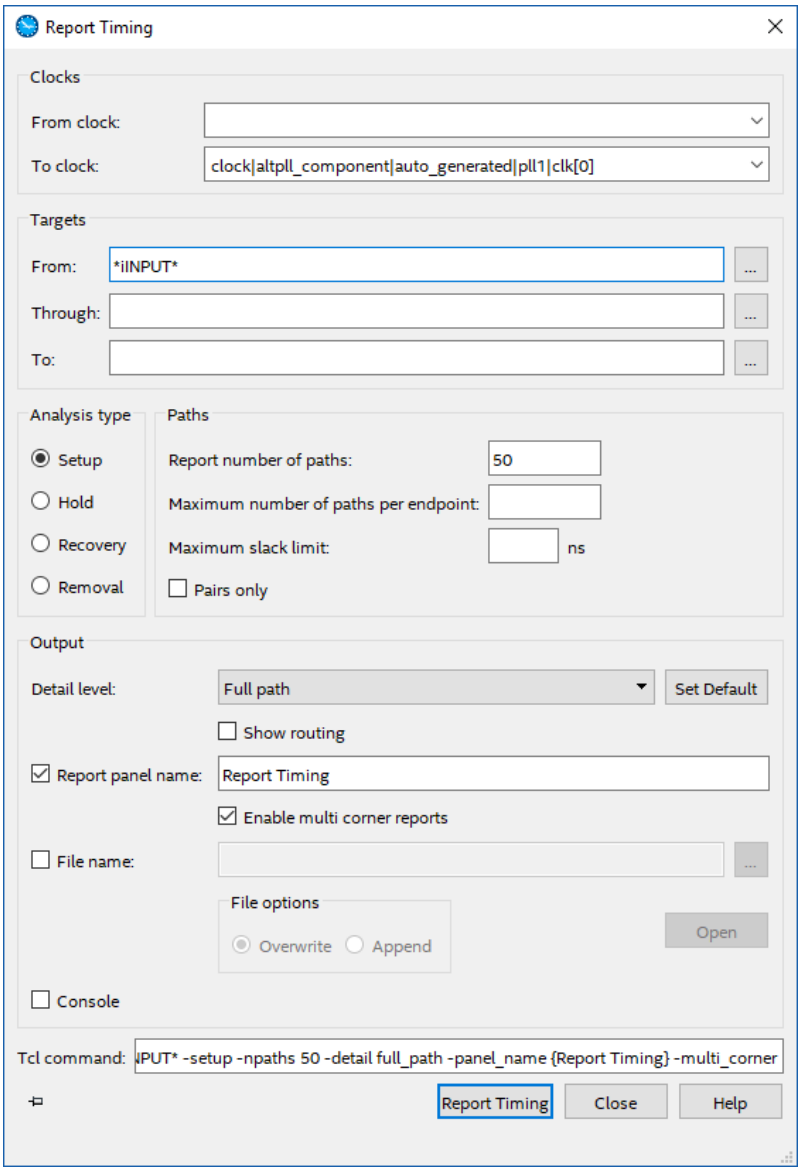
Slow 1200mV 85C Model

| | Slack | From Node | To Node | Launch Clock | Latch Clock | Relationship | Clock Skew | Data Delay |
|---|---|---|---|---|---|---|---|---|
| 1 | -16.540 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.851 |
| 2 | -16.521 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.832 |
| 3 | -16.493 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.804 |
| 4 | -16.474 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.785 |
| 5 | -16.447 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.758 |
| 6 | -16.428 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.739 |
| 7 | -16.400 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.711 |
| 8 | -16.396 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.707 |
| 9 | -16.382 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.693 |
| 10 | -16.381 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.692 |
| 11 | -16.380 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.691 |
| 12 | -16.377 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.688 |
| 13 | -16.364 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.675 |
| 14 | -16.363 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.674 |
| 15 | -16.361 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.672 |
| 16 | -16.345 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.656 |
| 17 | -16.336 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.647 |
| 18 | -16.333 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.644 |
| 19 | -16.333 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.644 |
| 20 | -16.317 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.628 |
| 21 | -16.314 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.625 |
| 22 | -16.314 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.625 |
| 23 | -16.314 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.625 |
| 24 | -16.303 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.614 |
| 25 | -16.301 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.612 |
| 26 | -16.295 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.606 |
| 27 | -16.289 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.600 |
| 28 | -16.284 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.595 |
| 29 | -16.282 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.593 |
| 30 | -16.270 | iINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.581 |
| 31 | -16.269 | iINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.580 |

3.3.2.9 In the Tasks window browse for **Reports\Custom Reports** and double click on **Report Timing…** The Report Timing dialog box will appear.

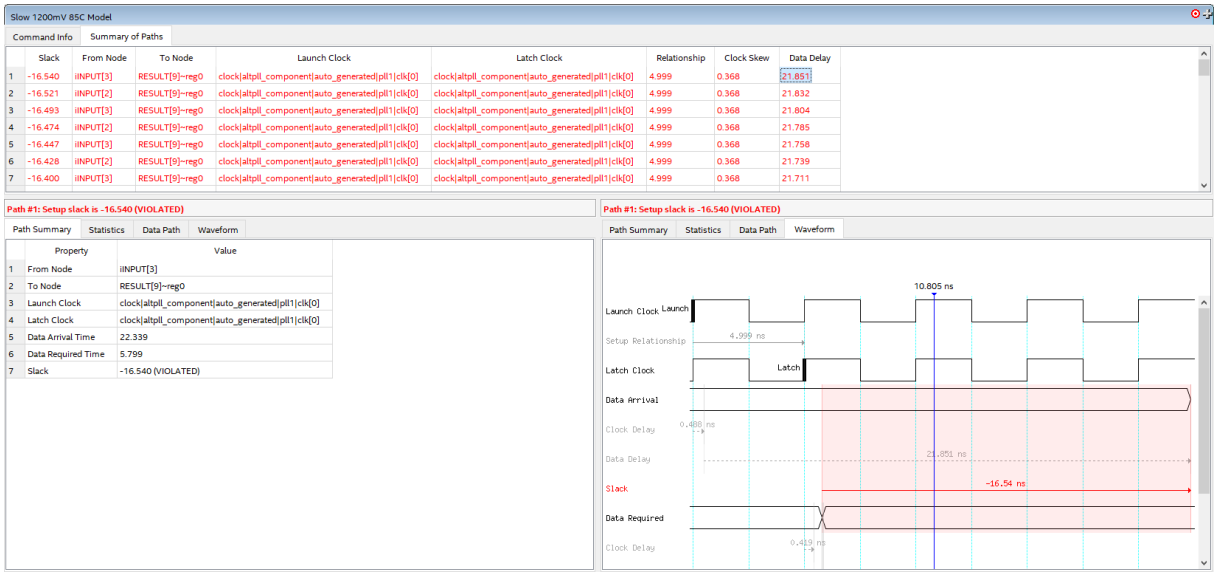


3.3.2.10 Select **clock|altpll_component|auto_generated|pll1|clk[0]** from the drop-down menu for the "To clock". In the target section type ***iINPUT*** for the "From" and set **50** the Report number of paths in the Paths section.




3.3.2.11 Make sure that the Detail level is **Full path** and press **Report Timing**.

3.3.2.12 Here you can see the specific timing check report of the top 50 worst paths. The Data Path tab of detailed report gives the delay break-downs, but there is also useful information in the Path Summary and Statistic tabs, while the Waveform tab is useful to help visualize the Data Path analysis.

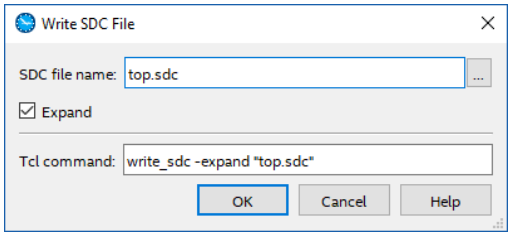



| | Slack | From Node | To Node | Launch Clock | Latch Clock | Relationship | Clock Skew | Data Delay |
|---|---|---|---|---|---|---|---|---|
| 1 | -16.540 | iiINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.851 |
| 2 | -16.521 | iiINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.832 |
| 3 | -16.493 | iiINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.804 |
| 4 | -16.474 | iiINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.785 |
| 5 | -16.447 | iiINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.758 |
| 6 | -16.428 | iiINPUT[2] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.739 |
| 7 | -16.400 | iiINPUT[3] | RESULT[9]~reg0 | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | 4.999 | 0.368 | 21.711 |

| | Property | Value |
|---|---|---|
| 1 | From Node | iiINPUT[3] |
| 2 | To Node | RESULT[9]~reg0 |
| 3 | Launch Clock | clock\|altpll_component\|auto_generated\|pll1\|clk[0] |
| 4 | Latch Clock | clock\|altpll_component\|auto_generated\|pll1\|clk[0] |
| 5 | Data Arrival Time | 22.339 |
| 6 | Data Required Time | 5.799 |
| 7 | Slack | -16.540 (VIOLATED) |

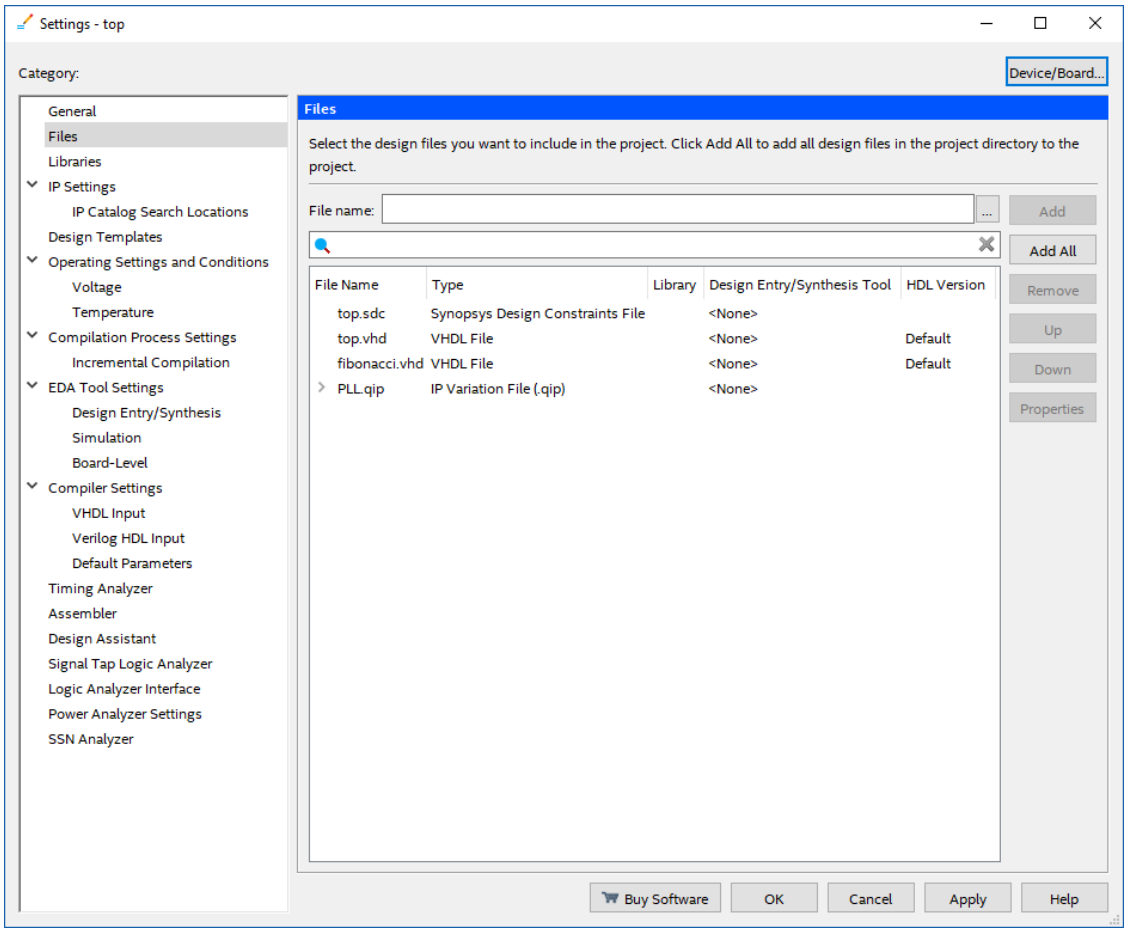
3.3.2.13 Select **Constraints → Write SDC File…** from the menu.

3.3.2.14 Type **top.sdc** for SDC file name and press **OK**.



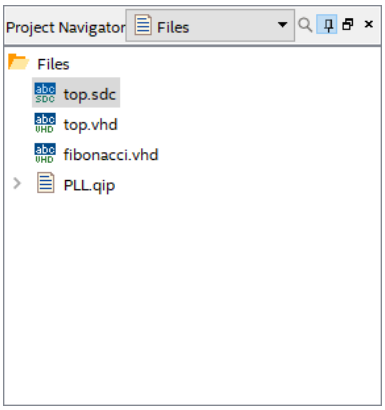

3.3.2.15 In Quartus Prime window, select **Project → Add/Remove Files in Project…**

3.3.2.16 Click on the ... button and add **top.sdc** file to the project.




3.3.2.17 Click **Apply** and **OK**.

3.3.2.18 Choose Files from the drop-down list in Project Navigator, and open **top.sdc**.

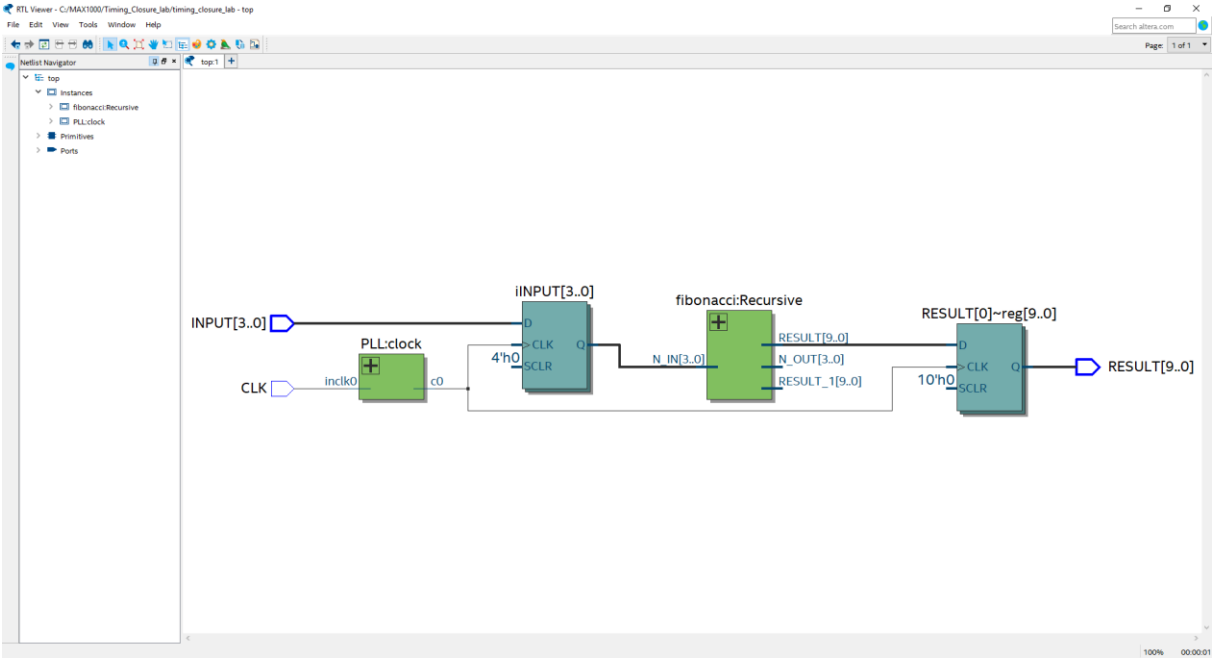



3.3.2.19 You should see that the Timing Analyzer automatically generated clock constrains for the input clock and for the PLL generated clock.

## 3.4 Design modification

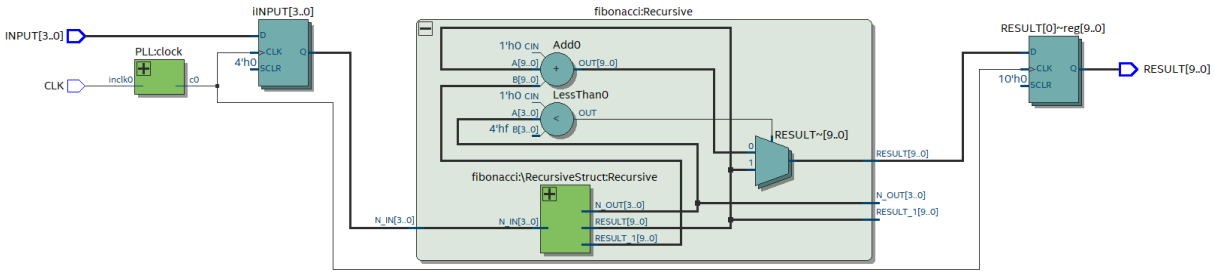
### 3.4.1 RTL Viewer

3.4.1.1 Open RTL Viewer by selecting **Tools → Netlist Viewers → RTL Viewer**.




3.4.1.2 The Timing Analyzer indicates that path between iINPUT and RESULT does not meet with the requirement. Click on  button of Fibonacci:Recursive module to see the internal logic.

**Note:** Because this is a recursive component, it is enough to open it once. The internal modules contain the same logic.




3.4.1.3 In the current design you can see that a big combinatorial logic causes the negative path slack. In order to reduce the slack, we will move some registers into the design in the following steps.

3.4.1.4 **Close** RTL Viewer.

### 3.4.2 Code modification

3.4.2.1 Open **fibonacci.vhd**.

3.4.2.2 Comment from line 72 to line 74 by writing "--" at the beginning of the lines or select the lines and click on button.

3.4.2.3 Add the following output register in the architecture section after the word 'begin'.

```
output_registers : process(CLK)
begin
      if(CLK = '1' and CLK'event) then
            RESULT_1 <= iRES_1;
            if(iN_IN < COUNT) then
                  RESULT <= iRES_1;
            else
                  RESULT <= iSUM;
            end if;
            if(COUNT > 0) then
                  N_OUT <= iN_IN;
            else
                  N_OUT <= N_IN;
            end if;
      end if;
end process;
```

3.4.2.4 Add the following connection to the Recursive : fibonacci association list, around line 66 after port map.

```
CLK => CLK,
```

3.4.2.5 Add the following port to the port list of fibonacci entity around line 34 and to the fibonacci component declaration, around line 54.

```
CLK    : in std_logic;
```

3.4.2.6 **Save** the modification by clicking on button or **File → Save**.

3.4.2.7 Open **top.vhd**.

3.4.2.8 Add the following connection to the Recursive : fibonacci association list, around line 72 after port map.

```
CLK => iCLK,
```

3.4.2.9 Add the following port to the fibonacci component declaration, around line 55.

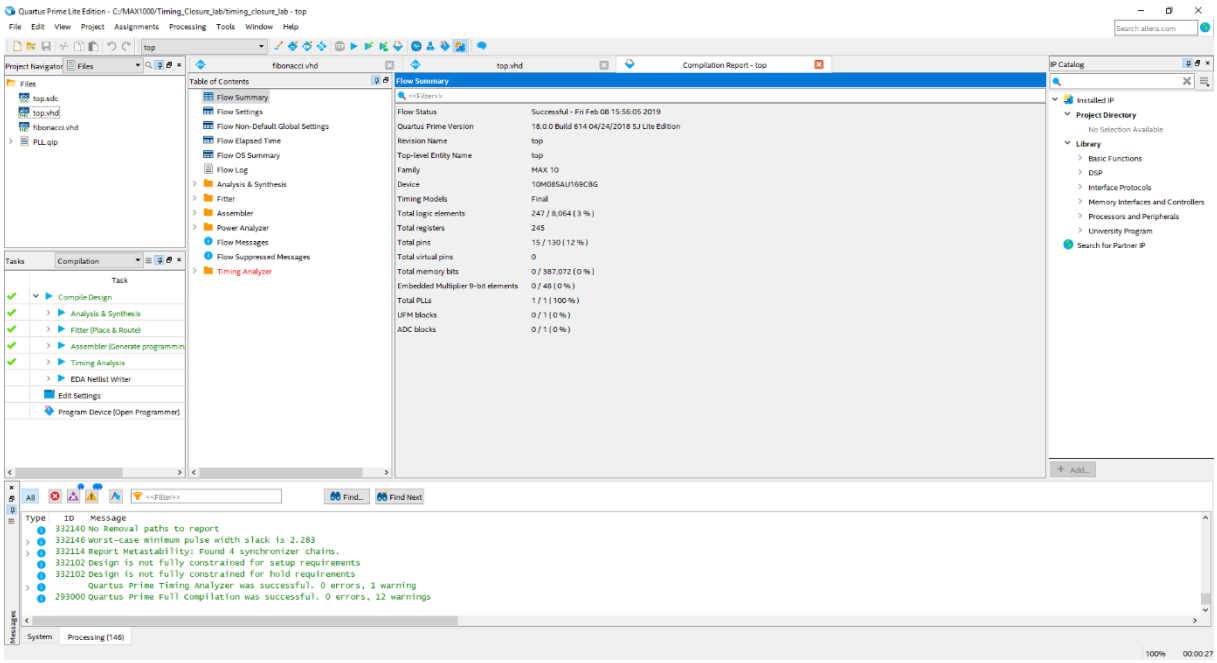
```
CLK    : in std_logic;
```

3.4.2.10 **Save** the modification by clicking on button or **File → Save**.

### 3.4.3 Compiling the design

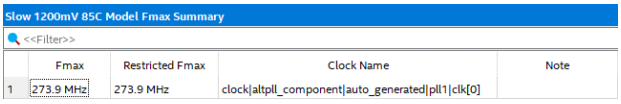
3.4.3.1 Start Compilation by clicking on button on the toolbars, or **Processing → Start Compilation**.

There should be no errors. If there are errors, they should be fixed before re-compilating.




### 3.4.4 Timing Analyzer report

3.4.4.1 In the Compilation Report window, expand **Timing Analyzer\Slow 1200mV 85C Model** folder and click on **Fmax Summary** to see the maximum frequency for the modified design.



Slow 1200mV 85C Model Fmax Summary

| | Fmax | Restricted Fmax | Clock Name | Note |
|---|---|---|---|---|
| 1 | 273.9 MHz | 273.9 MHz | clock\|altpll_component\|auto_generated\|pll1\|clk[0] | |

Our project should run on 200MHz and a new frequency is 273.9MHz, so the code modification was successful. If you would like, you can check the design in the RTL Viewer and compare it with the previous version.

3.4.4.2 The only problematic area is the unconstrained I/O. Expand **Unconstrained Paths** folder and click on **Summary**. In this report you can see how many I/O is not specified.

**Unconstrained Paths Summary**

<<Filter>>

| | Property | Setup | Hold |
|---|---|---|---|
| 1 | Illegal Clocks | 0 | 0 |
| 2 | Unconstrained Clocks | 0 | 0 |
| 3 | Unconstrained Input Ports | 4 | 4 |
| 4 | Unconstrained Input Port Paths | 4 | 4 |
| 5 | Unconstrained Output Ports | 10 | 10 |
| 6 | Unconstrained Output Port Paths | 10 | 10 |

3.4.4.3 To find more information about which I/O is affected and what is wrong, expand **Setup Analysis** folder and double click on **Unconstrained Input Ports**. With the **Unconstrained Output Ports** report you can see the affected output ports.

**Unconstrained Input Ports**

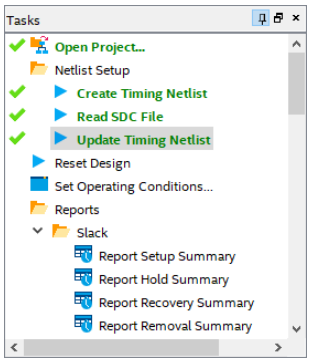
<<Filter>>

| | Input Port | Comment |
|---|---|---|
| 1 | INPUT[0] | No input delay, min/max delays, false-path exceptions, or max skew assignments found |
| 2 | INPUT[1] | No input delay, min/max delays, false-path exceptions, or max skew assignments found |
| 3 | INPUT[2] | No input delay, min/max delays, false-path exceptions, or max skew assignments found |
| 4 | INPUT[3] | No input delay, min/max delays, false-path exceptions, or max skew assignments found |

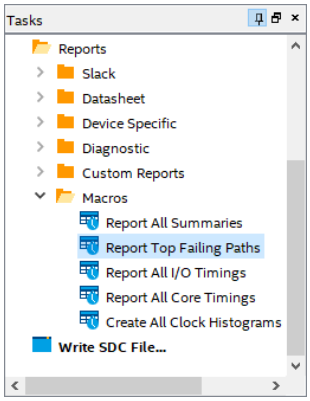
We will constrain these I/Os later.

3.4.4.4 Open **Timing Analyzer** window.

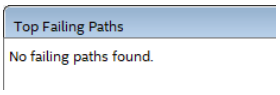
3.4.4.5 Double click on **Update Timing Netlist** in the Task window.



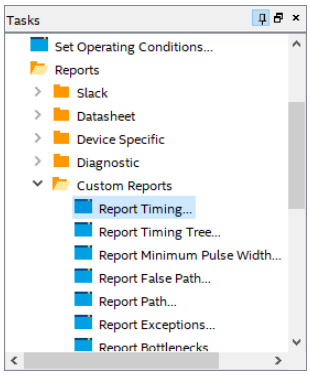
3.4.4.6 In the Task window, browse for **Reports\Macros** and double click on **Report Top Failing Path**.



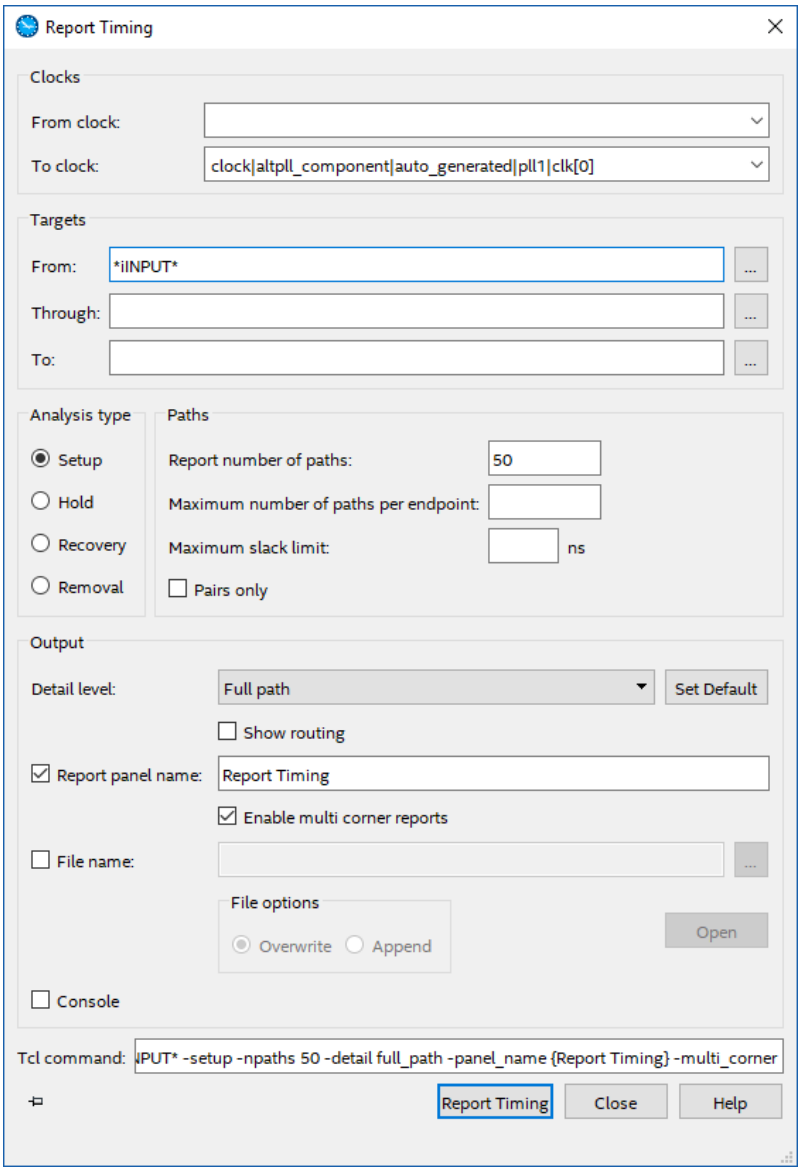

3.4.4.7 You should see "No failing paths found", which means all the paths meet the timing.




3.4.4.8 In the Tasks window browse for **Reports\Custom Reports** and double click on **Report Timing…** The Report Timing dialog box will appear.

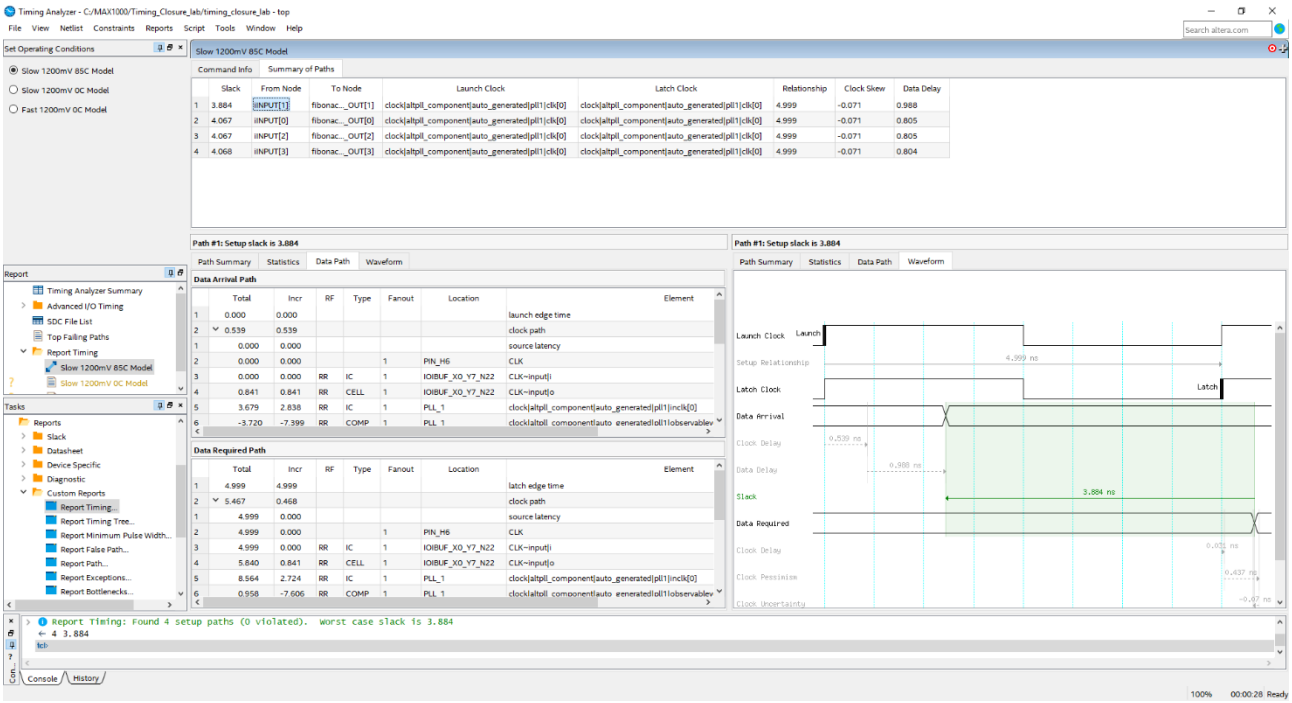


3.4.4.9 Select **clock|altpll_component|auto_generated|pll1|clk[0]** from the drop-down menu for the "To clock". In the target section type ***iINPUT*** for the "From" and set **50** the Report number of paths in the Paths section.




3.4.4.10 Make sure that the Detail level is **Full path** and press **Report Timing**.

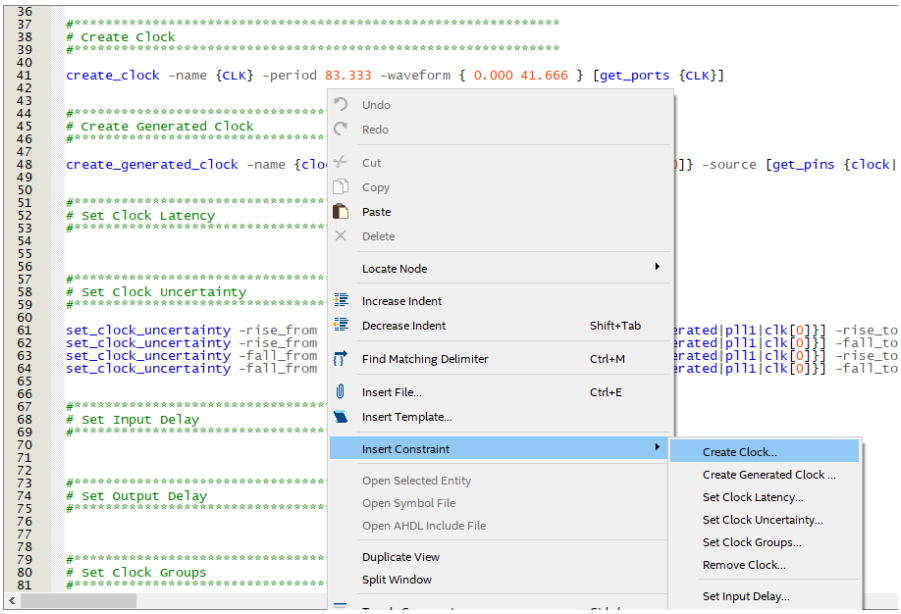
3.4.4.11 In the specific timing check report, you should see that there is not any negative slack, the data arrive in time.



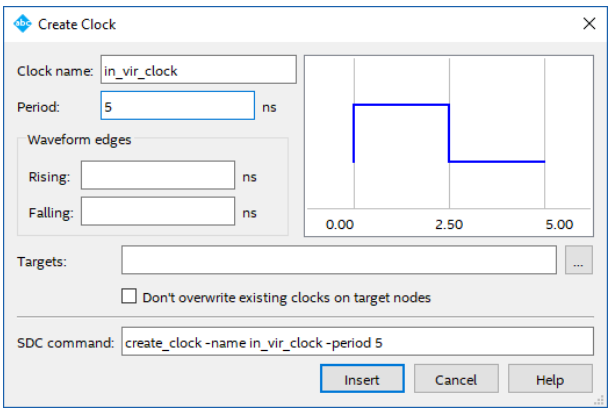
## 3.4.5 Constrain I/O

3.4.5.1 Open **top.sdc** in Quartus Prime.

3.4.5.2 Search for the Create Clock section which is indicated in the commented area, right click in the text editor and select **Insert Constraint → Create Clock…**



3.4.5.3 In the Create Clock dialog box, type **in_vir_clock** for the Clock name and set the period to **5 ns**.

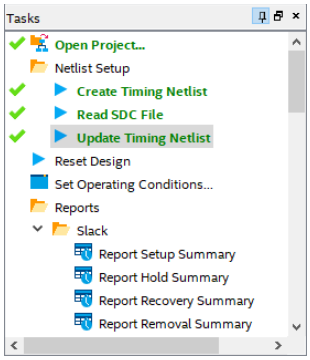



3.4.5.4 Click **Insert** and a new SDC command will be added to your SDC file.

3.4.5.5 Repeat the previous steps from 3.4.5.2 and add a new clock which name is **out_vir_clock**. Its period is **5 ns**.

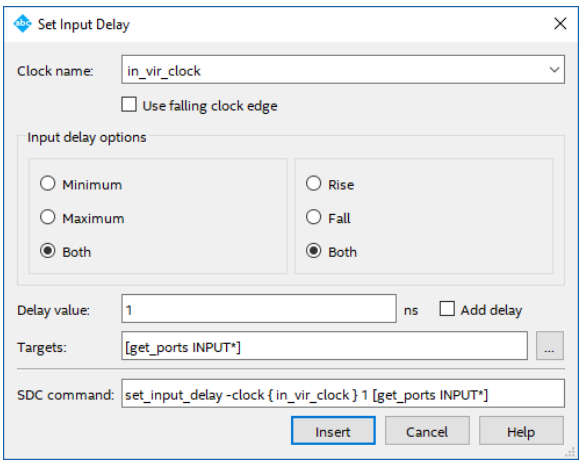
3.4.5.6 **Save** the modification by clicking on button or **File → Save**.

3.4.5.7 Open **Timing Analyzer** window and double click on **Update Timing Netlist** in the Task window.




3.4.5.8 Go back to the **top.sdc** file and search for the Set Input Delay section. Right click in the text editor and select **Insert Constraint → Set Input Delay…**

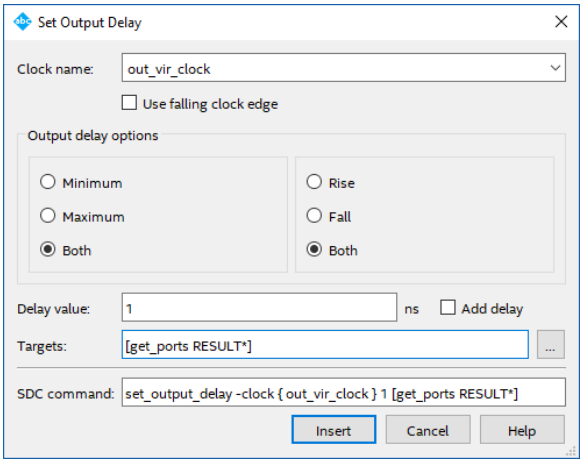
3.4.5.9 In the dialog box, select **in_vir_clock** from the drop-down menu for Clock name and set **1 ns** for the Delay value. Type **[get_ports INPUT*]** for the Targets.




3.4.5.10 Click **Insert**.

3.4.5.11 Search for the Set Output Delay section, right click in the text editor and select **Insert Constraint → Set Output Delay…**

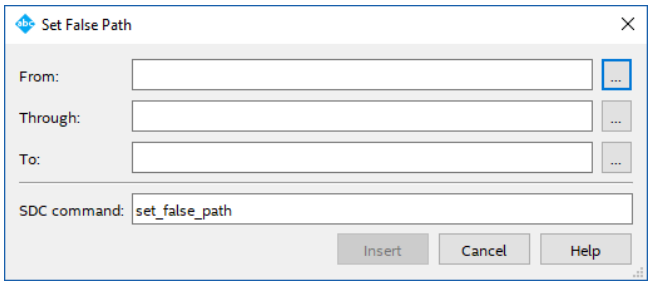
3.4.5.12 In the dialog box, select **out_vir_clock** from the drop-down menu for Clock name and set **1 ns** for the Delay value. Type **[get_ports RESULT*]** for the Targets.




3.4.5.13 Click **Insert**.

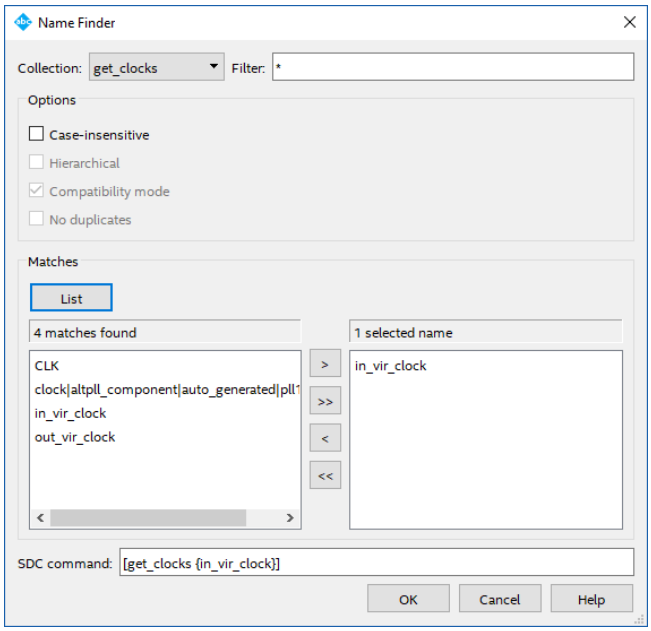
3.4.5.14 Search for the Set False Path section, right click in the text editor and select **Insert Constraint → Set False Path…**

3.4.5.15 In the dialog box, click button for the “From”.



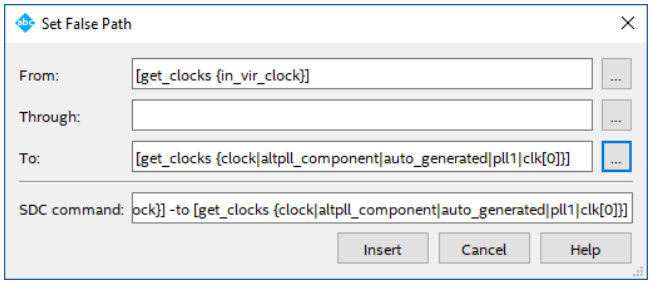

3.4.5.16 In the Name Finder dialog box, make sure that the Collection is **get_clocks** and click on **List**.

3.4.5.17 Select **in_vir_clock** and click on button.



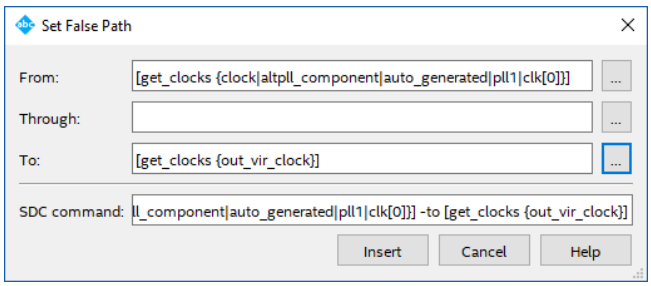

3.4.5.18 Press **OK**.

3.4.5.19 Click on button for the “To” and like in the previous steps, add the **clock|altpll_component|auto_generated|pll1|clk[0]**.




3.4.5.20 Click **Insert**.

3.4.5.21 Repeat the previous steps from 3.4.5.14 and set false path from **clock|altpll_component|auto_generated|pll1|clk[0]** to **out_vir_clock**.

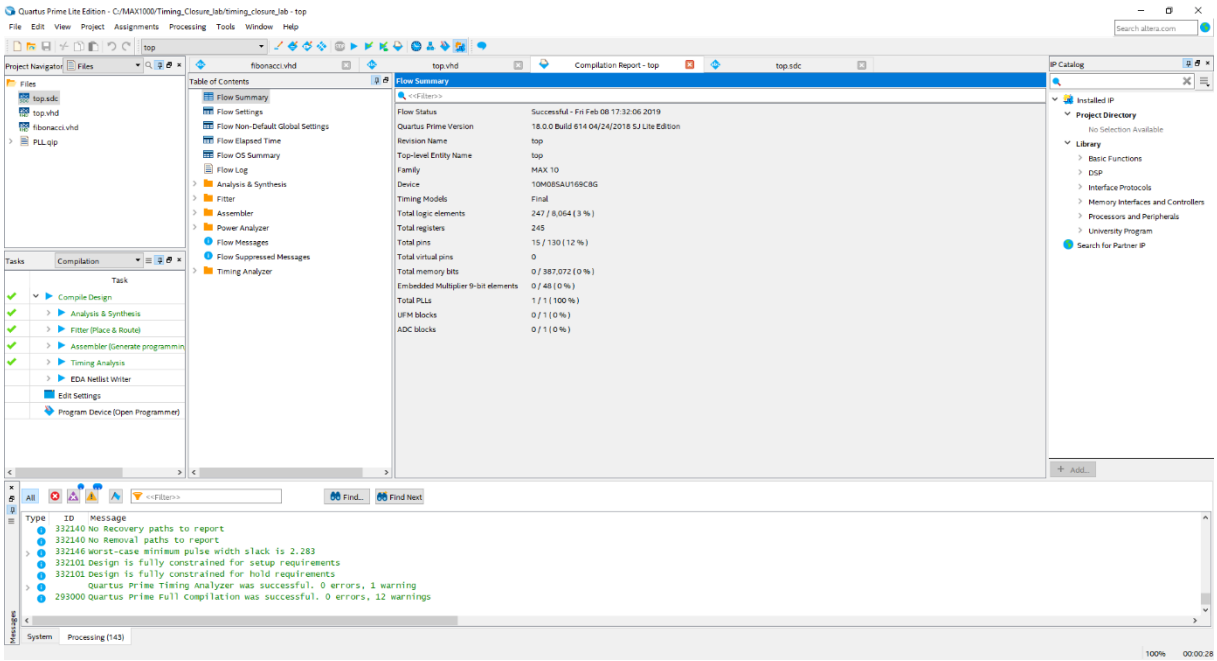



3.4.5.22 **Save** the modification by clicking on button or **File → Save**.

## 3.4.6 Compiling the design

3.4.6.1 Start Compilation by clicking on button on the toolbars, or **Processing → Start Compilation**.

There should be no errors. If there are errors, they should be fixed before re-compilating.




With this compilation all timing problem was resolved, our design meets with the timing requirements and there is no unconstrained path in the reports.

## 3.5 Design Test (optional)

**Note:** This section is optional, and only uses Quartus environment for the internal testing. If you have an external device, for example logic analyzer, you can use it after the configuration and skip the following steps.

### 3.5.1 VHDL code modification

3.5.1.1 Modify the io_reg process in top.vhd in order to able to generate input data

```
io_reg : process(iCLK)
begin
      if(iCLK = '1' and iCLK'event) then
            iINPUT <= iINPUT + '1';
            RESULT <= iRESULT;
      end if;
end process;
```

3.5.1.2 **Save** the modification by clicking on button or **File → Save**.

### 3.5.2 JTAG Signal Constraints

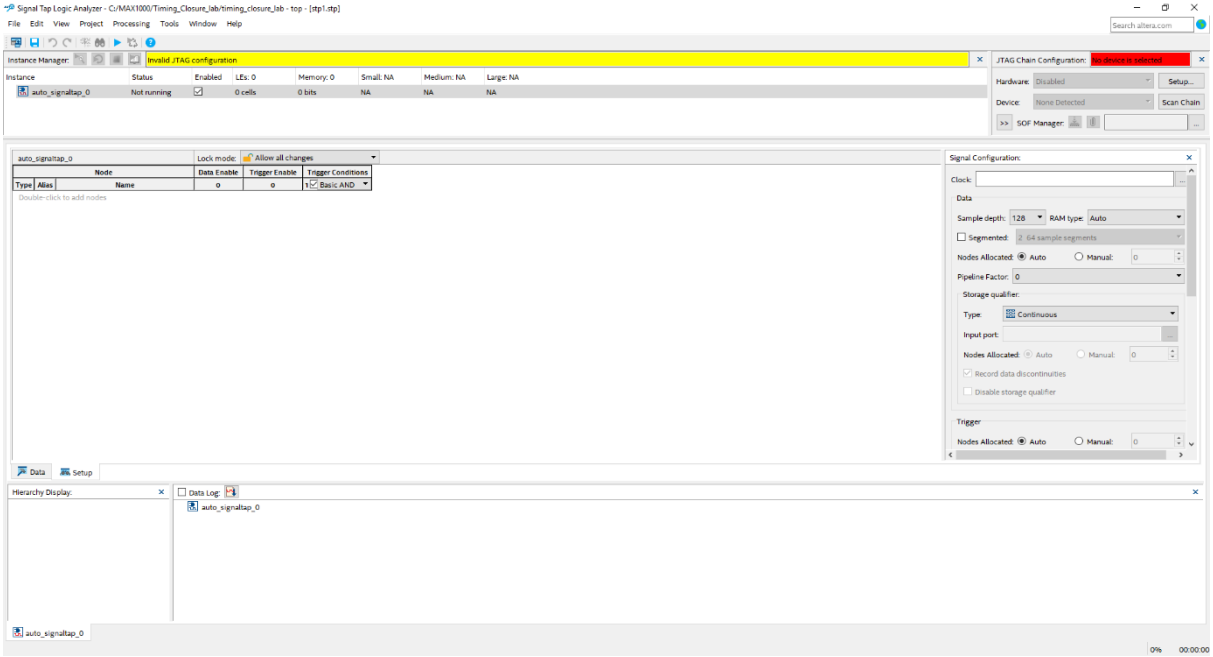
3.5.2.1 Many in-system debugging tools use the JTAG interface in Intel FPGAs. When you debug your design with the JTAG interface, the JTAG signals are implemented as part of the design. Because of this, the Timing Analyzer flags these signals as unconstrained when an unconstrained path report is generated. In order to avoid this in reports, add the following lines to the **top.scd**.

```
create_clock -name altera_reserved_tck \
             -period 166.667 [get_ports {altera_reserved_tck}]
set_input_delay -clock {altera_reserved_tck} 5 \
             [get_ports {altera_reserved_tdi}]
set_input_delay -clock {altera_reserved_tck} 5 \
             [get_ports {altera_reserved_tms}]
set_output_delay -clock {altera_reserved_tck} 5 \
             [get_ports {altera_reserved_tdo}]
```

3.5.2.2 **Save** the modification by clicking on button or **File → Save**.

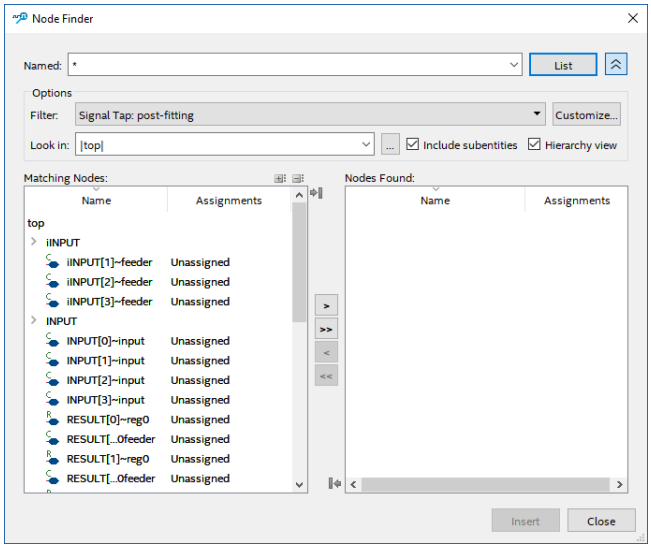
### 3.5.3 SignalTap setup

3.5.3.1 Select from the menu **Tools → Signal Tap Logic Analyzer**.



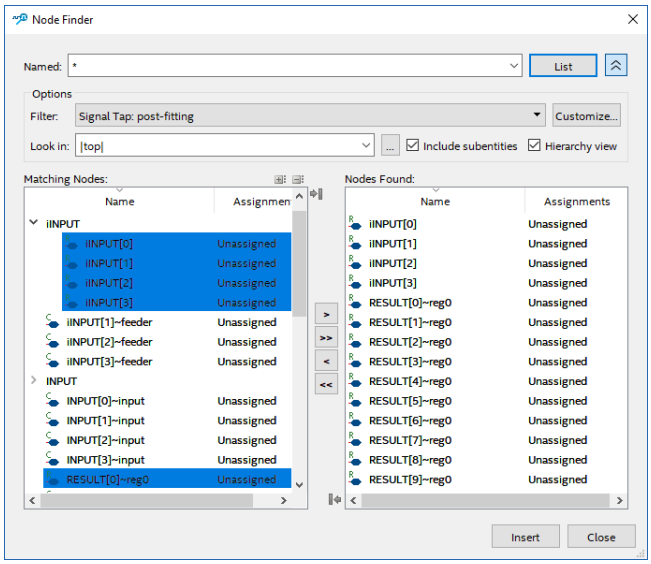
3.5.3.2 Double click in the Setup tab to add nodes.

3.5.3.3 In the Node Finder make sure, that **Signal Tap: post-fitting** is selected for the Filter and click on **List**.



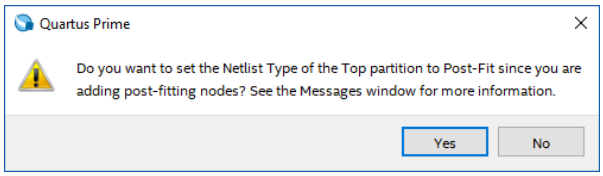

3.5.3.4 Expand "iINPUT" and select from **iINPUT[0]** to **iINPUT[3]** and click on button.

3.5.3.5 Select from **RESULT[0]~reg0** to **RESULT[9]~reg0** and lick on button.




3.5.3.6 Click **Insert.**

3.5.3.7 On the pop-up window, click **Yes** and **close** the Node Finder window.



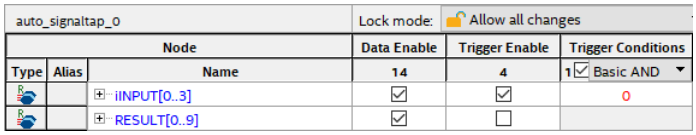

3.5.3.8 Select from **iINPUT[0]** to **iINPUT[3]**, right click on them and choose **Group**.

3.5.3.9 Select from **RESULT[0]~reg0** to **RESULT[9]~reg0**, right click on them and choose **Group**.

3.5.3.10 Select **iINPUT[0..3]** and **RESULT[0..9]** node group, right click on them and select **Bus Display Format → Unsigned Decimal**.

3.5.3.11 Uncheck Trigger Enable for **RESULT[0..9]**.

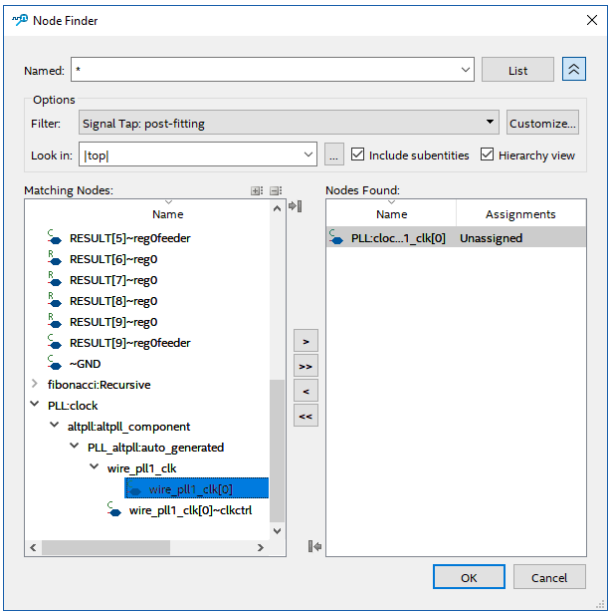
3.5.3.12 Double click in Trigger Conditions cell of **iINPUT[0..3]** and type **0**.




3.5.3.13 In the Signal Configuration window on the right side, click on the button to browse the clock signal.

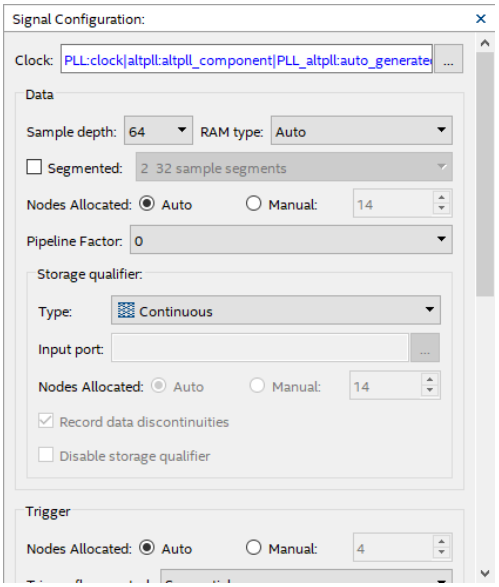
3.5.3.14 In the Node Finder window, make sure, that the **Signal Tap: post-fitting** is selected for the Filter, and click on the **List** button.

3.5.3.15 In the Matching Nodes window expand **PLL:clock\altpll:altpll_component\PLL_altpll:auto_generated\wire_pll1_clk** and select **wire_pll1_clk[0]**. Click on > button.



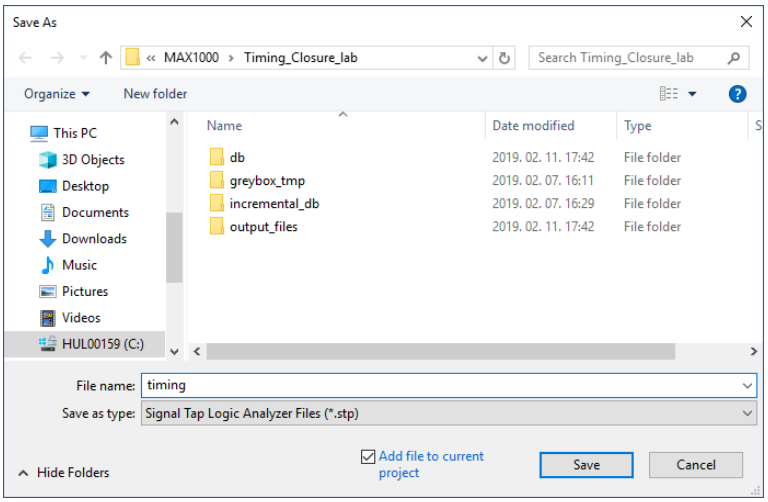

3.5.3.16 Click **OK** to close window.

3.5.3.17 Set Sample depth to 64 under Data and leave the other parameters by default.



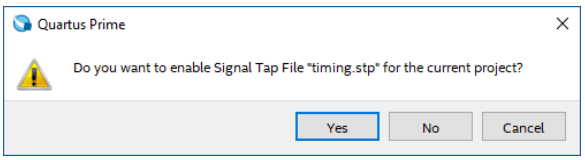
3.5.3.18 Save the analyzer setting by clicking on button or **File → Save** and enter the following information:

- File name: **timing**
- Save as type: **Signal Tap Logic Analyzer Files (*.stp)**
- Make sure that "**Add file to current project**" option is checked.




3.5.3.19 Click **Save**.

3.5.3.20 In the Quartus Prime pop-up window click **Yes** to add this file to the current project.




## 3.5.4 Compiling the Design

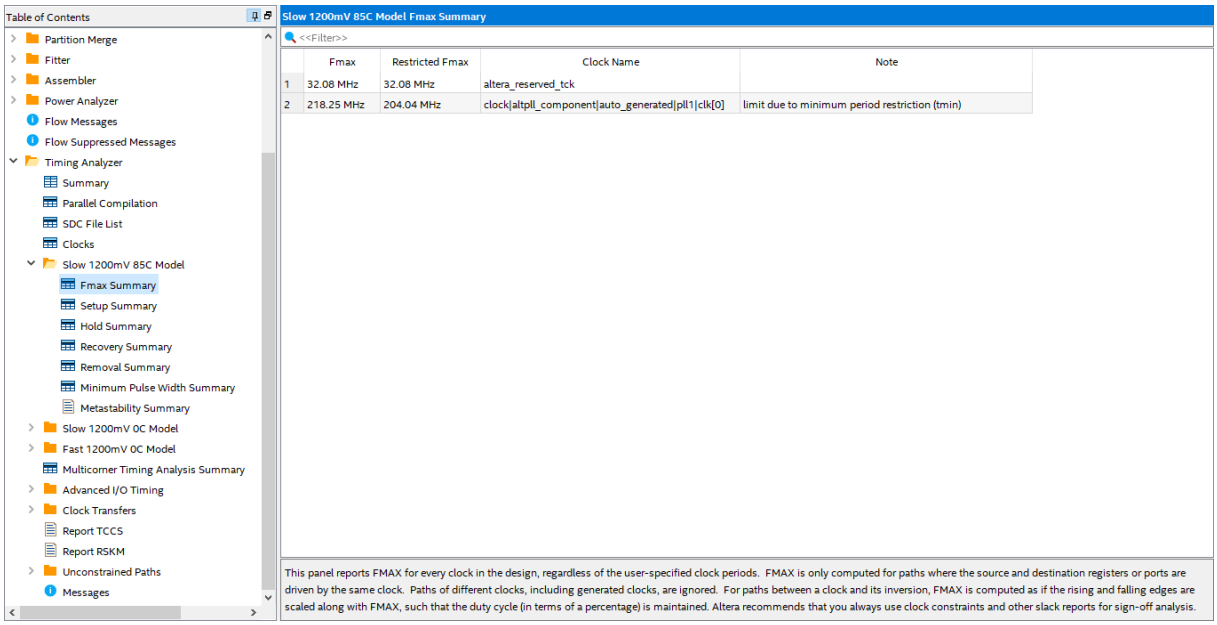
3.5.4.1 Before the compilation, make sure, that auto_signaltap_0 instance is enabled in SignalTap.

| Instance | Status | Enabled | LEs: 541 | Memory: 896 | Small: 0/0 | Medium: 1/42 | Large: 0/0 |
|---|---|---|---|---|---|---|---|
| auto_signaltap_0 | Not running | ☑ | 541 cells | 896 bits | 0 blocks | 1 blocks | 0 blocks |

3.5.4.2 Start Compilation by clicking on button on the toolbars, or **Processing → Start Compilation**.

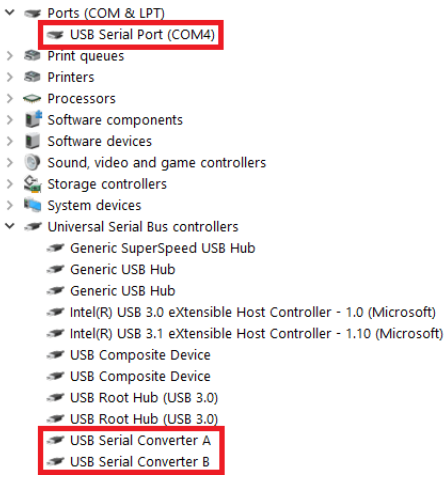
There should be no errors. If there are errors, they should be fixed before re-compilating.

If you check the compilation report, there should be no timing issues. If you expand Timing Analyzer → Slow 1200mV 85C Model and open Fmax Summary, you should see, that the minimum frequency is still higher than the requested 200 MHz.

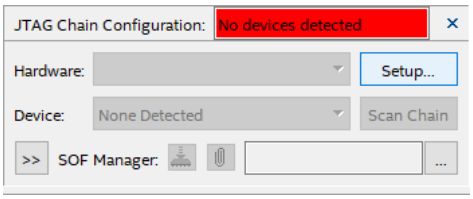



## 3.5.5 Configuration

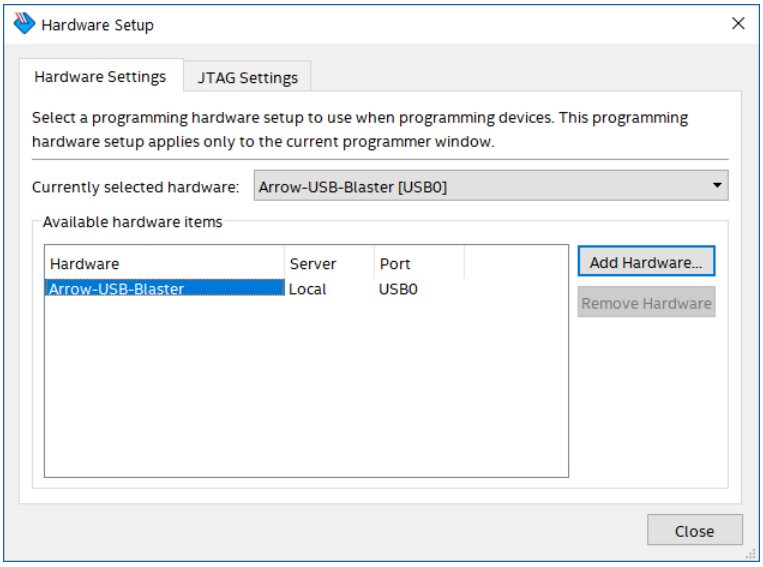
3.5.5.1 Connect your MAX1000 board to your PC using an USB cable. Since the Arrow USB Blaster should be already installed, the Window's Device Manager should display the following entries are highlighted in red (port number may differ depending on your PC):



3.5.5.2 Open the SignalTap window and click on the **Setup…** button in the top right corner.

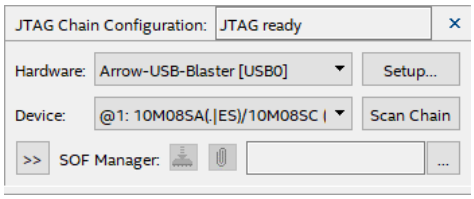



3.5.5.3 Double click **Arrow-USB-Blaster** entry in the Hardware Setup tab. The Currently selected hardware should now show Arrow-USB-Blaster [USB0] (depending on your PC, the USB port number may variant).




3.5.5.4 Click **Close**.

3.5.5.5 The hardware configuration window should be updated as follows.

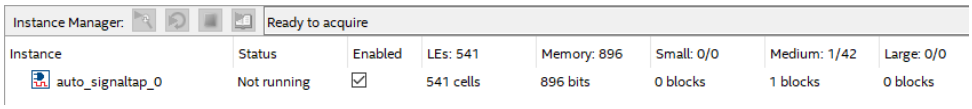



3.5.5.6 Click on ... button to choose the programming file.

3.5.5.7 Navigate to **<project_directory>/output_files/** and select the **top.sof** file.

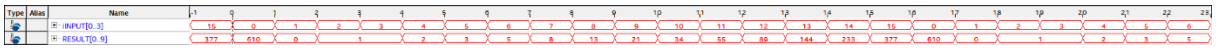
3.5.5.8 Click **Open**.

3.5.5.9 Click on button to program the board. When the configuration is complete, the message box of the Instance Manages should write “Ready to acquire”.




## 3.5.6 Analyze the design

3.5.6.1 Click on button to run single acquisition analysis. On the waveform you can see that the logic generates the correct value of Fibonacci numbers. Because of the registers between the components, there is a latency on the line and its value is 17 clock period.




**CONGRATULATIONS! YOU HAVE SUCCESSFULLY COMPLETED THE TIMING CLOSURE LAB!**

# 5 Revision History

| Version | Change Log | Date of Change |
|---|---|---|
| V1.0 | Initial Version | 12/02/2019 |

# 6 Legal Disclaimer

**ARROW ELECTRONICS**

**EVALUATION BOARD LICENSE AGREEMENT**

By using this evaluation board or kit (together with all related software, firmware, components, and documentation provided by Arrow, "Evaluation Board"), You ("You") are agreeing to be bound by the terms and conditions of this Evaluation Board License Agreement ("Agreement"). Do not use the Evaluation Board until You have read and agreed to this Agreement. Your use of the Evaluation Board constitutes Your acceptance of this Agreement.

**PURPOSE**

The purpose of this evaluation board is solely intended for evaluation purposes. Any use of the Board beyond these purposes is on your own risk. Furthermore, according the applicable law, the offering Arrow entity explicitly does not warrant, guarantee or provide any remedies to you with regard to the board.

**LICENSE**

Arrow grants You a non-exclusive, limited right to use the enclosed Evaluation Board offering limited features only for Your evaluation and testing purposes in a research and development setting. Usage in a live environment is prohibited. The Evaluation Board shall not be, in any case, directly or indirectly assembled as a part in any production of Yours as it is solely developed to serve evaluation purposes and has no direct function and is not a finished product.

**EVALUATION BOARD STATUS**

The Evaluation Board offers limited features allowing You only to evaluate and test purposes. The Evaluation Board is not intended for consumer or household use. You are not authorized to use the Evaluation Board in any production system, and it may not be offered for sale or lease, or sold, leased or otherwise distributed for commercial purposes.

**OWNERSHIP AND COPYRIGHT**

Title to the Evaluation Board remains with Arrow and/or its licensors. This Agreement does not involve any transfer of intellectual property rights ("IPR) for evaluation board. You may not remove any copyright or other proprietary rights notices without prior written authorization from Arrow or it licensors.

**RESTRICTIONS AND WARNINGS**

Before You handle or use the Evaluation Board, You shall comply with all such warnings and other instructions and employ reasonable safety precautions in using the Evaluation Board. Failure to do so may result in death, personal injury, or property damage.

You shall not use the Evaluation Board in any safety critical or functional safety testing, including but not limited to testing of life supporting, military or nuclear applications. Arrow expressly disclaims any responsibility for such usage which shall be made at Your sole risk.

**WARRANTY**

Arrow warrants that it has the right to provide the evaluation board to you. This warranty is provided by Arrow in lieu of all other warranties, written or oral, statutory, express or implied, including any warranty as to merchantability, non-infringement, fitness for any particular purpose, or uninterrupted or error-free operation, all of which are expressly disclaimed. The evaluation board is provided "as is" without any other rights or warranties, directly or indirectly.

You warrant to Arrow that the evaluation board is used only by electronics experts who understand the dangers of handling and using such items, you assume all responsibility and liability for any improper or unsafe handling or use of the evaluation board by you, your employees, affiliates, contractors, and designees.

**LIMITATION OF LIABILITIES**

In no event shall Arrow be liable to you, whether in contract, tort (including negligence), strict liability, or any other legal theory, for any direct, indirect, special, consequential, incidental, punitive, or exemplary damages with respect to any matters relating to this agreement. In no event shall arrow's liability arising out of this agreement in the aggregate exceed the amount paid by you under this agreement for the purchase of the evaluation board.

**IDENTIFICATION**

You shall, at Your expense, defend Arrow and its Affiliates and Licensors against a claim or action brought by a third party for infringement or misappropriation of any patent, copyright, trade secret or other intellectual property right of a third party to the extent resulting from (1) Your combination of the Evaluation Board with any other component, system, software, or firmware, (2) Your modification of the Evaluation Board, or (3) Your use of the Evaluation Board in a manner not permitted under this Agreement. You shall indemnify Arrow and its Affiliates and Licensors against and pay any resulting costs and damages finally awarded against Arrow and its Affiliates and Licensors or agreed to in any settlement, provided that You have sole control of the defense and settlement of the claim or action, and Arrow cooperates in the defense and furnishes all related evidence under its control at Your expense. Arrow will be entitled to participate in the defense of such claim or action and to employ counsel at its own expense.

**RECYCLING**

The Evaluation Board is not to be disposed as an urban waste. At the end of its life cycle, differentiated waste collection must be followed, as stated in the directive 2002/96/EC. In all the countries belonging to the European Union (EU Dir. 2002/96/EC) and those following differentiated recycling, the Evaluation Board is subject to differentiated recycling at the end of its life cycle, therefore: It is forbidden to dispose the Evaluation Board as an undifferentiated waste or with other domestic wastes. Consult the local authorities for more information on the proper disposal channels. An incorrect Evaluation Board disposal may cause damage to the environment and is punishable by the law.