

Data Compression

Lab Tutorial: Getting started with System Generator

This tutorial has three primary parts:

- In Step 1, you will implement a 4-state Moore-based FSM with Xilinx Vivado. The implemented VHDL module is then verified using behavioral simulation.
- In Step 2, you will work with the System Generator (SysGen) and learn how to use a VHDL module in a sysgen project.
- In Step 3, a Xilinx Vivado project will be generated from SysGen design from Step 2. You will run the implementation of this project and check the implementation results.

Step 1: Implementing a VHDL module with Xilinx Vivado.

In this part, you will implement a simple FSM (Finite State Machine) using VHDL language. This FSM is actually a 2-bits counter that counts the number of bit 1 in *data_i* signal. Fig. 1 shows the state diagram of this FSM. It is a Moore state machine since the change of output *data_o* only depends on the current state not on the current input *data_i*.

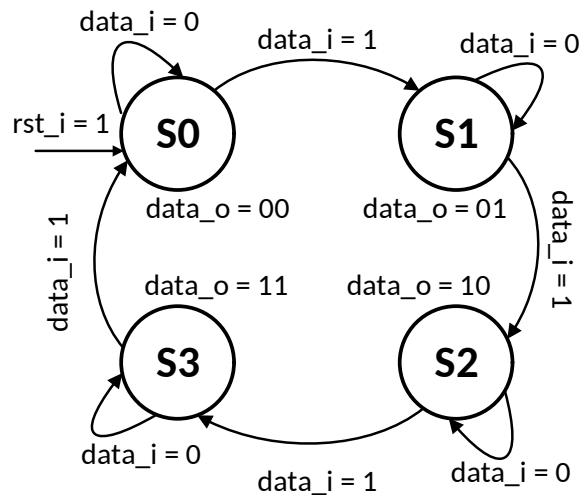


Fig. 1. 4-state Moore state machine

1. Open Vivado project.
Make sure the current directory is "`~/DC/P01/PART_A`". Run following command in terminal.
`./scripts/open_project_parta.sh`
2. This command will create a Vivado project and open it as in Fig. 2.

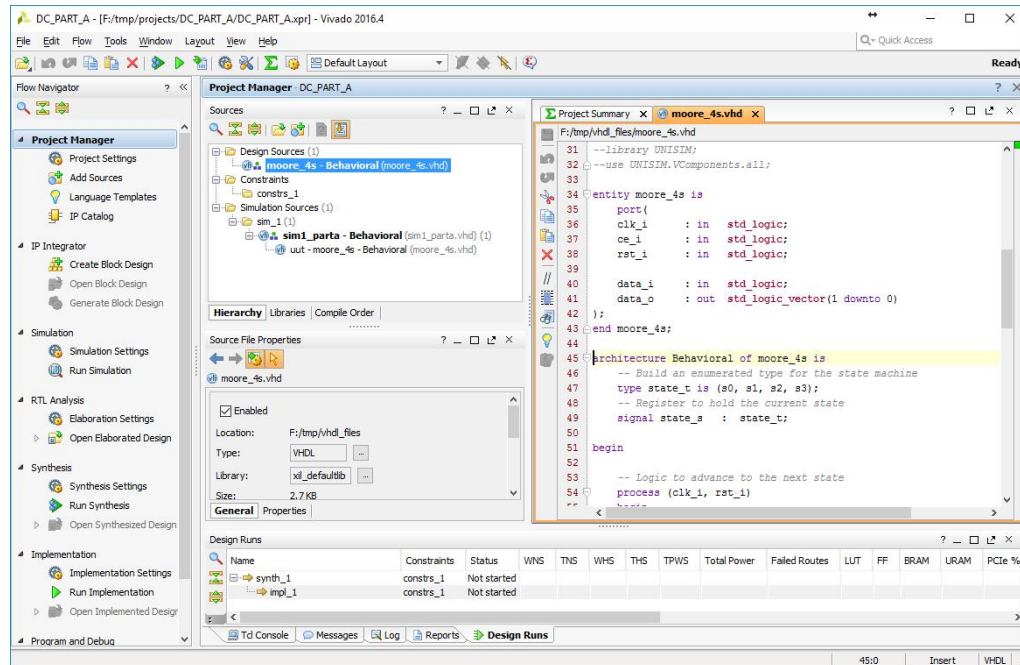


Fig. 2. Opened Vivado project.

3. The content of VHDL module *moore_4s* is missing. Finish the implementation of this FSM follow the state diagram in Fig. 1.
4. After finishing the implementation, synthesize this module by choosing **Synthesis** => **Run Synthesis** in the **Flow Navigator**.
5. Run a simulation to verify your implementation.
In the **Flow Navigator**, choose **Simulation** => **Run Simulation**. A pop-up window showing, choose **Run Behavioral Simulation** for behavioral simulation as in Fig. 3.

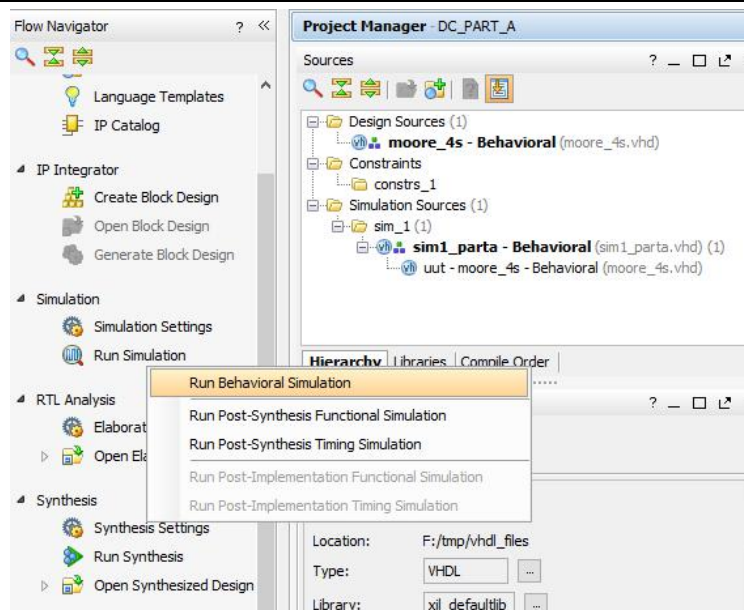


Fig. 3. Running Behavioral Simulation.

6. Check the waveform to verify if the FSM work correctly.

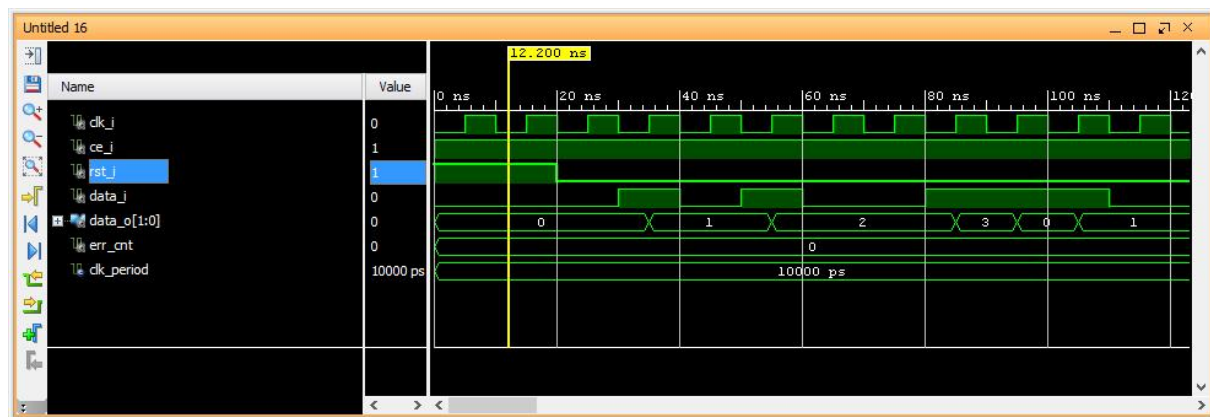


Fig. 4. Simulated waveform

Step 2: Using VHDL module in System Generator.

1. Open System Generator.
Make sure the current directory is "`~/DC/P01/PART_A/`". Run following command in terminal
`./scripts/open_sysgen.sh`
2. After this command, Matlab will be showed. In the **Command Window**, type following to open a system generator design.
`open moore_state_machine.slx`
3. A simulink project is opened as in Fig. 5. There are several missing components that need to be added before you can run the simulation.

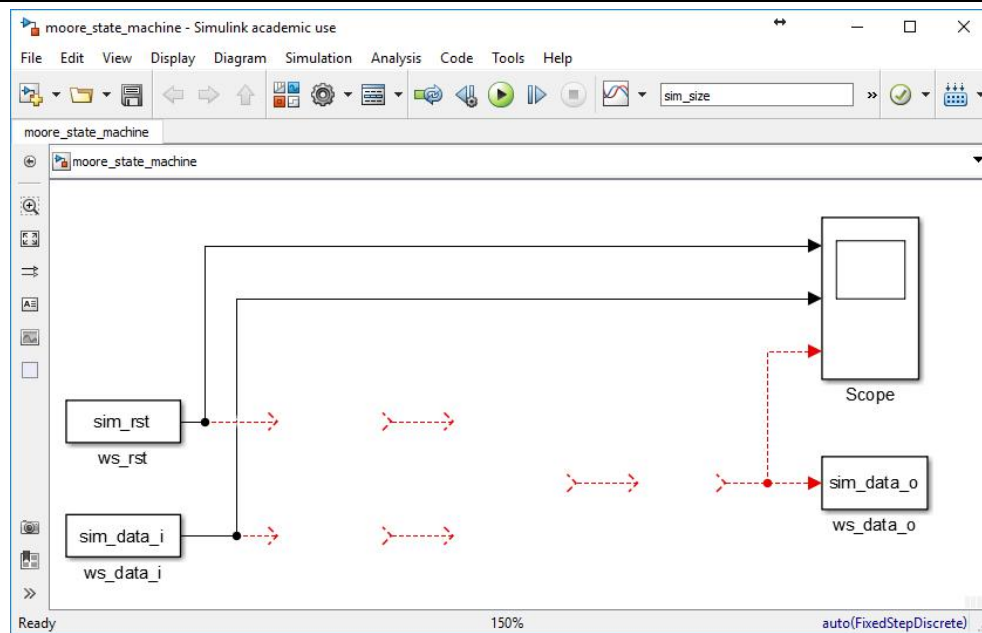


Fig. 5. Opened simulink project

4. In the menu **View** choose **Library Browser** to choose some building blocks for this design. The Simulink Library Browser window is open as in Fig. 6.

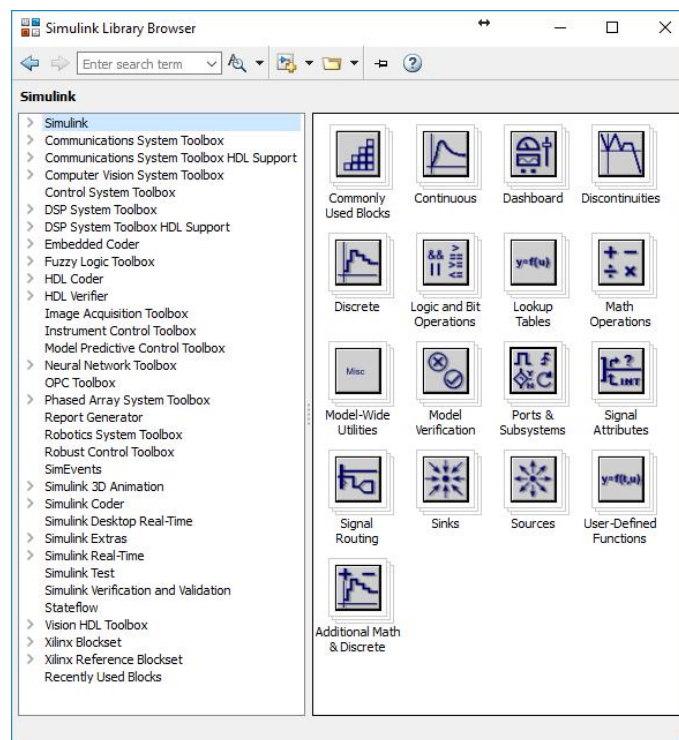


Fig. 6. Simulink Library Browser.

5. In the *Xilinx Blockset* => *Basic Elements* group, choose and add following blocks into current design.
 - System Generator Token
 - 2 Gateway In
 - 1 Gateway Out.
6. Connect the work space variables (*sim_rst*, *sim_data_i*, *sim_data_o*) with input and output Gateway as in Fig. 7.

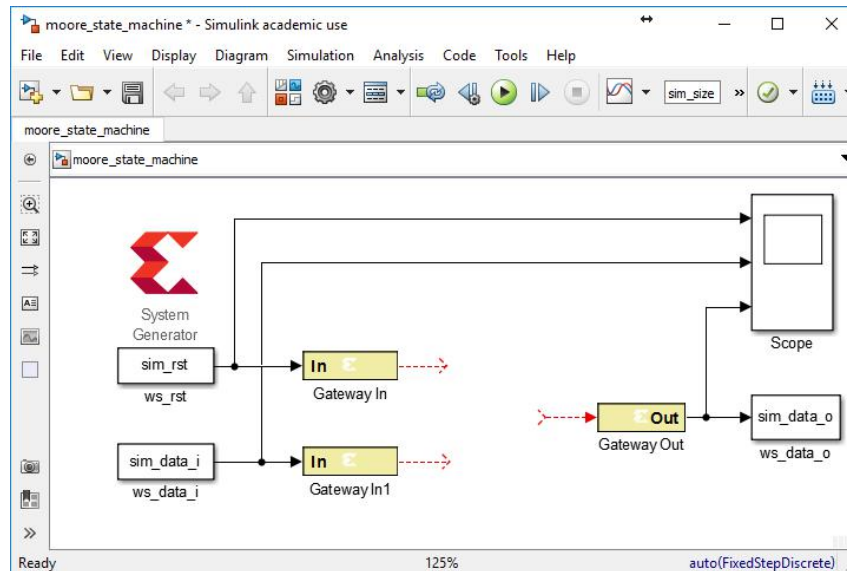


Fig. 7. Connected Gateway In/Out.

7. Adding blackbox for VHDL module.
 Right click in the middle of Simulink design and choose *Xilinx Block Add*.
 Search for **Black Box** and then double click to it. (Fig. 8.)
 A browser window opens for choosing the VHDL source file that will be associated with this black box. Choose the *moore_4s.vhd* from previous step.

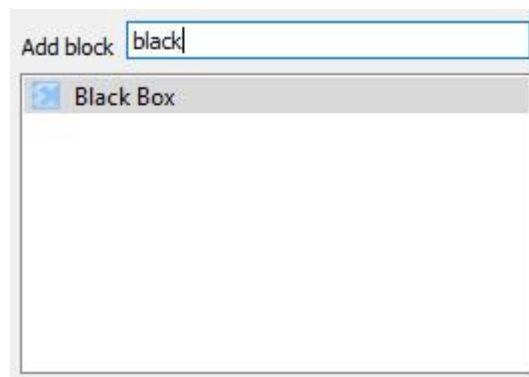


Fig. 8. Add a Black Box block.

8. The associated configuration M-code *moore_4s_config.m* will be created and open in an Editor for modification.
9. Connect the Black Box with Gateway as in Fig. 9.

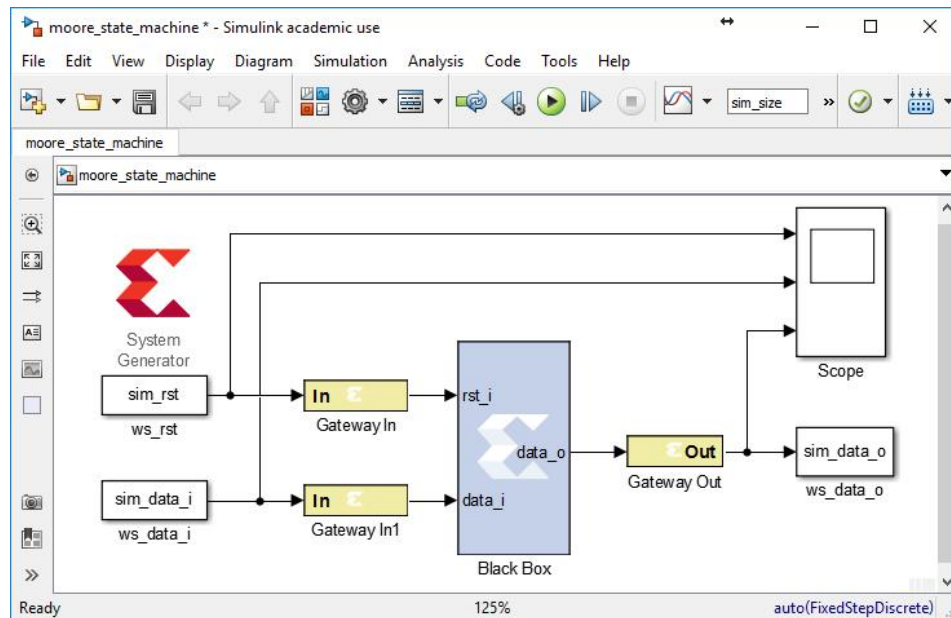


Fig. 9. Connected all simulink blocks

10. Double click on Gateway In block to bring up the configuration window and configure it as in Fig. 10.

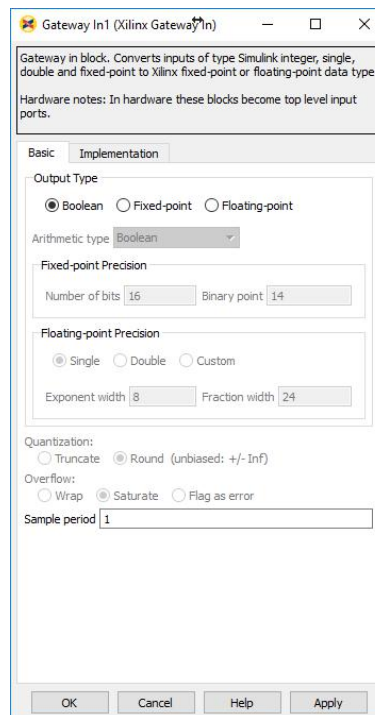


Fig. 10. Set Gateway In as boolean signal.

11. Load the simulation data by typing following command in Matlab **Command Window**
`loadSimData`
12. Look into file loadSimData.m to understand the format of input data `sim_rst` and `sim_data_i`
13. Run simulation by choosing menu **Simulation** => **Run**. The expected result in the Scope window should look like the Fig. 11.

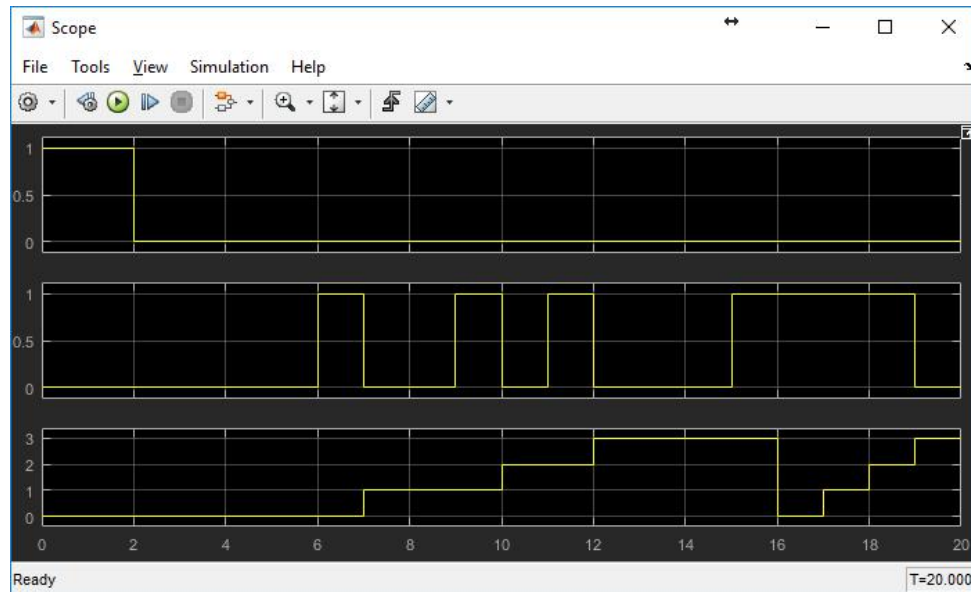


Fig. 11. Scope view.

Step 3: Generate a Vivado Project using System Generator

1. Double click on *System Generator Token* to bring up the configuration window.
2. On **Compilation** tab, set
 - **Board** to *Zybo B.3*.
 - **Compilation** to *IP Catalog*
 - **Hardware Description Language** to *VHDL*

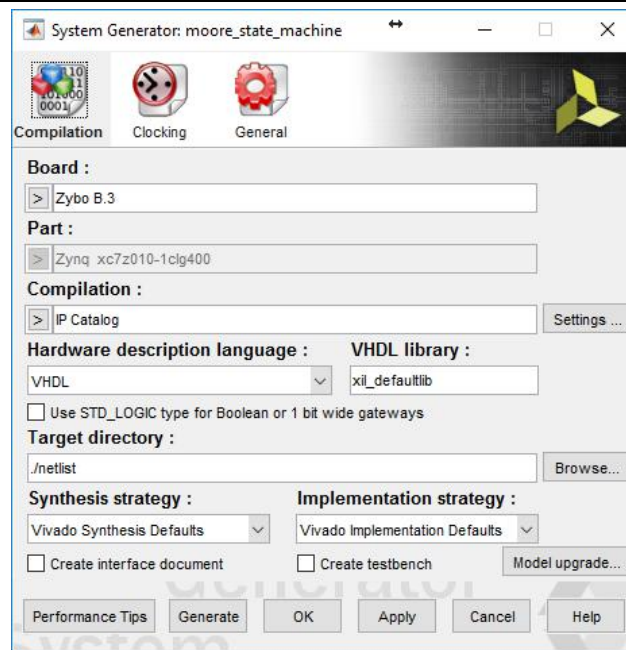


Fig. 12. Configure the System Generator

3. On Clocking tab, set **FPGA clock period (ns)** to 5.

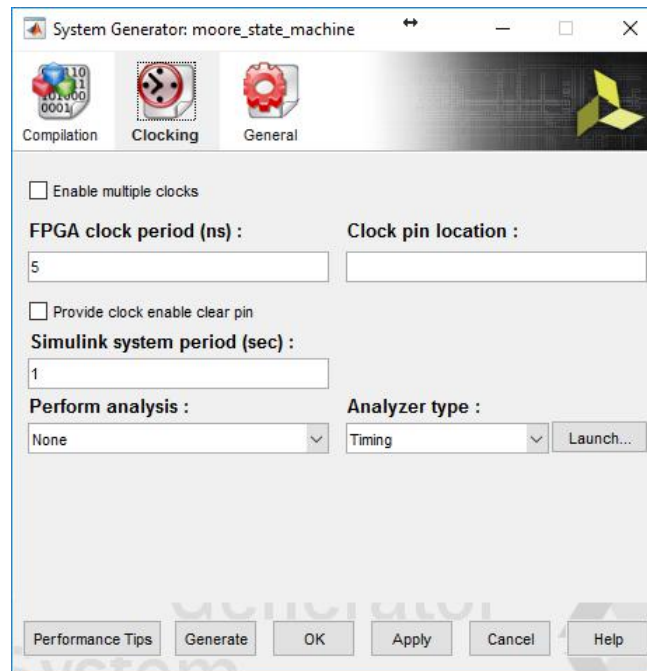


Fig. 13. Configure System Generator

4. Click **Generate** to generate a Vivado project.

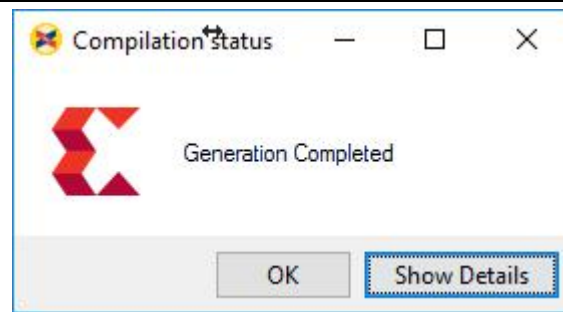


Fig. 14. The generation is finished

5. Take a look in the folder "`~/DC/P01/PARTA/sysgen/netlist`" to see generated files. The output from design generation process is written to the *netlist* directory. This directory contains three subdirectories:
 - *sysgen*: This contains the RTL design description written in the industry standard VHDL format. This is provided for users experienced in hardware design who wish to view the detailed results.
 - *ip*: This directory contains the design IP, captured in Xilinx IP Catalog format, which is used to transfer the design into the Xilinx Vivado Design Suite. Lab 8: Using AXI Interfaces and IP Integrator, presented later in this document, explains in detail how to transfer your design IP into the Vivado Design Suite for implementation in an FPGA.
 - *ip_catalog*: This directory contains an example Vivado project with the design IP already included. This project is provided only as a means of quick analysis.
6. Open Xilinx Vivado using following command.


```
./scripts/open_vivado.sh
```
7. In the menu **File** choose **Open Project** to open the example Vivado project in folder:


```
"~/DC/P01/PARTA/sysgen/netlist/ip_catalog"
```
8. Take a look into design constraint file. There is a timing constraint generated due to our configuration in 3.

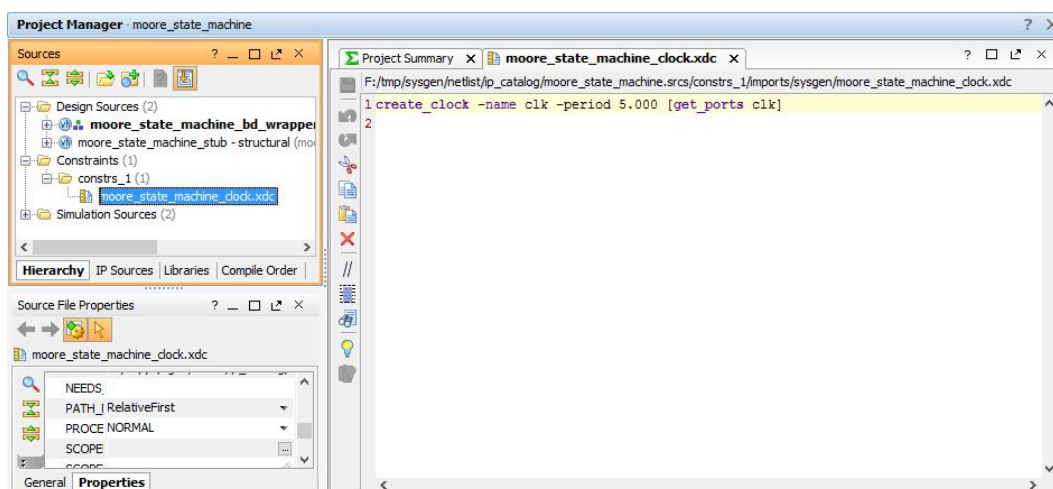


Fig. 15. Generated timing constraint.

9. Under Flow Navigator, choose **Implementation** => **Run Implementation**.

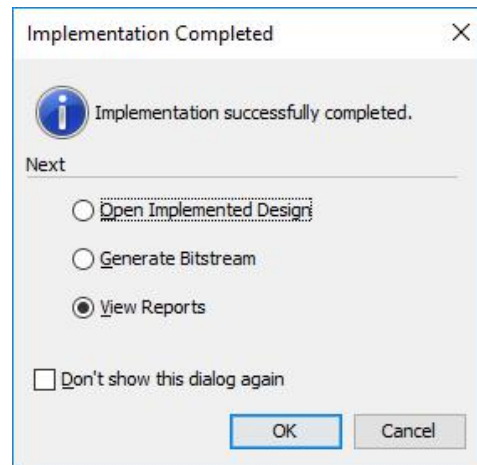
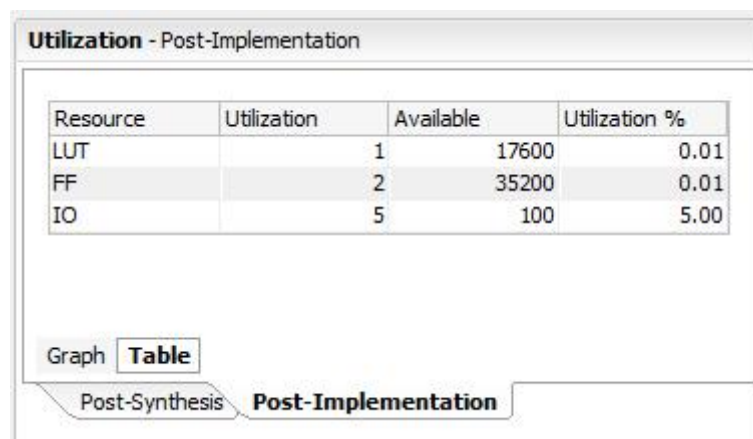


Fig. 16. The implementation is finished

10. Check the implementation result. In menu **Windows** choose **Project Summary**. The hardware resource consumption report can be seen in Utilization sub-window as in Fig. 17.

A window titled "Utilization - Post-Implementation" showing a table of hardware resource utilization. The table has four columns: Resource, Utilization, Available, and Utilization %. Below the table are two tabs: "Graph" and "Table" (which is selected). At the bottom, there are two sub-window tabs: "Post-Synthesis" and "Post-Implementation" (which is selected).

Resource	Utilization	Available	Utilization %
LUT	1	17600	0.01
FF	2	35200	0.01
IO	5	100	5.00

Fig. 17. Hardware resource utilization.

11. Check timing report. In the **Timing** sub-window click **Implemented Timing Report** as in Fig. 18. The detail timing report will be showed as in Fig. 19.

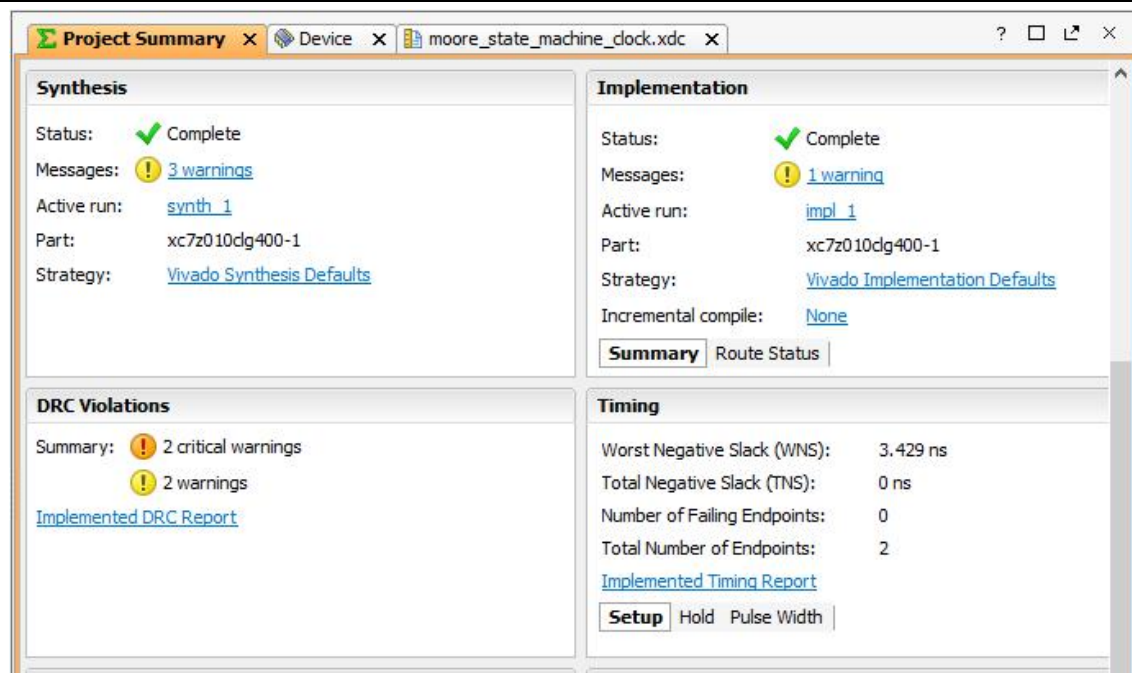


Fig. 18. Check timing report

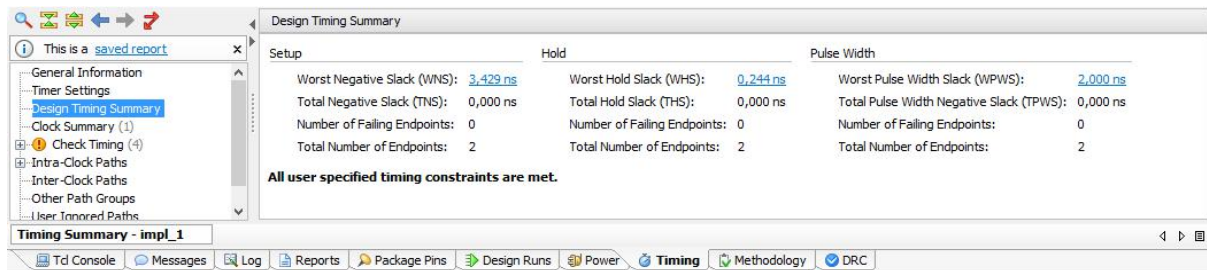


Fig. 19. Timing report detail.