Implementation in Vivado

After we have finished simulating, it is now time to move on and run the implementation.In Vivado, **implementation** is the process that maps your synthesized HDL design onto the physical resources of a Xilinx FPGA. This step is similar to what Quartus refers to as **fitting**.

After synthesis is completed, Vivado proceeds with:

1. **Placement** – Vivado places the logic elements (LUTs, flip-flops, etc.) onto the FPGA fabric.
2. **Routing** – Vivado connects the placed elements using the FPGA's routing architecture.
3. **Timing Analysis** – Vivado checks whether all signal paths meet the timing constraints (setup/hold).

Once implementation is successful, you can proceed to **bitstream generation** and **programming the FPGA**.

📌 Constraints and I/O Assignments

Vivado uses a **.xdc file** (Xilinx Design Constraints) to assign pins and define I/O standards. This is equivalent to the .qsf file in Quartus.

We will also go through:

- How to add a .xdc file to your project
- How to assign ports to FPGA pins using the .xdc file or via the graphical interface

This will ensure that your design is properly connected to the hardware on your FPGA board.

Let's start!

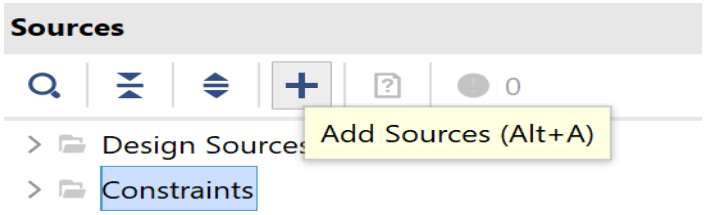
To avoid complicating things at the beginning, I will continue using a simple design, just like last time(and_gate).

We will start by adding the xdc file to the design sources, but first you need to download the xdc file. Click on the link below to download it.

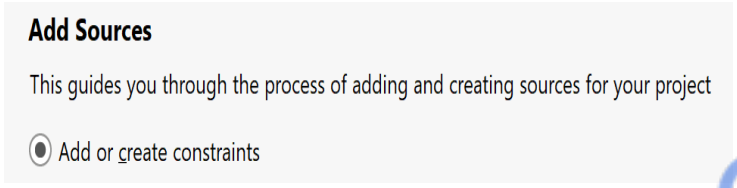
https://github.com/Digilent/digilent-xdc/blob/master/Arty-S7-50-Master.xdc

After you download and save the xdc file:

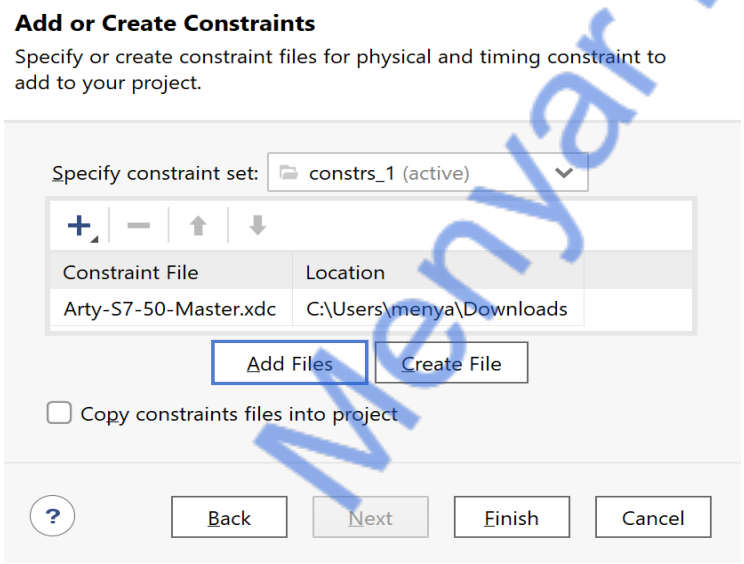
In the **sources window,** click on **Add Sources.**



Choose **Add or Create Constraints,** then click **Next**

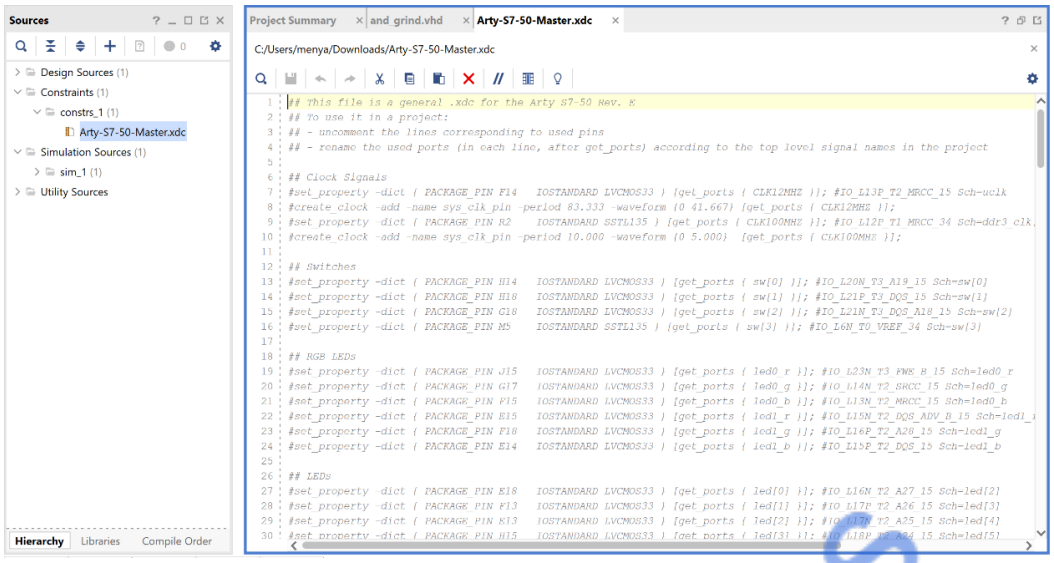


Select Create File, choose **xdc** as the file type, and click on finish.



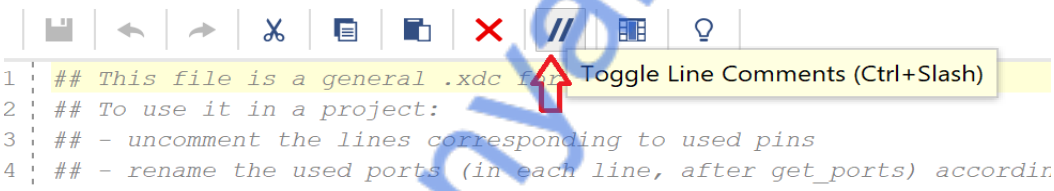
The file has now been added to the constraint's files.

Double click on it to open the file. Now you will see a list of available signals (ports), each one preceded by a comment symbol (#).



These are placeholders for assignments. To activate a constraint, simply **remove the comment symbol** and **modify the line** to match your desired pin and standard.
This is how you assign a physical pin to a specific signal in your design.

Mark the lines you want to comment out and then press this symbol



In this case we need to assign 2 inputs and one output for the and_gate and I chose to have 2 Switches for **a** and **b** inputs and one Led for the output **y.**

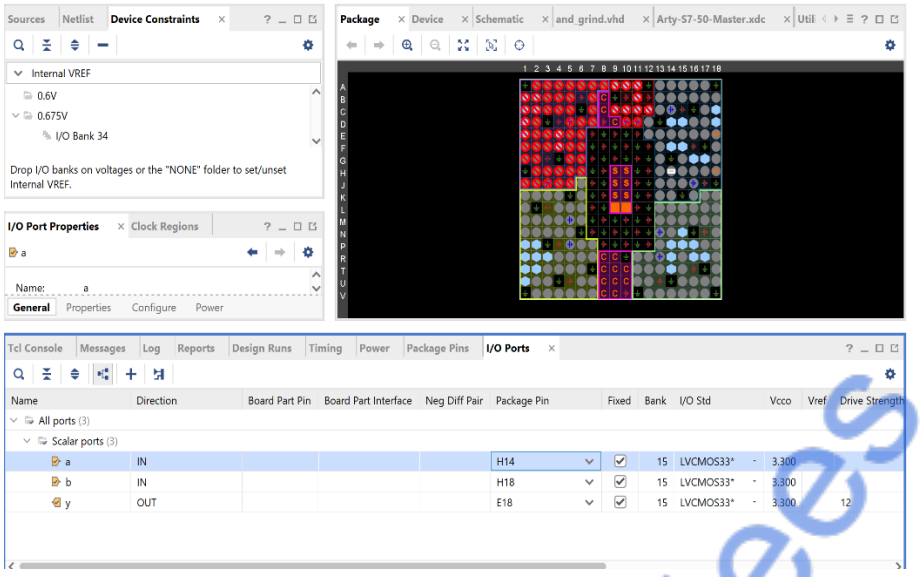
```
## Switches
set_property -dict { PACKAGE_PIN H14   IOSTANDARD LVCMOS33 } [get_ports { a }]; #IO_L20N_T3_A19_15 Sch=sw[0]     port a
set_property -dict { PACKAGE_PIN H18   IOSTANDARD LVCMOS33 } [get_ports { b }]; #IO_L21P_T3_DQS_15 Sch=sw[1]     port b
#set_property -dict { PACKAGE_PIN G18   IOSTANDARD LVCMOS33 } [get_ports { sw[2] }]; #IO_L21N_T3_DQS_A18_15 Sch=sw[2]
#set_property -dict { PACKAGE_PIN M5    IOSTANDARD SSTL135 } [get_ports { sw[3] }]; #IO_L6N_T0_VREF_34 Sch=sw[3]

## RGB LEDs
#set_property -dict { PACKAGE_PIN J15   IOSTANDARD LVCMOS33 } [get_ports { led0_r }]; #IO_L23N_T3_FWE_B_15 Sch=led0_r
#set_property -dict { PACKAGE_PIN G17   IOSTANDARD LVCMOS33 } [get_ports { led0_g }]; #IO_L14N_T2_SRCC_15 Sch=led0_g
#set_property -dict { PACKAGE_PIN F15   IOSTANDARD LVCMOS33 } [get_ports { led0_b }]; #IO_L13N_T2_MRCC_15 Sch=led0_b
#set_property -dict { PACKAGE_PIN E15   IOSTANDARD LVCMOS33 } [get_ports { led1_r }]; #IO_L15N_T2_DQS_ADV_B_15 Sch=led1_
#set_property -dict { PACKAGE_PIN F18   IOSTANDARD LVCMOS33 } [get_ports { led1_g }]; #IO_L16P_T2_A28_15 Sch=led1_g
#set_property -dict { PACKAGE_PIN E14   IOSTANDARD LVCMOS33 } [get_ports { led1_b }]; #IO_L15P_T2_DQS_15 Sch=led1_b

## LEDs
set_property -dict { PACKAGE_PIN E18   IOSTANDARD LVCMOS33 } [get_ports { y }]; #IO_L16N_T2_A27_15 Sch=led[2]     port y
```

Manuell pin-assignment i Vivado

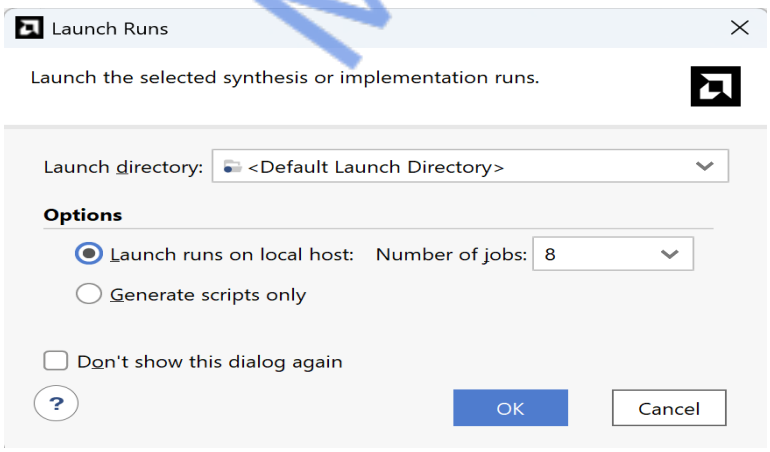
There is another way to do this in vivado. Click on **Layout** and then choose **I/O Planing.**



Here you can see the entire packaging and your inputs and outputs in your design and under the pin package row you can select and change the pin.

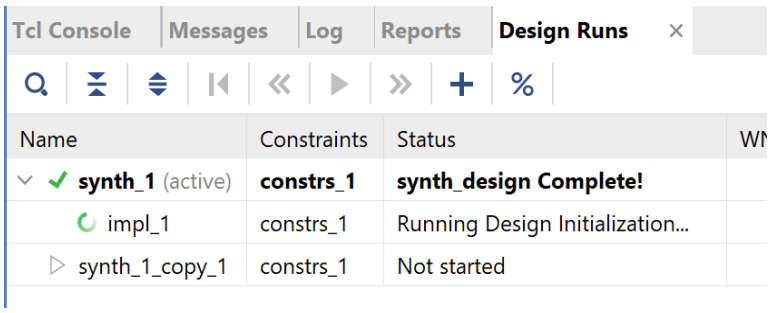
We have already run synthesis so now we can click on implementation.

When you click on Run Implementation you will get a optional to change the number of jobs on the launch runs on local host. and that means the number of available cores you have and want to use for the process. It is recommended to use all of them for faster processing. In my case I have 8.

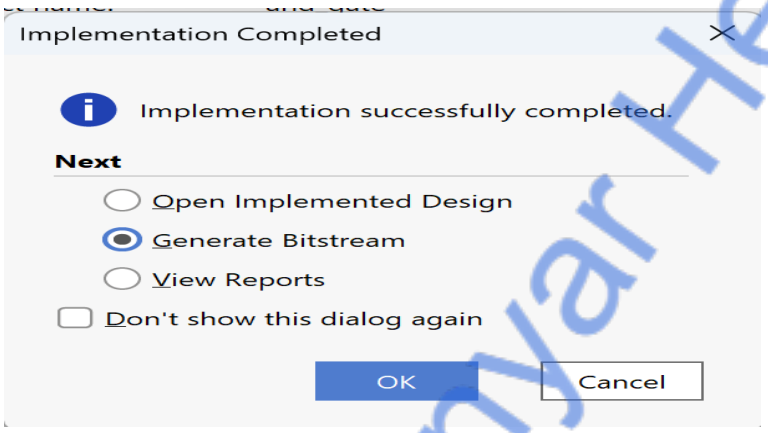


Click on OK.

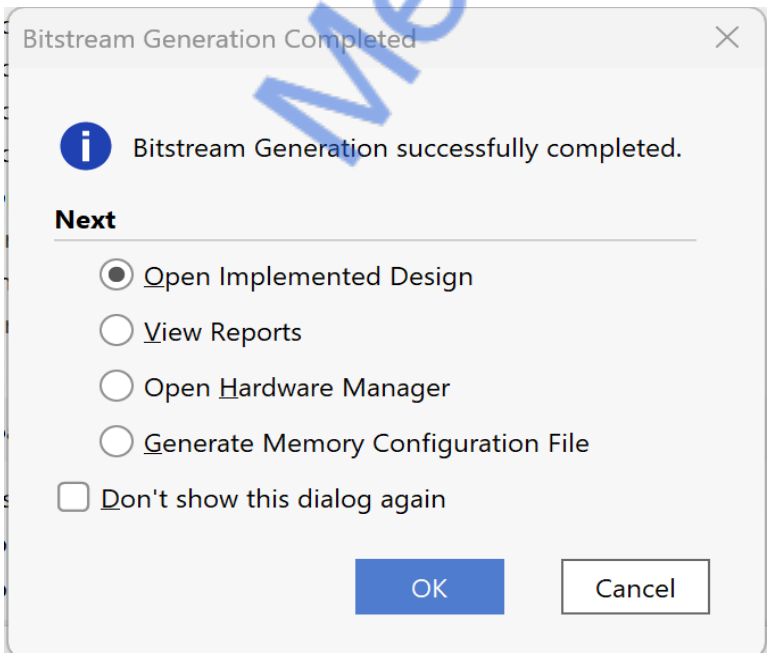
Under the Design Runs you can see the running processes.



Once the implementation process is complete, you will see this window. Here, you choose 'Generate Bitstream'. This is the file we will use to program the FPGA board. And click OK.



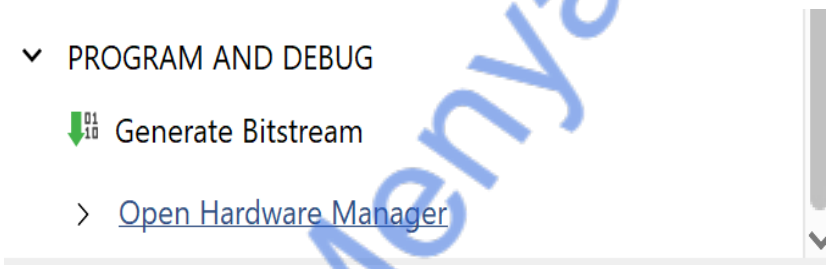
When the generation of the Bitstream is complete, this window will appear.



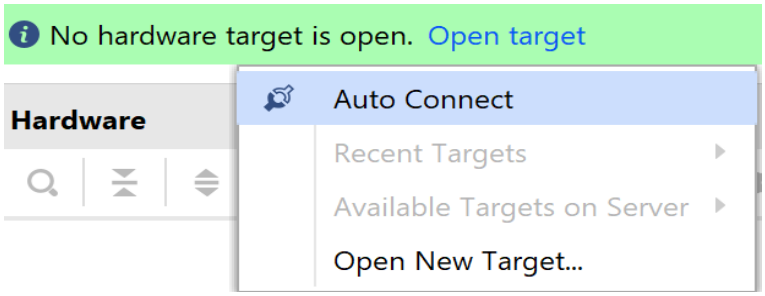
- **Open Implemented Design**: To view your design after implementation.
- **View Reports**: To check timing, resource, or other reports.
- **Open Hardware Manager**: to program the FPGA.
- **Generate Memory Configuration File**: To create files for external memory.

You can also close it without choosing any option for now. To view the design or edit timing and constraints, you can always click on **'Open Implemented Design'**. We well go through this later on the Demo.

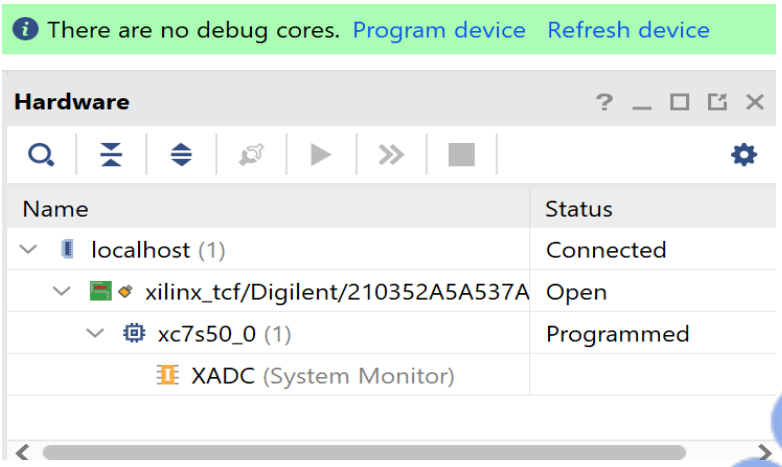
To program the FPGA board, connect it to the computer click on **Open Hardware Manger**.



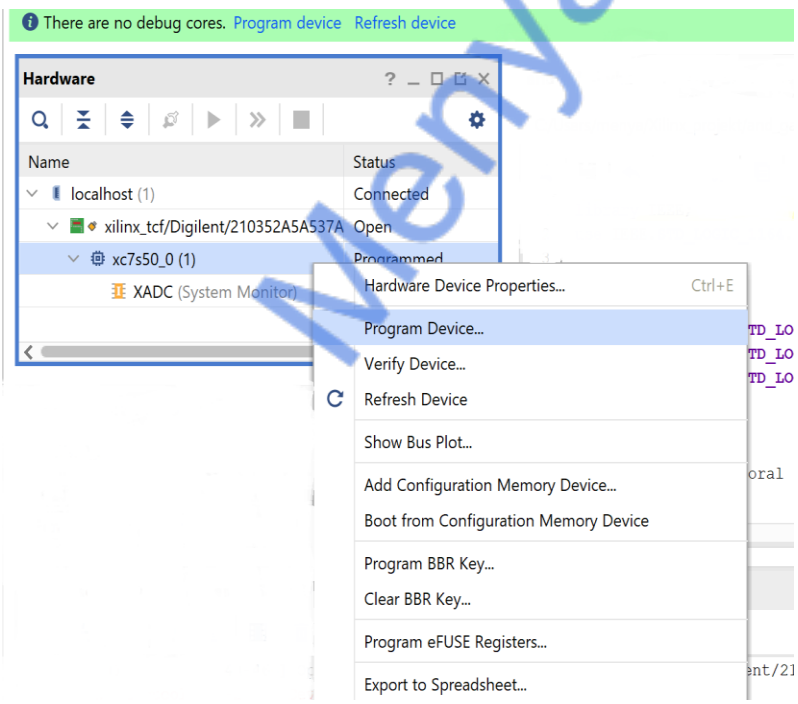
Click on **Open Target** and choose **Auto Connect.**



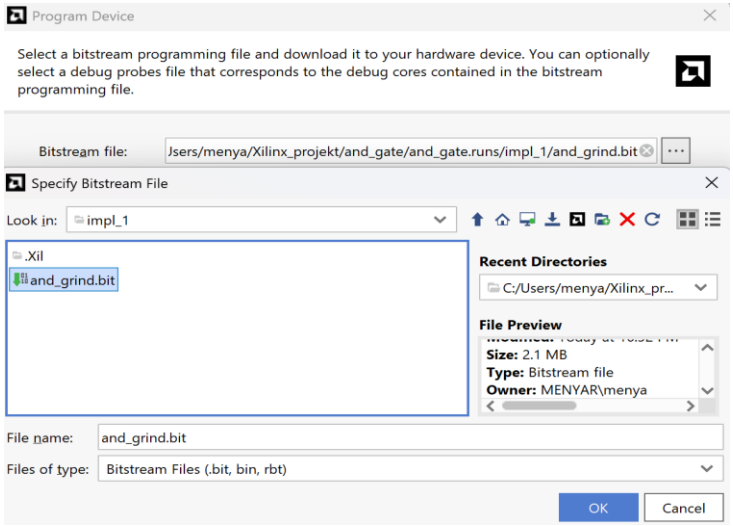
Vivado will automatically connect directly with your FPGA



To program the FPGA right click on fpga-an serial number and choose **Program Device.**



Choose th **bit.file** and click **OK** then **Program.**



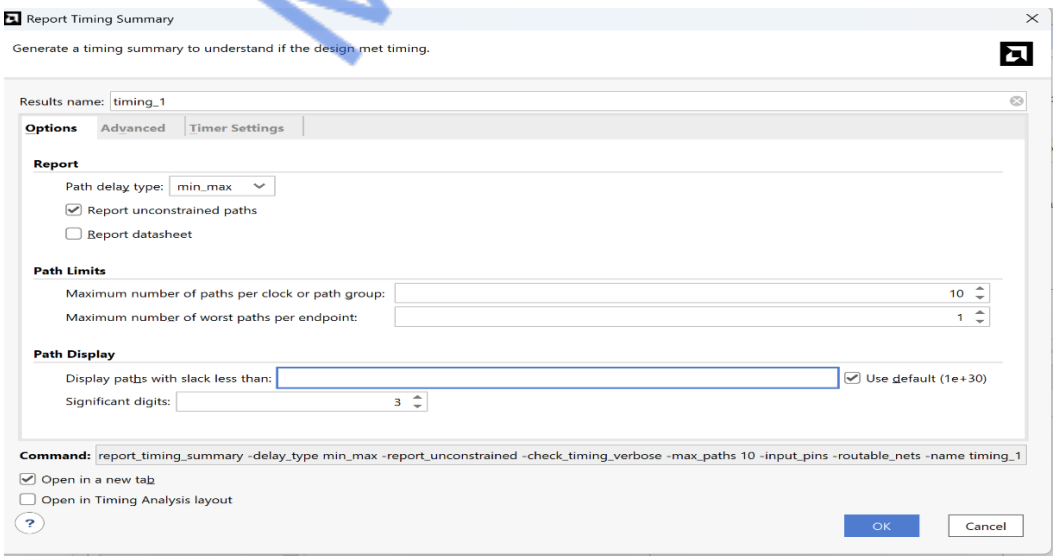
Now you can validate if your design works on the FPGA board !

Timing Analysis in Vivado

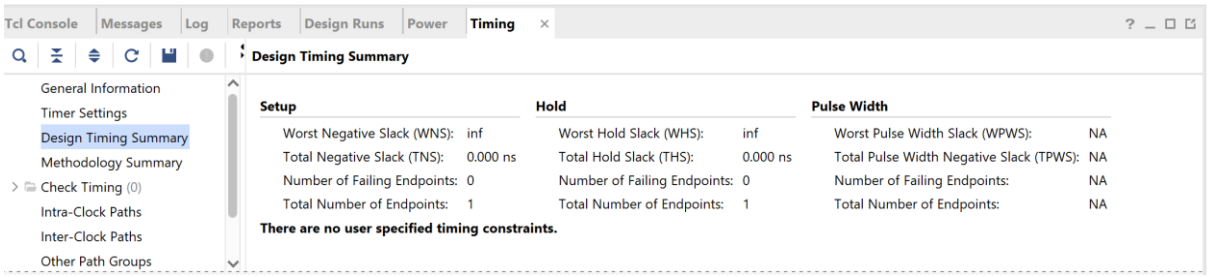
To check if the design meets timing, I clicked on "Report Timing Summary" in Vivado. This window appeared.

- I kept the delay type as min_max, which checks both setup and hold timing.
- I checked "Report unconstrained paths" to include paths that are not covered by timing constraints.
- You can also limit the number of paths shown or filter by slack value under Path Display.
- The command at the bottom shows the exact TCL command Vivado will run.

Then, click **OK** to generate the report and view timing results**.**



This will generate a time report for you



Demo for synthesis implementation, time analyzing and PIN assignment.

https://www.youtube.com/watch?v=4oTwGYvK8Ug

Demo for implementation, generate bit file and programming the FPGA board.

https://www.youtube.com/watch?v=gM2B3UIOZm0

Demo for implementation and Comparison of Different Strategies (Using project example, synthesized CPU)

https://www.youtube.com/watch?v=aSIQoeDdAzA
