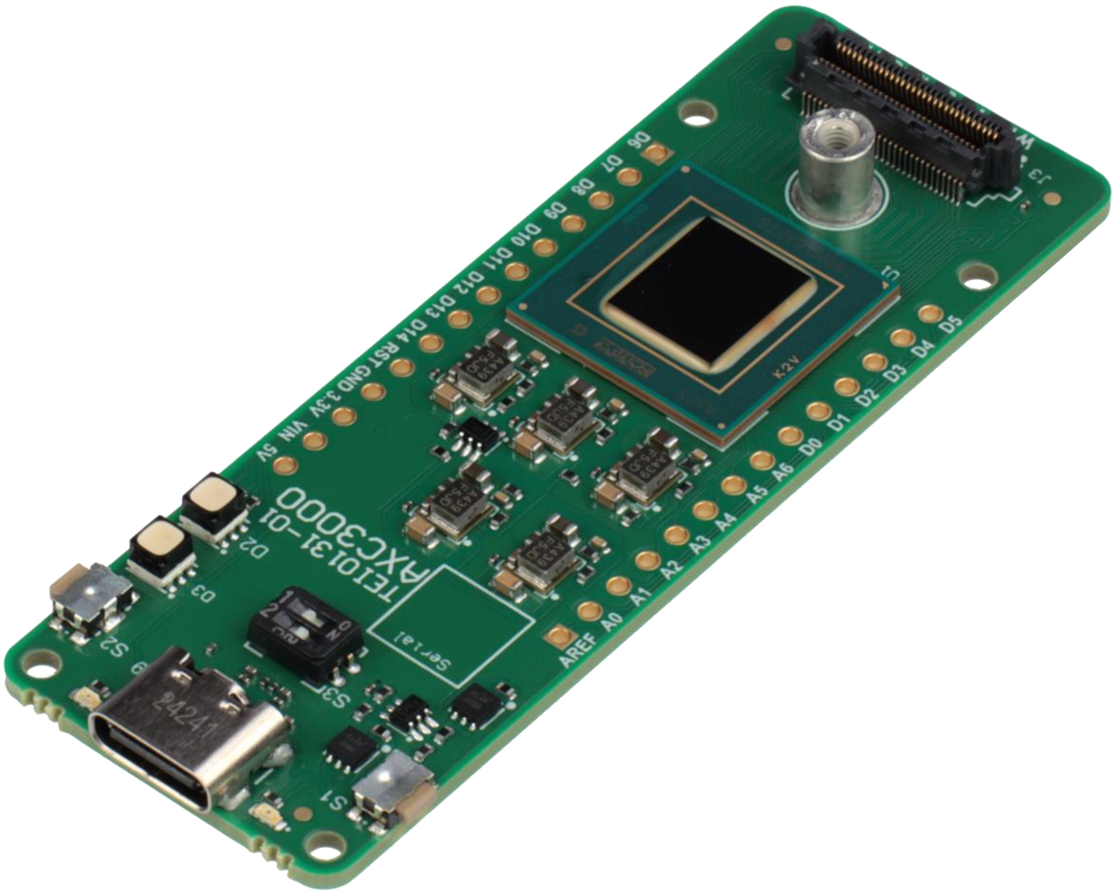
AXC3000 Timing Workshop



Software and hardware requirements to complete all exercises.

Software Requirements: Quartus® Prime Pro version 25.1.1

Hardware Requirements: AXC3000 Agilex™ 3C Development Kit

Document Control

Document Version:	Version 1.0
Document Date:	08/12/2025
Document Author(s):	Steven Kravatsky, Erika Peter, Naji Naufel, , ESC Team
Document Classification:	Released
Document Distribution:	This document is still under development. All specifications, procedures, and processes described in this document are subject to change without prior notice
Prior Version History:	0.1

Please read the legal disclaimer at the end of this document.

Table of Contents

1 Introduction

This workshop provides comprehensive information to help you understand how to set and use the FPGA design environment to ensure that your design meets timing requirements. The Quartus Prime Timing Analyzer is a powerful ASIC-style timing analysis tool that validates the timing performance of the FPGA using industry standard constraint methodology.

1.1 Objective

Create a design

1. Parameterize IPs
2. Add timing constraints
3. Review various timing errors
4. Implement solutions to remove the timing errors

Lab Notes: Many of the names that the lab asks you to use for files, components, and other objects in this exercise must be spelled exactly as directed. This nomenclature is necessary because the pre-written files use certain names. Naming the components differently can cause errors.

This lab can be compiled on a local machine or on a remote machine hosted by CloudLabs.

2 Getting Started

The first objective is to ensure that you have all the necessary software installed so that the lab can be completed successfully. Below is a list of items required to complete this lab. A number of options are provided.

Option 1: Completing the lab using CloudLabs tools and the board farm.

- Personal computer or laptop running 64-bit Linux / Windows 10 or later with at least an Intel i3 core (or equivalent), 8GB RAM.

Option 2: Completing the lab using CloudLabs tools and local hardware.

- AXC3000 Board
- USB C cable
- Quartus Prime 25.1.1 Pro Standalone Programmer.
- Personal computer or laptop running 64-bit Linux / Windows 10 or later with at least an Intel i3 core (or equivalent), 8GB RAM.
- Terminal emulator (Putty or Tera term)

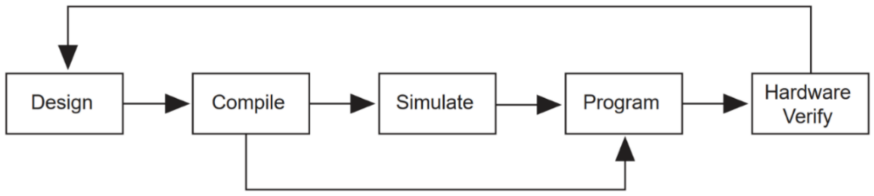
Option 3: Completing the lab using a laptop and local hardware.

- AXC3000 Board
- USB C cable
- Quartus Prime 25.1.1 Pro
- Ashling RiscFree IDE
- Personal computer or laptop running 64-bit Linux / Windows 10 or later with at least an Intel i3 core (or equivalent), 8GB RAM.
- Terminal emulator (Putty or Tera term)

A desire to learn!

3 FPGA Flow and Timing

In the FPGA flow, there are various steps such as design, compile, simulate, program and hardware verify as:



The workshop will focus on the Design and Compile section where a design is created and timing constraints are entered, reviewed and then changes are implemented to resolve issues.

Accurate timing constraints need to be entered before the compile stage so that the timing driven synthesis and place and route can optimize the design. Timing constraints are needed to ensure that the timing requirements are met. If timing constraints are not entered, Quartus make default assumptions about the timings which will not be correct.

Quartus will report if the timing requirements are met/not met. If the timing requirements are not met, there will be timing violations. (which will be seen in this workshop)

Timing Violations in FPGA designs are primarily categorized into two types: setup time violations and hold time violations.

- **Setup Time Violations:** Occur when a signal arrives at a flip-flop too late, missing the setup time requirement before the clock edge.
- **Hold time Violations:** Occur when a signal changes too soon after the clock edge, violating the hold time requirement.

Both types of violations can lead to incorrect data being captured and propagated through the design, causing functional failures.

3.1 Timing Analysis Tools within Quartus

Timing analysis is a critical step in identifying potential timing violations. The following tools and techniques are commonly used within Quartus.

- **Static Timing Analysis (STA):** STA is a method of validating the timing performance of a design by analyzing all possible paths and ensuring they meet specified timing constraints without requiring simulation vectors.

- **Timing Reports:** Generated by STA tools like the Quartus Prime Timing Analyzer, timing reports provide detailed information about the critical paths, slack times, and violated constraints.

3.2 How to enter timing Design Constraints

Properly defining and applying design constraints is crucial for mitigating timing violations. This involves:

- **Constraint files:** Utilize Synopsys® Design Constraints (SDC) files to specify timing requirements, such as clock frequencies, input/output delays, and multi-cycle paths.
- **Clock Constraints:** Define accurate clock constraints, including primary clocks, generated clocks, and clock groups, to ensure precise timing analysis.
- **Path-Specific Constraints:** Apply constraints to specific paths or exceptions to handle unique timing requirements and optimize critical paths.

Some design constraints will be covered in this workshop.

There are different ways to achieve timing closure such as changing the design, synthesis and placement optimization, routing and timing closure techniques.

3.3 Synthesis and Placement Optimization

Synthesis and placement play a pivotal role in achieving timing closure:

- **Logic Synthesis:** Optimize the RTL design during synthesis to improve the overall timing performance. Techniques include retiming, logic replication, and resource sharing.
- **Placement Optimization:** During the placement phase, the tool positions logic elements on the FPGA fabric to minimize critical path delays. Manual floor planning can be employed to guide placement for better performance.

3.4 Routing Optimization

Routing significantly impacts the timing of a design:

- **Timing-Driven Routing:** Utilize timing-driven routing algorithms to prioritize critical paths during routing, reducing delays and improving timing closure.
- **Incremental Routing:** Apply incremental routing techniques to iteratively refine the routing of critical paths without affecting the entire design.

3.5 Timing Closure Techniques

Several techniques can be employed to achieve timing closure such as:

- **Pipelining:** Introduce pipeline stages to break long combinational paths, reducing the delay and meeting timing constraints. This will increase the latency of the path that has been pipelined as a trade-off.
- **Clock Domain Crossing (CDC):** Properly manage signals crossing different clock domains using synchronizers and FIFOs to prevent metastability and timing issues.

Some of these techniques will be covered in this workshop.

4 Quartus Prime Project

This chapter gives instructions for creating the hardware project in Quartus® Prime Pro 25.1.1

4.1 New Quartus Prime project

The hardware implementation starts with a Quartus® Prime project. A Quartus project defines the location for the HDL and IP files used, target family, target device, and initial settings. After the initial creation of the project, changes can be made. Quartus uses the project location for the generated files.

4.1.1 New project creation

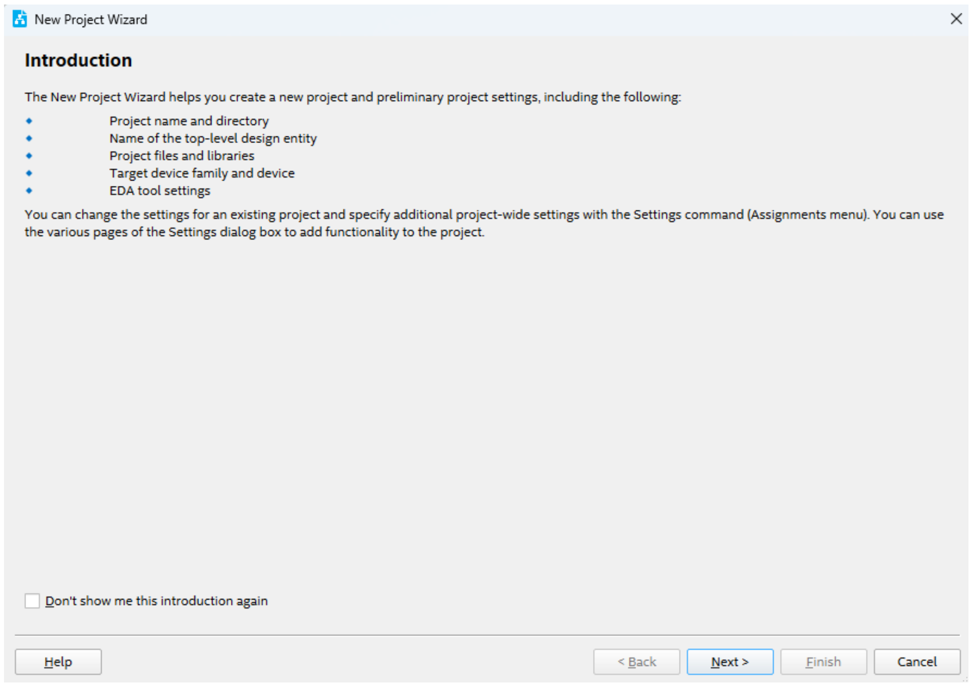
There are screens that need to be filled in when creating a new project. The following steps will take you through all of them.

4.1.1.1 Open Quartus Pro 25.1.1

If not already open, from the Start menu or the Desktop, open Quartus Prime Pro 25.1. 1

4.1.1.2 Create a new project

Using the New Project Wizard: **File → New Project Wizard**. This screen shows an introduction window.



The New Project Wizard helps you create a new project and preliminary project settings, including the following:

- Project name and directory
- Name of the top-level design entity
- Project files and libraries
- Target device family and device
- EDA tool settings

You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.

4.1.1.3 Click Next

4.1.1.4 Configure the New Project Wizard

4.1.1.5 Specify project directory, name, and top-level entity information:

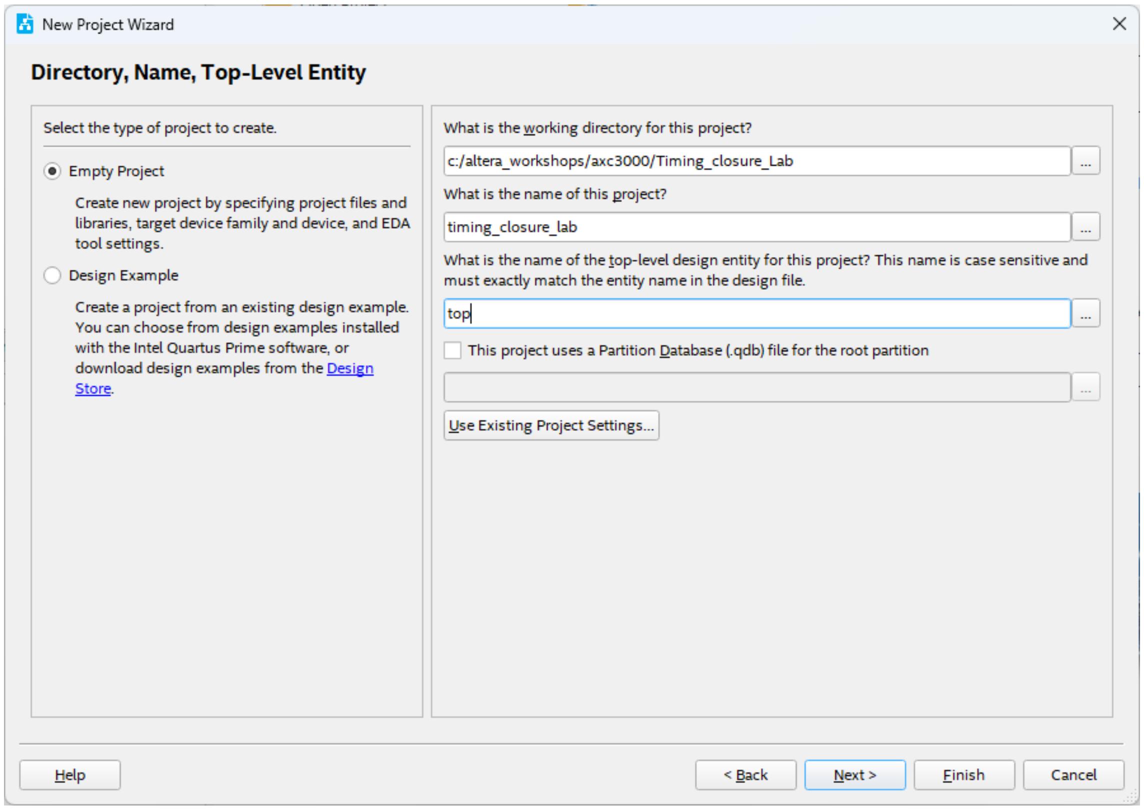
Note: No spaces are allowed in path or filenames.

4.1.1.6 Ensure that the Empty Project radio button is selected on the left side of the window, as this is the new project (not an existing project)

4.1.1.7 Type the directory name in which you will store your Quartus project files as **c:/altera_workshops/axc3000/Timing_closure_Lab**

4.1.1.8 Specify the name of the project: **timing_closure_lab**

4.1.1.9 Specify the name of the top-level entity: **top**



Select the type of project to create.

● Empty Project

Create new project by specifying project files and libraries, target device family and device, and EDA tool settings.

○ Design Example

Create a project from an existing design example. You can choose from design examples installed with the Intel Quartus Prime software, or download design examples from the Design Store.

What is the working directory for this project?

c:/altera_workshops/axc3000/Timing_closure_Lab

What is the name of this project?

timing_closure_lab

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

top

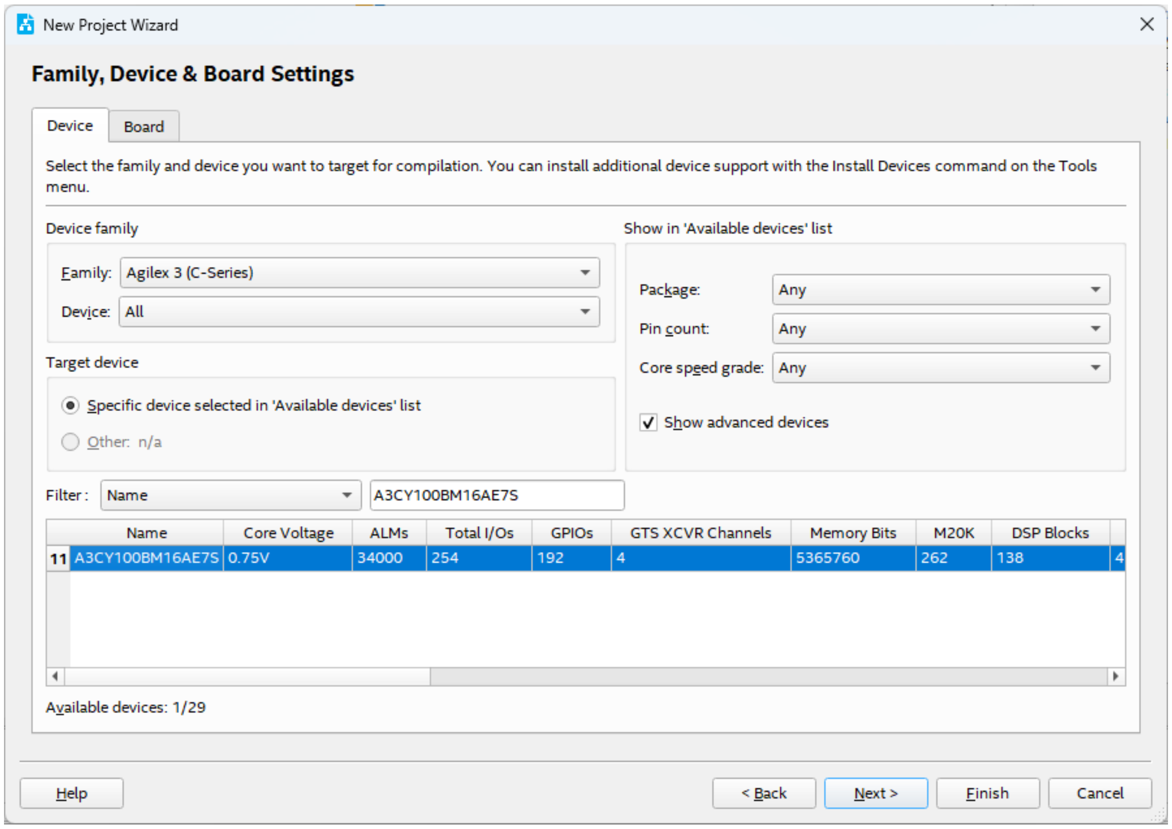
☐ This project uses a Partition Database (.qdb) file for the root partition

Use Existing Project Settings...

4.1.1.10 Click Next

4.1.1.11 Specify Family and Device Settings

In the Family field, select ***Agilex 3 (C-series)*** and enter the part number (**A3CY100BM16AE7S**) in the Name Filter text box and select it. Ensure that the device is highlighted in blue. It has been seen if the device is not highlighted, another (default) device will be selected. This part is the same part as on the AXE-5 board but there will be no programming in this case.



New Project Wizard

Family, Device & Board Settings

Device | Board

Select the family and device you want to target for compilation. You can install additional device support with the Install Devices command on the Tools menu.

Device family

Family: Agilex 3 (C-Series)

Device: All

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

✓ Show advanced devices

Target device

● Specific device selected in 'Available devices' list

○ Other: n/a

Filter: Name A3CY100BM16AE7S

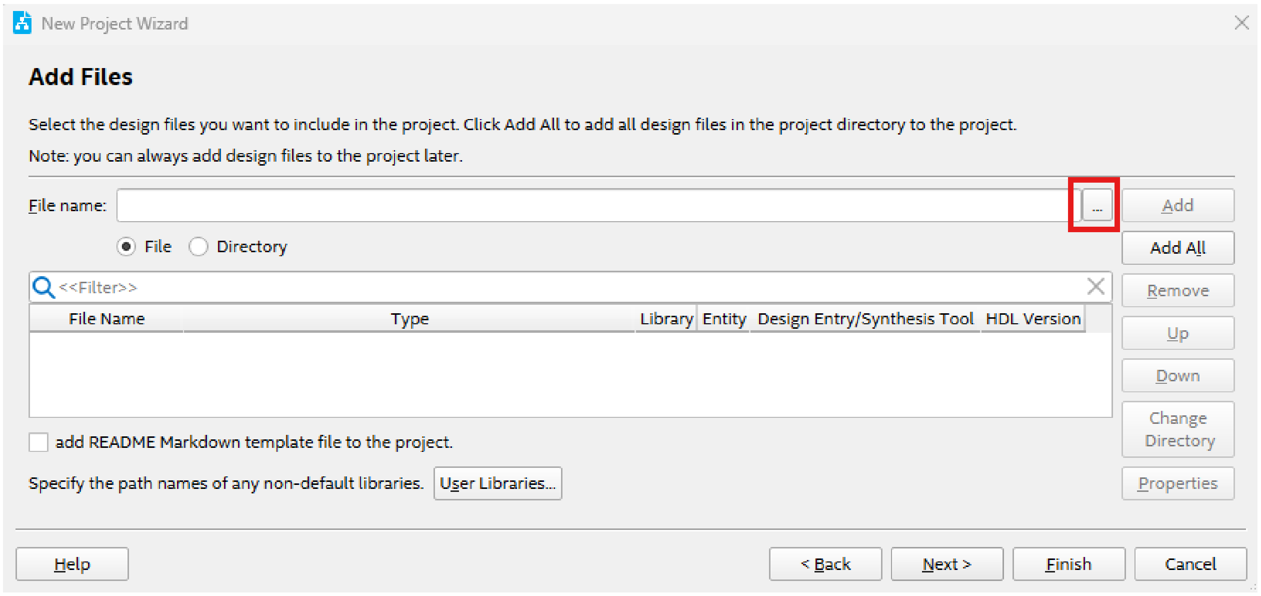
	Name	Core Voltage	ALMs	Total I/Os	GPIOs	GTS XCVR Channels	Memory Bits	M20K	DSP Blocks
11	A3CY100BM16AE7S	0.75V	34000	254	192	4	5365760	262	138

Available devices: 1/29

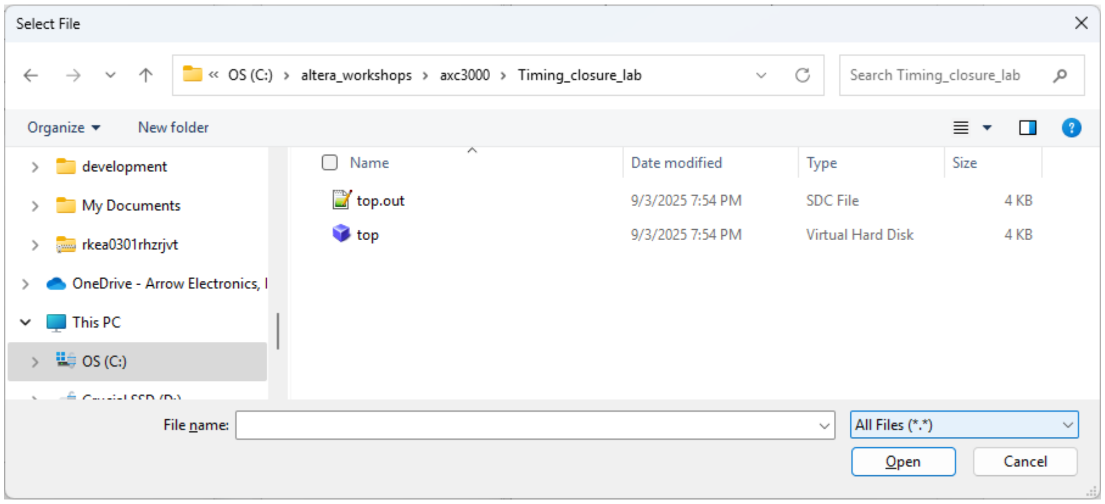
4.1.1.12 Click Next.

4.1.1.13 Add Files to the Project

In the Quartus → Add Files section, select the ... by the file name, change to the c:/altera_workshops/axc3000/Timing_closure_Lab

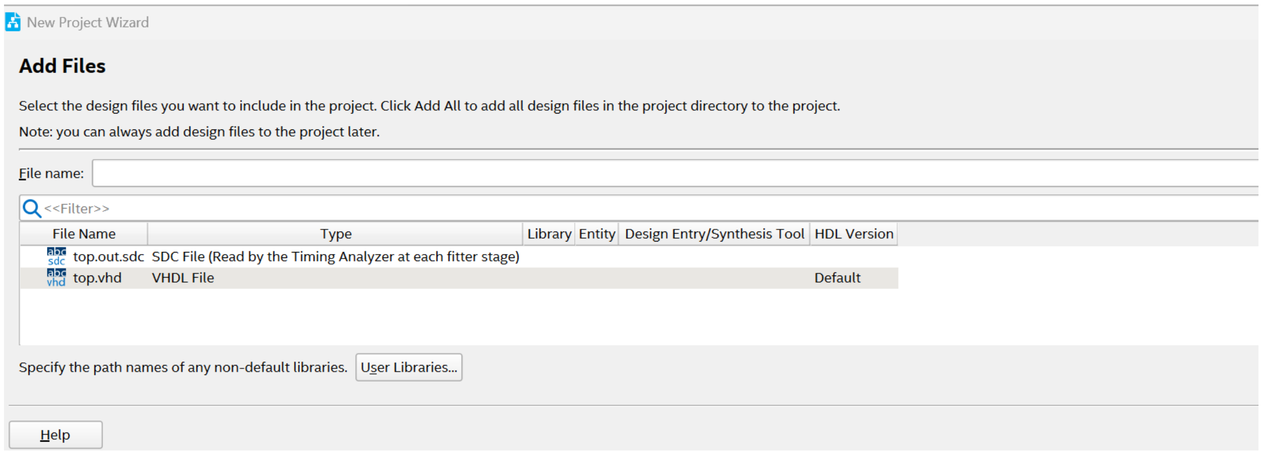

New Project Wizard
Add Files
Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.
Note: you can always add design files to the project later.
File name:
...
Add
Add All
File
Directory
<<Filter>>
Remove
File Name
Type
Library
Entity
Design Entry/Synthesis Tool
HDL Version
Up
Down
Change Directory
add README Markdown template file to the project.
Specify the path names of any non-default libraries.
User Libraries...
Properties
Help
< Back
Next >
Finish
Cancel

Note by default, you will not see both files unless you **select all Files** in the bottom right corner as



Name	Date modified	Type	Size
top.out	9/3/2025 7:54 PM	SDC File	4 KB
top	9/3/2025 7:54 PM	Virtual Hard Disk	4 KB

4.1.1.14 Select both files: **top.vhd** and **top.out.sdc** file and Select Open. The files will then be seen as


New Project Wizard

Add Files

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

Note: you can always add design files to the project later.

File name:

<<Filter>>

File Name	Type	Library	Entity	Design Entry/Synthesis Tool	HDL Version
top.out.sdc	SDC File (Read by the Timing Analyzer at each fitter stage)				
top.vhd	VHDL File				Default

Specify the path names of any non-default libraries. User Libraries...

Help

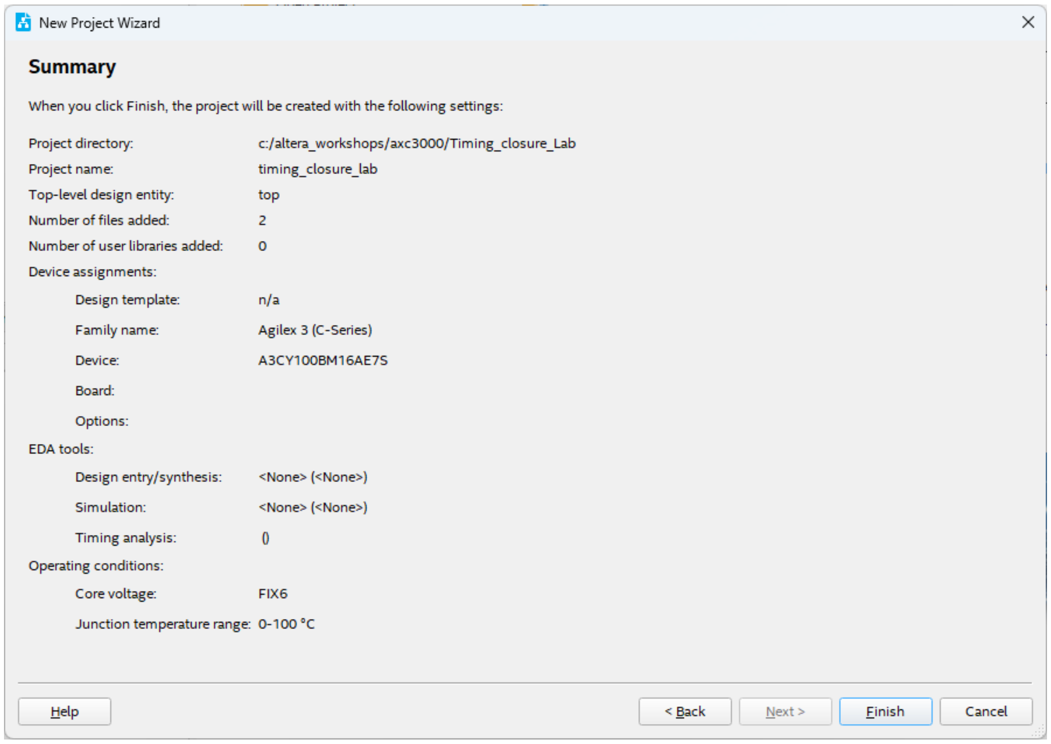
4.1.1.15 Select Next

4.1.1.16 EDA Tools Settings

On the EDA Tools Settings page, select **Next**. No third party (EDA) software will be used in this workshop.

4.1.1.17 Summary Screen

The Summary screen will show the file location and the settings for the device. These settings can be changed if needed, later.



Summary

When you click Finish, the project will be created with the following settings:

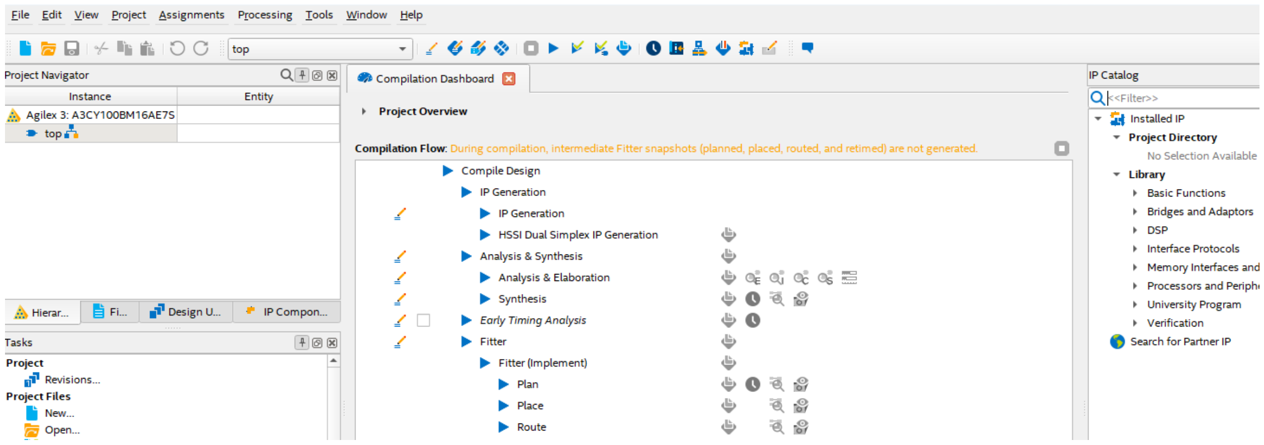
Project directory:	c:/altera_workshops/axc3000/Timing_closure_Lab
Project name:	timing_closure_lab
Top-level design entity:	top
Number of files added:	2
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	Agilex 3 (C-Series)
Device:	A3CY100BM16AE7S
Board:	
Options:	
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	<None> (<None>)
Timing analysis:	0
Operating conditions:	
Core voltage:	FIX6
Junction temperature range:	0-100 °C

4.1.1.18 Click Finish.   The Project is now set up.

4.2 Design Entry

In this workshop you create components and then use the top-level file to compile the design.

4.2.1 Project Top Level

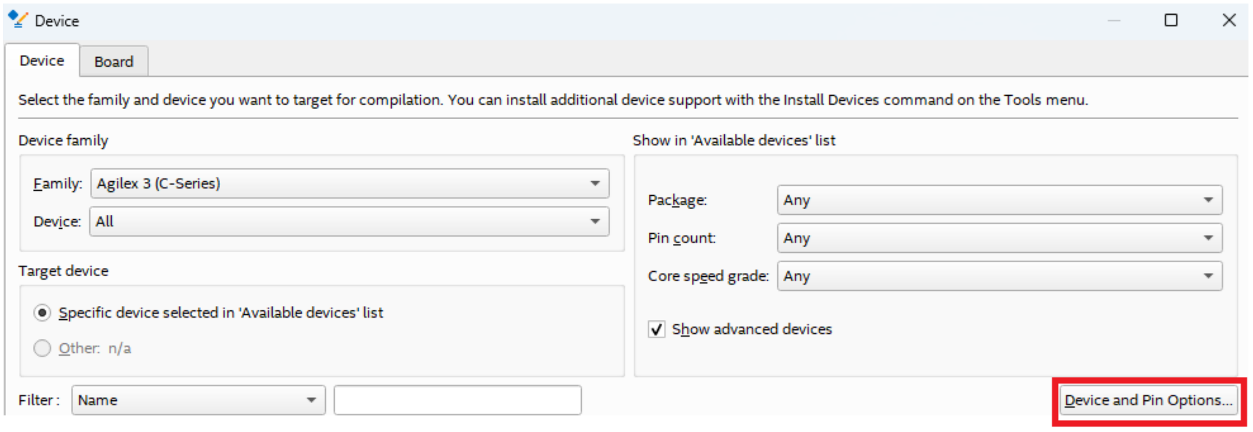


Compilation Flow: During compilation, intermediate Fitter snapshots (planned, placed, routed, and retimed) are not generated.

The top-level file is seen in the **Project Navigator** pane (top left). The middle section (**Compilation Dashboard**) shows the FPGA steps which will be taken by the compiler. These steps include analysis and synthesis, fitter, etc.

The Agilex 3 device needs to implement a Reset Release IP (to ensure that the FPGA design comes out of reset correctly), thus enabling the option to connect the Reset Release IP to the init_done should be done. More details on the Reset Release IP will be described later in this workshop. The conf_done pin (which is a dual-purpose pin) can also be set the same as for the other workshops.

To do so, select the menu **Assignments** → **Device...** then click the **Device and Pin Options** button.



Select the family and device you want to target for compilation. You can install additional device support with the Install Devices command on the Tools menu.

Device family
Family: Agilex 3 (C-Series)
Device: All

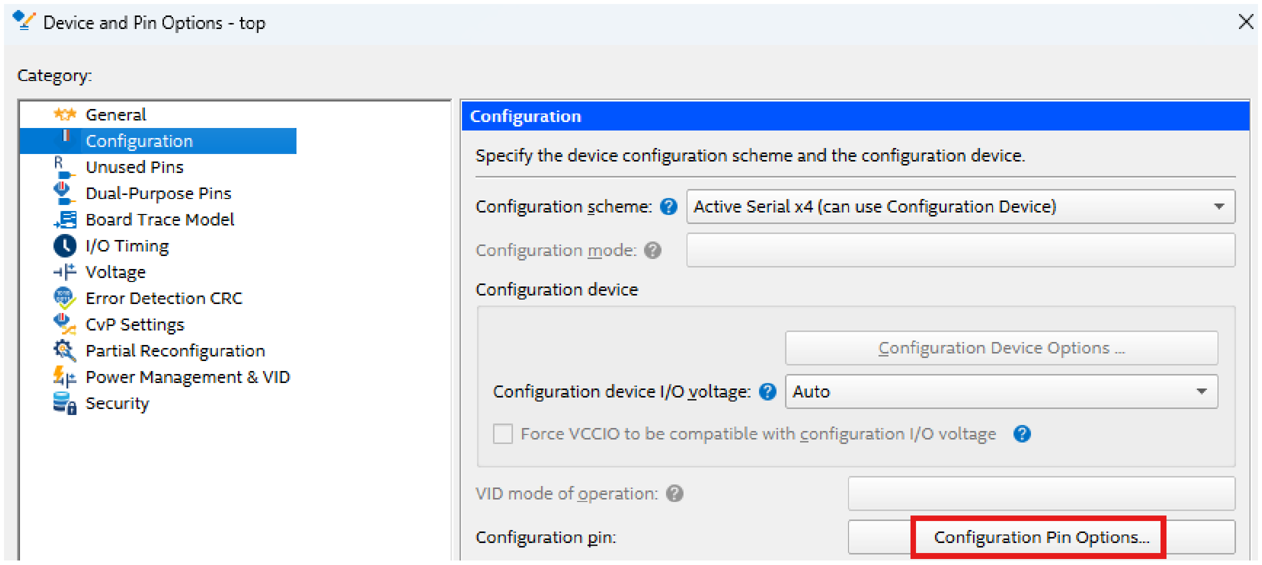
Target device
[✓ Specific device selected in 'Available devices' list]
Other: n/a

Show in 'Available devices' list
Package: Any
Pin count: Any
Core speed grade: Any
[✓ Show advanced devices]

Filter: Name

When a window opens:

4.2.1.1 Select **Configuration** in the Category pane and Click the **Configuration Pin Options** bar.



Specify the device configuration scheme and the configuration device.

Configuration scheme: Active Serial x4 (can use Configuration Device)

Configuration mode:

Configuration device

Configuration device I/O voltage: Auto

Force VCCIO to be compatible with configuration I/O voltage

VID mode of operation:

Configuration pin:

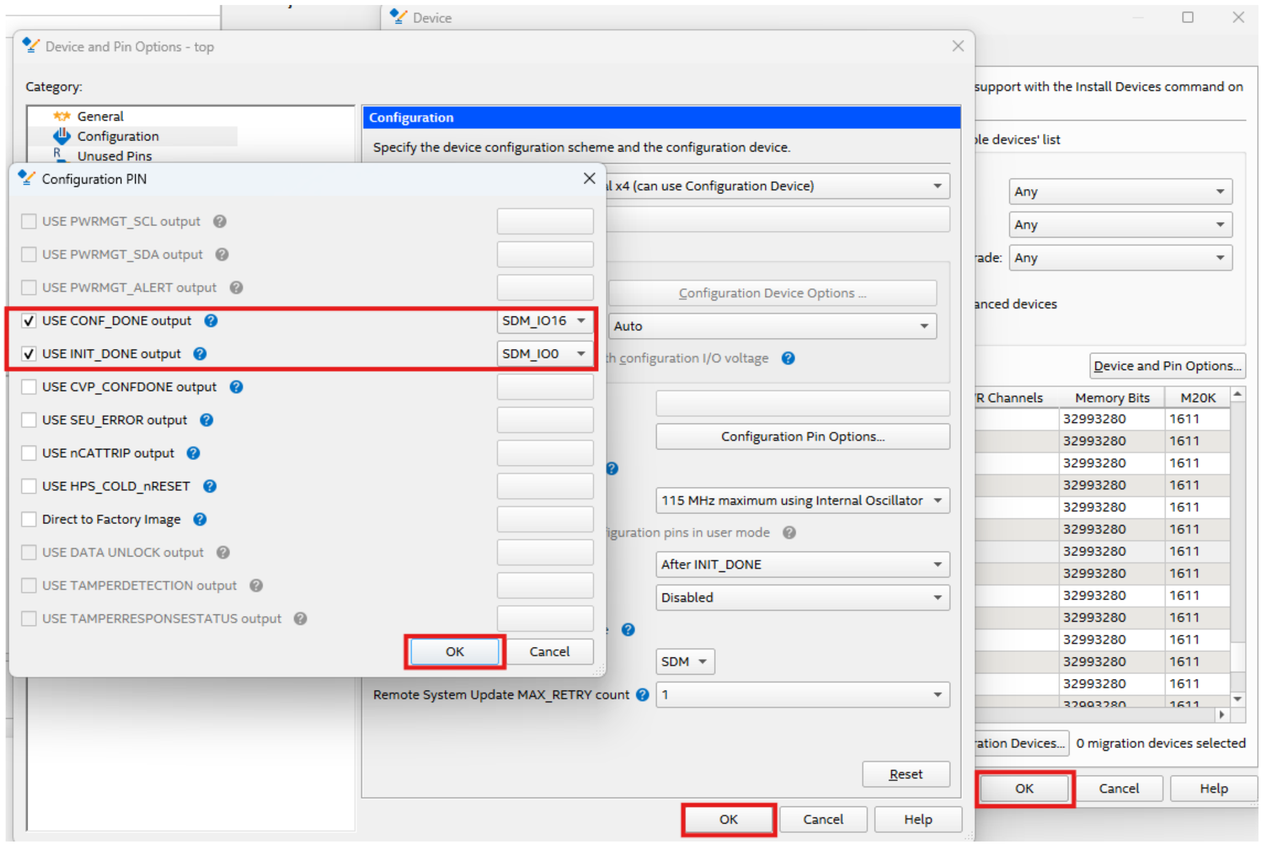
4.2.1.2 Check the box to the left of **USE CONF_DONE** output and select ***SDM_IO16***

The Conf_done pin is used to tell that the configuration is complete and after the configuration, it can be used as general purpose I/O.

4.2.1.3 Check the box to the left of **USE INIT_DONE** output and select ***SDM_IO0***

The init_done pin indicates that the FPGA initialization is complete. The Reset Release IP that will be described and created in this workshop can be connected to the Init_done pin.

4.2.1.4 Click **OK** in multiple windows (3 times) until the windows are closed.



Device

Device and Pin Options - top

Category:

- General
- Configuration
- Unused Pins

Configuration

Specify the device configuration scheme and the configuration device.

Configuration PIN

☐ USE PWRMGT_SCL output	
☐ USE PWRMGT_SDA output	
☐ USE PWRMGT_ALERT output	
✓ USE CONF_DONE output	SDM_IO16
✓ USE INIT_DONE output	SDM_IO0
☐ USE CVP_CONFDONE output	
☐ USE SEU_ERROR output	
☐ USE nCATTRIP output	
☐ USE HPS_COLD_nRESET	
☐ Direct to Factory Image	
☐ USE DATA UNLOCK output	
☐ USE TAMPERDETECTION output	
☐ USE TAMPERRESPONSESTATUS output	

OK Cancel

l x4 (can use Configuration Device)

Configuration Device Options ...

Auto

h configuration I/O voltage

Configuration Pin Options...

115 MHz maximum using Internal Oscillator

iguration pins in user mode

After INIT_DONE

Disabled

SDM

Remote System Update MAX_RETRY count: 1

Reset

OK Cancel Help

support with the Install Devices command on

le devices' list

Any

Any

ade: Any

anced devices

Device and Pin Options...

R Channels	Memory Bits	M20K
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611
	32993280	1611

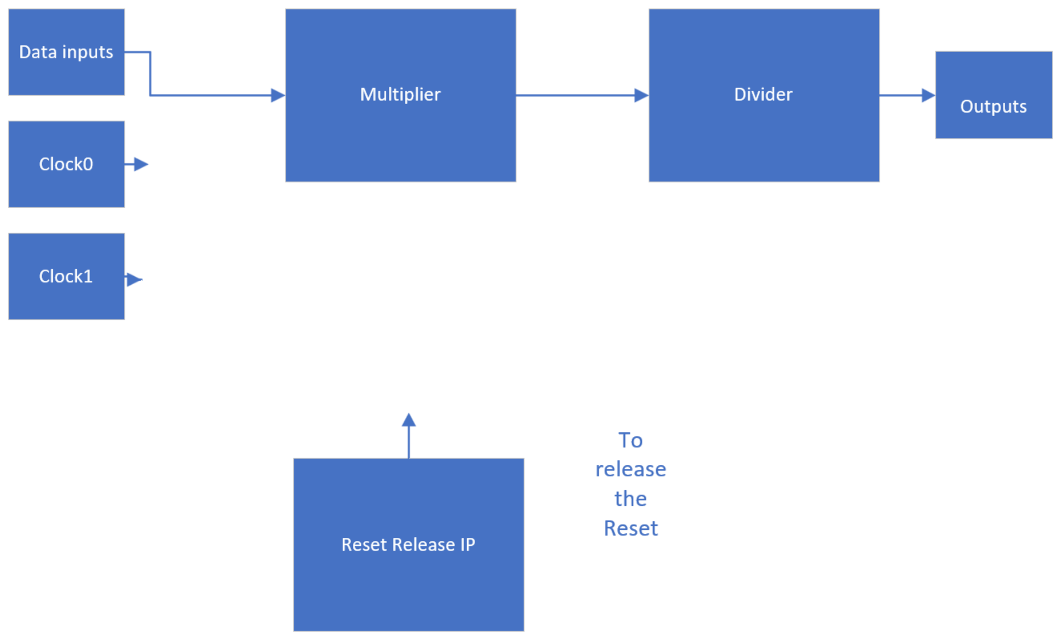
ation Devices... 0 migration devices selected

OK Cancel Help

4.3 Design Flow

What will be implemented in the next steps are IPs (multiplier, divider and reset IP) and a top-level file to implement and compile a small design.

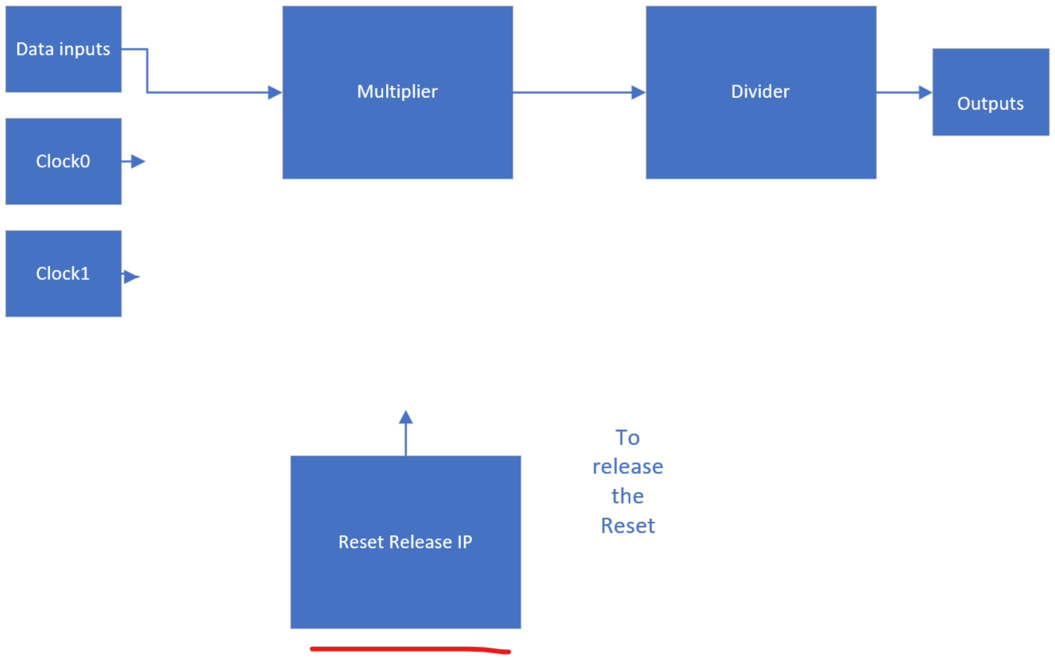
At a high level, the FPGA design will look like the following:



4.3.1 Add a Reset Release IP

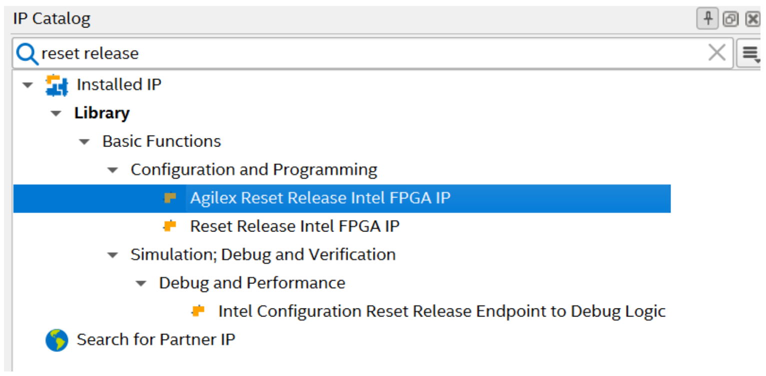
For Agilex 3 devices (and other families), it is recommended to use the Reset Release IP. The Reset Release is an IP that generates a signal to indicate the device configuration is complete. It holds the control circuit in reset until the configuration is complete and user mode is entered. If this IP is not implemented, Quartus will report a warning that the configuration might not be correct.

The Reset Release IP in the top-level view is

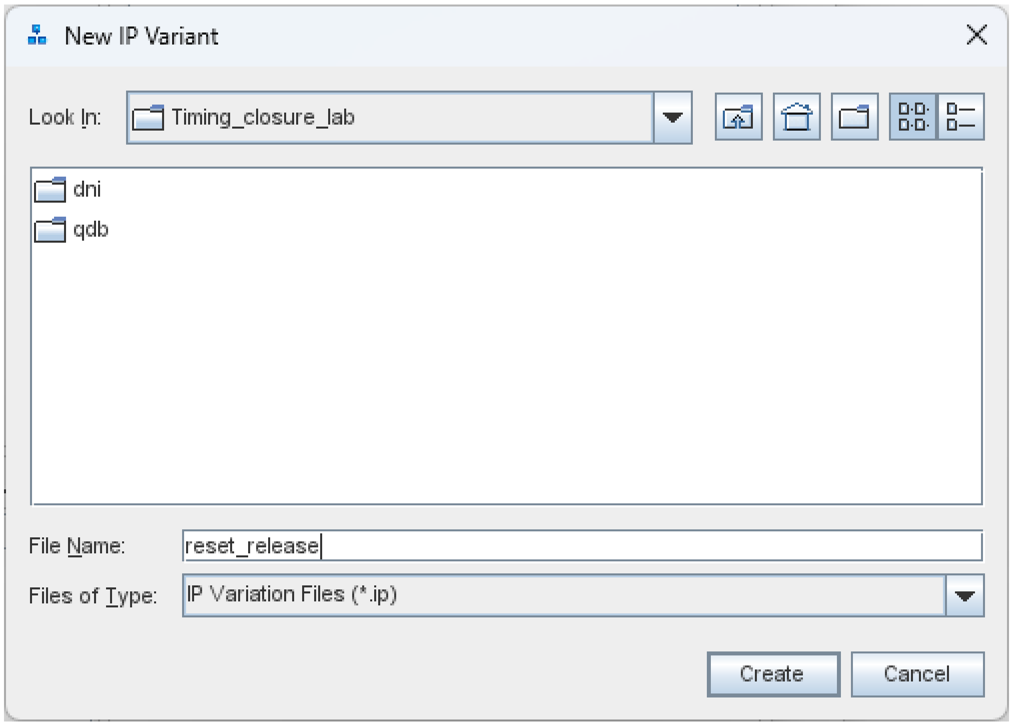


4.3.1.1 Open IP Catalog

4.3.1.2 In the search bar of the IP Catalog, type "**reset release**", and **double click** on **Agilex Reset Release Intel FPGA IP** as below. Even if part of the word "reset" is typed, Quartus will show the closest IP.


IP Catalog
reset release
Installed IP
Library
Basic Functions
Configuration and Programming
Agilex Reset Release Intel FPGA IP
Reset Release Intel FPGA IP
Simulation; Debug and Verification
Debug and Performance
Intel Configuration Reset Release Endpoint to Debug Logic
Search for Partner IP

4.3.1.3 Type the file name as **reset_release**. Select **Create** (bottom right of the window) as


New IP Variant

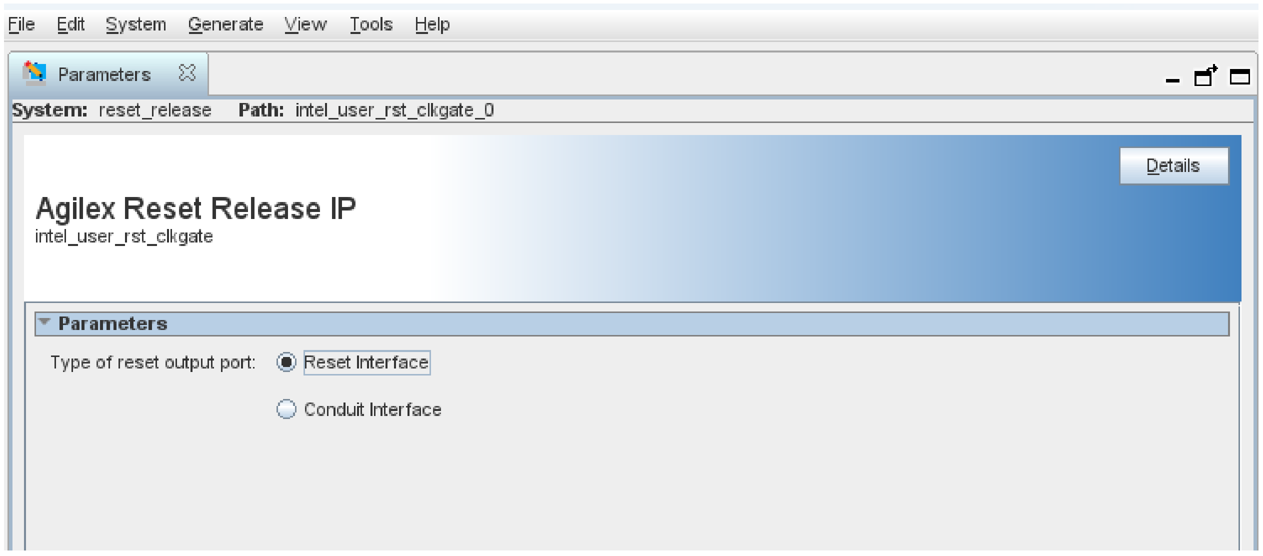
Look In: Timing_closure_lab

dni
qdb

File Name: reset_release

Files of Type: IP Variation Files (*.ip)

4.3.1.4 Select the type of reset output port as **Reset interface**



System: reset_release Path: intel_user_rst_clkgate_0

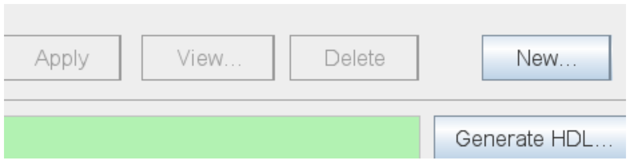
Agilex Reset Release IP
intel_user_rst_clkgate

Parameters

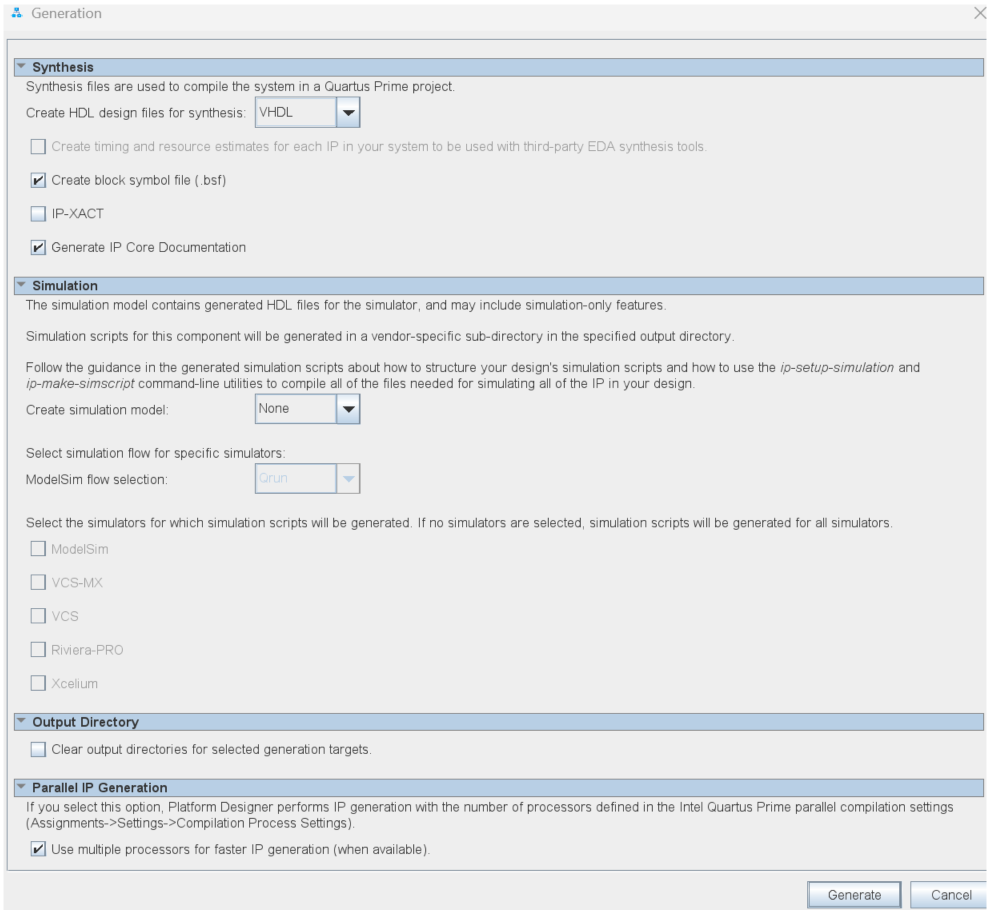
Type of reset output port: (● Reset Interface) ○ Conduit Interface

Conduit interface is a legacy reset output, thus the Reset interface should be selected.

4.3.1.5 Select **Generate HDL** in bottom right of the window



4.3.1.6 Change the **Default settings to be VHDL**. Verilog could have been used as well but this workshop was created with VHDL files. The focus for this workshop is on synthesis (not on simulation).


Generation

Synthesis

Synthesis files are used to compile the system in a Quartus Prime project.

Create HDL design files for synthesis: VHDL

☐ Create timing and resource estimates for each IP in your system to be used with third-party EDA synthesis tools.

☑ Create block symbol file (.bsf)

☐ IP-XACT

☑ Generate IP Core Documentation

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model: None

Select simulation flow for specific simulators:

ModelSim flow selection: Qrun

Select the simulators for which simulation scripts will be generated. If no simulators are selected, simulation scripts will be generated for all simulators.

☐ ModelSim

☐ VCS-MX

☐ VCS

☐ Riviera-PRO

☐ Xcelium

Output Directory

☐ Clear output directories for selected generation targets.

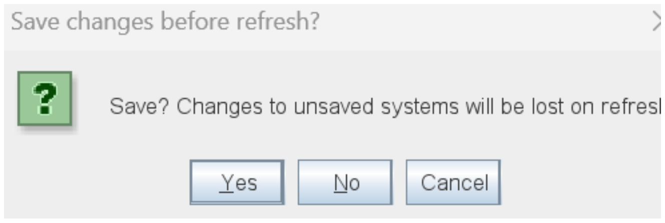
Parallel IP Generation

If you select this option, Platform Designer performs IP generation with the number of processors defined in the Intel Quartus Prime parallel compilation settings (Assignments->Settings->Compilation Process Settings).

☑ Use multiple processors for faster IP generation (when available).

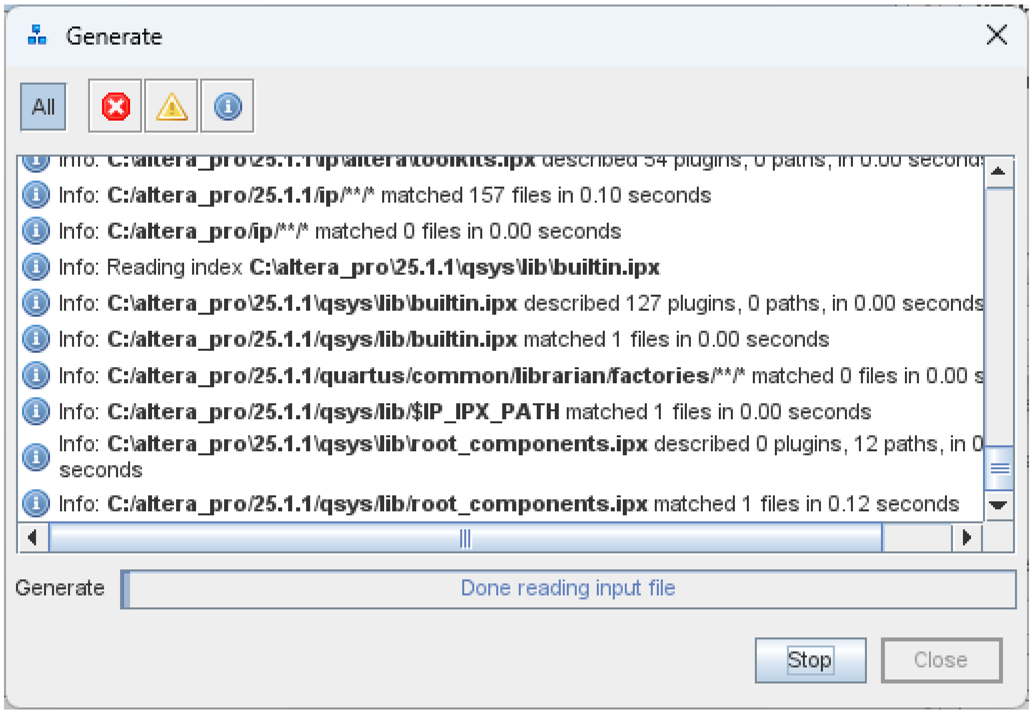
4.3.1.7 Select **Generate** (bottom right of the window).

4.3.1.8 Select **Yes** to the window of Save? Changes to unsaved systems will be lost on refresh.


Save changes before refresh?

Save? Changes to unsaved systems will be lost on refresl

4.3.1.9 Select **Close** after the Generation is complete (ie green text) as



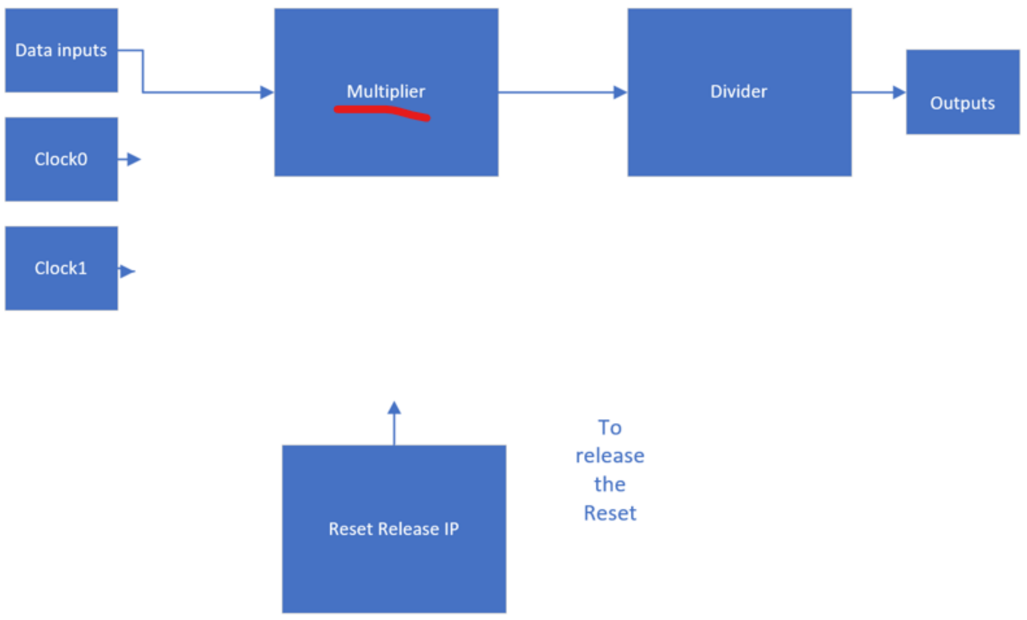
```
Info: C:\altera_pro\25.1.1\ip\altera\toolkits.ipx described 54 plugins, 0 paths, in 0.00 second
Info: C:/altera_pro/25.1.1/ip/**/* matched 157 files in 0.10 seconds
Info: C:/altera_pro/ip/**/* matched 0 files in 0.00 seconds
Info: Reading index C:\altera_pro\25.1.1\qsys\lib\builtin.ipx
Info: C:\altera_pro\25.1.1\qsys\lib\builtin.ipx described 127 plugins, 0 paths, in 0.00 seconds
Info: C:/altera_pro/25.1.1/qsys/lib/builtin.ipx matched 1 files in 0.00 seconds
Info: C:/altera_pro/25.1.1/quartus/common/librarian/factories/**/* matched 0 files in 0.00 s
Info: C:/altera_pro/25.1.1/qsys/lib/$IP_IPX_PATH matched 1 files in 0.00 seconds
Info: C:\altera_pro\25.1.1\qsys\lib\root_components.ipx described 0 plugins, 12 paths, in 0
seconds
Info: C:/altera_pro/25.1.1/qsys/lib/root_components.ipx matched 1 files in 0.12 seconds
```

Generate: Done reading input file

4.3.1.10 Close the IP Parameter Window by selecting **File** → **Exit**

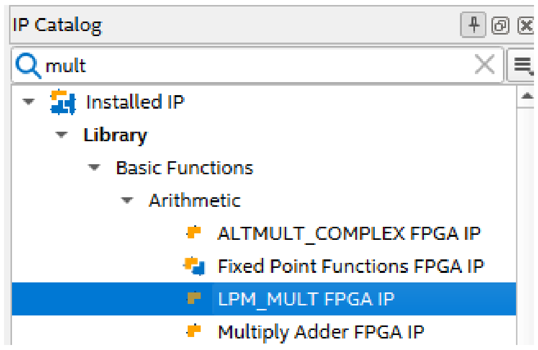
4.3.2 Add a Multiplier

There will be multiplier block to multiply input data together. The multiplier in the top-level view is seen underlined below.

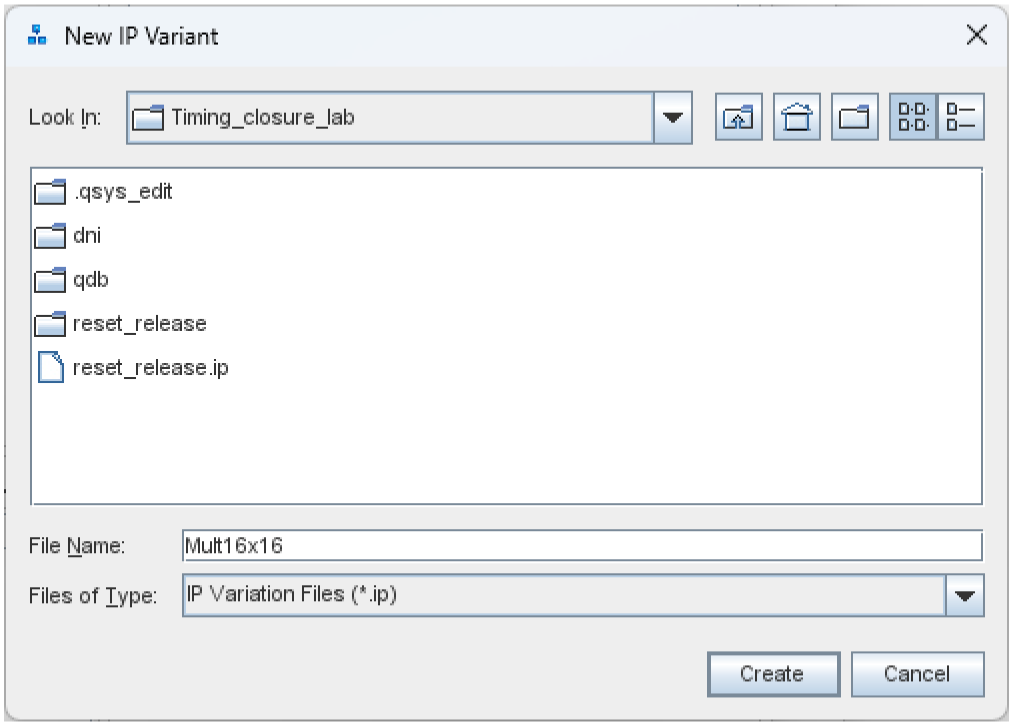


4.3.2.1 IP Catalog

In the search bar of the IP Catalog, type “**mult**”, and **double click** on **LPM_Mult Intel FPGA** as below. Even if part of the word “mult” is typed, Quartus will show the closest IP In the search bar of the IP Catalog, select LPM_Mult Intel FPGA IP.



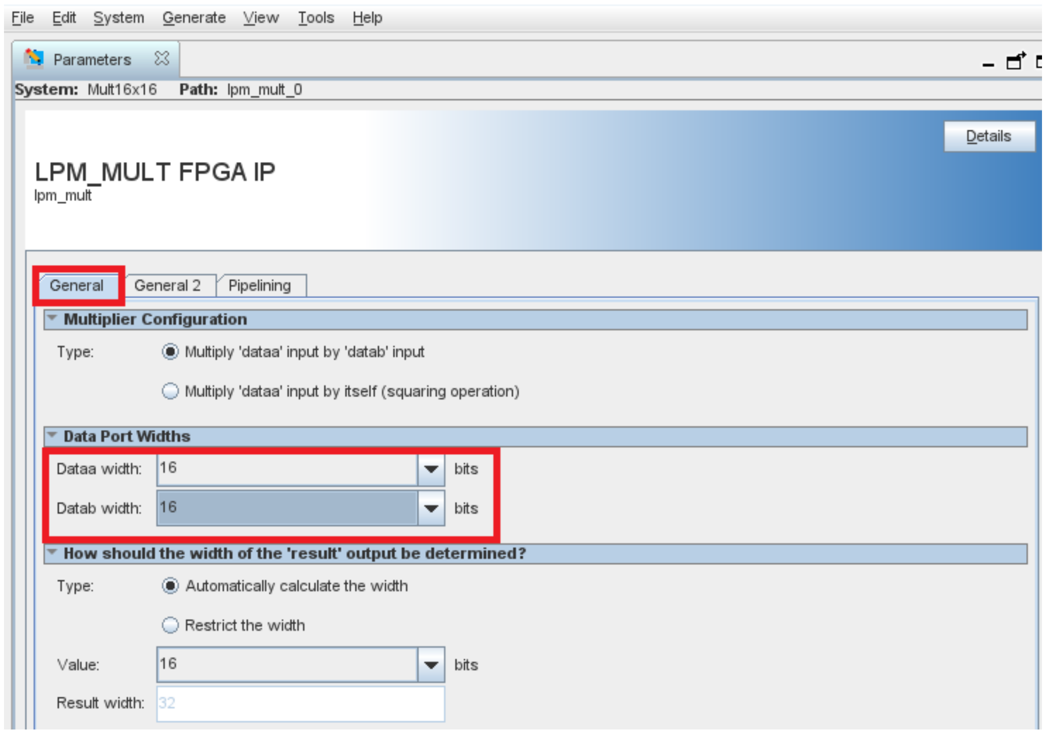
4.3.2.2 Provide the File Name. Select **Create**



File Name: Mult16x16

Configuration of the multiplier will be next. It will be configured as an 16 x 16 multiplier.

4.3.2.3 Under the General tab, change the **Dataa width =16 and Datab width =16** (Larger number of bits will take longer to compile).



System: Mult16x16 Path: lpm_mult_0

LPM_MULT FPGA IP
lpm_mult

Multiplier Configuration

Type: [✓ Multiply 'dataa' input by 'datab' input] Multiply 'dataa' input by itself (squaring operation)

Data Port Widths

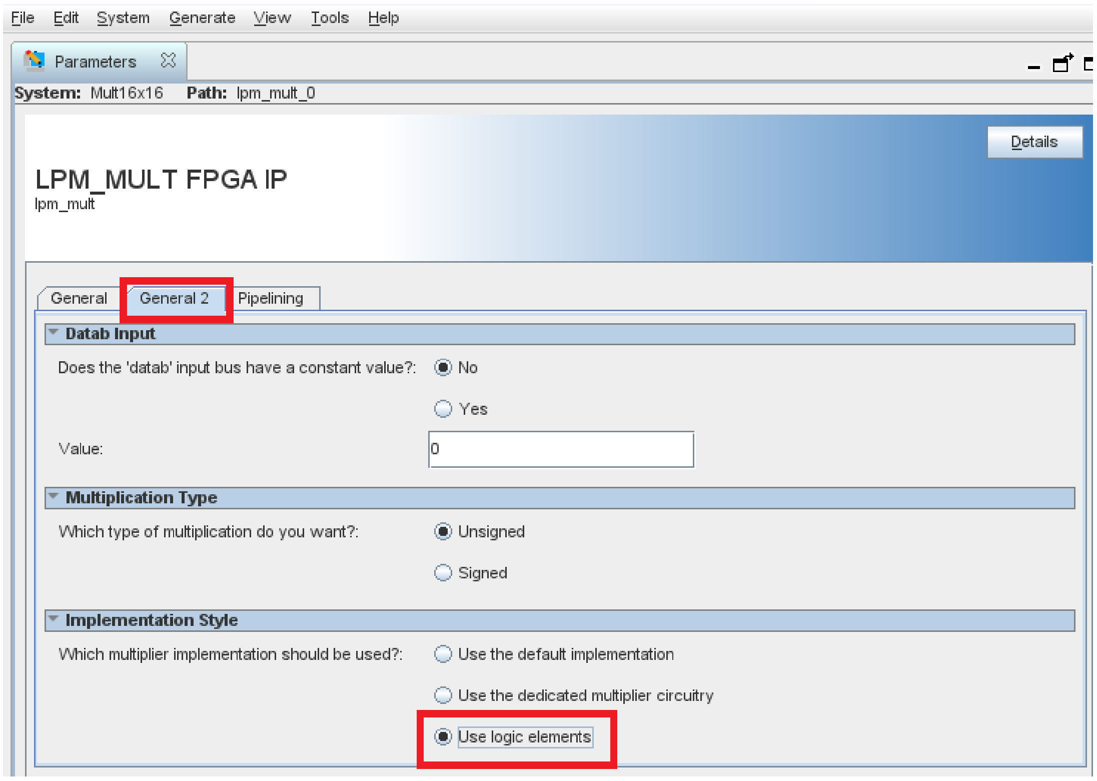
Dataa width: 16 bits
Datab width: 16 bits

How should the width of the 'result' output be determined?

Type: [✓ Automatically calculate the width] Restrict the width

Value: 16 bits
Result width: 32

4.3.2.4 Under the **General 2 tab**, select **Use Logic Elements**. The Agilex 3, which runs very fast, could implement the resources in a multiplier block, but in this case, use logic elements. (At the end of the workshop, the dedicated multiplier can be tried).



System: Mult16x16 Path: lpm_mult_0

LPM_MULT FPGA IP
lpm_mult

Datab Input

Does the 'datab' input bus have a constant value?: [✓ No] Yes

Value: 0

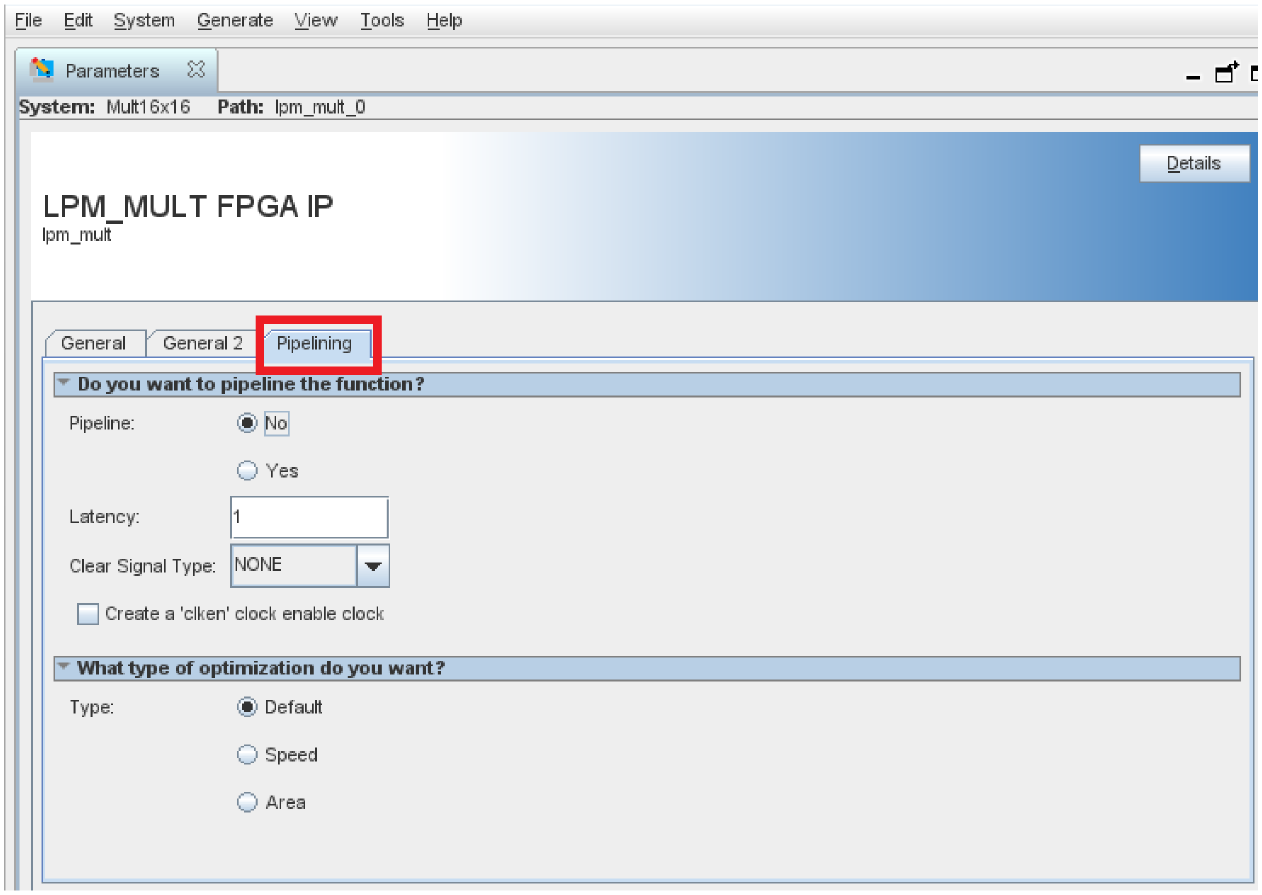
Multiplication Type

Which type of multiplication do you want?: [✓ Unsigned] Signed

Implementation Style

Which multiplier implementation should be used?: Use the default implementation / Use the dedicated multiplier circuitry / [✓ Use logic elements]

4.3.2.5 Under the **Pipelining** tab ensure that no pipelining is enabled (which is the default)



System: Mult16x16 Path: lpm_mult_0

LPM_MULT FPGA IP
lpm_mult

Do you want to pipeline the function?

Pipeline: [✓ No] Yes

Latency: 1

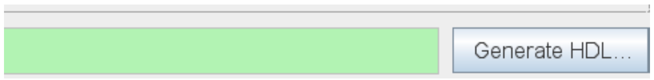
Clear Signal Type: NONE

Create a 'clken' clock enable clock

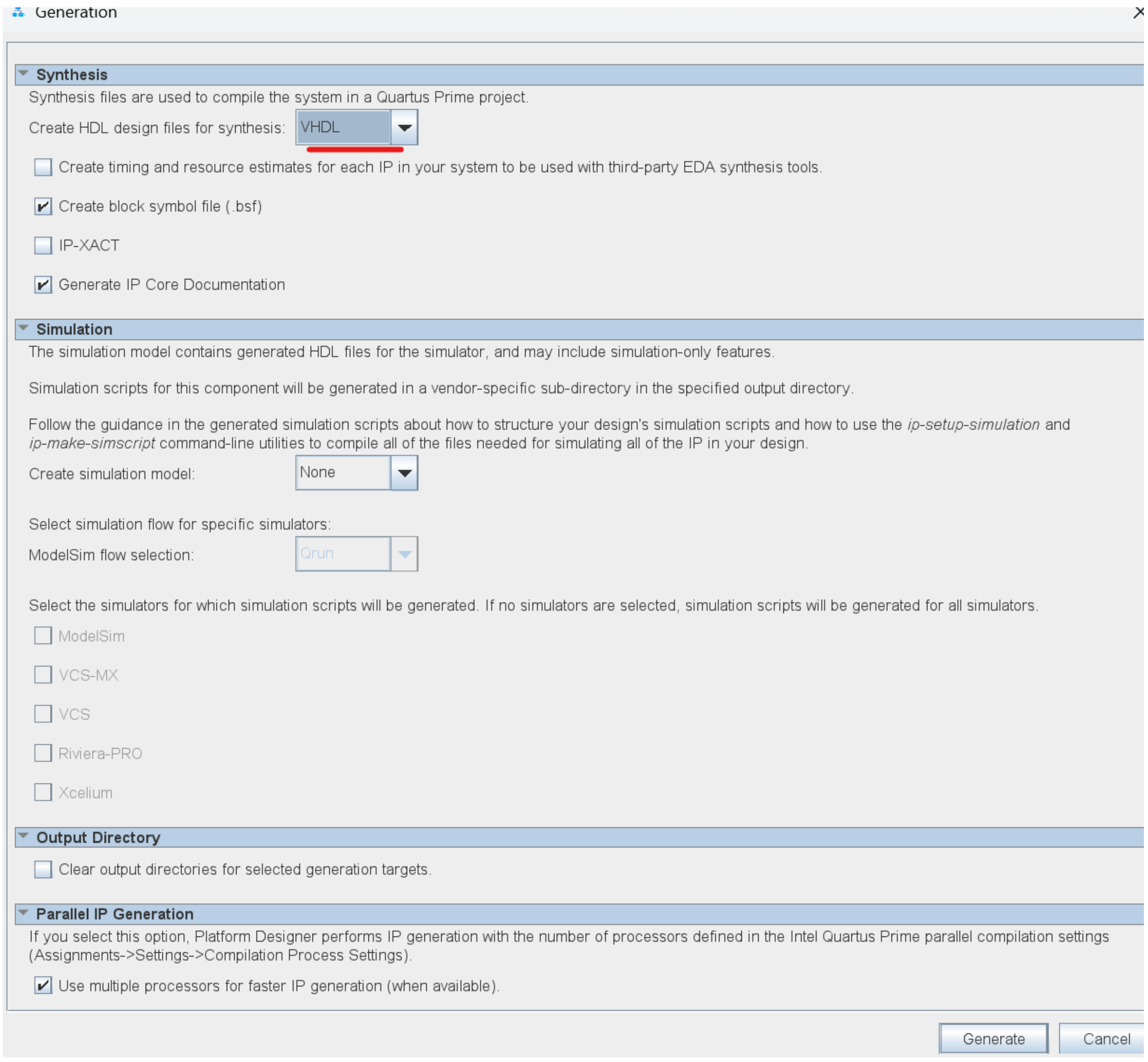
What type of optimization do you want?

Type: [✓ Default] Speed Area

4.3.2.6 Select **Generate HDL** in the bottom right corner as



4.3.2.7 Change the **HDL design files to be VHDL** (if it is not set). Verilog files could be used, but this workshop was created with VHDL.



Generation

Synthesis

Synthesis files are used to compile the system in a Quartus Prime project.

Create HDL design files for synthesis: VHDL

- [ ] Create timing and resource estimates for each IP in your system to be used with third-party EDA synthesis tools.
- [x] Create block symbol file (.bsf)
- [ ] IP-XACT
- [x] Generate IP Core Documentation

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model: None

Select simulation flow for specific simulators:

ModelSim flow selection: Qrun

Select the simulators for which simulation scripts will be generated. If no simulators are selected, simulation scripts will be generated for all simulators.

- [ ] ModelSim
- [ ] VCS-MX
- [ ] VCS
- [ ] Riviera-PRO
- [ ] Xcelium

Output Directory

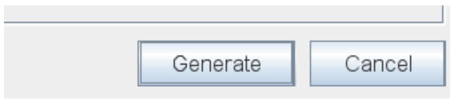
- [ ] Clear output directories for selected generation targets.

Parallel IP Generation

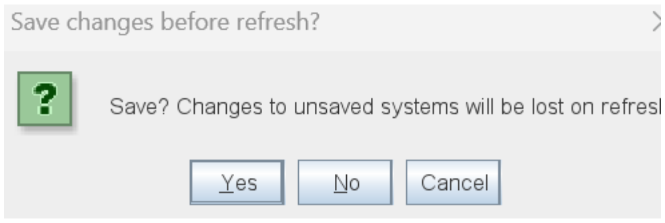
If you select this option, Platform Designer performs IP generation with the number of processors defined in the Intel Quartus Prime parallel compilation settings (Assignments->Settings->Compilation Process Settings).

- [x] Use multiple processors for faster IP generation (when available).

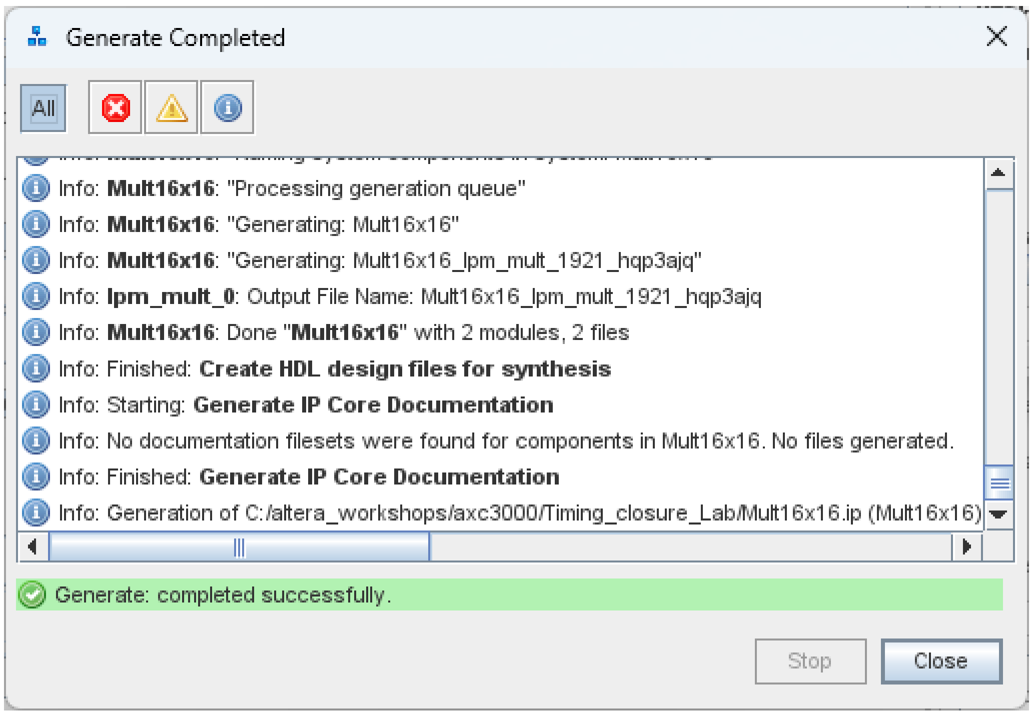
4.3.2.8 Select **Generate** in the bottom right as



4.3.2.9 Select **Yes** when prompted to Save? Changes to unsaved systems will be lost on refresh.


Save changes before refresh?
Save? Changes to unsaved systems will be lost on refres
Yes
No
Cancel

4.3.2.10 Select **Close** after the Generation is complete (ie green text) as

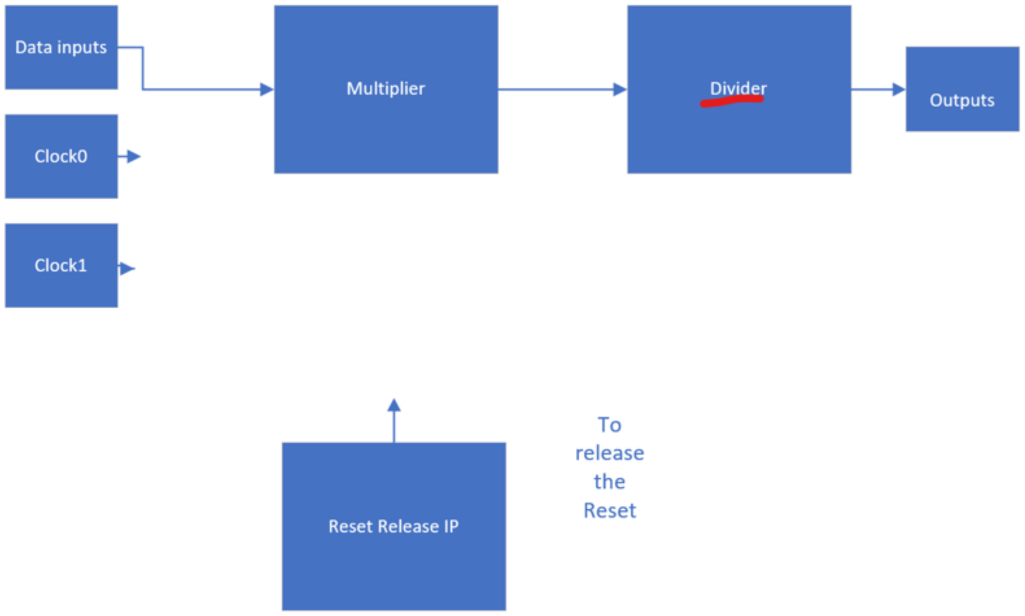


```
Info: Mult16x16: "Processing generation queue"
Info: Mult16x16: "Generating: Mult16x16"
Info: Mult16x16: "Generating: Mult16x16_lpm_mult_1921_hqp3ajq"
Info: lpm_mult_0: Output File Name: Mult16x16_lpm_mult_1921_hqp3ajq
Info: Mult16x16: Done "Mult16x16" with 2 modules, 2 files
Info: Finished: Create HDL design files for synthesis
Info: Starting: Generate IP Core Documentation
Info: No documentation filesets were found for components in Mult16x16. No files generated.
Info: Finished: Generate IP Core Documentation
Info: Generation of C:/altera_workshops/axc3000/Timing_closure_Lab/Mult16x16.ip (Mult16x16)
Generate: completed successfully.
```

4.3.2.11 Close the IP Parameter Window by selecting **File** → **Exit**

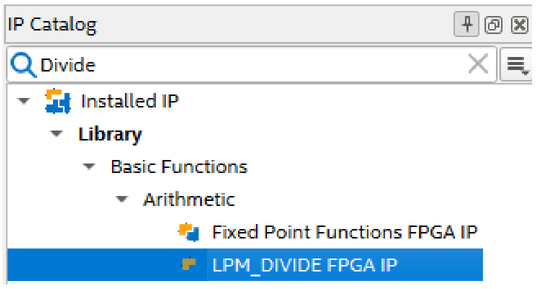
4.3.3 Add a Divider

The last IP to add is the divider. The reason for the divider is to make the design run slower, because a divider does require considerable processing. In the overview of the FPGA design, the divider is seen below.

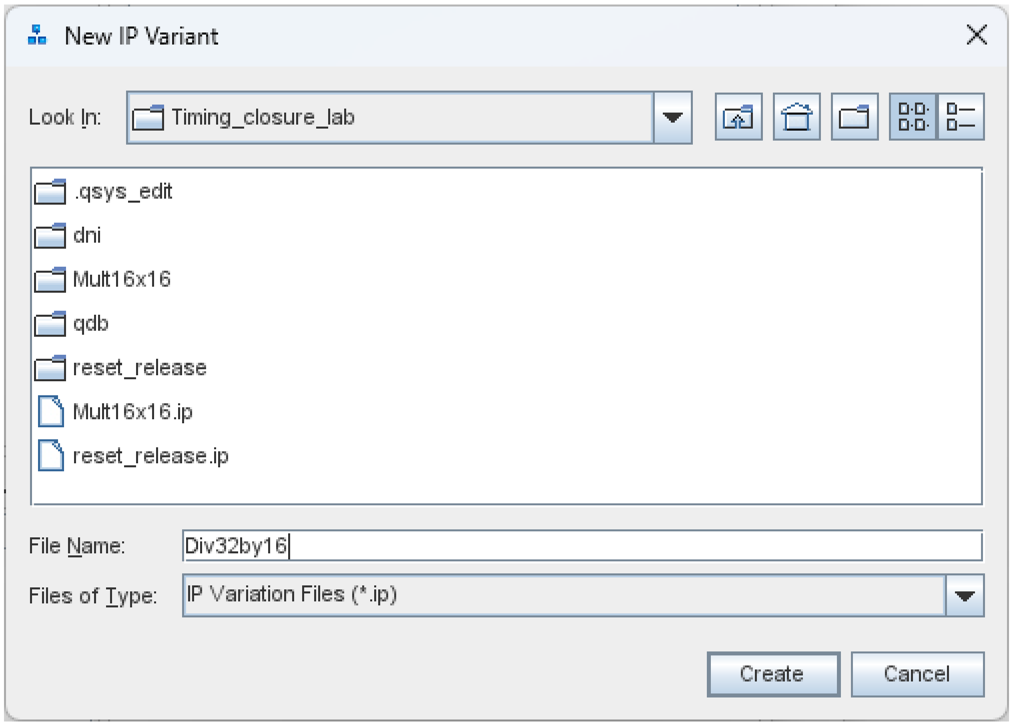


4.3.3.1Open IP Catalog

In the search bar of the IP Catalog, type "**divide**", and **double click** on **LPM_Divide FPGA IP** as below. Even if part of the word "divide" is typed, Quartus will show the closest IP In the search bar of the IP Catalog.



4.3.3.2 Provide the File Name as **Div32by16**. Select **Create** in the bottom right of the corner.

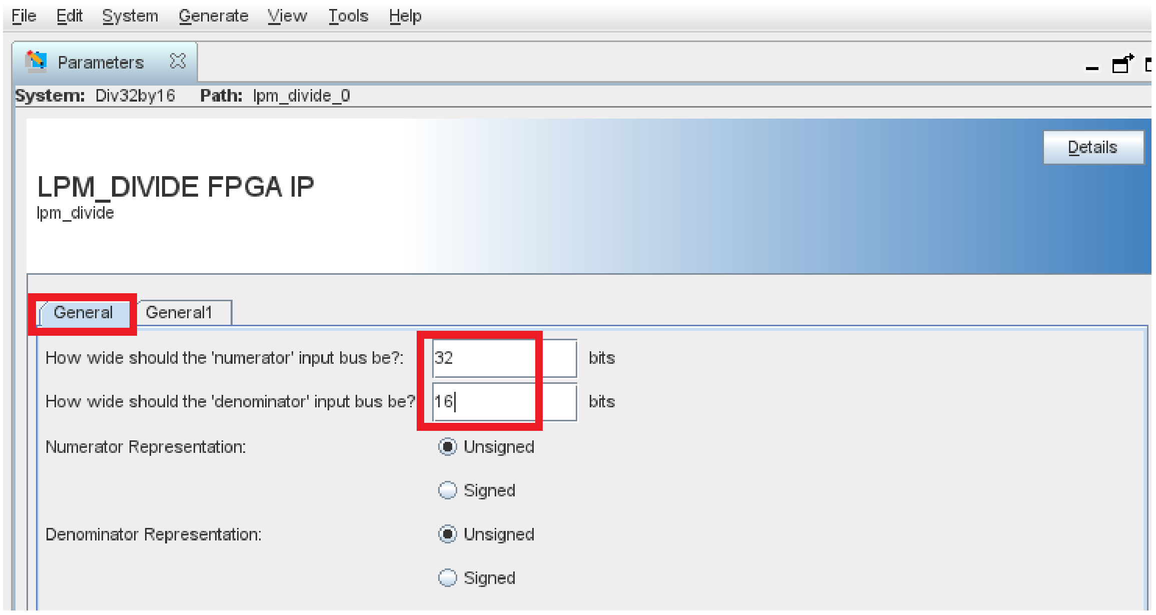


.qsys_edit
dni
Mult16x16
qdb
reset_release
Mult16x16.ip
reset_release.ip

File Name: Div32by16

Files of Type: IP Variation Files (*.ip)

4.3.3.3Under the **General** tab, change the **Numerator = 32** (this value of 32 will be connected to the output of the multiplier that has the output of 16x16 (which is set to 32) and **denominator = 16** as



System: Div32by16 Path: lpm_divide_0

LPM_DIVIDE FPGA IP
lpm_divide

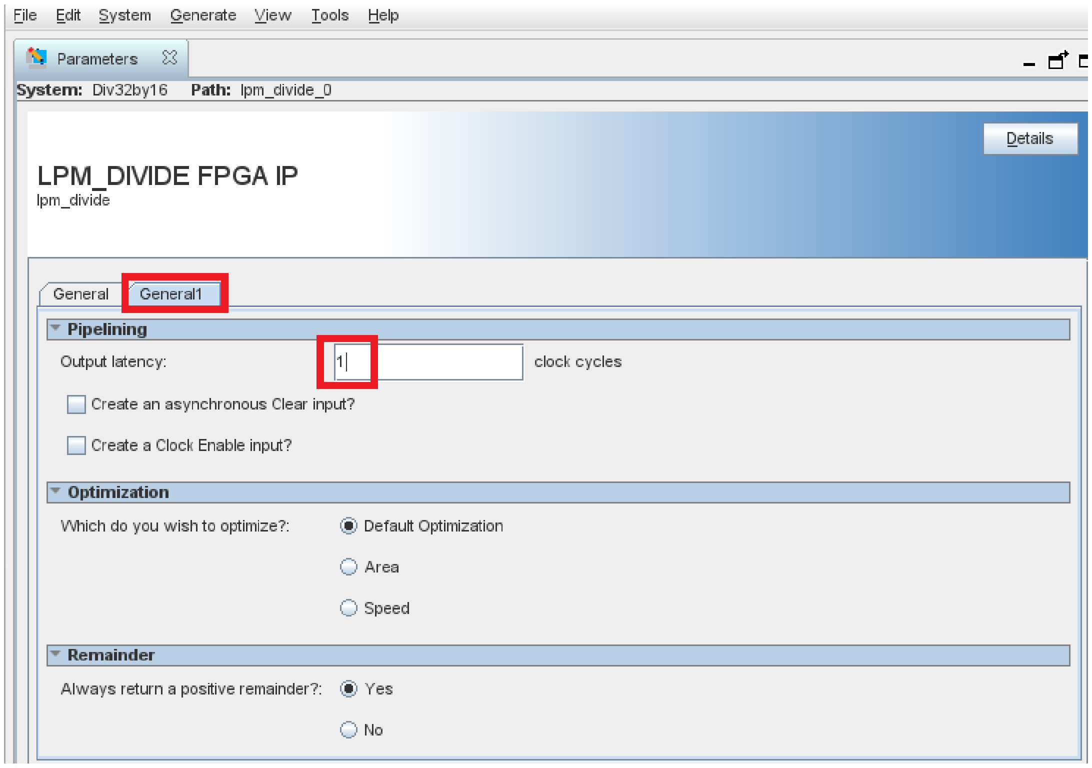
How wide should the 'numerator' input bus be?: 32 bits

How wide should the 'denominator' input bus be? 16 bits

Numerator Representation: [✓ Unsigned] Signed

Denominator Representation: [✓ Unsigned] Signed

4.3.3.4 Under the **General1 tab**, set the **output latency as 1** (just one pipeline) as



System: Div32by16 Path: lpm_divide_0

LPM_DIVIDE FPGA IP
lpm_divide

Pipelining

Output latency: 1 clock cycles

Create an asynchronous Clear input?

Create a Clock Enable input?

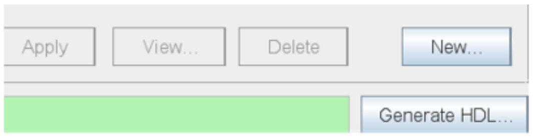
Optimization

Which do you wish to optimize?: (● Default Optimization) ○ Area ○ Speed

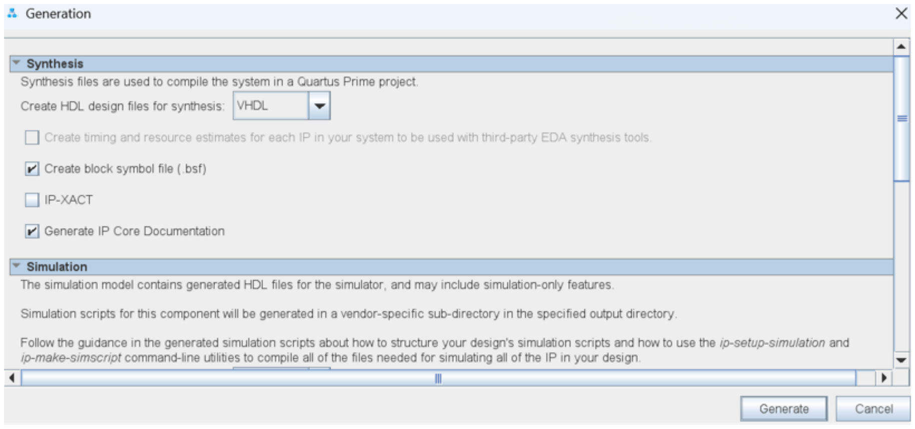
Remainder

Always return a positive remainder?: (● Yes) ○ No

4.3.3.5 Select **Generate HDL** (bottom right of the window) as



4.3.3.6 Leave the defaults (as it will be set to VHDL) and select **Generate**



Synthesis

Synthesis files are used to compile the system in a Quartus Prime project.

Create HDL design files for synthesis: VHDL

☐ Create timing and resource estimates for each IP in your system to be used with third-party EDA synthesis tools.

☑ Create block symbol file (.bsf)

☐ IP-XACT

☑ Generate IP Core Documentation

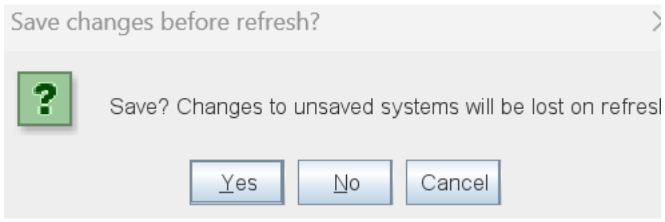
Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

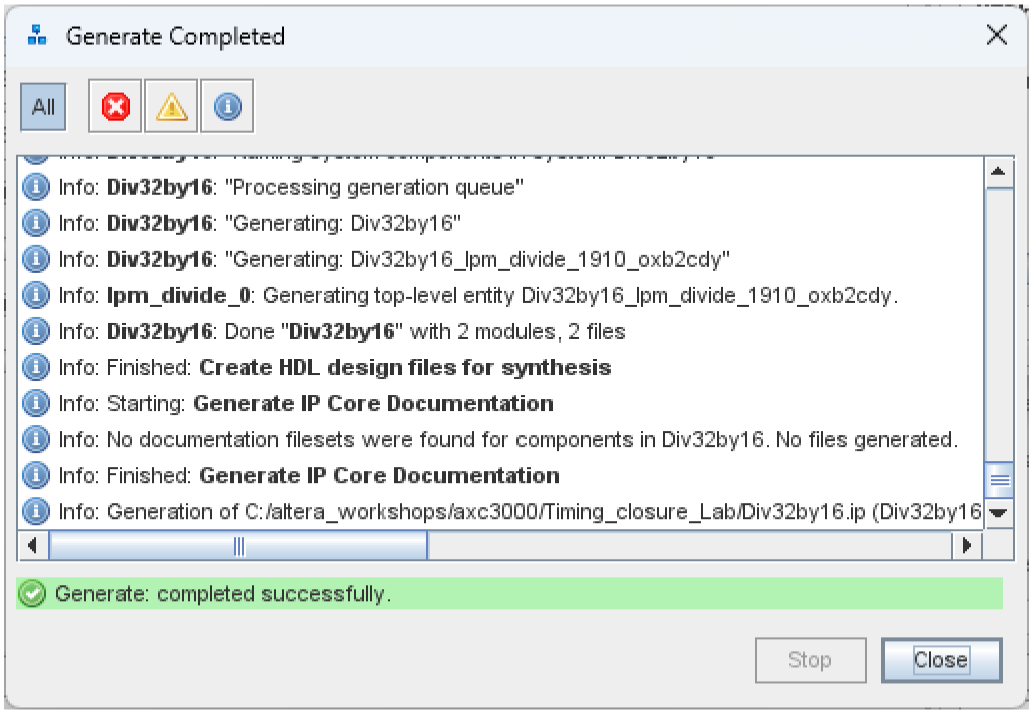
Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

4.3.3.7 Select **Yes** to the window of Save? Changes to unsaved systems will be lost on refresh.


Save changes before refresh?
Save? Changes to unsaved systems will be lost on refresl
Yes
No
Cancel

4.3.3.8 Select **Close** after the Generation is complete (ie green text) as



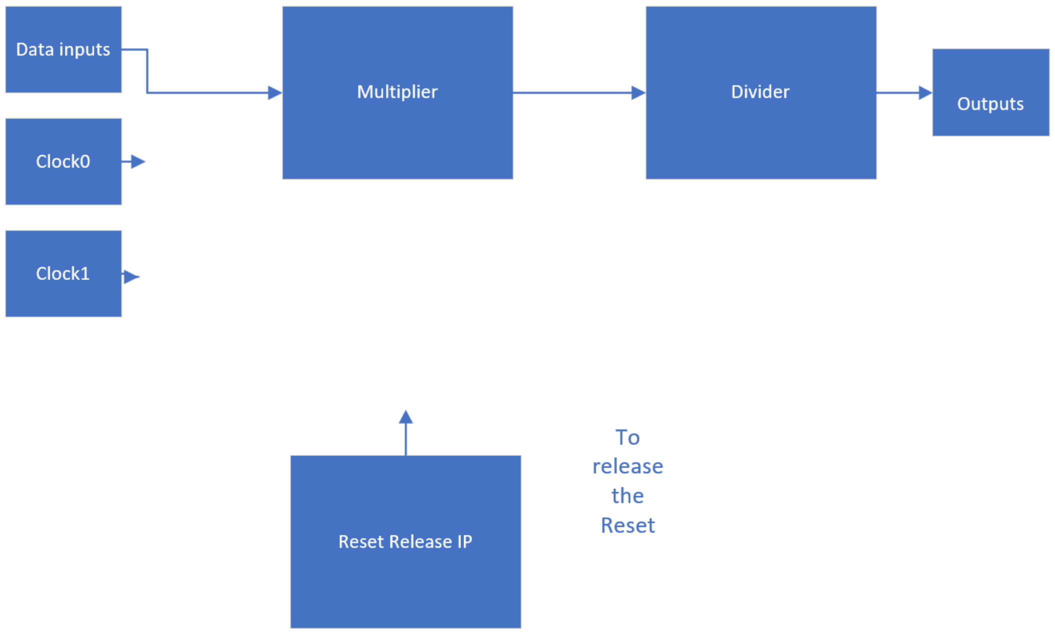
```
Info: Div32by16: "Processing generation queue"
Info: Div32by16: "Generating: Div32by16"
Info: Div32by16: "Generating: Div32by16_lpm_divide_1910_oxb2cdy"
Info: lpm_divide_0: Generating top-level entity Div32by16_lpm_divide_1910_oxb2cdy.
Info: Div32by16: Done "Div32by16" with 2 modules, 2 files
Info: Finished: Create HDL design files for synthesis
Info: Starting: Generate IP Core Documentation
Info: No documentation filesets were found for components in Div32by16. No files generated.
Info: Finished: Generate IP Core Documentation
Info: Generation of C:/altera_workshops/axc3000/Timing_closure_Lab/Div32by16.ip (Div32by16
Generate: completed successfully.
```

4.3.3.9 Select **Close**

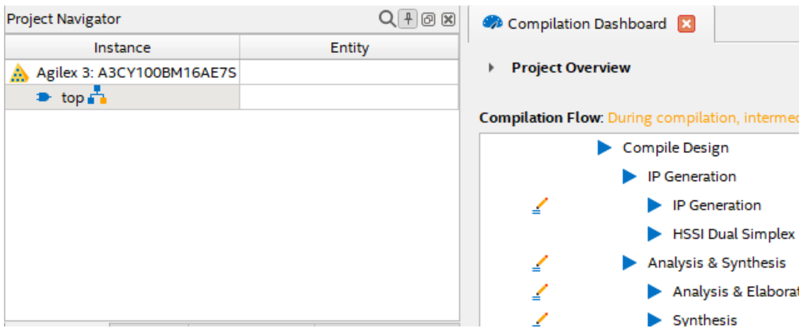
4.3.3.10 Close the IP Parameter Window by selecting **File** → **Exit**

4.3.4 Review the Top-Level Design

A top-level file is needed for each Quartus project. In this case, the top-level file, which is completed for you, will "connect" the various generated IP.



In the Project Navigator window, **double click on the top.vhd file** so that the top-level file opens.



```
library IEEE;
library altera_mult_add_1921;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
```

The file has different sections i.e. library, entity, and architecture section.

The library section calls the IEEE defined library files, which are used for the definition of handling bits, and arrays.

The entity section of top defines the inputs (seen with the word in) and output signals (seen with the word out). The signal lengths are either 1 bit (std_logic) or a vector (ie std_logic_vector).

```
entity top is
    port (
        dataa  : in  std_logic_vector(15 downto 0) := (others => '0');
        datab  : in  std_logic_vector(15 downto 0) := (others => '0');
        denom  : in  std_logic_vector(15 downto 0) := (others => '0');
        result : out std_logic_vector(31 downto 0);
        remain : out std_logic_vector(15 downto 0);
        clock0 : in  std_logic := '0';
        clock1 : in  std_logic := '0'
    );
end entity top;
```

The component section tells Quartus that there are sub-files. The components are the multiplier and divider. (For reference, the Reset Release IP should have been added here and in the port map section to ensure that device comes out of configuration correctly. The Reset Release IP does not affect timing, and the focus of the workshop is on timing).

The signal section defines the signals that are "used" to connect the signals from the multiplier or the divider to the top-level file.

Note the two clocks (clock0 and clock1) are used to clock different sections as below. These are the different clock domains.

```
CAPTURE_DATA0 : process (clock1)
begin
    if rising_edge(clock1) then
        dataa_r <= dataa;
        datab_r <= datab;
    end if;
end process CAPTURE_DATA0;

CAPTURE_DATA1 : process (clock0)
begin
    if rising_edge(clock0) then
        denom_r <= denom;
        denom_r2 <= denom_r;
    end if;
end process CAPTURE_DATA1;

CAPTURE_OUT0 : process (clock0)
begin
    if rising_edge(clock0) then
        result <= result_u;
        remain <= remain_u;
    end if;
end process CAPTURE_OUT0;
```

The final section is the port map section (for example) where there is the actual connection of sub-files signals to the top-level file signals. For example, the multiplier and divider signals are connected to the top-level file.

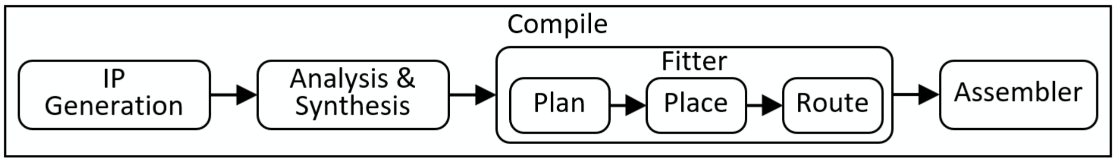
```
u0 : component Mult16x16
    port map (
        dataa  => dataa_r,
        result => mult_result,
        datab  => datab_r
    );

u1 : component Div32by16
    port map (
        numer    => mult_result,
        denom    => denom_r2,
        clock    => clock0,
        quotient => result_u,
        remain   => remain_u
    );
```

Since the file may not be easy to visualize, the following step using RTL viewer will show this better and to help with your understanding.

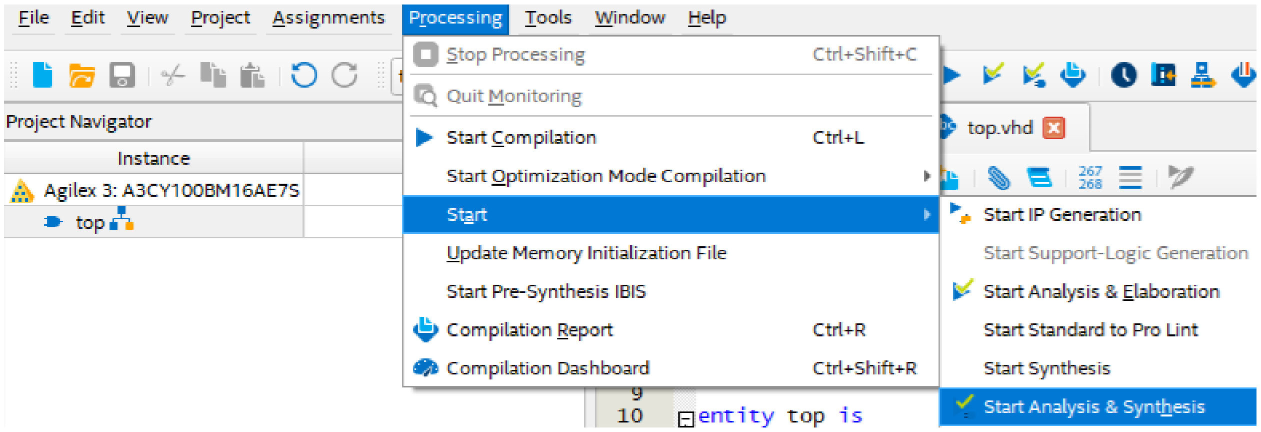
4.3.5 RTL Viewer

The RTL Viewer is a way to graphically show how Quartus sees the code. Before using the RTL Viewer, Quartus needs to check for possible syntax errors and build a database.

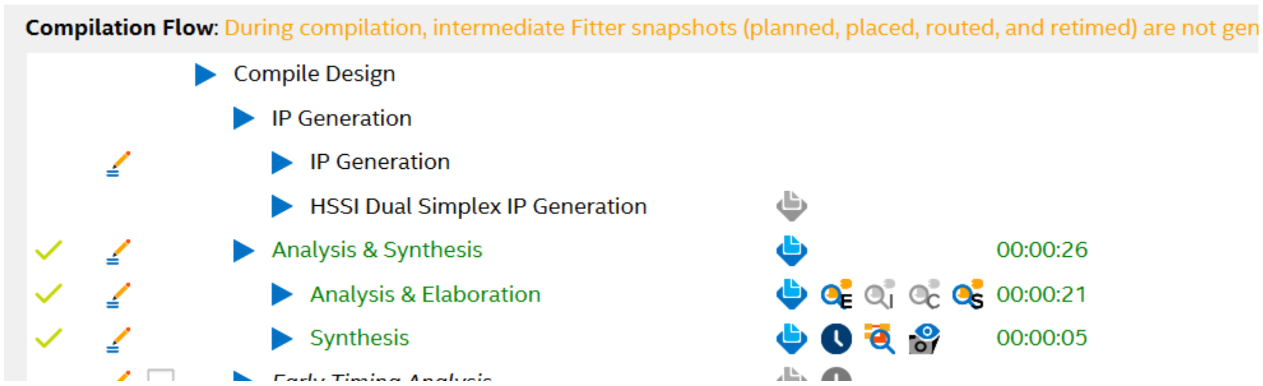
To build a database, the Analysis and Synthesis which is part of the compile flow can be selected.



4.3.5.1 Select Analysis and Synthesis by selecting

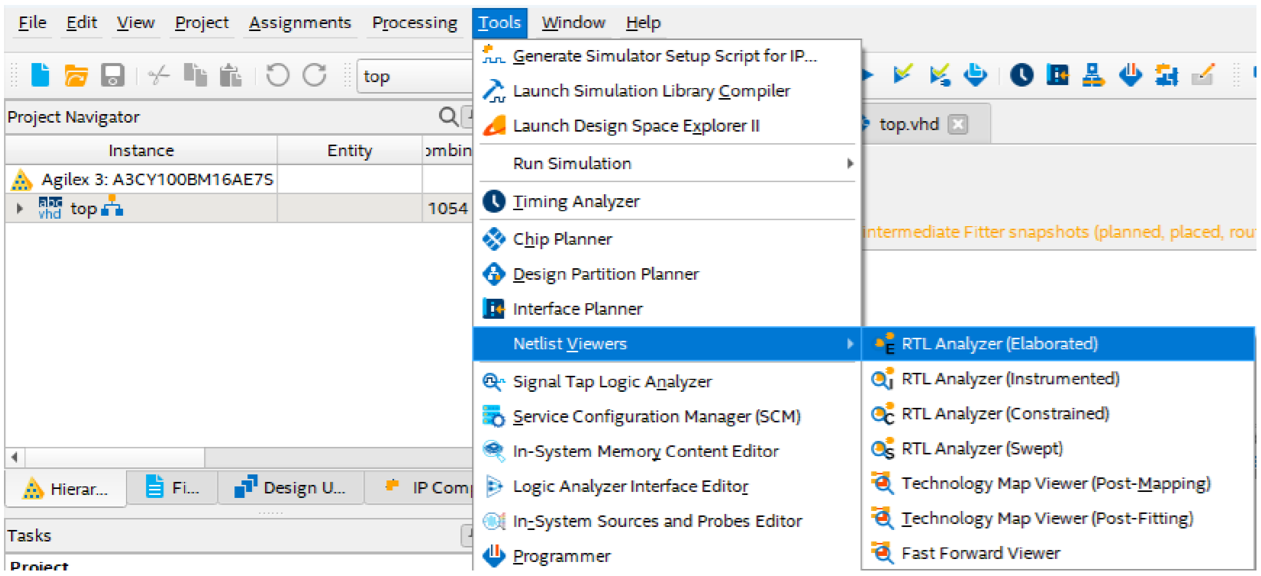

File
Edit
View
Project
Assignments
Processing
Tools
Window
Help
Project Navigator
Instance
Agilex 3: A3CY100BM16AE7S
top
Stop Processing Ctrl+Shift+C
Quit Monitoring
Start Compilation Ctrl+L
Start Optimization Mode Compilation
Start
Update Memory Initialization File
Start Pre-Synthesis IBIS
Compilation Report Ctrl+R
Compilation Dashboard Ctrl+Shift+R
top.vhd
Start IP Generation
Start Support-Logic Generation
Start Analysis & Elaboration
Start Standard to Pro Lint
Start Synthesis
Start Analysis & Synthesis
entity top is

Once the Synthesis step is complete (with the green checkmark)

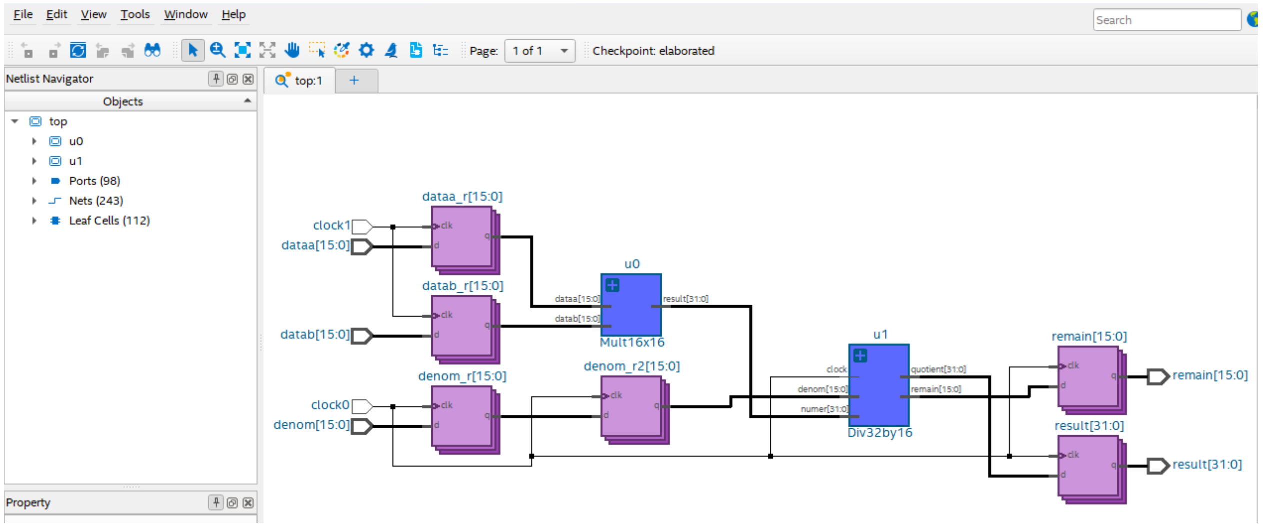

Compilation Flow: During compilation, intermediate Fitter snapshots (planned, placed, routed, and retimed) are not gen

		Compile Design	
		IP Generation	
		IP Generation	
		HSSI Dual Simplex IP Generation	
✓		Analysis & Synthesis	00:00:26
✓		Analysis & Elaboration	00:00:21
✓		Synthesis	00:00:05
		Early Timing Analysis	

4.3.5.2 Select Netlist Viewers ➔ RTL Analyzer (Elaborated) as


File
Edit
View
Project
Assignments
Processing
Tools
Window
Help
Generate Simulator Setup Script for IP...
Launch Simulation Library Compiler
Launch Design Space Explorer II
Run Simulation
Timing Analyzer
Chip Planner
Design Partition Planner
Interface Planner
Netlist Viewers
Signal Tap Logic Analyzer
Service Configuration Manager (SCM)
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
Programmer
RTL Analyzer (Elaborated)
RTL Analyzer (Instrumented)
RTL Analyzer (Constrained)
RTL Analyzer (Swept)
Technology Map Viewer (Post-Mapping)
Technology Map Viewer (Post-Fitting)
Fast Forward Viewer
Project Navigator
Instance
Entity
Agilex 3: A3CY100BM16AE7S
top
1054
top.vhd
Tasks

RTL Analyzer (Elaborated) is the step where Quartus shows graphically a view of the design close to what you have implemented. The other views of the RTL Analyzer are a result of further Quartus compilation steps. The view of the design is



Note that the registers are in pink, while the Multiplier and Divider are in blue. The input signals and output signals are registered. Registers can be used to synchronize parts of the code. Registers on the input (data) and output (result) are needed so that the fmax can be determined. Without these registers, the clock rate of the design cannot be calculated.

The input and output registers and the two clocks can be seen for example in the top-level code as

```
CAPTURE_DATA0 : process (clock1)
begin
```

```
    if rising_edge(clock1) then
        dataa_r <= dataa;
        datab_r <= datab;
    end if;
end process CAPTURE_DATA0;

CAPTURE_DATA1 : process (clock0)
begin
    if rising_edge(clock0) then
        denom_r <= denom;
        denom_r2 <= denom_r;
    end if;
end process CAPTURE_DATA1;

CAPTURE_OUT0 : process (clock0)
begin
    if rising_edge(clock0) then
        result <= result_u;
        remain <= remain_u;
    end if;
end process CAPTURE_OUT0;
```

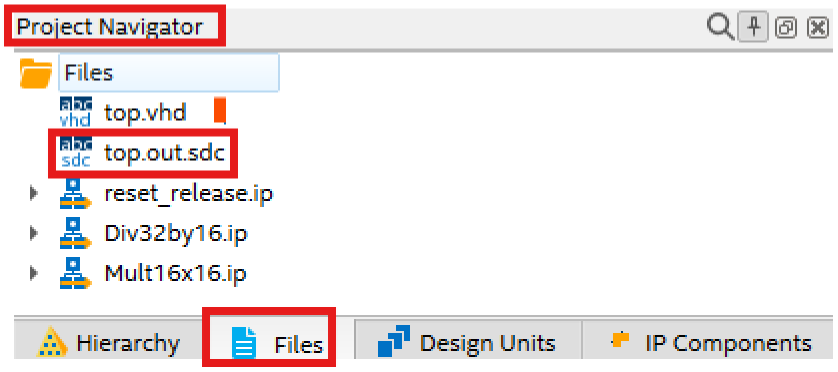
4.3.5.3 In the RTL Viewer, select File ➔ **Close.**

4.3.6 Timing Constraints

Timing Constraints are very important in a FPGA design because they instruct Quartus on how fast/slow different parts of the design are to run. Quartus uses these constraints to meet/exceed the assignments. Providing accurate timing will enable the software to easily create a FPGA design. If the timing is over constrained the software will spend longer trying to meet timing.

Timing constraints used for Quartus design is .sdc, where sdc stands for **Synopsys Design Constraint**. The .sdc file was imported in this design.

4.3.6.1 Go to the Project Navigator (top left of the Quartus window) and select the Files tab



Double-**click on the top.out.sdc file** so that the file opens. There are various lines in this file. The lines in green are comment lines. The lines that are not in green are what Quartus uses.

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

#create_clock -name {clock1-derived} -period 1.000 -waveform { 0.000 0.500 } [get_ports {clock1}] -add
#create_clock -name {clock0-derived} -period 1.000 -waveform { 0.000 0.500 } [get_ports {clock0}] -add
create_clock -name {clock0} -period 4.000 -waveform { 0.000 2.000 } [get_ports {clock0}] -add
create_clock -name {clk1} -period 4.000 -waveform { 0.200 2.200 } [get_ports {clock1}] -add
set_clock_groups -asynchronous -group [get_clocks {clk1}] -group [get_clocks {clock0}]
```

The lines that state -period 4.00 mean that 1/period (4 ns) equals 250 MHz for the clock0 and the clock1 signals.

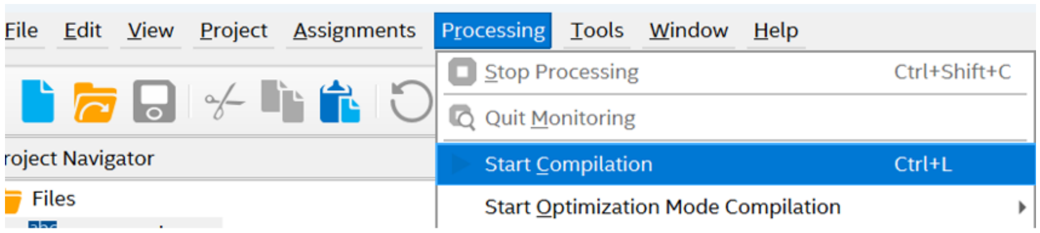
The values after the word waveform in the **{ }** state how the clocks toggle in terms high vs low.

The last line that states that asynchronous tells Quartus that the clocks are not related to each other.

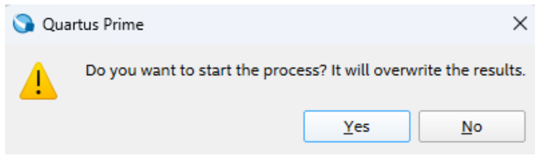
4.3.7 Compilation

Compilation is where the software takes the database that it created during the earlier step (Analysis and Synthesis) along with the timing constraints and then the software places and routes (connect) the design into the device.

The next step is to see the timing results. Select **Processing ➔ Start Compilation** as



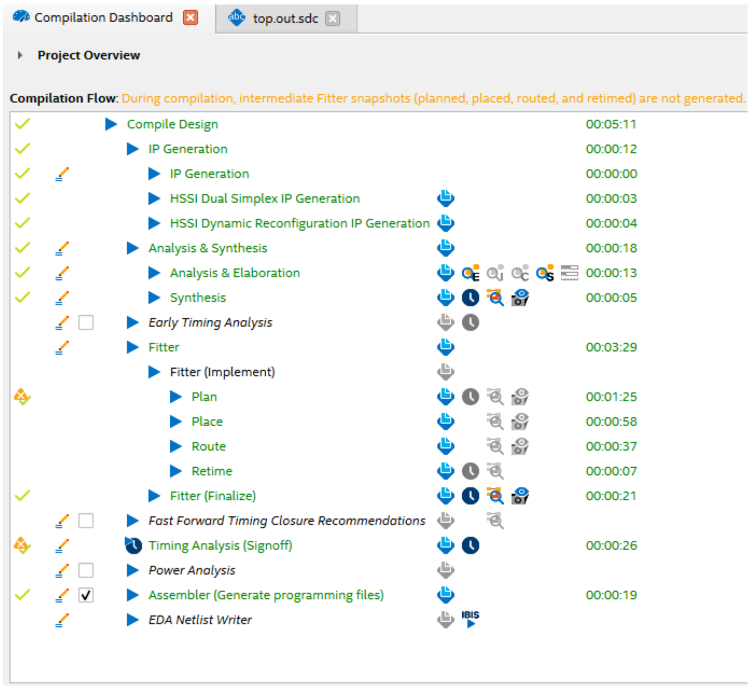
4.3.7.1 Select **Yes** to the window of


Quartus Prime

Do you want to start the process? It will overwrite the results.

During the compilation, Quartus runs through various steps such as placing and routing the design. Since this design will not be programmed into the device, there are no pin assignments and as result, there will be a warning that there are no pin assignments for almost 100 pins.

The time for the compilation depends on the computer used. On a relatively fast computer, it runs less than 6 minutes through the compile process.



Compilation Dashboard | top.out.sdc

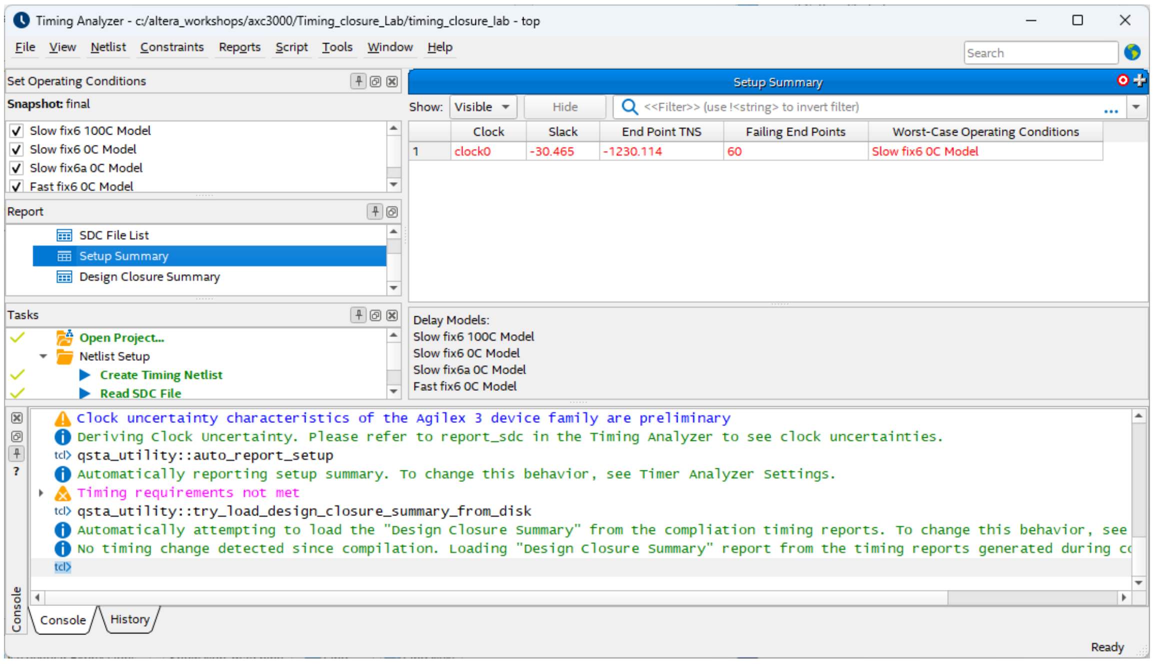
Project Overview

Compilation Flow: During compilation, intermediate Fitter snapshots (planned, placed, routed, and retimed) are not generated.

Task	Time
Compile Design	00:05:11
IP Generation	00:00:12
IP Generation	00:00:00
HSSI Dual Simplex IP Generation	00:00:03
HSSI Dynamic Reconfiguration IP Generation	00:00:04
Analysis & Synthesis	00:00:18
Analysis & Elaboration	00:00:13
Synthesis	00:00:05
Early Timing Analysis	
Fitter	00:03:29
Fitter (Implement)	
Plan	00:01:25
Place	00:00:58
Route	00:00:37
Retime	00:00:07
Fitter (Finalize)	00:00:21
Fast Forward Timing Closure Recommendations	
Timing Analysis (Signoff)	00:00:26
Power Analysis	
Assembler (Generate programming files)	00:00:19
EDA Netlist Writer	

The Timing Analyzer window will open automatically.

Note in the Timing Analyzer window that it states that the timing requirements are not met and there are setup issues (indicated with a red line). Items in red show what failed.



Timing Analyzer - c:/altera_workshops/axc3000/Timing_closure_Lab/timing_closure_lab - top

Set Operating Conditions — Snapshot: final

- ✓ Slow fix6 100C Model
- ✓ Slow fix6 0C Model
- ✓ Slow fix6a 0C Model
- ✓ Fast fix6 0C Model

Report: SDC File List, Setup Summary, Design Closure Summary

Tasks: Open Project..., Netlist Setup, Create Timing Netlist, Read SDC File

Setup Summary

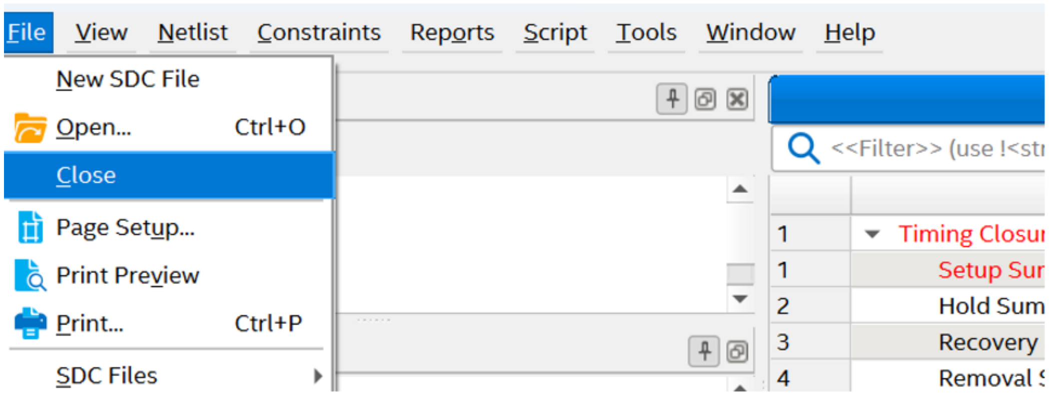
	Clock	Slack	End Point TNS	Failing End Points	Worst-Case Operating Conditions
1	clock0	-30.465	-1230.114	60	Slow fix6 0C Model

Delay Models:
Slow fix6 100C Model
Slow fix6 0C Model
Slow fix6a 0C Model
Fast fix6 0C Model

```
Clock uncertainty characteristics of the Agilex 3 device family are preliminary
Deriving Clock Uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
tcl> qsta_utility::auto_report_setup
Automatically reporting setup summary. To change this behavior, see Timer Analyzer Settings.
Timing requirements not met
tcl> qsta_utility::try_load_design_closure_summary_from_disk
Automatically attempting to load the "Design Closure Summary" from the compliation timing reports. To change this behavior, see
No timing change detected since compilation. Loading "Design Closure Summary" report from the timing reports generated during co
tcl>
```

Ready

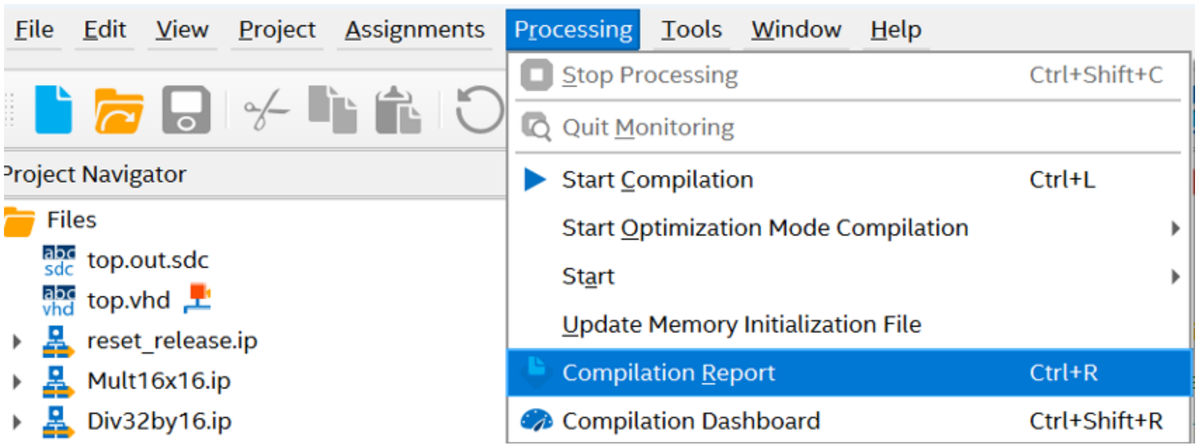
4.3.7.2 Close the Timing Analyzer window by selecting **File ➔ Close**



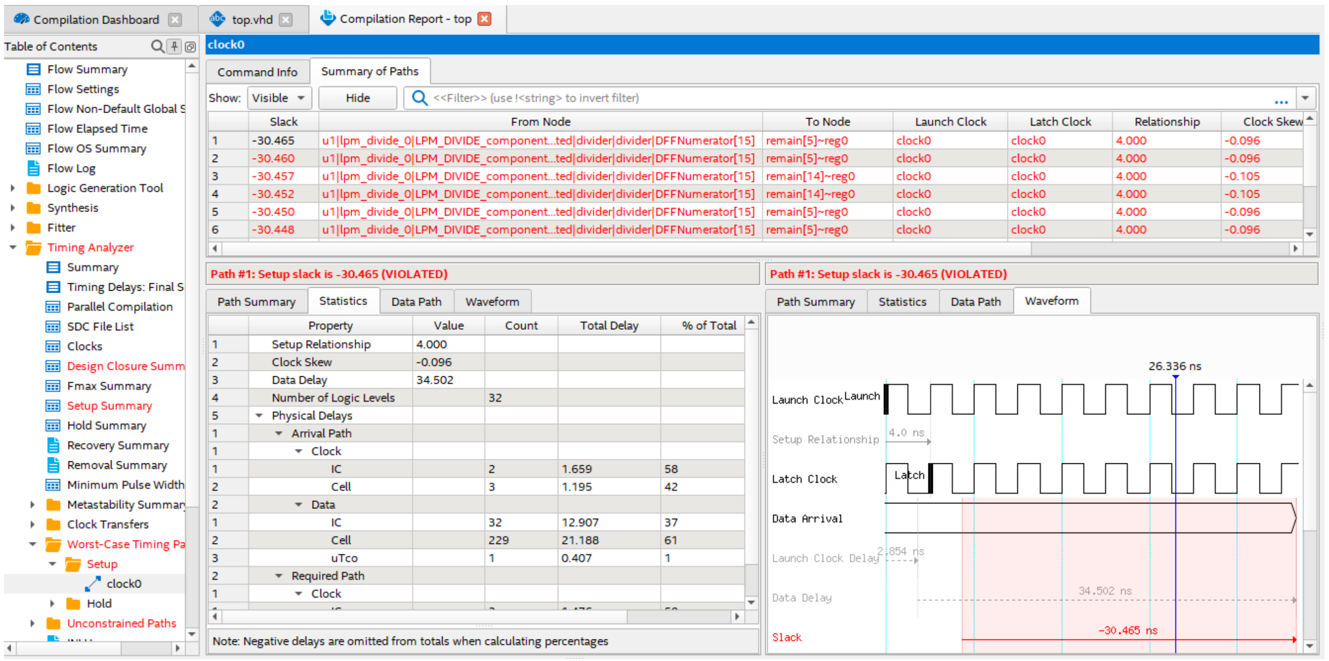
4.3.8 Compilation Results and Timing Results

One way to see the timing path results is to open the Compilation Report...

4.3.8.1 Select Processing ➔ Compilation Report as



4.3.8.2 And then expand the Timing Analyzer ➔ Worst-Case Timing Paths ➔ Setup as



	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew
1	-30.465	u1|lpm_divide_0|LPM_DIVIDE_component...ted|divider|divider|DFFNumerator[15]	remain[5]~reg0	clock0	clock0	4.000	-0.096
2	-30.460	u1|lpm_divide_0|LPM_DIVIDE_component...ted|divider|divider|DFFNumerator[15]	remain[5]~reg0	clock0	clock0	4.000	-0.096
3	-30.457	u1|lpm_divide_0|LPM_DIVIDE_component...ted|divider|divider|DFFNumerator[15]	remain[14]~reg0	clock0	clock0	4.000	-0.105
4	-30.452	u1|lpm_divide_0|LPM_DIVIDE_component...ted|divider|divider|DFFNumerator[15]	remain[14]~reg0	clock0	clock0	4.000	-0.105
5	-30.450	u1|lpm_divide_0|LPM_DIVIDE_component...ted|divider|divider|DFFNumerator[15]	remain[5]~reg0	clock0	clock0	4.000	-0.096
6	-30.448	u1|lpm_divide_0|LPM_DIVIDE_component...ted|divider|divider|DFFNumerator[15]	remain[5]~reg0	clock0	clock0	4.000	-0.096

Path #1: Setup slack is -30.465 (VIOLATED)

	Property	Value	Count	Total Delay	% of Total
1	Setup Relationship	4.000			
2	Clock Skew	-0.096			
3	Data Delay	34.502			
4	Number of Logic Levels		32		
5	Physical Delays				
1	Arrival Path				
1	Clock				
1	IC		2	1.659	58
2	Cell		3	1.195	42
2	Data				
1	IC		32	12.907	37
2	Cell		229	21.188	61
3	uTco		1	0.407	1
2	Required Path				
1	Clock				

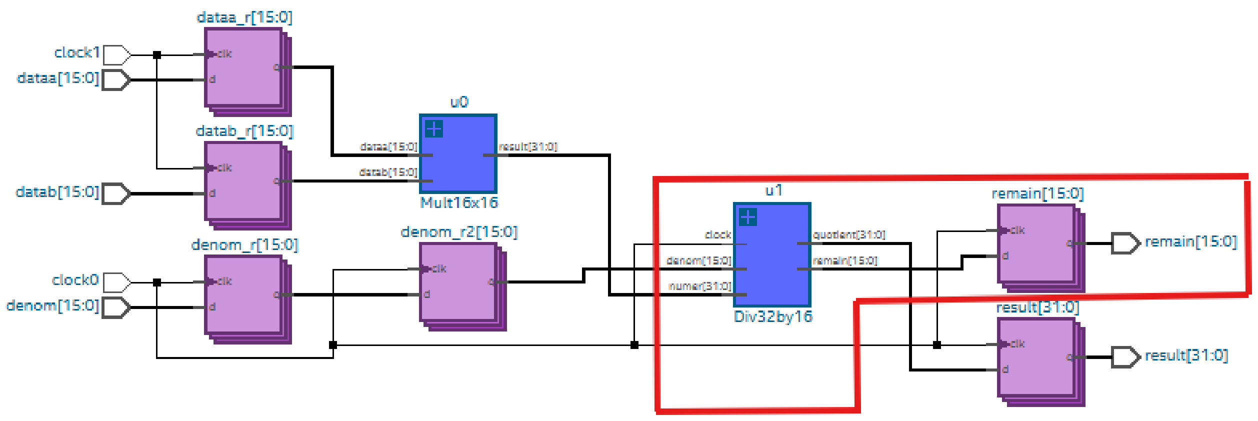
Note: Negative delays are omitted from totals when calculating percentages

Path #1: Setup slack is -30.465 (VIOLATED)

Note the many negative slack violations of -30.465 ns. This is a large negative slack which means the data will not be caught correctly. The reason for 10 failing paths is that the limit is set (by default) to be 10. The number of 10 should be set to larger to see more results. To ensure a design will work, the slack needs to be above zero for all corner cases.

Setup violations occur when a signal arrives at a flip-flop too late, missing the setup time requirement before the clock edge.

One thing to note is the nodes that are failing are from the divider (lpm_divide) to the remain[6]/remain[5] registers. Another way to look at this in the RTL Viewer (view seen earlier) is the stage from the div32by16 to remain[15..0] is the issue.



Another item to note: High negative slack means that items in the design need to be changed. If the negative slack had been very small i.e. picoseconds range, there are ways within Quartus to try to change the settings (i.e. by changing the initial seed setting of the compiler) to try to make it meet timing rather than changing the code.

4.3.9 How to resolve Timing issues Attempt One

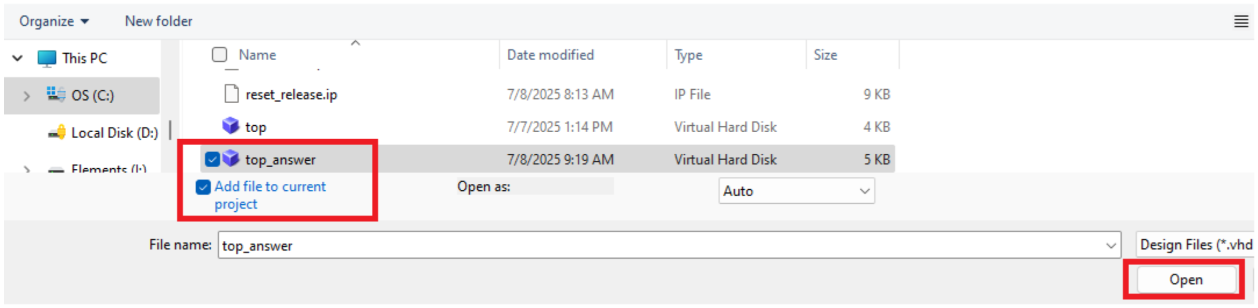
There are different ways to solve timing issues.

One way is to add pipelines (registers). The clock makes the registers synchronized and provided more time for the data to arrive.

To avoid possible typing issues of creating pipelines in the code, a VHDL file was created where the registers were pre-added.

4.3.9.1 Add the file **top_answer.vhd** to the project

4.3.9.2 Within Quartus, select File → Open → select top_answer and Select Add File to Current Project.

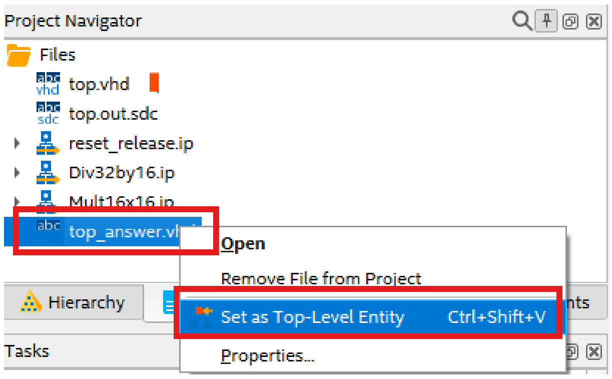


Name	Date modified	Type	Size
reset_release.ip	7/8/2025 8:13 AM	IP File	9 KB
top	7/7/2025 1:14 PM	Virtual Hard Disk	4 KB
top_answer	7/8/2025 9:19 AM	Virtual Hard Disk	5 KB

File name: top_answer

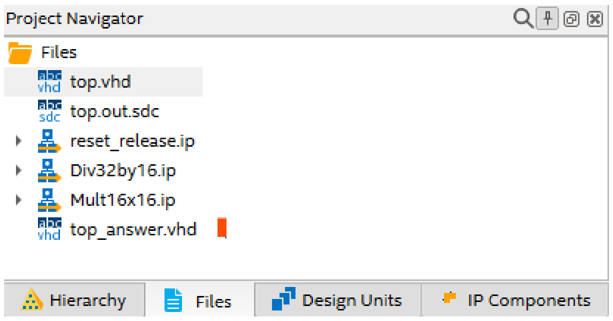
4.3.9.3 Select **Open**

The file should now look like the following figure in the Project Navigator. Right click on the top_answer.vhd and set as Top-Level Entity.

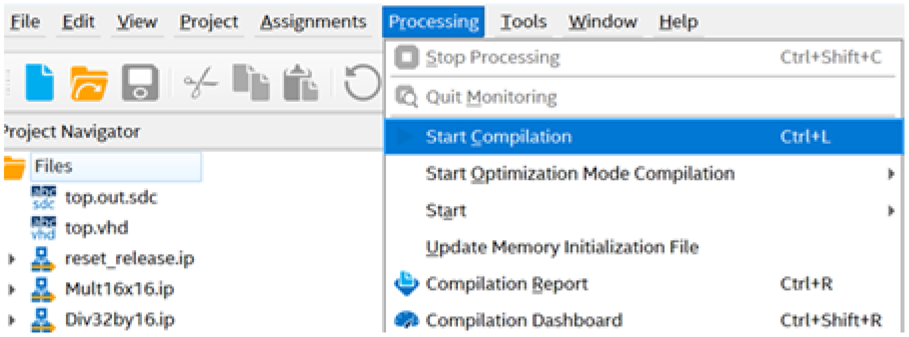


The icon of the top-level entity now changes to the top_answer file as follows.

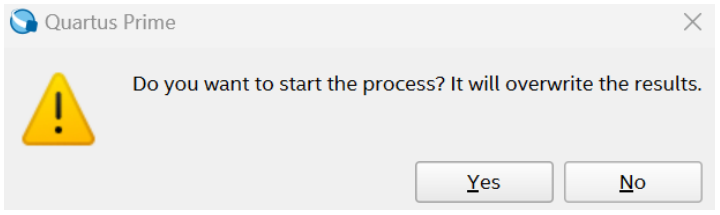
Top Level Entity means the Quartus will compile from that file down so that it compiles the sub files.



4.3.9.4 Select Start Compilation as



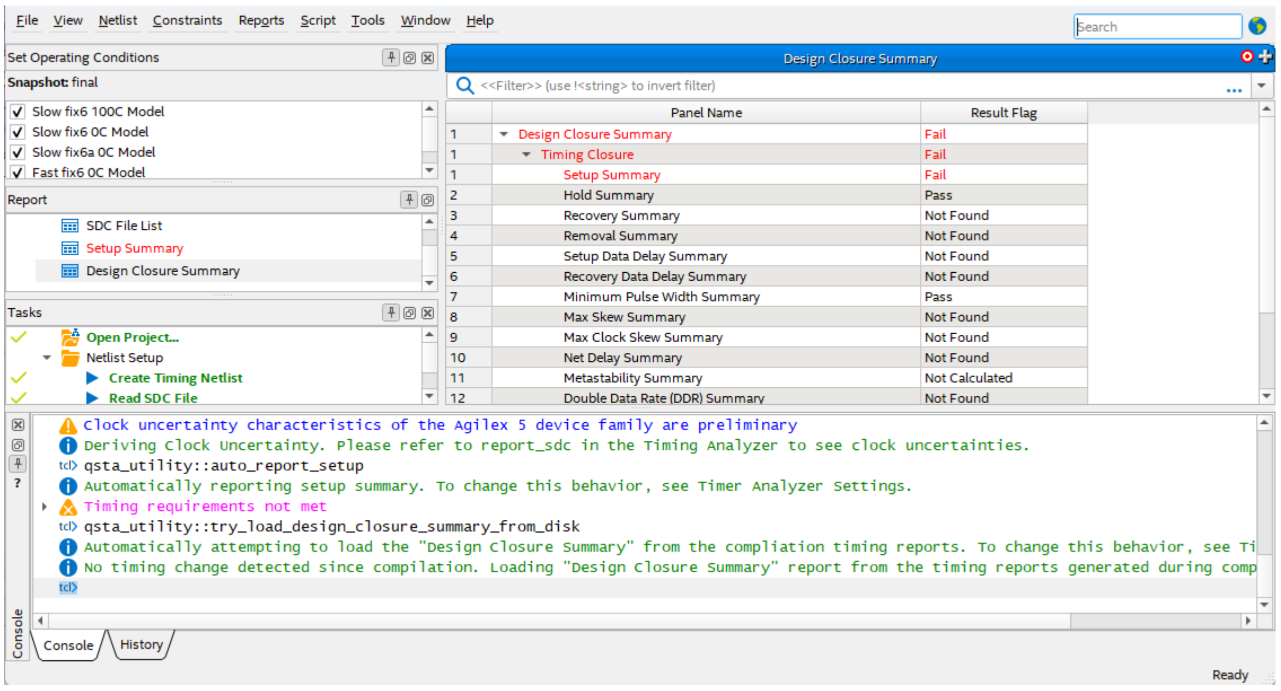
4.3.9.5 Select **Yes** to the pop-up window of


Quartus Prime

Do you want to start the process? It will overwrite the results.

It will then compile.

The Timing Analyzer window opens, and it will show that there are still timing violations as seen in red (top right) and the bottom line in purple.



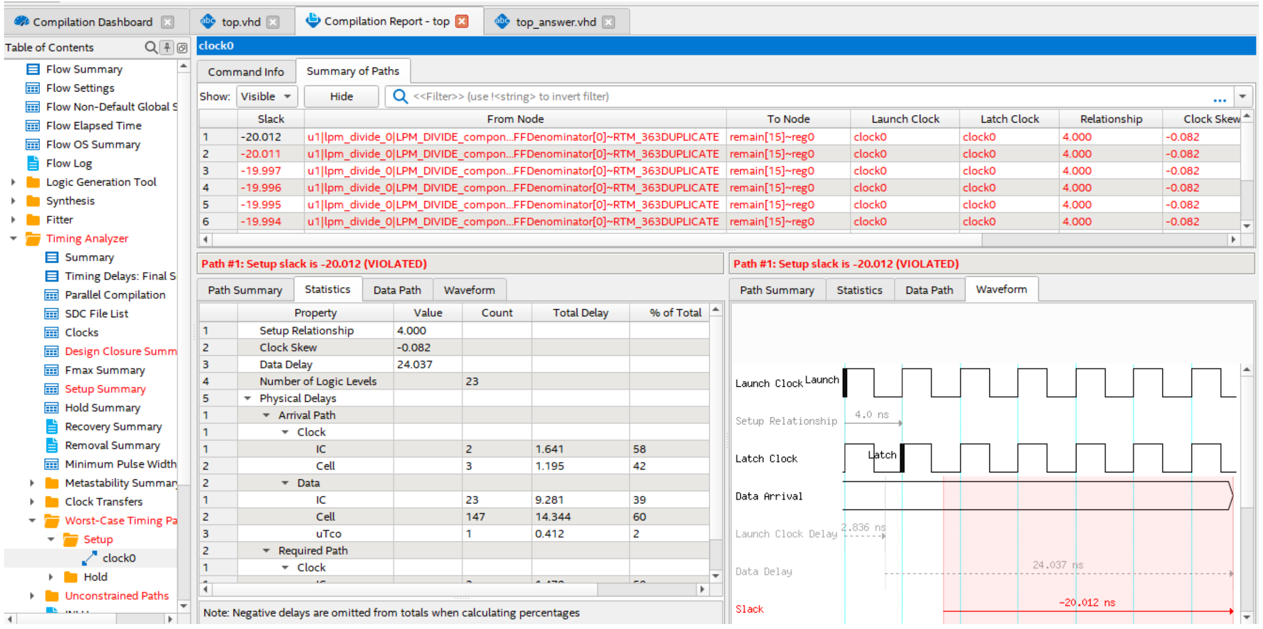
Design Closure Summary

	Panel Name	Result Flag
1	Design Closure Summary	Fail
1	Timing Closure	Fail
1	Setup Summary	Fail
2	Hold Summary	Pass
3	Recovery Summary	Not Found
4	Removal Summary	Not Found
5	Setup Data Delay Summary	Not Found
6	Recovery Data Delay Summary	Not Found
7	Minimum Pulse Width Summary	Pass
8	Max Skew Summary	Not Found
9	Max Clock Skew Summary	Not Found
10	Net Delay Summary	Not Found
11	Metastability Summary	Not Calculated
12	Double Data Rate (DDR) Summary	Not Found

```
Clock uncertainty characteristics of the Agilex 5 device family are preliminary
Deriving Clock Uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
tcl> qsta_utility::auto_report_setup
Automatically reporting setup summary. To change this behavior, see Timer Analyzer Settings.
Timing requirements not met
tcl> qsta_utility::try_load_design_closure_summary_from_disk
Automatically attempting to load the "Design Closure Summary" from the compliation timing reports. To change this behavior, see Ti
No timing change detected since compilation. Loading "Design Closure Summary" report from the timing reports generated during comp
tcl>
```

4.3.9.6 Close the **Timing Analyzer Window** that opens automatically.

4.3.9.7 Select in Quartus ➔ Processing ➔ Compilation Report and open the Timing Analyzer results as



	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew
1	-20.012	u1|lpm_divide_0|LPM_DIVIDE_compon...FFDenominator[0]~RTM_363DUPLICATE	remain[15]~reg0	clock0	clock0	4.000	-0.082
2	-20.011	u1|lpm_divide_0|LPM_DIVIDE_compon...FFDenominator[0]~RTM_363DUPLICATE	remain[15]~reg0	clock0	clock0	4.000	-0.082
3	-19.997	u1|lpm_divide_0|LPM_DIVIDE_compon...FFDenominator[0]~RTM_363DUPLICATE	remain[15]~reg0	clock0	clock0	4.000	-0.082
4	-19.996	u1|lpm_divide_0|LPM_DIVIDE_compon...FFDenominator[0]~RTM_363DUPLICATE	remain[15]~reg0	clock0	clock0	4.000	-0.082
5	-19.995	u1|lpm_divide_0|LPM_DIVIDE_compon...FFDenominator[0]~RTM_363DUPLICATE	remain[15]~reg0	clock0	clock0	4.000	-0.082
6	-19.994	u1|lpm_divide_0|LPM_DIVIDE_compon...FFDenominator[0]~RTM_363DUPLICATE	remain[15]~reg0	clock0	clock0	4.000	-0.082

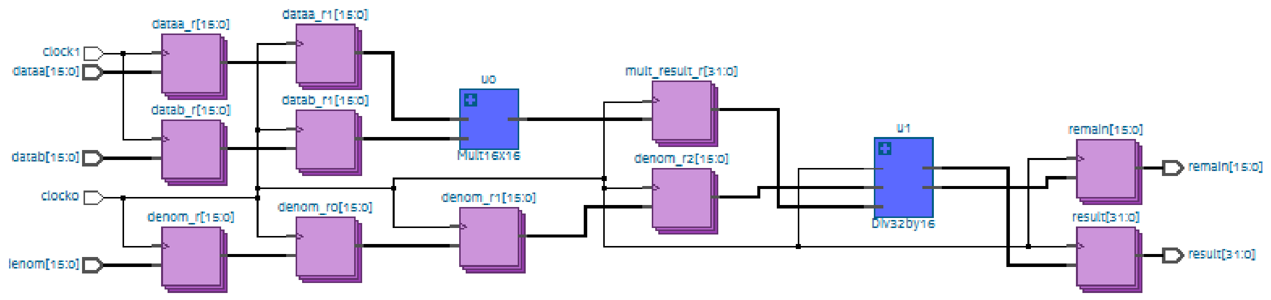
Path #1: Setup slack is -20.012 (VIOLATED)

	Property	Value	Count	Total Delay	% of Total
1	Setup Relationship	4.000			
2	Clock Skew	-0.082			
3	Data Delay	24.037			
4	Number of Logic Levels		23		
5	Physical Delays				
1	Arrival Path				
1	Clock				
1	IC		2	1.641	58
2	Cell		3	1.195	42
2	Data				
1	IC		23	9.281	39
2	Cell		147	14.344	60
3	uTco		1	0.412	2
2	Required Path				
1	Clock				

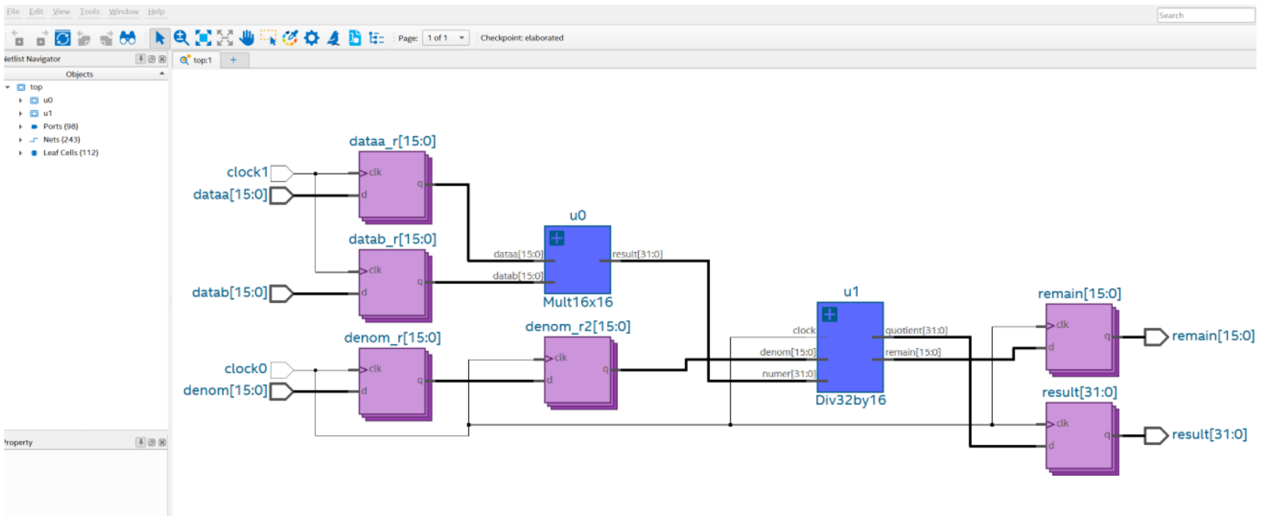
Note: Negative delays are omitted from totals when calculating percentages

Note the slack has decreased from -30.465 ns to -20.012 ns. This is still negative large slack but the slack value has decreased.

Select **Tools** → **Netlist Viewer** → **RTL Analyzer Netlist** viewer. The view looks like



(For comparison the original Netlist Viewer shows)



There are many more registers now.

The reason for adding more input registers is to have synchronization of the clocks. The registers between the multiplier and divider IPs serve as pipelining between the two IPs.

However, the timing issue is still between the divider and the output registers.

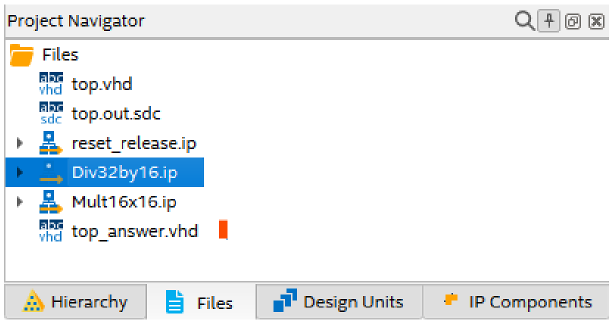
4.3.9.8 In the RTL Analyzer Window **select File** → **Close**.

4.3.10 How to resolve Timing issues -Attempt Two

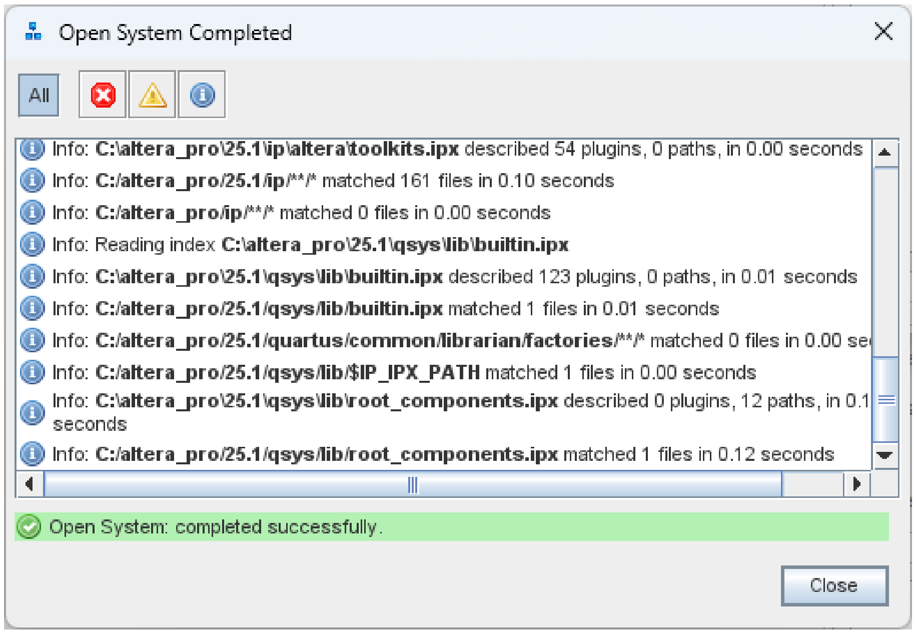
Since the slack is still large, the next step is to look for other ways to add more pipelining.

One way to add pipelining is to add latency within the Divider IP itself.

4.3.10.1 In the Project Navigator window, **double-click the Div32x16.ip**



4.3.10.2 Select **Close** for the Open System Completed as


Open System Completed

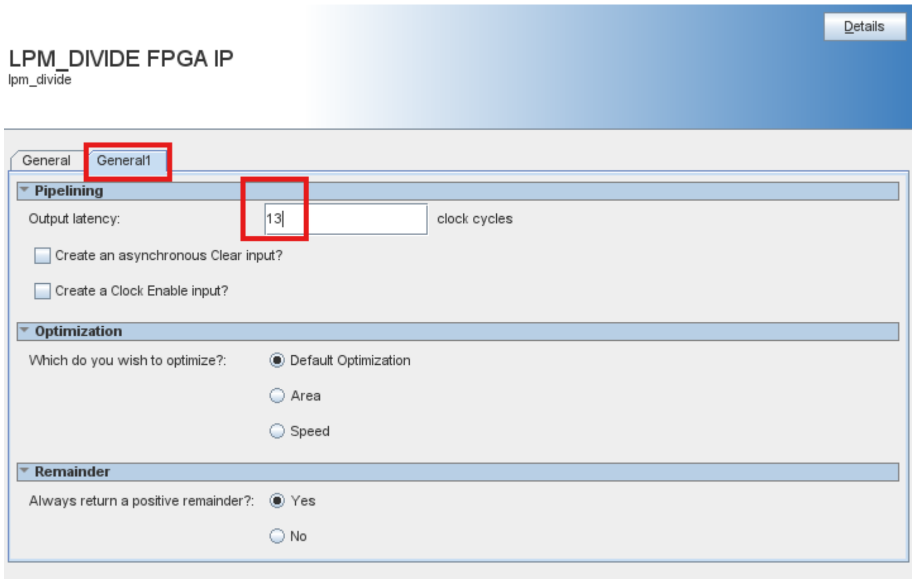
All

```
Info: C:\altera_pro\25.1\ip\altera\toolkits.ipx described 54 plugins, 0 paths, in 0.00 seconds
Info: C:/altera_pro/25.1/ip/**/* matched 161 files in 0.10 seconds
Info: C:/altera_pro/ip/**/* matched 0 files in 0.00 seconds
Info: Reading index C:\altera_pro\25.1\qsys\lib\builtin.ipx
Info: C:\altera_pro\25.1\qsys\lib\builtin.ipx described 123 plugins, 0 paths, in 0.01 seconds
Info: C:/altera_pro/25.1/qsys/lib/builtin.ipx matched 1 files in 0.01 seconds
Info: C:/altera_pro/25.1/quartus/common/librarian/factories/**/* matched 0 files in 0.00 se
Info: C:/altera_pro/25.1/qsys/lib/$IP_IPX_PATH matched 1 files in 0.00 seconds
Info: C:\altera_pro\25.1\qsys\lib\root_components.ipx described 0 plugins, 12 paths, in 0.1
seconds
Info: C:/altera_pro/25.1/qsys/lib/root_components.ipx matched 1 files in 0.12 seconds
```

Open System: completed successfully.

Close

4.3.10.3 Under the **General 1 tab** of the LPM_divide, change the output latency from 1 to 13


LPM_DIVIDE FPGA IP
lpm_divide

Details

General | General1

Pipelining

Output latency: 13 clock cycles

☐ Create an asynchronous Clear input?

☐ Create a Clock Enable input?

Optimization

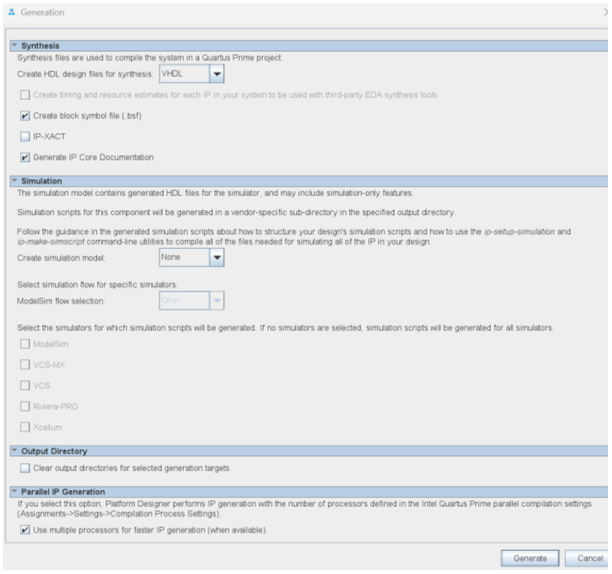
Which do you wish to optimize?: [✓ Default Optimization] Area Speed

Remainder

Always return a positive remainder?: [✓ Yes] No

This means that there will be 13 clock cycles to the output. The pipelining within the divider will now be 13.

4.3.10.4 Click on Generate HDL button. Select **Generate** (bottom right corner of the window) again.



Generation

Synthesis

Synthesis files are used to compile the system in a Quartus Prime project.

Create HDL design files for synthesis: VHDL

☐ Create timing and resource estimates for each IP in your system to be used with third-party EDA synthesis tools.

☑ Create block symbol file (.bsf)

☐ IP-XACT

☑ Generate IP Core Documentation

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model: None

Select simulation flow for specific simulators:

ModelSim flow selection: Qrun

Select the simulators for which simulation scripts will be generated. If no simulators are selected, simulation scripts will be generated for all simulators.

☐ ModelSim

☐ VCS-MX

☐ VCS

☐ Riviera-PRO

☐ Xcelium

Output Directory

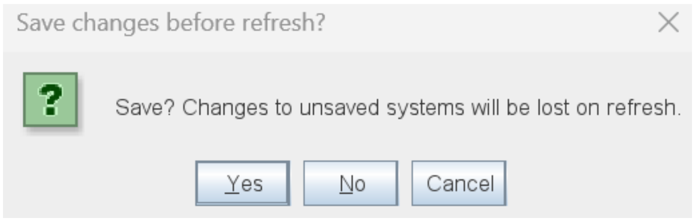
☐ Clear output directories for selected generation targets.

Parallel IP Generation

If you select this option, Platform Designer performs IP generation with the number of processors defined in the Intel Quartus Prime parallel compilation settings (Assignments->Settings->Compilation Process Settings).

☑ Use multiple processors for faster IP generation (when available).

4.3.10.5 Select **Yes** for



Save changes before refresh?

Save? Changes to unsaved systems will be lost on refresh.

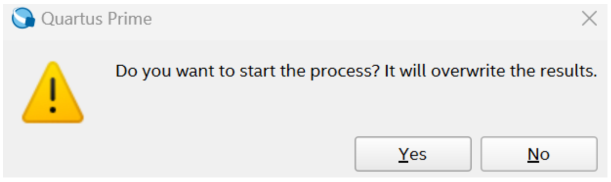
4.3.10.6 After the generation is complete, select **Close.**

4.3.10.7 Select in the IP Parameter Editor for the LPM_Divide, **File** → **Exit**.

4.3.10.8 In the Quartus main window, select **Processing** → **Start Compilation**

4.3.10.9 Select in the Timing Analyzer Window, **File ➔ Close**.

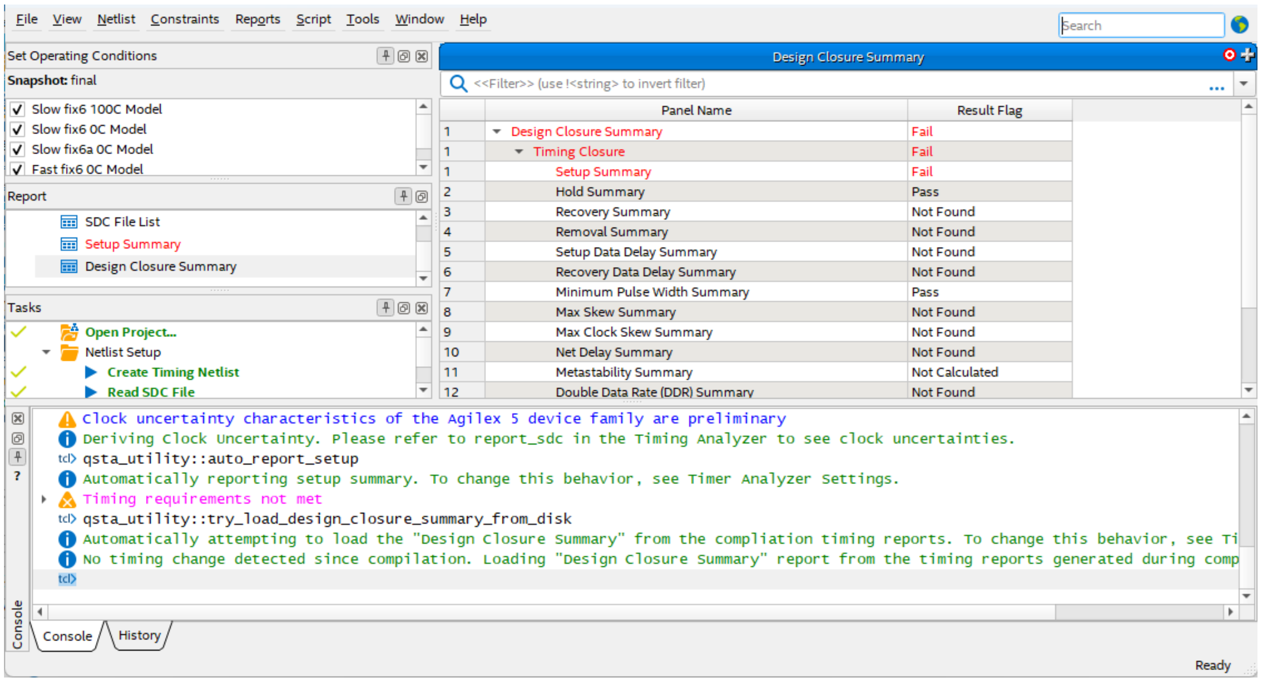
4.3.10.10 Select **Yes** to the pop window



Quartus Prime

Do you want to start the process? It will overwrite the results.

After the Timing Analyzer window opens, it is seen that the timing still does not meet.



	Panel Name	Result Flag
1	Design Closure Summary	Fail
1	Timing Closure	Fail
1	Setup Summary	Fail
2	Hold Summary	Pass
3	Recovery Summary	Not Found
4	Removal Summary	Not Found
5	Setup Data Delay Summary	Not Found
6	Recovery Data Delay Summary	Not Found
7	Minimum Pulse Width Summary	Pass
8	Max Skew Summary	Not Found
9	Max Clock Skew Summary	Not Found
10	Net Delay Summary	Not Found
11	Metastability Summary	Not Calculated
12	Double Data Rate (DDR) Summary	Not Found

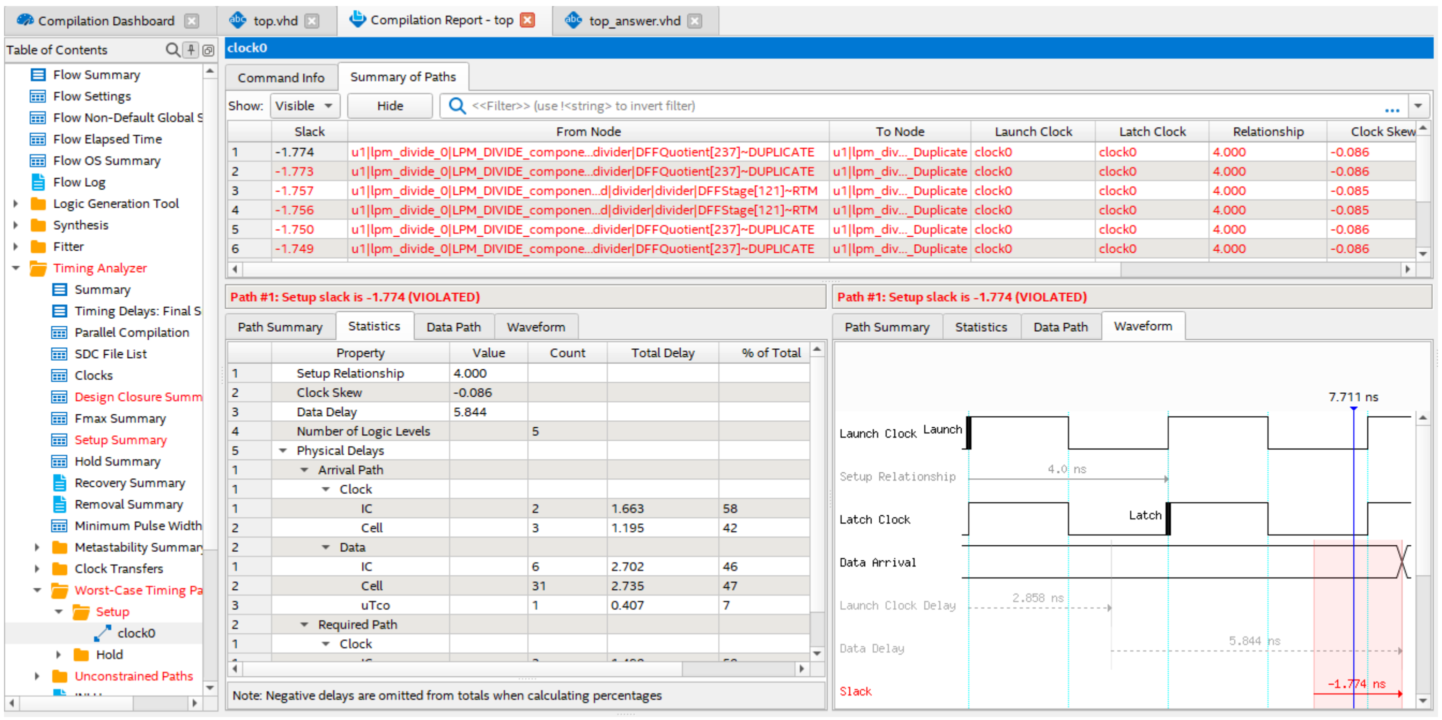
```
Clock uncertainty characteristics of the Agilex 5 device family are preliminary
Deriving Clock Uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
tcl> qsta_utility::auto_report_setup
Automatically reporting setup summary. To change this behavior, see Timer Analyzer Settings.
Timing requirements not met
tcl> qsta_utility::try_load_design_closure_summary_from_disk
Automatically attempting to load the "Design Closure Summary" from the compliation timing reports. To change this behavior, see Ti
No timing change detected since compilation. Loading "Design Closure Summary" report from the timing reports generated during comp
tcl>
```

4.3.10.11 Select in the Timing Analyzer Window, **File ➔ Close**.

4.3.10.12 Within Quartus, select **Processing** → **Compilation Report** to see the timing results.

There are other ways to see the timing results, but this is one way to do this.

4.3.10.13 Expand the Timing Analyzer and then the Worst Case Timing Paths ➔ Setup :



	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew
1	-1.774	u1|lpm_divide_0|LPM_DIVIDE_compone...divider|DFFQuotient[237]~DUPLICATE	u1|lpm_div..._Duplicate	clock0	clock0	4.000	-0.086
2	-1.773	u1|lpm_divide_0|LPM_DIVIDE_compone...divider|DFFQuotient[237]~DUPLICATE	u1|lpm_div..._Duplicate	clock0	clock0	4.000	-0.086
3	-1.757	u1|lpm_divide_0|LPM_DIVIDE_componen...d|divider|divider|DFFStage[121]~RTM	u1|lpm_div..._Duplicate	clock0	clock0	4.000	-0.085
4	-1.756	u1|lpm_divide_0|LPM_DIVIDE_componen...d|divider|divider|DFFStage[121]~RTM	u1|lpm_div..._Duplicate	clock0	clock0	4.000	-0.085
5	-1.750	u1|lpm_divide_0|LPM_DIVIDE_compone...divider|DFFQuotient[237]~DUPLICATE	u1|lpm_div..._Duplicate	clock0	clock0	4.000	-0.086
6	-1.749	u1|lpm_divide_0|LPM_DIVIDE_compone...divider|DFFQuotient[237]~DUPLICATE	u1|lpm_div..._Duplicate	clock0	clock0	4.000	-0.086

Path #1: Setup slack is -1.774 (VIOLATED)

	Property	Value	Count	Total Delay	% of Total
1	Setup Relationship	4.000			
2	Clock Skew	-0.086			
3	Data Delay	5.844			
4	Number of Logic Levels		5		
5	Physical Delays				
1	Arrival Path				
1	Clock				
1	IC		2	1.663	58
2	Cell		3	1.195	42
2	Data				
1	IC		6	2.702	46
2	Cell		31	2.735	47
3	uTco		1	0.407	7
2	Required Path				
1	Clock				

Note: Negative delays are omitted from totals when calculating percentages

Path #1: Setup slack is -1.774 (VIOLATED)

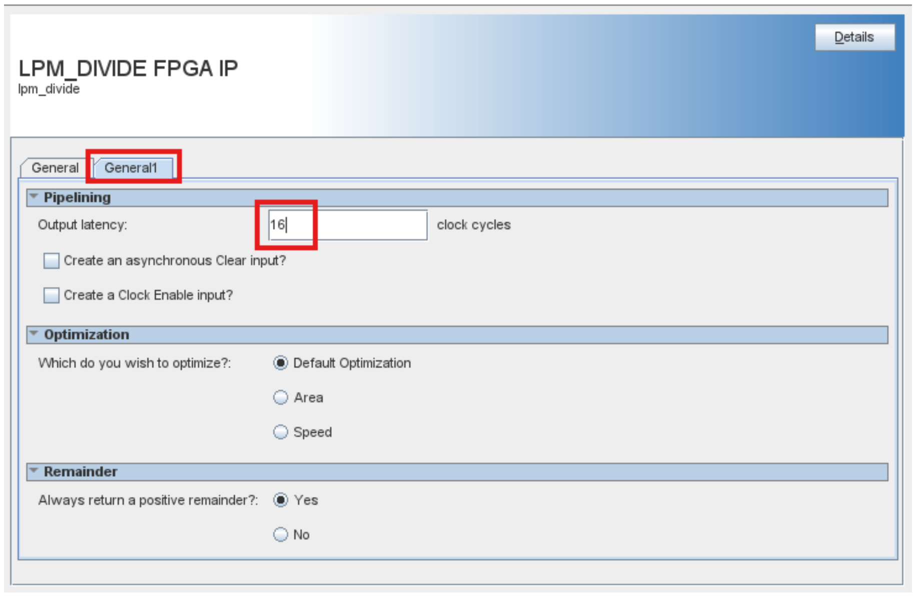
Note the timing slack has now decreased to -1.774 ns. The initial timing violations were –30.465 ns, then with added pipelining in the code, it decreased to -20.012, and now in this case, the timing violations decreased to -1.774 ns. This is now getting much closer to meeting the slack requirement.

4.3.10.14 How to resolve Timing Issues-Attempt Three

Since the output latency in the Divider helped considerably to almost resolving the issue, the next step is try adding more latency in the Divider.

4.3.10.15 Within the Project Navigator window, **double-click on the Div32x16.ip**. Select **Close** in the Open System Completed dialog box.

4.3.10.16 In the LPM_Divide Intel FPGA IP under the **General 1 tab**, change the 13 clock cycles to be **16 clock cycles** as



LPM_DIVIDE FPGA IP
lpm_divide

General | General1

Pipelining

Output latency: 16 clock cycles

☐ Create an asynchronous Clear input?

☐ Create a Clock Enable input?

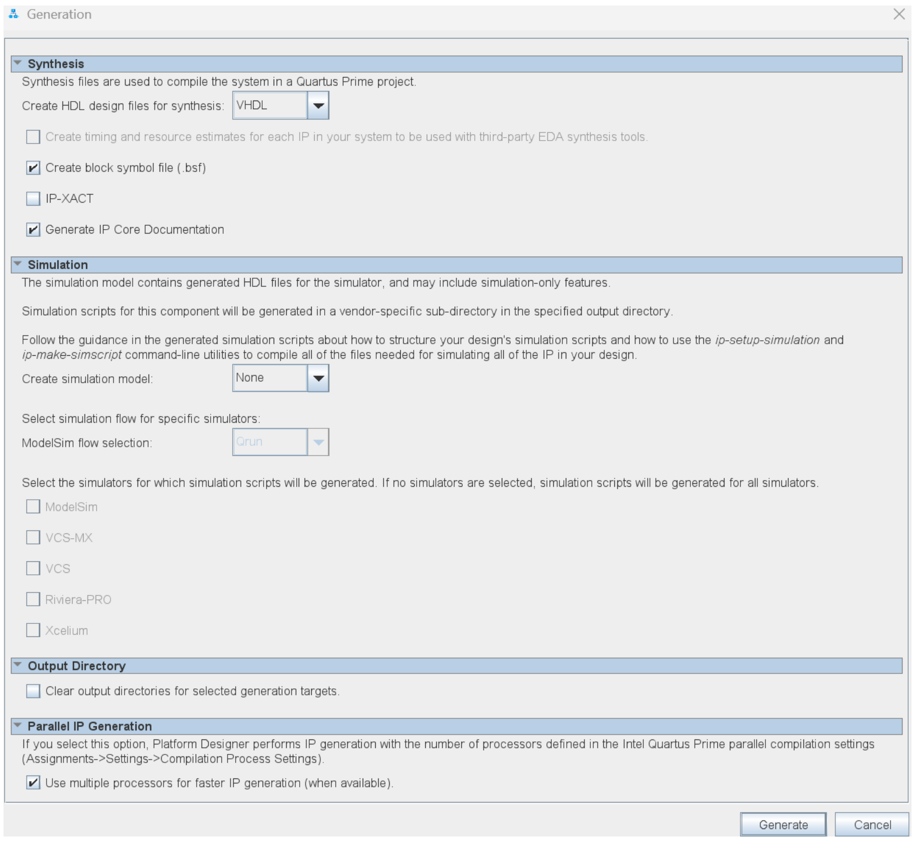
Optimization

Which do you wish to optimize?: [✓ Default Optimization] Area Speed

Remainder

Always return a positive remainder?: [✓ Yes] No

4.3.10.17 Select **Generate HDL** (bottom right corner of the window) and select **Generate** (bottom right corner) on the **Generation** dialog box.



Synthesis

Synthesis files are used to compile the system in a Quartus Prime project.

Create HDL design files for synthesis: VHDL

☐ Create timing and resource estimates for each IP in your system to be used with third-party EDA synthesis tools.

☑ Create block symbol file (.bsf)

☐ IP-XACT

☑ Generate IP Core Documentation

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model: None

Select simulation flow for specific simulators:

ModelSim flow selection: Qrun

Select the simulators for which simulation scripts will be generated. If no simulators are selected, simulation scripts will be generated for all simulators.

☐ ModelSim

☐ VCS-MX

☐ VCS

☐ Riviera-PRO

☐ Xcelium

Output Directory

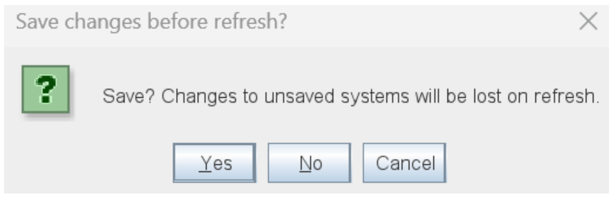
☐ Clear output directories for selected generation targets.

Parallel IP Generation

If you select this option, Platform Designer performs IP generation with the number of processors defined in the Intel Quartus Prime parallel compilation settings (Assignments->Settings->Compilation Process Settings).

☑ Use multiple processors for faster IP generation (when available).

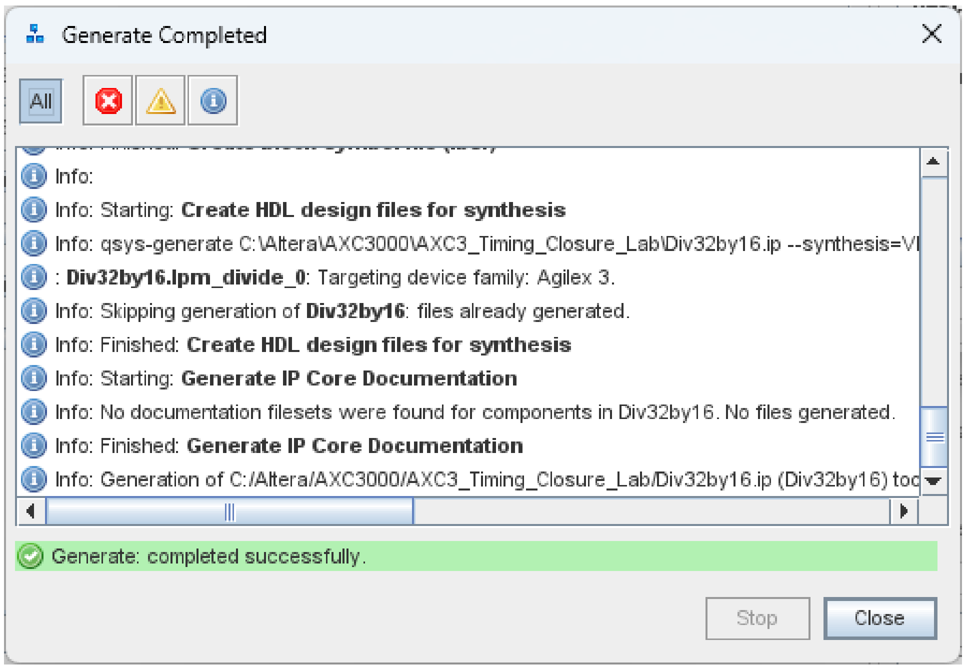
4.3.10.18 Select **Yes** when prompted



Save changes before refresh?

Save? Changes to unsaved systems will be lost on refresh.

4.3.10.19 After the generation is complete, select **Close**



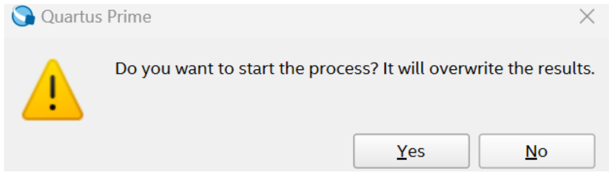
Generate Completed

```
Info:
Info: Starting: Create HDL design files for synthesis
Info: qsys-generate C:\Altera\AXC3000\AXC3_Timing_Closure_Lab\Div32by16.ip --synthesis=V
: Div32by16.lpm_divide_0: Targeting device family: Agilex 3.
Info: Skipping generation of Div32by16: files already generated.
Info: Finished: Create HDL design files for synthesis
Info: Starting: Generate IP Core Documentation
Info: No documentation filesets were found for components in Div32by16. No files generated.
Info: Finished: Generate IP Core Documentation
Info: Generation of C:/Altera/AXC3000/AXC3_Timing_Closure_Lab/Div32by16.ip (Div32by16) too
Generate: completed successfully.
```

4.3.10.20 Within the IP Parameter Window, select **File** → **Exit.**

4.3.10.21 From the Quartus Main menu, select **Processing** → **Start Compilation**.

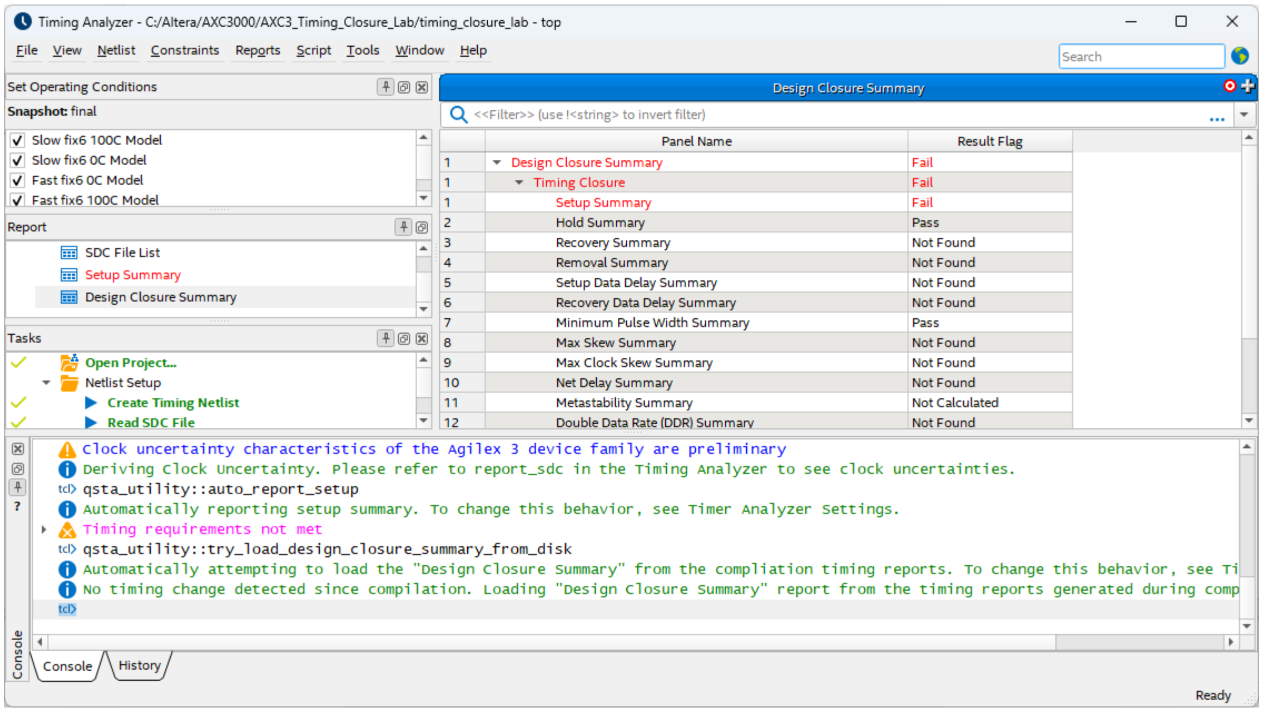
4.3.10.22 Select **Yes.**



Quartus Prime

Do you want to start the process? It will overwrite the results.

Note when the Timing Analyzer window opens, now there are no longer failing paths as it shows pass for the setup values.



Timing Analyzer - C:/Altera/AXC3000/AXC3_Timing_Closure_Lab/timing_closure_lab - top

Set Operating Conditions

Snapshot: final

- ✓ Slow fix6 100C Model
- ✓ Slow fix6 0C Model
- ✓ Fast fix6 0C Model
- ✓ Fast fix6 100C Model

Report

- SDC File List
- Setup Summary
- Design Closure Summary

Tasks

- ✓ Open Project...
- Netlist Setup
  - ✓ Create Timing Netlist
  - ✓ Read SDC File

Design Closure Summary

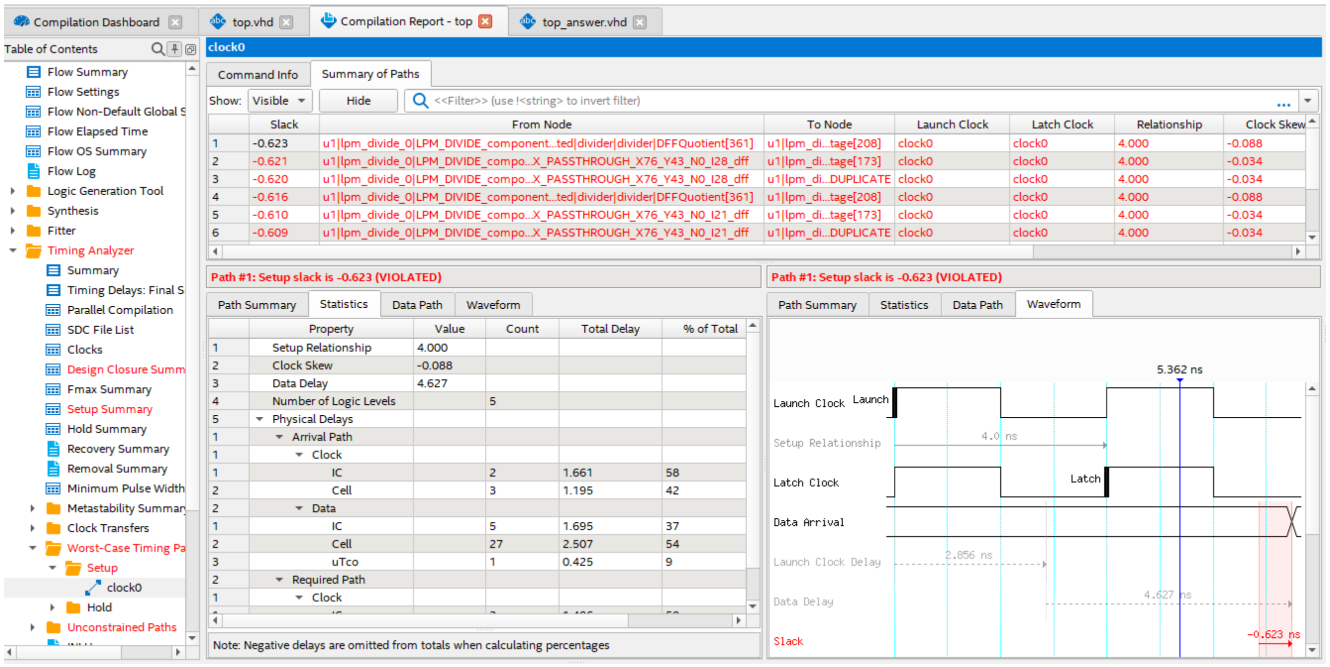
	Panel Name	Result Flag
1	Design Closure Summary	Fail
1	Timing Closure	Fail
1	Setup Summary	Fail
2	Hold Summary	Pass
3	Recovery Summary	Not Found
4	Removal Summary	Not Found
5	Setup Data Delay Summary	Not Found
6	Recovery Data Delay Summary	Not Found
7	Minimum Pulse Width Summary	Pass
8	Max Skew Summary	Not Found
9	Max Clock Skew Summary	Not Found
10	Net Delay Summary	Not Found
11	Metastability Summary	Not Calculated
12	Double Data Rate (DDR) Summary	Not Found

```
Clock uncertainty characteristics of the Agilex 3 device family are preliminary
Deriving Clock Uncertainty. Please refer to report_sdc in the Timing Analyzer to see clock uncertainties.
tcl> qsta_utility::auto_report_setup
Automatically reporting setup summary. To change this behavior, see Timer Analyzer Settings.
Timing requirements not met
tcl> qsta_utility::try_load_design_closure_summary_from_disk
Automatically attempting to load the "Design Closure Summary" from the compliation timing reports. To change this behavior, see Ti
No timing change detected since compilation. Loading "Design Closure Summary" report from the timing reports generated during comp
tcl>
```

After the Timing Analyzer window opens, it is seen that the timing still does not meet.

4.3.10.23 Close the Timing Analyzer window by selecting **File** → **Close**.

4.3.10.24 Within Quartus, select **Processing** → **Compilation Report** and expand the Timing Analyzer as



	Slack	From Node	To Node	Launch Clock	Latch Clock	Relationship	Clock Skew
1	-0.623	u1|lpm_divide_0|LPM_DIVIDE_component...ted|divider|divider|DFFQuotient[361]	u1|lpm_di...tage[208]	clock0	clock0	4.000	-0.088
2	-0.621	u1|lpm_divide_0|LPM_DIVIDE_compo...X_PASSTHROUGH_X76_Y43_N0_I28_dff	u1|lpm_di...tage[173]	clock0	clock0	4.000	-0.034
3	-0.620	u1|lpm_divide_0|LPM_DIVIDE_compo...X_PASSTHROUGH_X76_Y43_N0_I28_dff	u1|lpm_di...DUPLICATE	clock0	clock0	4.000	-0.034
4	-0.616	u1|lpm_divide_0|LPM_DIVIDE_component...ted|divider|divider|DFFQuotient[361]	u1|lpm_di...tage[208]	clock0	clock0	4.000	-0.088
5	-0.610	u1|lpm_divide_0|LPM_DIVIDE_compo...X_PASSTHROUGH_X76_Y43_N0_I21_dff	u1|lpm_di...tage[173]	clock0	clock0	4.000	-0.034
6	-0.609	u1|lpm_divide_0|LPM_DIVIDE_compo...X_PASSTHROUGH_X76_Y43_N0_I21_dff	u1|lpm_di...DUPLICATE	clock0	clock0	4.000	-0.034

Path #1: Setup slack is -0.623 (VIOLATED)

	Property	Value	Count	Total Delay	% of Total
1	Setup Relationship	4.000			
2	Clock Skew	-0.088			
3	Data Delay	4.627			
4	Number of Logic Levels		5		
5	Physical Delays				
1	Arrival Path				
1	Clock				
1	IC		2	1.661	58
2	Cell		3	1.195	42
2	Data				
1	IC		5	1.695	37
2	Cell		27	2.507	54
3	uTco		1	0.425	9
2	Required Path				
1	Clock				

Note: Negative delays are omitted from totals when calculating percentages

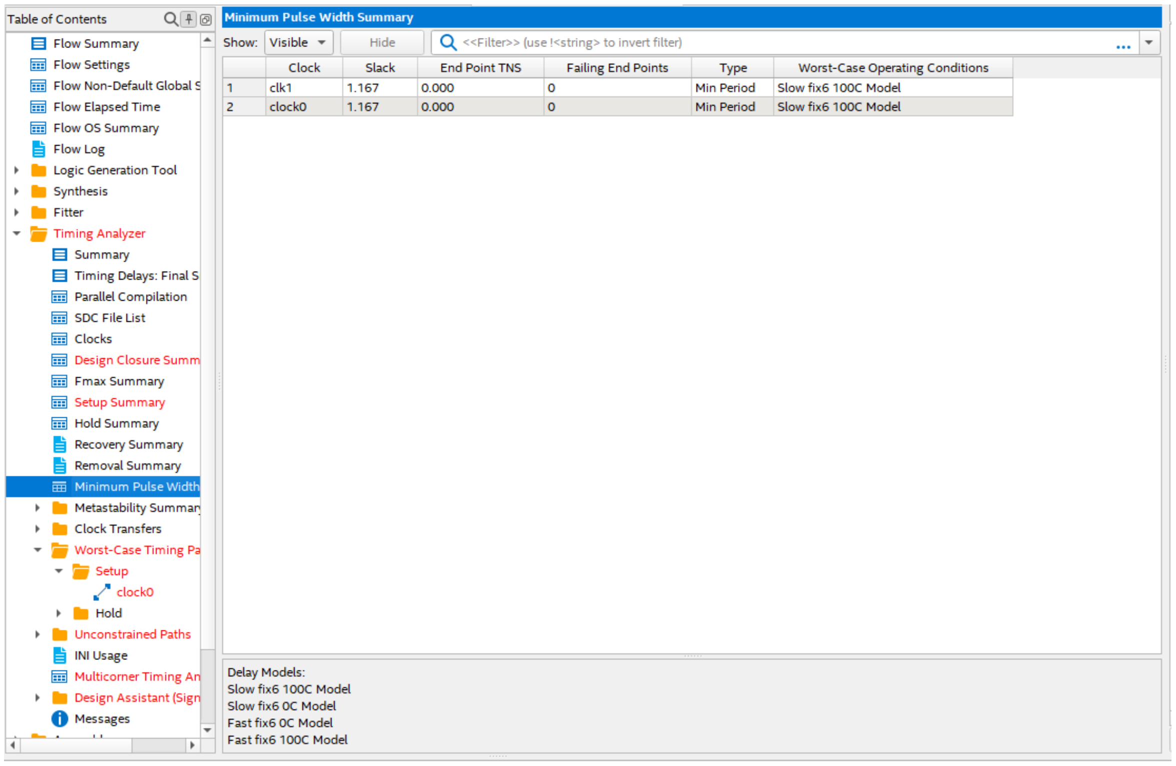
Path #1: Setup slack is -0.623 (VIOLATED)

Note the timing slack has now decreased to -0.623 ns. The initial timing violations were -30.465 ns, then with added pipelining in the code, it decreased to -20.012 ns, and now in this case, the timing violations decreased to -0.623 ns. This is now getting much closer to meeting the slack requirement. However, experimentation with deeper pipelines (up to 32 levels) indicate that it will still not meet timing. It appears that this speed grade device cannot close timing at 250MHz.

A solution would be to reduce the clock0 and clock1 timing requirements in the SDC file from a period of 4ns to perhaps 5ns. This would reduce the fmax from 250 MHz to 200 MHz. Another solution might be to use a faster speed grade device.

There are unconstrained paths because the input and output ports were not assigned.

The next item is to expand the individual settings in the Timing Analyzer, and look at the various items such as minimum pulse width, unconstrained paths, hold summary and more



Minimum Pulse Width Summary

	Clock	Slack	End Point TNS	Failing End Points	Type	Worst-Case Operating Conditions
1	clk1	1.167	0.000	0	Min Period	Slow fix6 100C Model
2	clock0	1.167	0.000	0	Min Period	Slow fix6 100C Model

Delay Models:
Slow fix6 100C Model
Slow fix6 0C Model
Fast fix6 0C Model
Fast fix6 100C Model

Under fmax summary, it is seen that the clock0 runs almost 216.31 MHz. Now for the clk1 setting, there is a section in the next, optional timing section that can be looked at.

Optional timing experiments that could be done

The following are optional items that could be looked at by using the same flow as before.

There are other items for timing. You can try these items or other items.

1) As mentioned previously, use a faster speed grade device or reduce the timing requirements

2) Try using the Netlist Analyzer i.e. the Technology Map Viewer to see more details for clk1.

CONGRATULATIONS! YOU HAVE SUCCESSFULLY COMPLETED THE TIMING CLOSURE LAB!

5 Legal Disclaimer

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.