Tutorial
ModelSim & Quartus first time enabling

1 Objective

- To enable ModelSim and Quartus software.
- To understand the basics steps for compiling, simulating and synthesizing a SystemVerilog design using ModelSim and Quartus.

2 Teamwork policy

This is an individual assignment.

3 Background

ModelSim and Quartus software are used for simulating, validating and synthesizing digital hardware designs using Hardware Description Languages (HDLs). In this tutorial, you will refresh your knowledge on the basic use of these two tools.

4 Pre-requisites

It is assumed that ModelSim and Quartus are installed in your computer. If this is not the case, the attached document `ModelSim_Quartus_install.pdf` provides a quick guide on how to install these tools. After you complete the installation of the tools, come back to this document and proceed with Section 5.

DISCLAIMER. The versions of ModelSim and Quartus stated in the tutorial instructions `ModelSim_Quartus_install.pdf` are for reference only. You might find more recent versions of the software. However, the steps provided in this tutorial should still be relevant. If you find a mistake in this tutorial due to a version update, please do let me know.

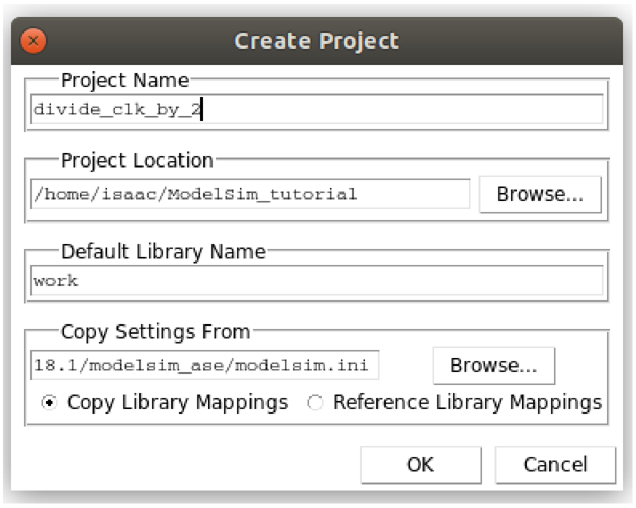
5 ModelSim project

This section reinforces the steps necessary in order to create a ModelSim project, compile and simulate a design. If you are confident enough with your ModelSim knowledge, you may skip Section 5.1 and go straight to the exercise in Section 5.2.

5.1 Tutorial Instructions

For this part of the tutorial, please use the source files `divide_clk_by_2.sv` and `tb_divide_clk_by_2.sv`

1. Open ModelSim.
2. From the top menu select **File** → **New** → **Project**
3. Provide a name and a location for your project and click **OK**. You can select any project name you want. However, as shown in Figure 1, it is a good practice to match the name of your project with the name of your top-level SystemVerilog module.



Create Project

Project Name: divide_clk_by_2

Project Location: /home/isaac/ModelSim_tutorial

Default Library Name: work

Copy Settings From: 18.1/modelsim_ase/modelsim.ini

[✓ Copy Library Mappings] Reference Library Mappings

Figure 1: Project name and location.

4. You should be prompted with the window shown in Figure 2. For this tutorial, you are provided with some SystemVerilog files, for this reason, select **Add Existing File**
5. Browse to the location of the provided files for this tutorial, select `divide_clk_by_2.sv` and `tb_divide_clk_by_2.sv` files and click **Open**. As shown in Figure 3, I've placed both files under the same location, which corresponds to the project's location. However, this is not necessary and we can refer to files located elsewhere in our file system.

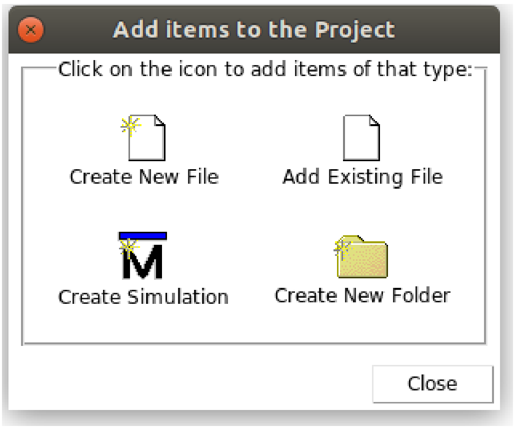
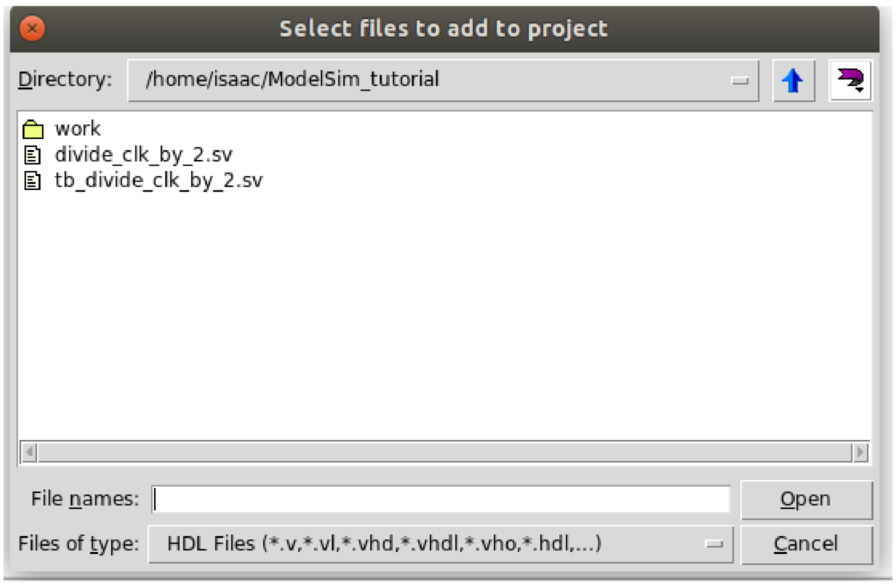


Figure 2: Add files prompt.



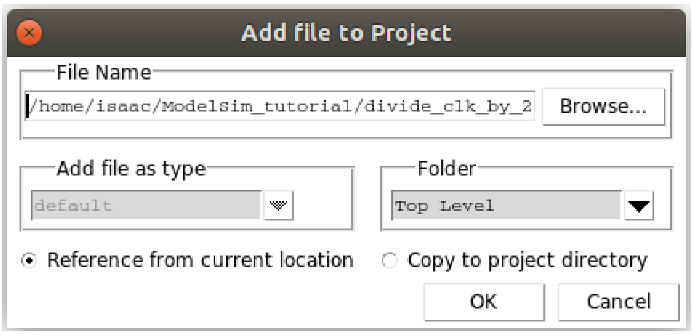
Directory: /home/isaac/ModelSim_tutorial

work
divide_clk_by_2.sv
tb_divide_clk_by_2.sv

File names:
Files of type: HDL Files (*.v,*.vl,*.vhd,*.vhdl,*.vho,*.hdl,...)

Figure 3: Files to be added.

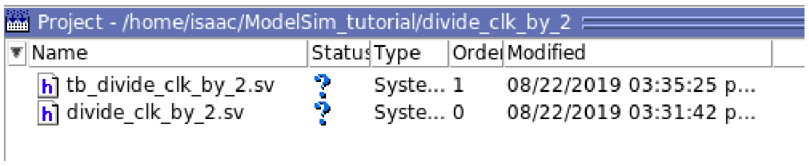
6. As shown in Figure 4, make sure that **Reference from current location** is selected and select **OK**. After this, you should see something similar to Figure 5.



File Name: /home/isaac/ModelSim_tutorial/divide_clk_by_2
Add file as type: default
Folder: Top Level
[✓ Reference from current location] Copy to project directory

Figure 4: Adding project files.

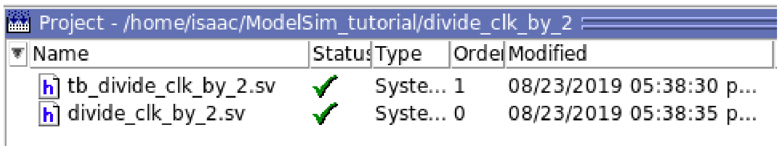
Note that there's a question mark (**?**) next to the name of the files. This indicates that the files have not been compiled since its creation or last modification.



```
Project - /home/isaac/ModelSim_tutorial/divide_clk_by_2
Name                  Status Type     Order Modified
tb_divide_clk_by_2.sv   ?    Syste... 1     08/22/2019 03:35:25 p...
divide_clk_by_2.sv      ?    Syste... 0     08/22/2019 03:31:42 p...
```

Figure 5: Files added to project.

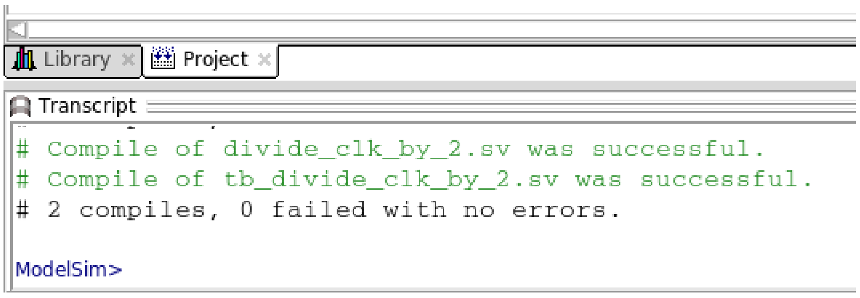
7. In order to compile, select **Compile** → **Compile all** from the top menu. As shown in Figure 6, a green check mark next to the name of the file indicates a successful compilation.



```
Project - /home/isaac/ModelSim_tutorial/divide_clk_by_2
Name                  Status Type     Order Modified
tb_divide_clk_by_2.sv   ✓    Syste... 1     08/23/2019 05:38:30 p...
divide_clk_by_2.sv      ✓    Syste... 0     08/23/2019 05:38:35 p...
```

Figure 6: Files compiled successfully.

ModelSim also provides a command and message window, which is embedded in the bottom part of the screen. Here, ModelSim also indicates if a compilation has been successful as shown in Figure 7.

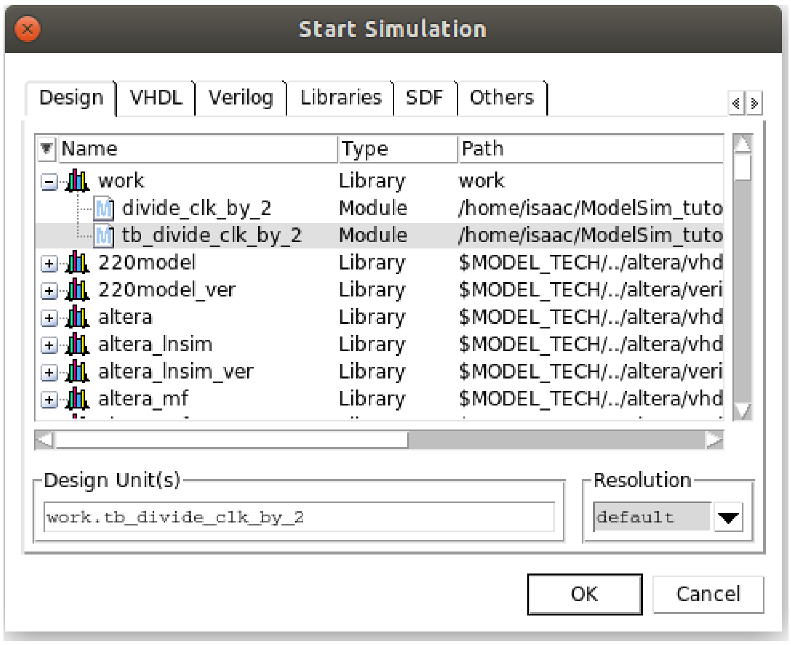


```
# Compile of divide_clk_by_2.sv was successful.
# Compile of tb_divide_clk_by_2.sv was successful.
# 2 compiles, 0 failed with no errors.

ModelSim>
```

Figure 7: Command prompt showing successful compilation.

8. The next step is to simulate our testbench. On the top menu, select **Simulate** → **Start Simulation**. You should be prompted with a window as shown in Figure 8.
   Click on the + sign next to **work**, select `tb_divide_clk_by_2.sv` and click **OK**. You should see something similar to Figure 9.

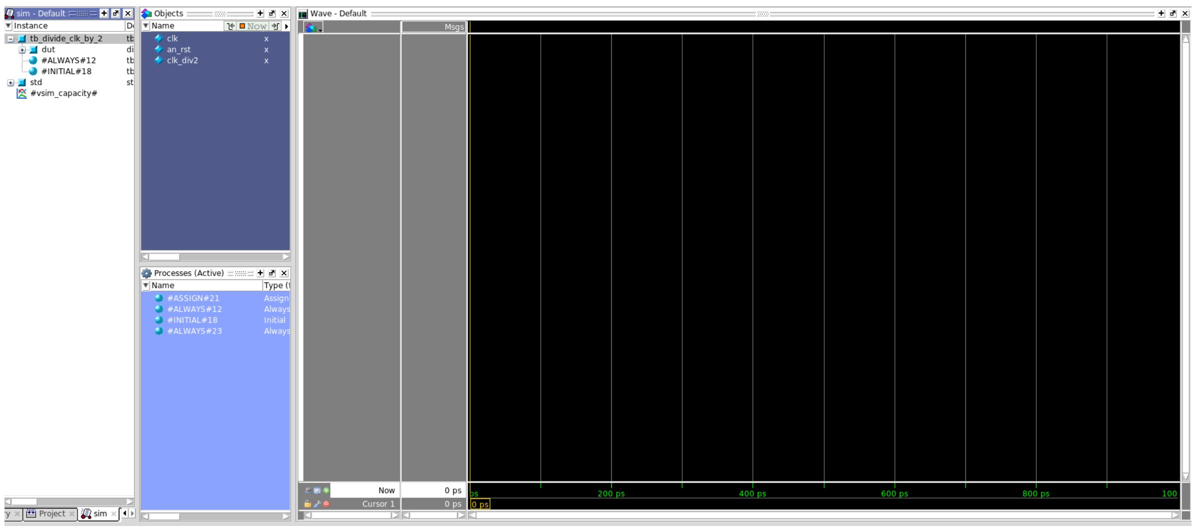


Name	Type	Path
work	Library	work
divide_clk_by_2	Module	/home/isaac/ModelSim_tuto
tb_divide_clk_by_2	Module	/home/isaac/ModelSim_tuto
220model	Library	$MODEL_TECH/../altera/vhd
220model_ver	Library	$MODEL_TECH/../altera/veri
altera	Library	$MODEL_TECH/../altera/vhd
altera_lnsim	Library	$MODEL_TECH/../altera/vhd
altera_lnsim_ver	Library	$MODEL_TECH/../altera/veri
altera_mf	Library	$MODEL_TECH/../altera/vhd

Design Unit(s): work.tb_divide_clk_by_2

Resolution: default

Figure 8: Start simulation.



Instance
- tb_divide_clk_by_2
  - dut
  - #ALWAYS#12
  - #INITIAL#18
- std
- #vsim_capacity#

Name	Now
clk	x
an_rst	x
clk_div2	x

Name	Type
#ASSIGN#21	Assign
#ALWAYS#12	Always
#INITIAL#18	Initial
#ALWAYS#23	Always

Figure 9: Simulation environment.

9. In the **Objects** panel select all signals, right click and select **Add Wave** as shown in Figure 10.

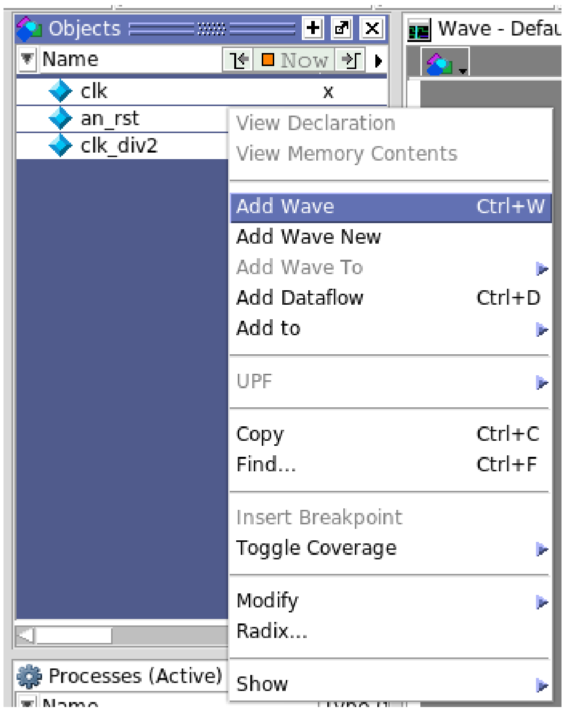


Figure 10: Adding waveforms.

Our testbench signals are now added to the **Wave** panel as shown in Figure 11.

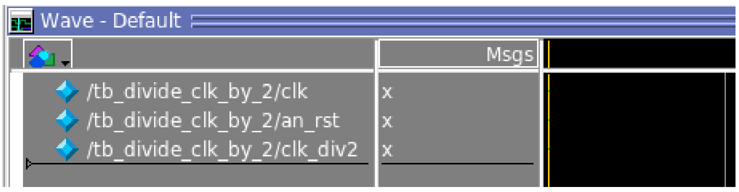
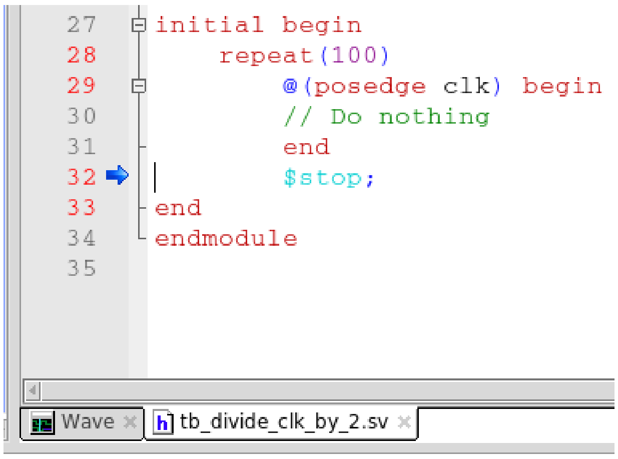

Wave - Default
Msgs
/tb_divide_clk_by_2/clk x
/tb_divide_clk_by_2/an_rst x
/tb_divide_clk_by_2/clk_div2 x

Figure 11: Signals added to waveform panel.

10. So far, we've compiled our source files, loaded our simulation environment and our testbench. However, we haven't actually started our simulation. In order to start our simulation, on the top menu select **Simulation** → **Run** → **Run -All**. ModelSim will perform the simulation and it should open the testbench file pointing to a `$stop` construct as shown in Figure 12. This is a control statement used in testbenches in order to prevent the simulator to continue with its simulation indefinitely. Go back to the waveform panel by clicking on **Wave** next to the name of the testbench, as shown in the lower left corner of Figure 12.



```
initial begin
    repeat(100)
        @(posedge clk) begin
        // Do nothing
        end
        $stop;
end
endmodule
```

Figure 12: `$stop` construct in testbenches.

11. In order to properly view our waveforms, we must zoom out. We can do this by either clicking on the icon to the right to the magnifier with the - sign, or by simply clicking anywhere in the wave panel and pressing **F**, as shown in Figure 13.

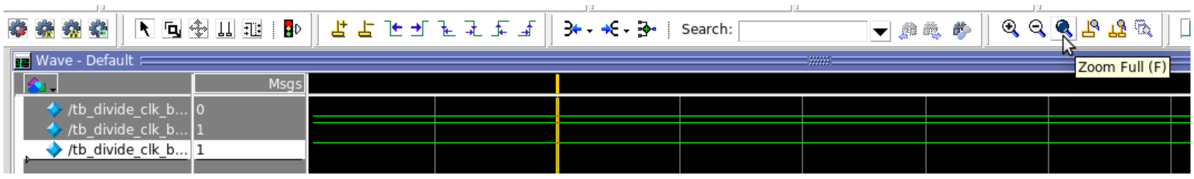


Figure 13: Zooming out in wave panel.

You should be able to see waveforms for the entire simulation time as shown in Figure 14.

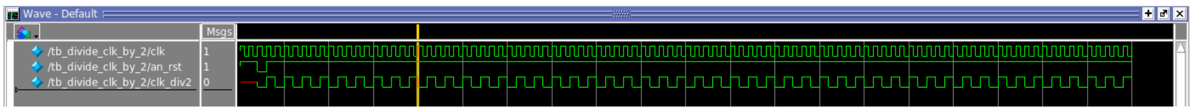

Wave - Default
Msgs
/tb_divide_clk_by_2/clk 1
/tb_divide_clk_by_2/an_rst 1
/tb_divide_clk_by_2/clk_div2 0

Figure 14: Full waveform view.

Use the magnifiers in order to zoom in and zoom out as you please. It is also possible to zoom in using the middle click of the mouse. Moreover, there are defined shortcut keys for zooming in and out. Right click on the **wave** region and you can find out the different zoom shortcuts.

12. Hover over the icons shown in Figure 15 in order to see different simulation run options. Click on these buttons and identify what each of them does.

    HINT: You might have to **manually** stop the simulation by clicking on the red stop sign from Figure 15.

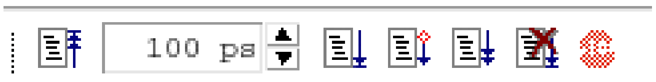


Figure 15: Quick simulation buttons.

13. Identify which button restarts the simulation. Click on it and when prompted with a confirmation dialogue box, click **OK**. How could you run the simulation again? What happens when you change the default simulation step and click on the **Run** button? In my case, the default simulation value is set to `100 ps`. However, this value might be different in your environment. Play around with these buttons, as well as with the simulation step and make sure you understand what each of them does.

14. Close ModelSim and open it again, it will automatically load your last project. Try to run your last simulation starting from step 8 from above in order to reinforce what you have learned.

15. Once you are familiar with how to launch and restart a simulation, finish your simulation by selecting **Simulate** → **End Simulation** → **Yes**.

16. You have finished the ModelSim part of the tutorial. You may now proceed to the ModelSim exercise in Section 5.2.

5.2 ModelSim exercise

1. Create a new ModelSim project using the file `counter.sv` and its testbench `tb_counter.sv`.

2. Once you create the project, compile the files. **There should be syntax errors!**

3. Debug the errors by double-clicking on the error messages that will appear in red font in the command prompt inside ModelSim. A window should be opened with information about which line of code contains the error and a hint on how to fix it.

4. Fix all the errors until all error messages are gone.

5. Continue with the steps necessary in order to simulate the design.

6. Zoom in to a relevant time window in order to properly observe the waveforms.

7. Analyse the code in both `counter.sv` and its testbench `tb_counter.sv` and answer the questions embedded in the code comments.

8. The provided code models a 3-bit counter. Modify your file(s) in order to simulate an 8-bit counter.

9. You may now proceed to Section 6.

6 Quartus project

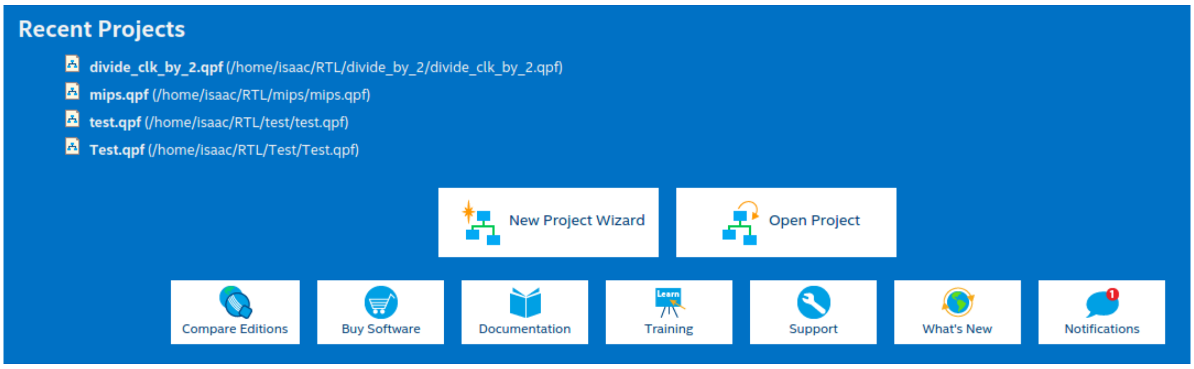
This section reinforces the steps necessary in order to create a Quartus project and synthesize a design If you are confident enough, you may skip Section 6.1 and go straight to the exercise in Section 6.2.

6.1 Tutorial Instructions

For this part of the tutorial, please use the source file `divide_clk_by_2.sv`.

1. Open Quartus.

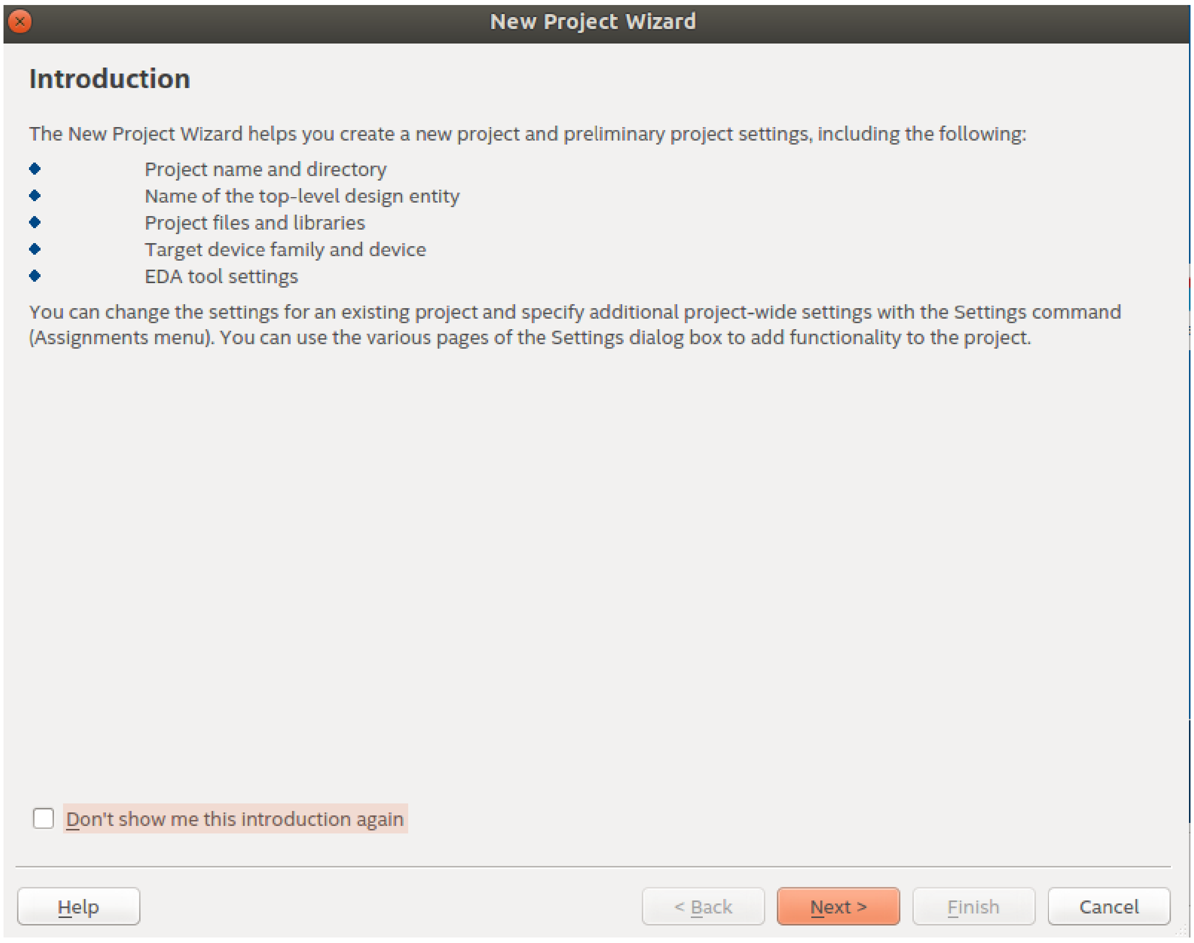
2. Select **New Project Wizard** from the menu, as shown in Figure 16.


Recent Projects

divide_clk_by_2.qpf (/home/isaac/RTL/divide_by_2/divide_clk_by_2.qpf)
mips.qpf (/home/isaac/RTL/mips/mips.qpf)
test.qpf (/home/isaac/RTL/test/test.qpf)
Test.qpf (/home/isaac/RTL/Test/Test.qpf)

Figure 16: Project wizard.

3. When prompted with the **Introduction** window, click **Next** as shown in Figure 17.



Introduction

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

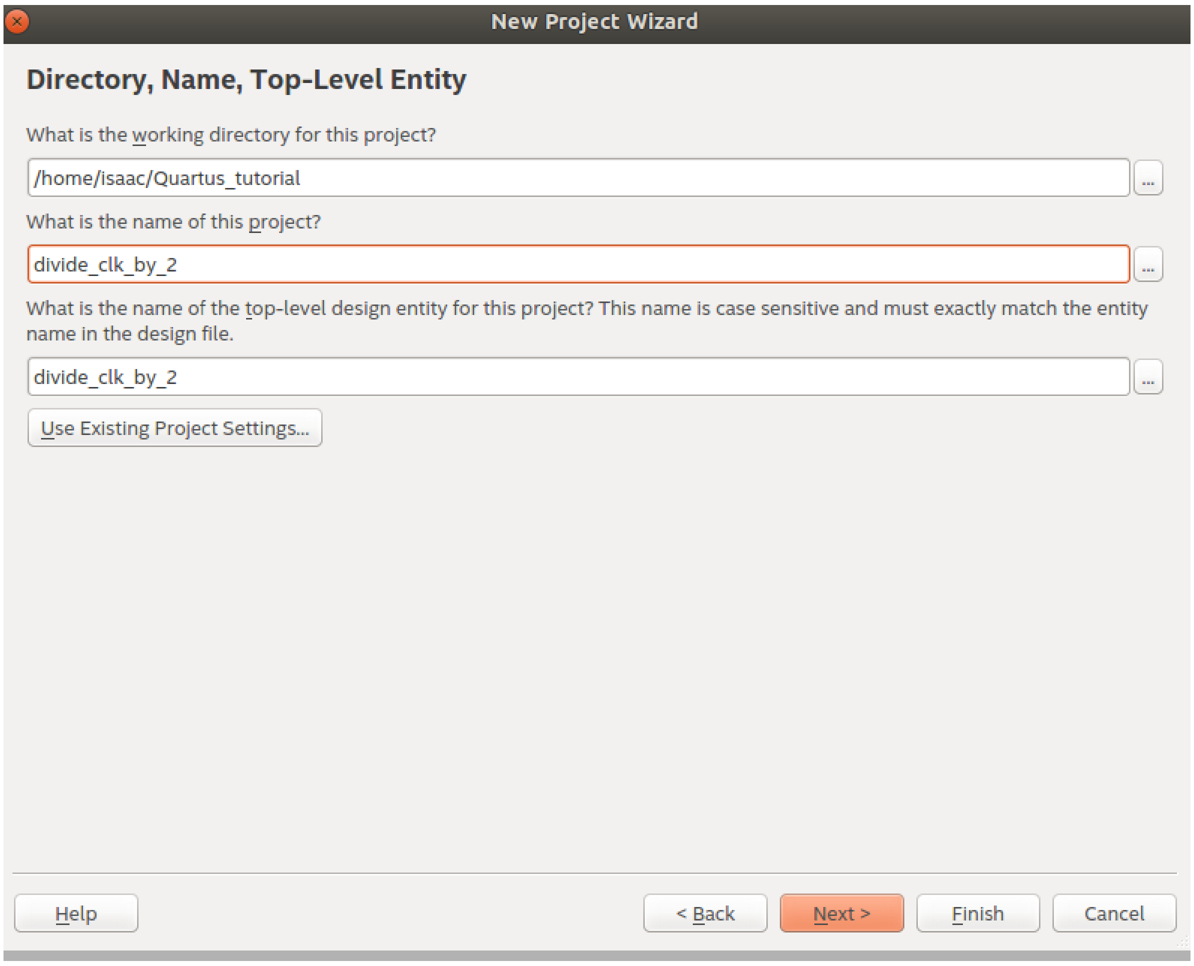
You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.

☐ Don't show me this introduction again

Figure 17: Project intro.

4. Provide a name for the project and click **Next**.

   Important: Quartus requires that the name given to the project is the same as the top-level SystemVerilog module. As a result of this, name the project `divide_clk_by_2.sv`, as shown in Figure 18.



New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

/home/isaac/Quartus_tutorial

What is the name of this project?

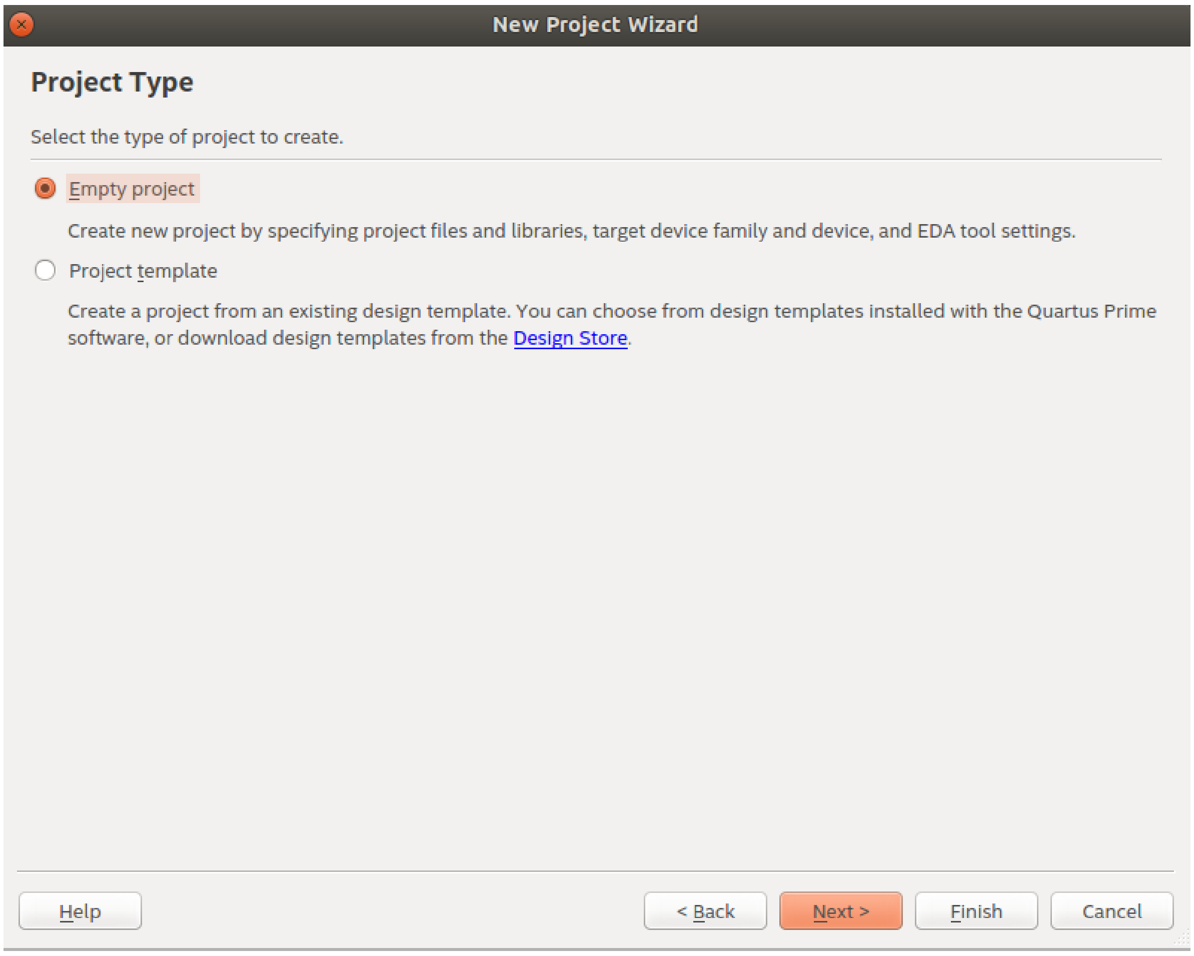
divide_clk_by_2

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.

divide_clk_by_2

Figure 18: Project name.

5. Select **Empty project** as shown in Figure 19 and click **Next**.


New Project Wizard

Project Type

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.

Project template

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

Figure 19: Project type.

6. When prompted to add files, browse to the location of the downloaded files and select `divide_clk_by_2.sv`, as shown in Figure 20. Notice that testbenches are not synthesizable, as a result of this, we can simply omit our testbench.

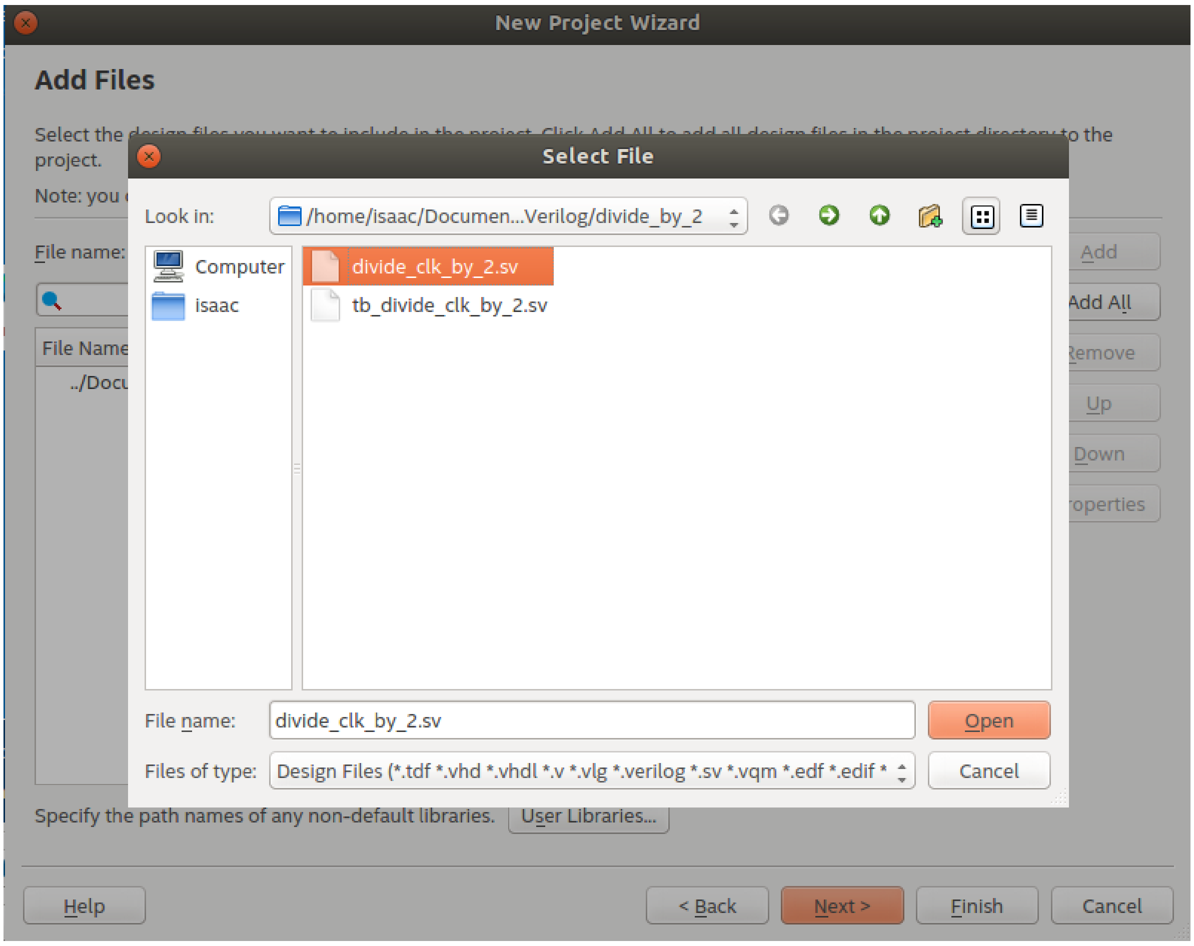
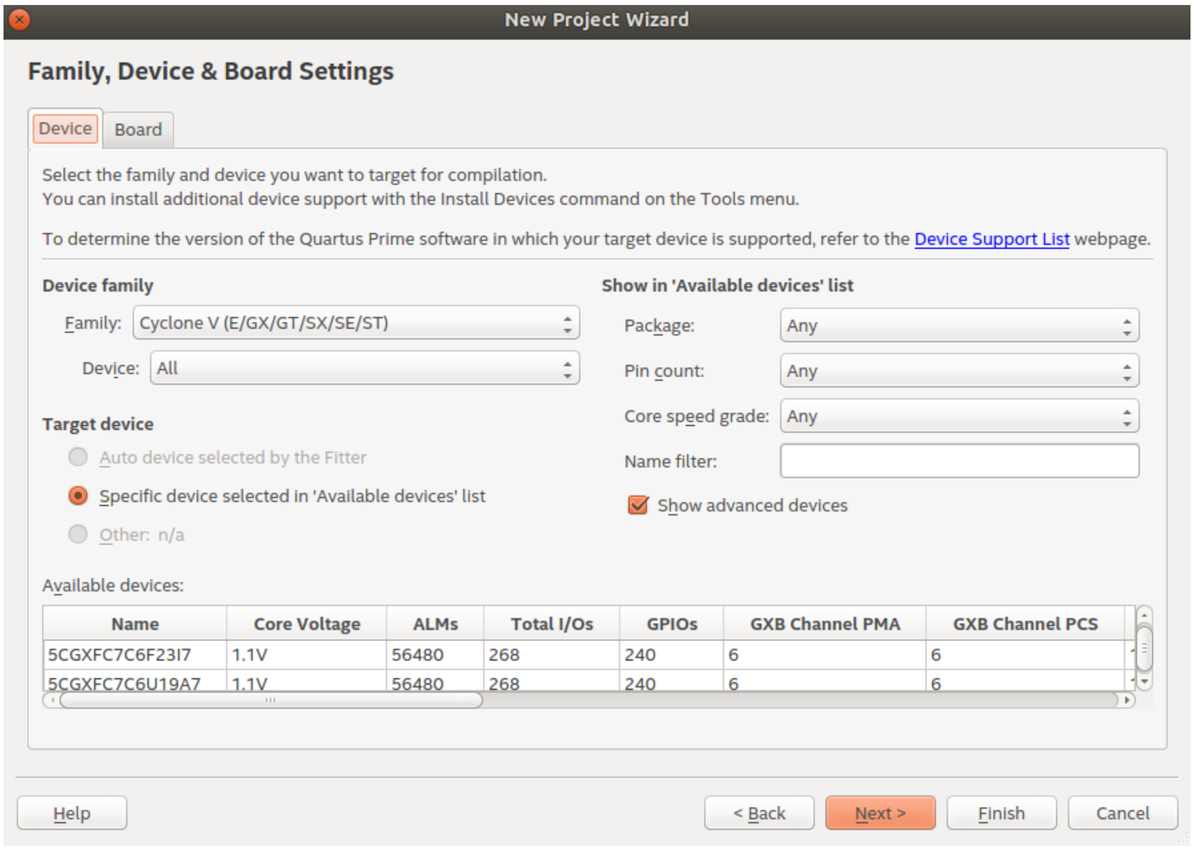

New Project Wizard
Add Files
Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.
Note: you
File name:
File Name
../Docu
Add
Add All
Remove
Up
Down
Properties
Select File
Look in: /home/isaac/Documen...Verilog/divide_by_2
Computer
isaac
divide_clk_by_2.sv
tb_divide_clk_by_2.sv
File name: divide_clk_by_2.sv
Open
Files of type: Design Files (*.tdf *.vhd *.vhdl *.v *.vlg *.verilog *.sv *.vqm *.edf *.edif *
Cancel
Specify the path names of any non-default libraries.
User Libraries...
Help
< Back
Next >
Finish
Cancel

Figure 20: Adding files.

7. Quartus allows us to select our target device, *i.e.*, our target Field-Programmable Gate Array (FPGA). For this tutorial, we can simply click on **Next** in order to select the default device, as shown in Figure 21.



Family, Device & Board Settings

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.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the Device Support List webpage.

Device family

Family: Cyclone V (E/GX/GT/SX/SE/ST)

Device: All

Target device

- Auto device selected by the Fitter
- [✓ Specific device selected in 'Available devices' list]
- Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

Name filter:

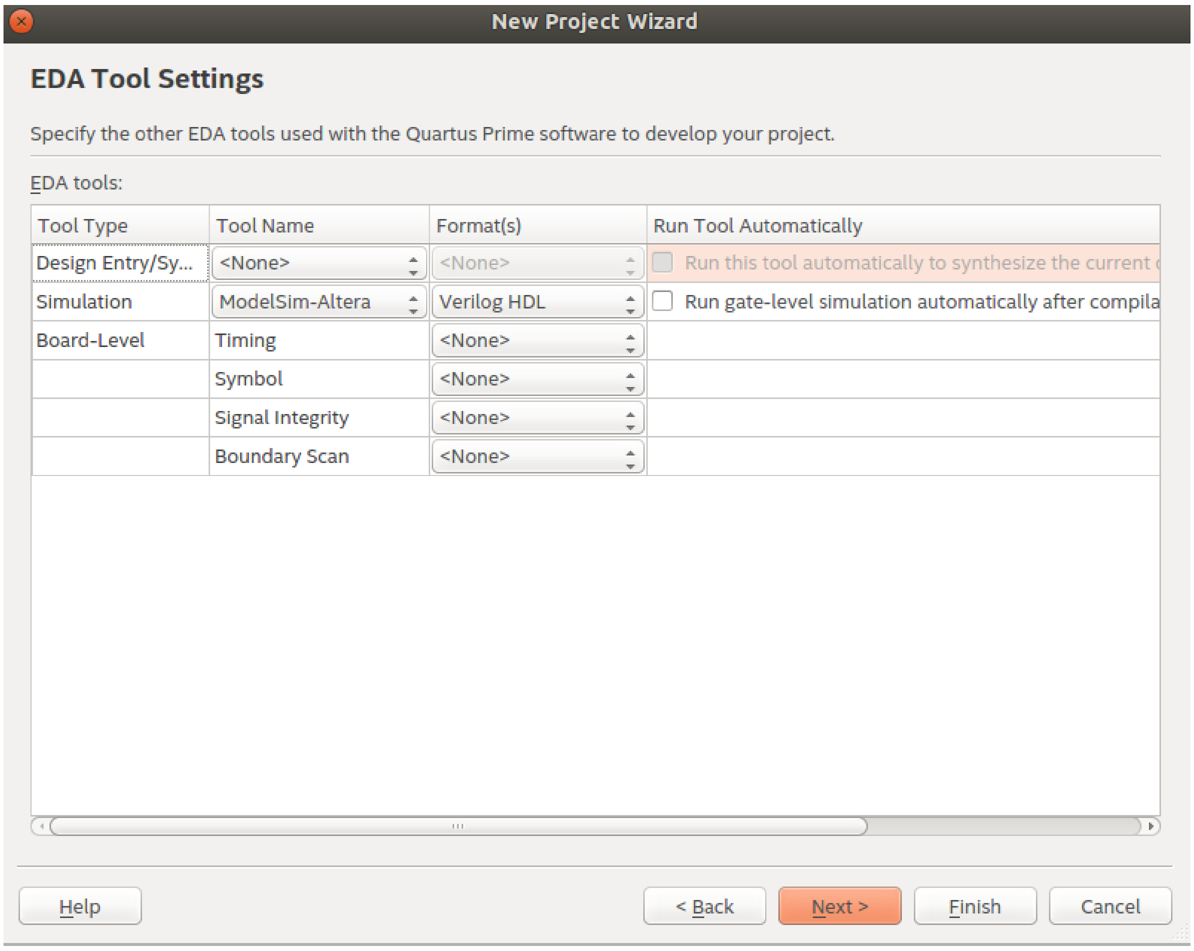
[✓ Show advanced devices]

Available devices:

Name	Core Voltage	ALMs	Total I/Os	GPIOs	GXB Channel PMA	GXB Channel PCS
5CGXFC7C6F23I7	1.1V	56480	268	240	6	6
5CGXFC7C6U19A7	1.1V	56480	268	240	6	6

Figure 21: Selecting device.

8. The last step for creating a project is to select which other Electronic Design Automation (EDA) tools will be used. We can choose our simulation tool as well as some static check tools for timing and signal integrity. However, for this tutorial, and for the majority of the course, we can simply click on **Next** without specifying any of the EDAs, as shown in Figure 22.


New Project Wizard

EDA Tool Settings

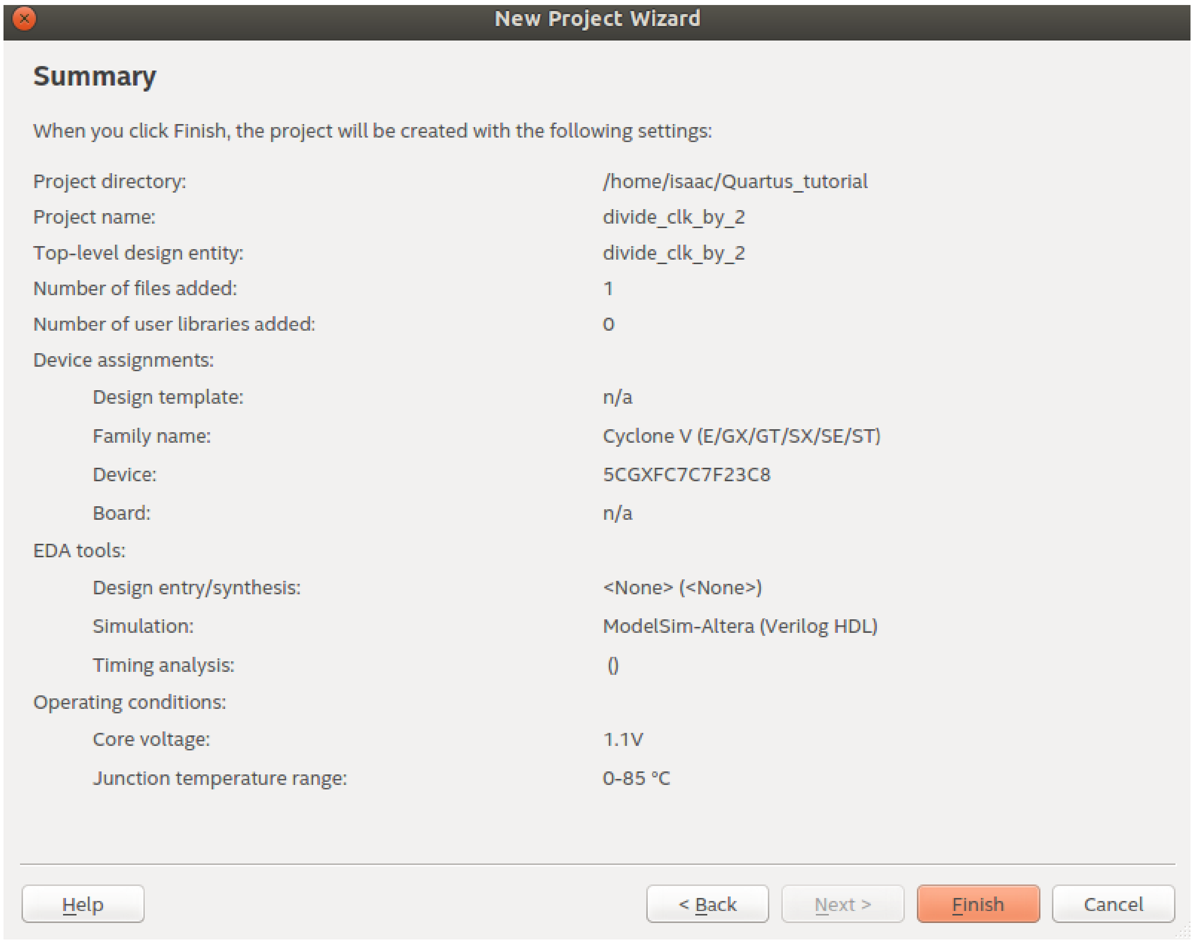
Specify the other EDA tools used with the Quartus Prime software to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)	Run Tool Automatically
Design Entry/Sy...	<None>	<None>	Run this tool automatically to synthesize the current
Simulation	ModelSim-Altera	Verilog HDL	Run gate-level simulation automatically after compila
Board-Level	Timing	<None>	
	Symbol	<None>	
	Signal Integrity	<None>	
	Boundary Scan	<None>	

Figure 22: Selecting EDA.

9. Finally, Quartus will prompt you with a summary of the project configuration. Click on **Finish** as shown in Figure 23.



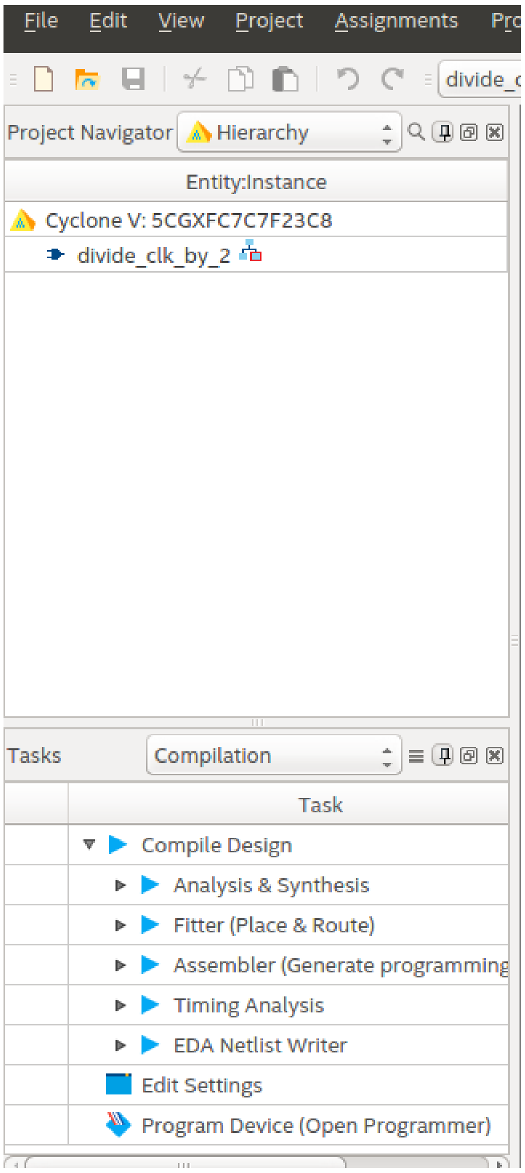
Summary

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

Project directory:	/home/isaac/Quartus_tutorial
Project name:	divide_clk_by_2
Top-level design entity:	divide_clk_by_2
Number of files added:	1
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	Cyclone V (E/GX/GT/SX/SE/ST)
Device:	5CGXFC7C7F23C8
Board:	n/a
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	ModelSim-Altera (Verilog HDL)
Timing analysis:	()
Operating conditions:	
Core voltage:	1.1V
Junction temperature range:	0-85 °C

Figure 23: Project summary.

10. The project is now ready to be compiled. You should see something similar to Figure 24. Here, the top part of the panel indicates the chosen device and the top-level unit to be compiled. The bottom part shows the synthesis stages that Quartus has performed.



Entity:Instance
Cyclone V: 5CGXFC7C7F23C8
divide_clk_by_2

	Task
	Compile Design
	Analysis & Synthesis
	Fitter (Place & Route)
	Assembler (Generate programming
	Timing Analysis
	EDA Netlist Writer
	Edit Settings
	Program Device (Open Programmer)

Figure 24: Project ready.

11. The next step is to perform synthesis. Double click on the blue triangle/arrow next to **Compile Design**. Quartus should start the synthesis process and you should see some progress bars constantly updating. This process can take several seconds or even minutes, depending on the computing power of your computer.

    Quartus indicates that a design has been successfully synthesized by using green check marks next to each stage, as shown in Figure 25. Quartus will also show a summary in a new panel. If a synthesis stage did not meet our constrains, Quartus will use a red font to point to the failing stage. In this case, you should see that we didn't meet the **Timing Analyzer** stage, as shown in the right side of Figure 25. This is expected since we didn't specify any timing constrain. Despite this, Quartus was able to synthesise our design.

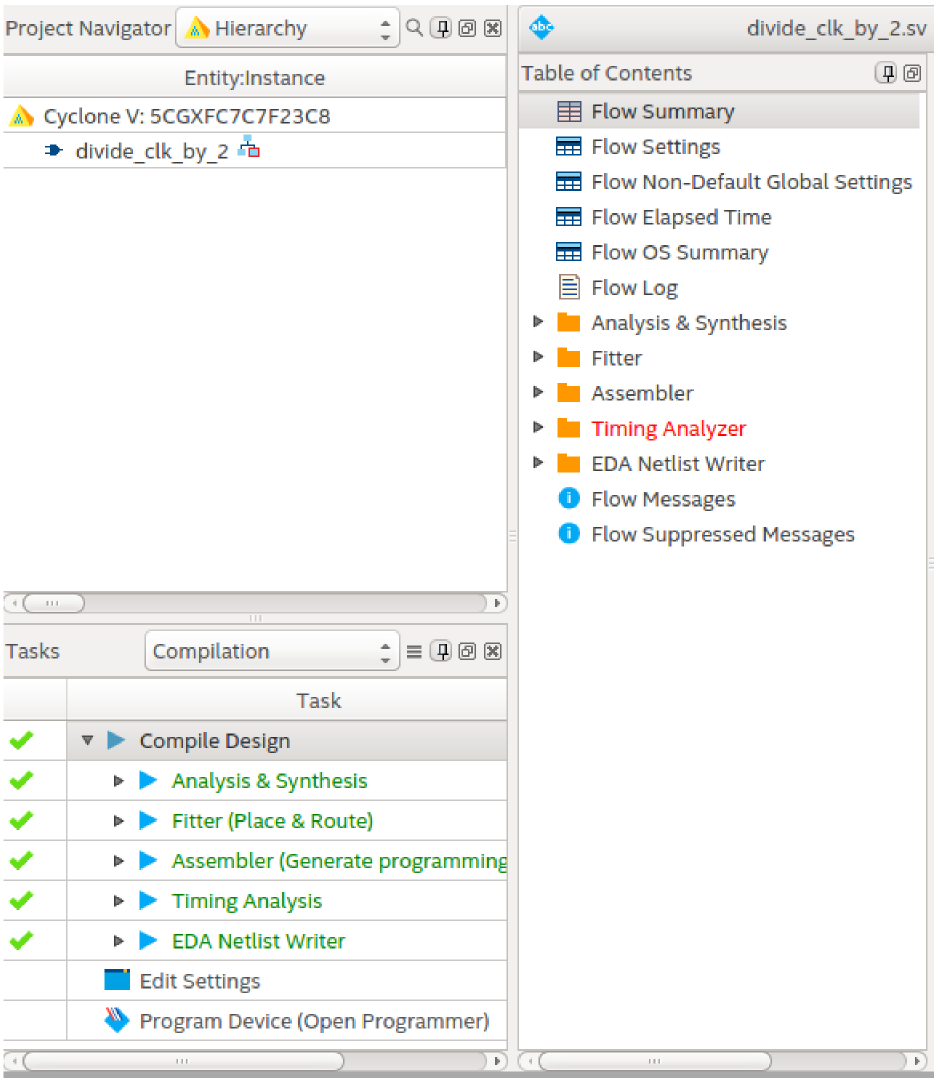


Figure 25: Compiled project.

12. The next step is to have a look at the inferred Register-Transfer Level (RTL). From the top menu, select **Tools** → **Netlist Viewers** → **RTL Viewer**, as shown in Figure 26.

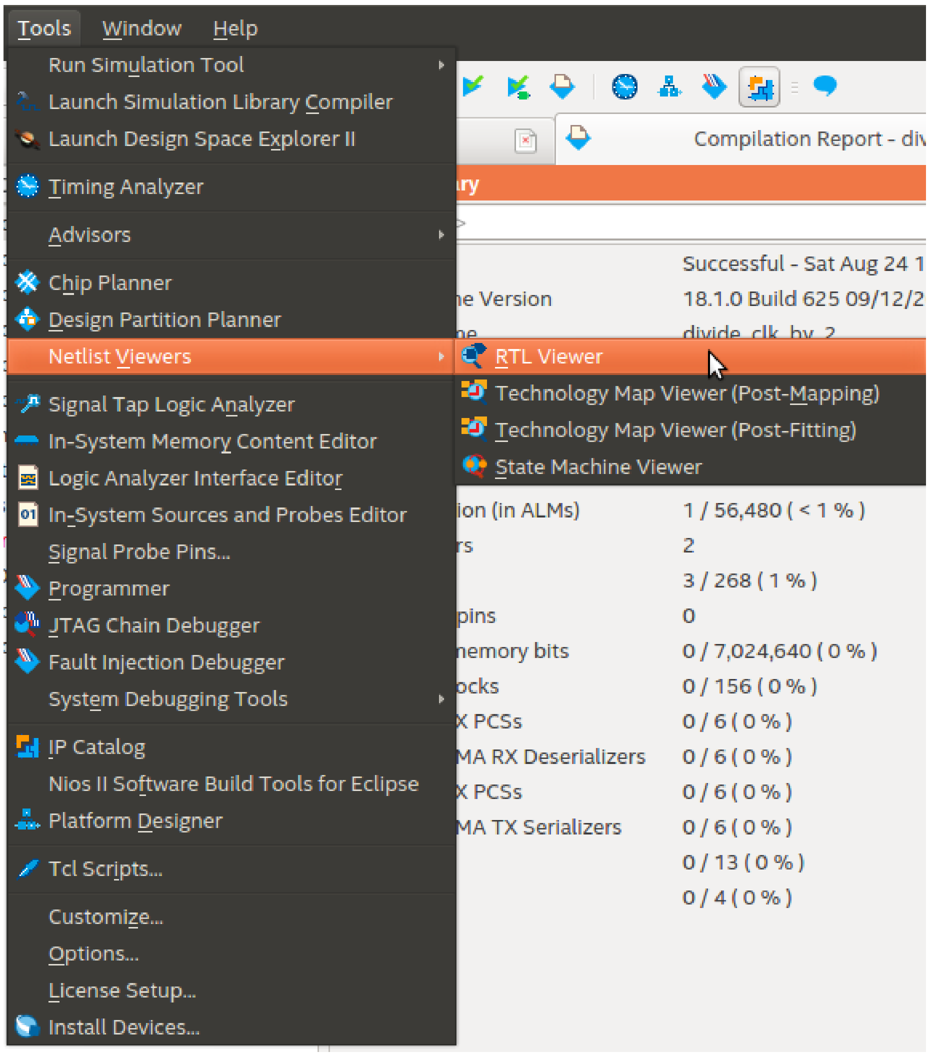


Figure 26: RTL view menu.

13. You should see something similar to Figure 27. Here, Figure 27 shows how Quartus logically interprets our SystemVerilog design.

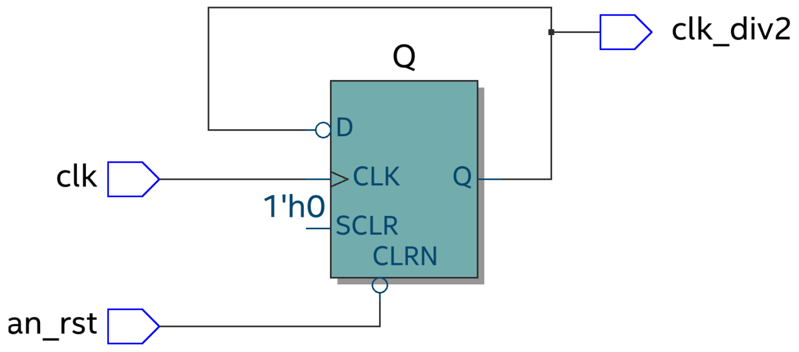


Figure 27: RTL view.

14. You may now close Quartus and conclude the Quartus part of this tutorial.
15. You may now proceed to the Quartus exercise part of the tutorial in Section 6.2.

6.2 Quartus exercise

1. Create a new Quartus project using the file `counter.sv`.
2. Modify `counter.sv` in order to synthesise a 6-bit counter.
3. Compile and synthesize the design.
4. **Congratulations!** You have finished this tutorial.