

# Creating a Digital Comparator Circuit

Electronics 1 Laboratory

Dr Brian Cogan & Dr Patrick McEvoy

# Comparator

- In this laboratory, the Vivado Design Environment will be used to design, simulate and synthesise a digital comparator circuit
- Finally, the functionality of the finished design will be tested by implementing it and programming it onto the Artix-7 FPGA of the Digilent Basys3 board
- To complete this lab you will need the Report Template, headphones to listen to online videos and the Digilent Basys3 FPGA development board

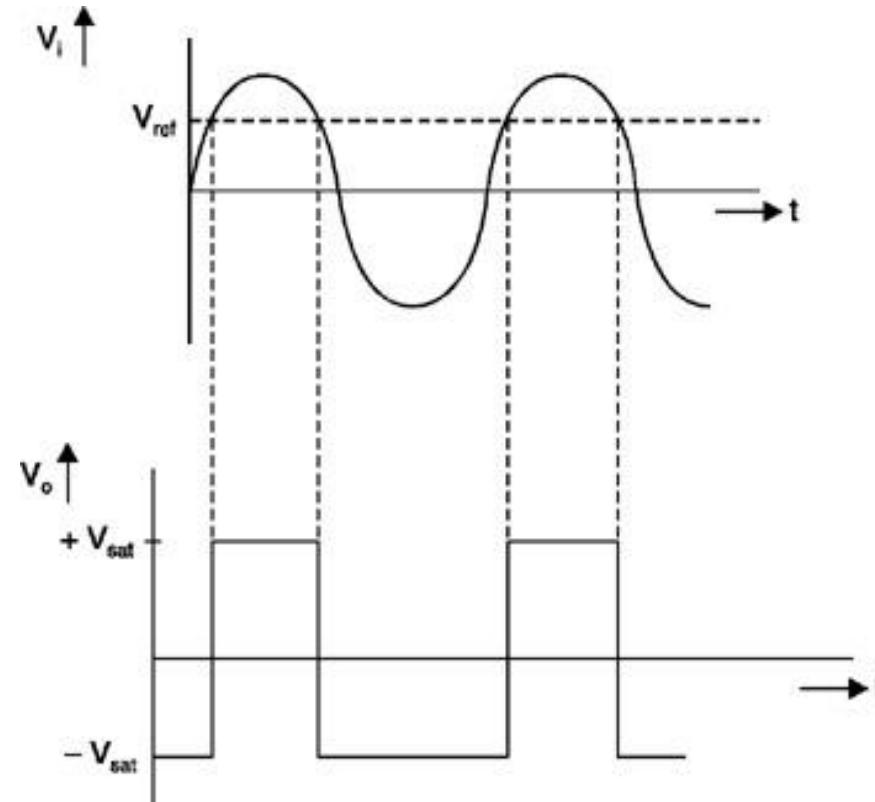
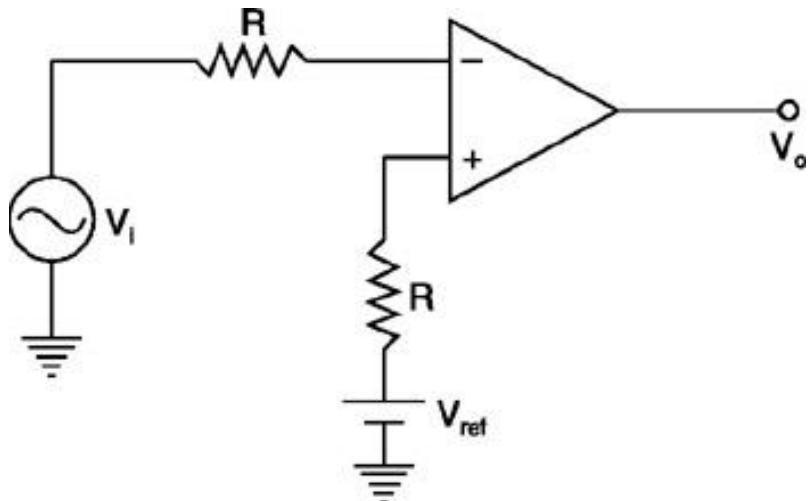
# Learning Objectives

In this laboratory, you will:

- Create VHDL design source code
- Create a VHDL simulation code to test the design
- Implement the design on the Basys 3 development board based on Artix-7 FPGA
- Apply signals to the board via the switches to confirm that the design works as planned

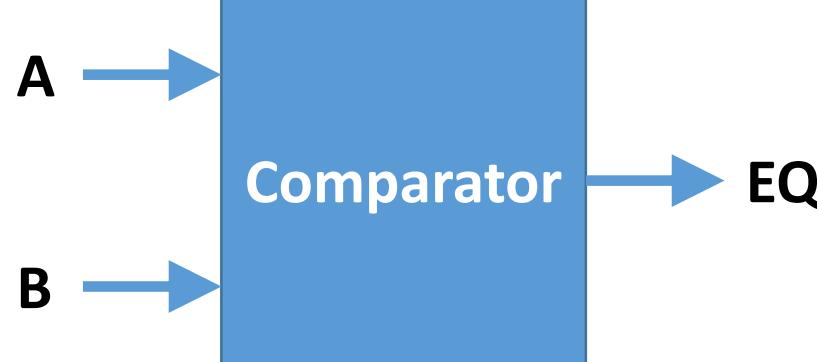
# An Analogue Comparator

- The output of an analogue comparator is high when the input signal equals or exceeds the reference signal



# A Digital Comparator

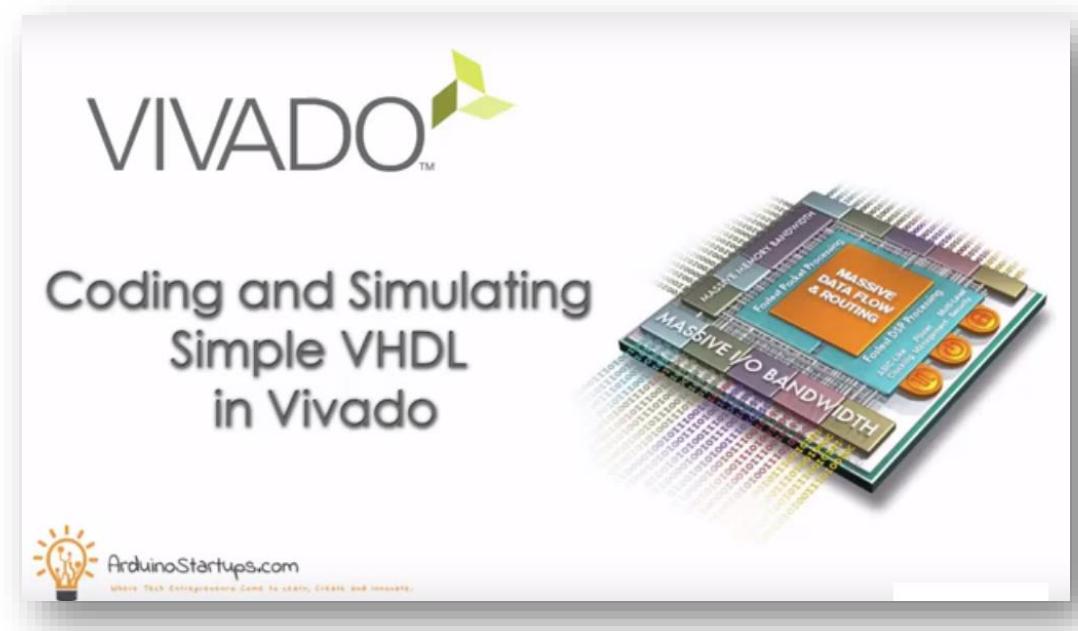
- A 1-bit digital comparator provides an “1” output when inputs A and B are equal i.e. two “1’s” or two “0’s”



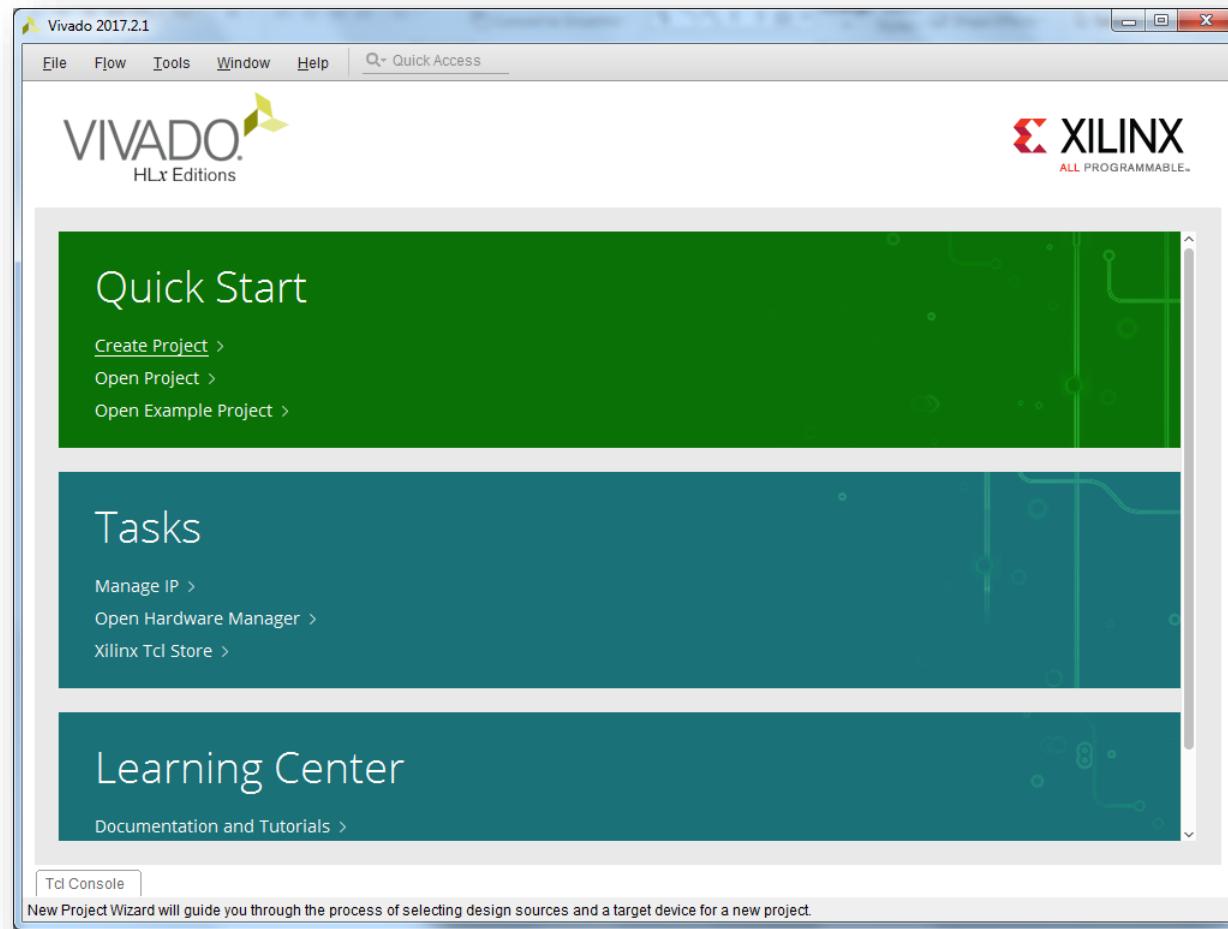
A	B	EQ
0	0	1
0	1	0
1	0	0
1	1	1

# YouTube – Look at old work flow (ver. 2016.2)

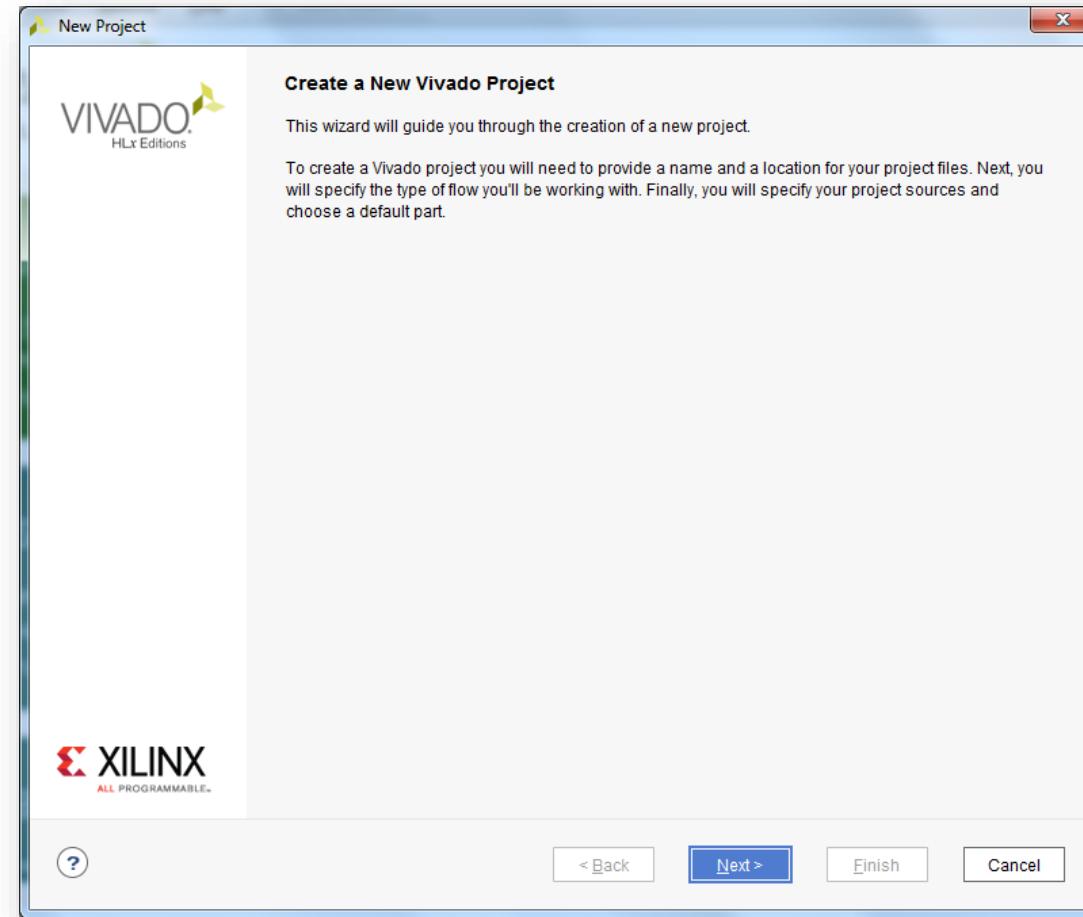
- Take a look at the following video to get oriented on the work flow
- <http://www.youtube.com/watch?v=ShjXQdKdxsE> (7:51)



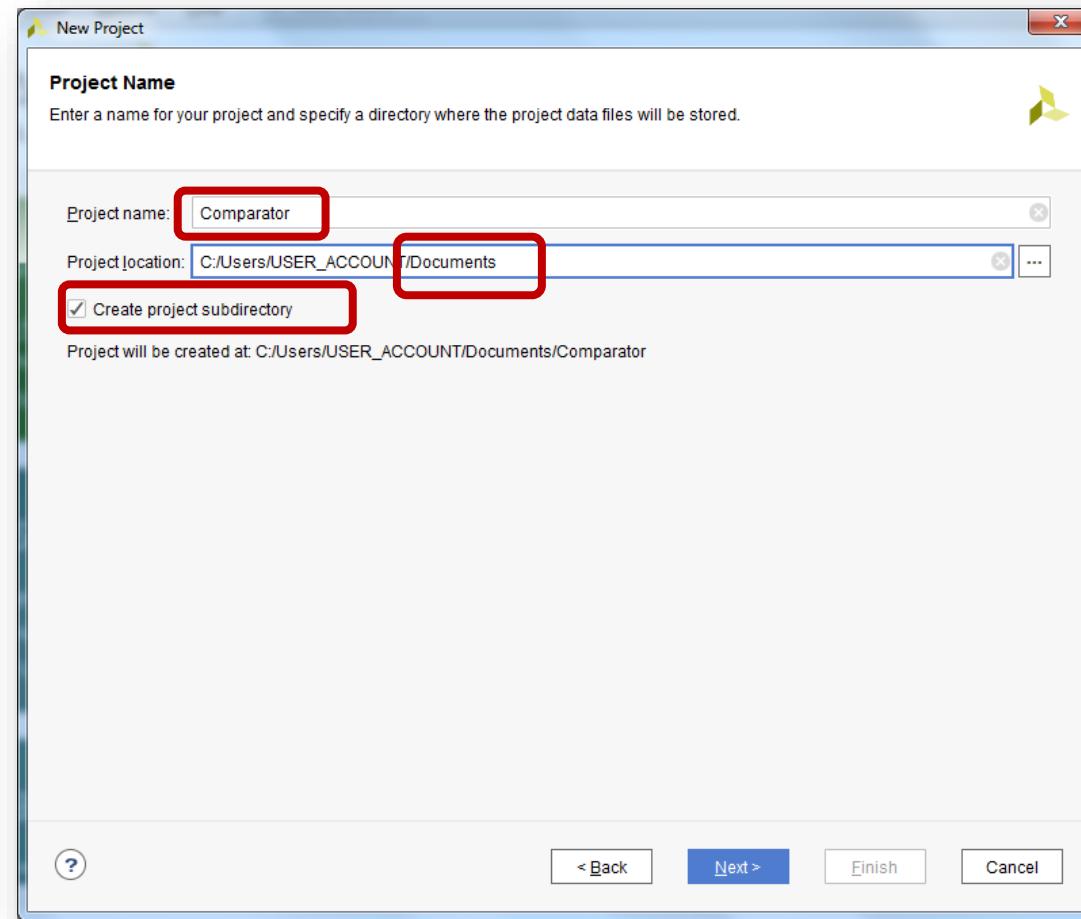
# Create a new project



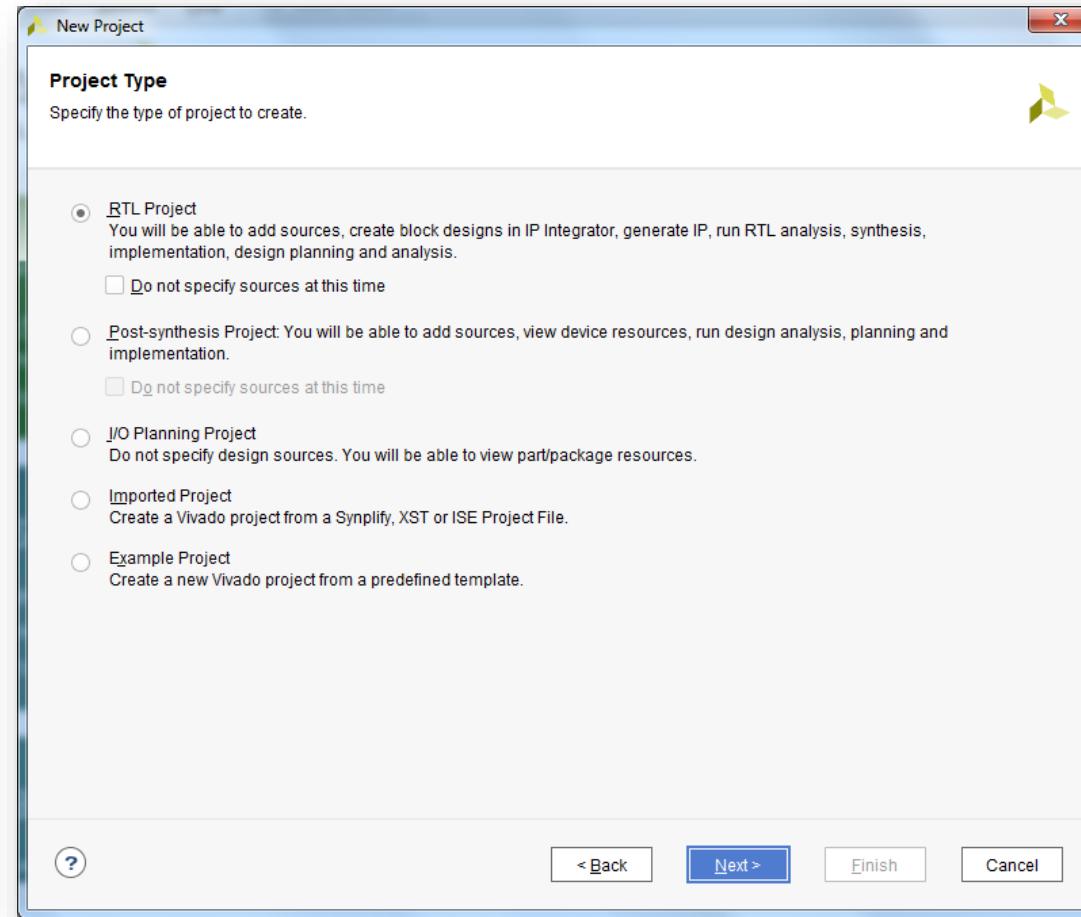
# Create a new project



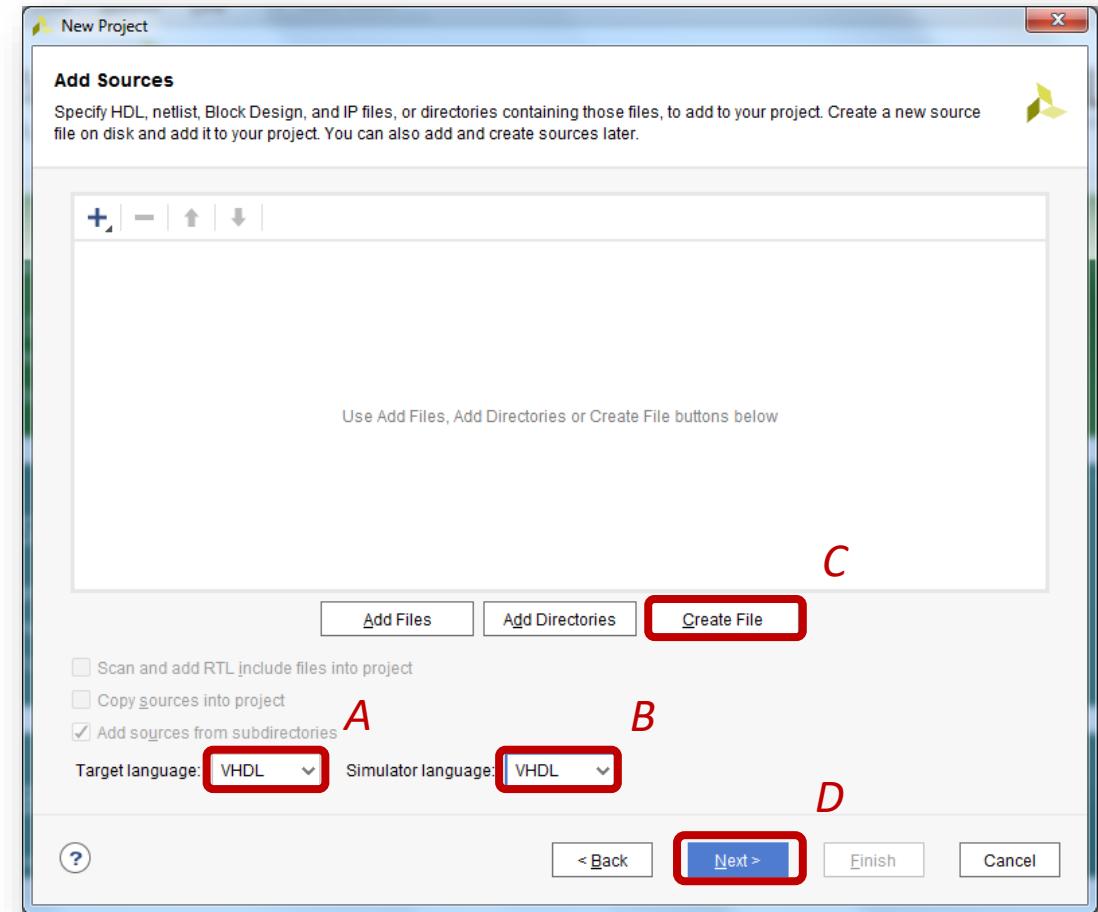
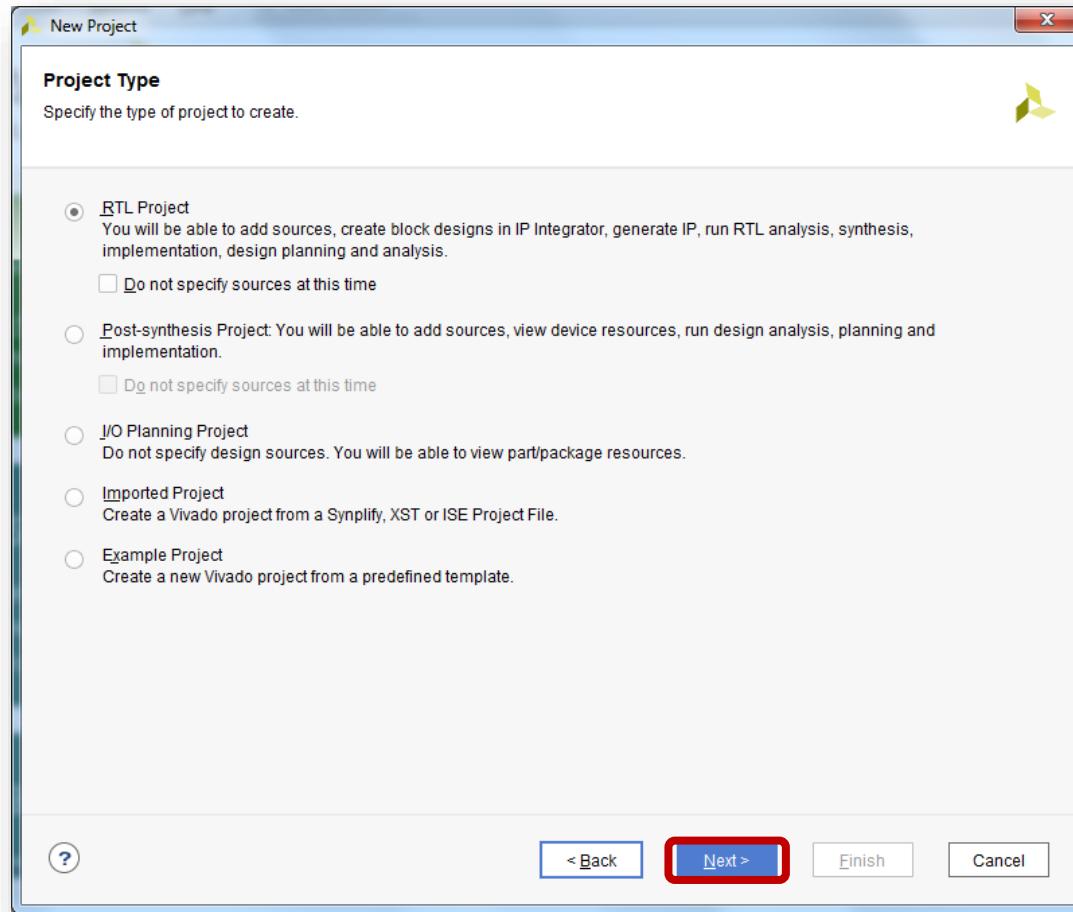
# Select a project name and location



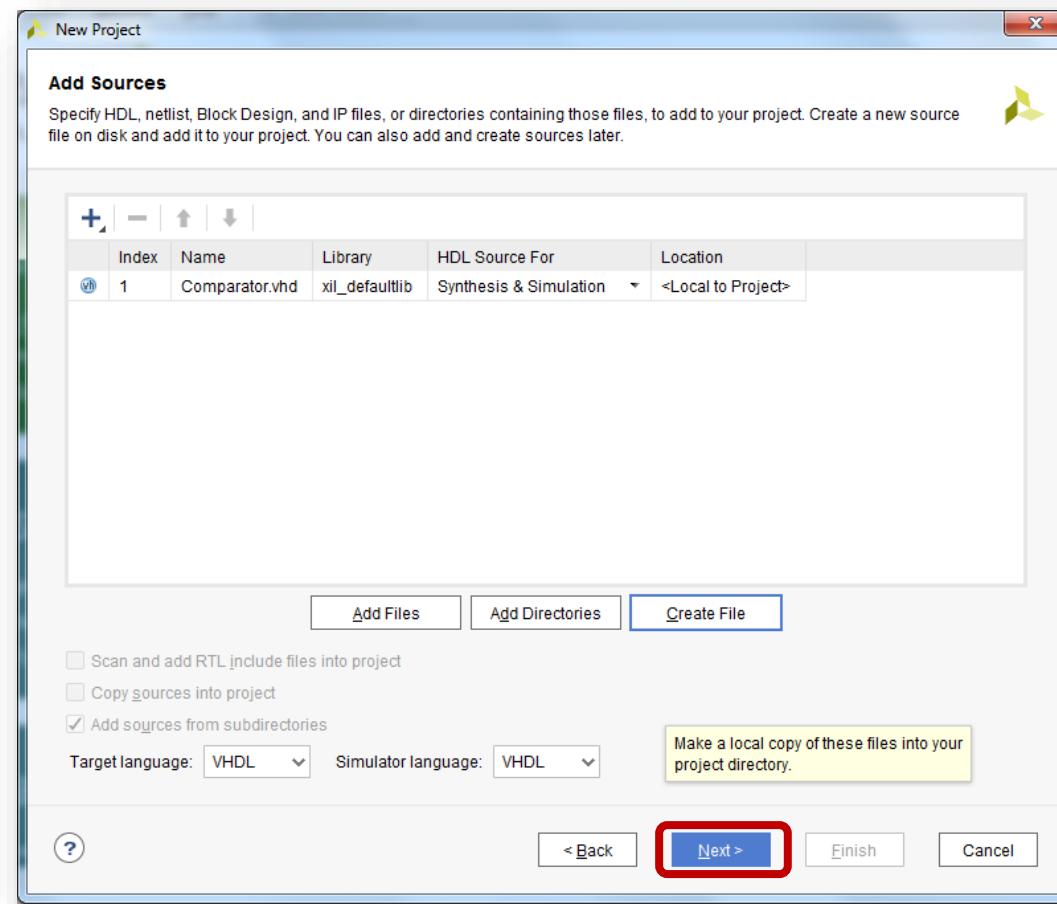
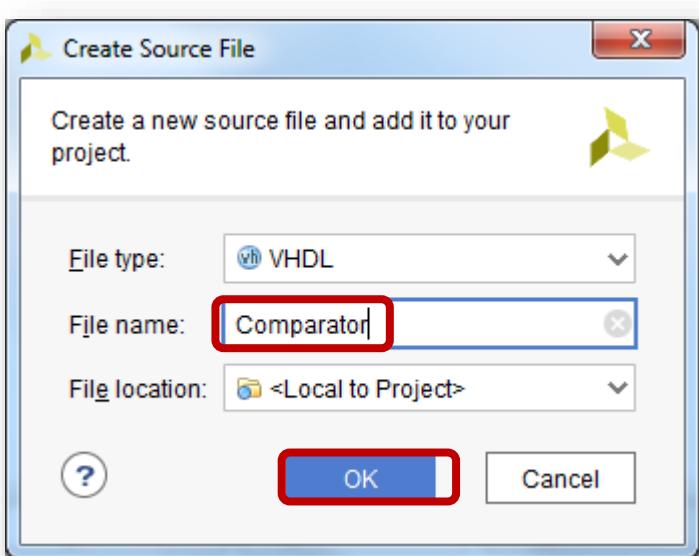
# Select Register-Transfer Level (RTL) project



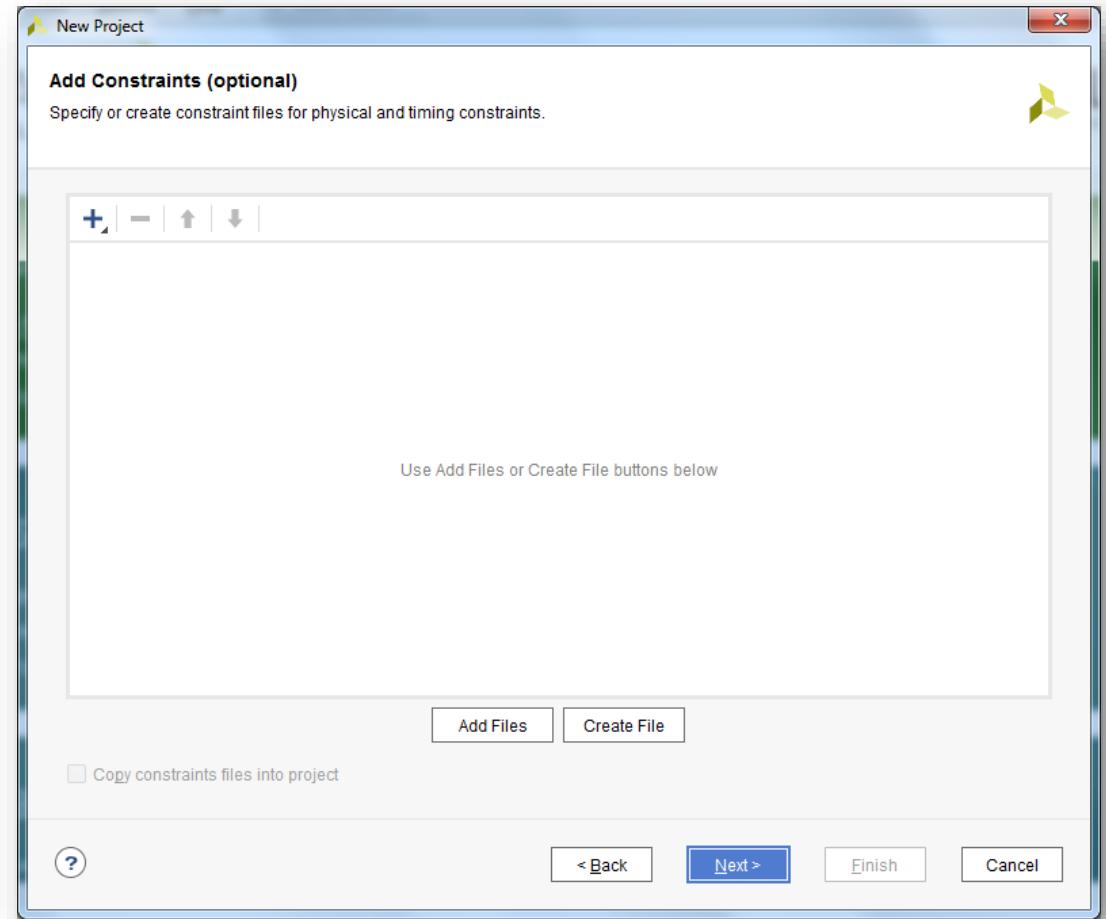
# Create a source file



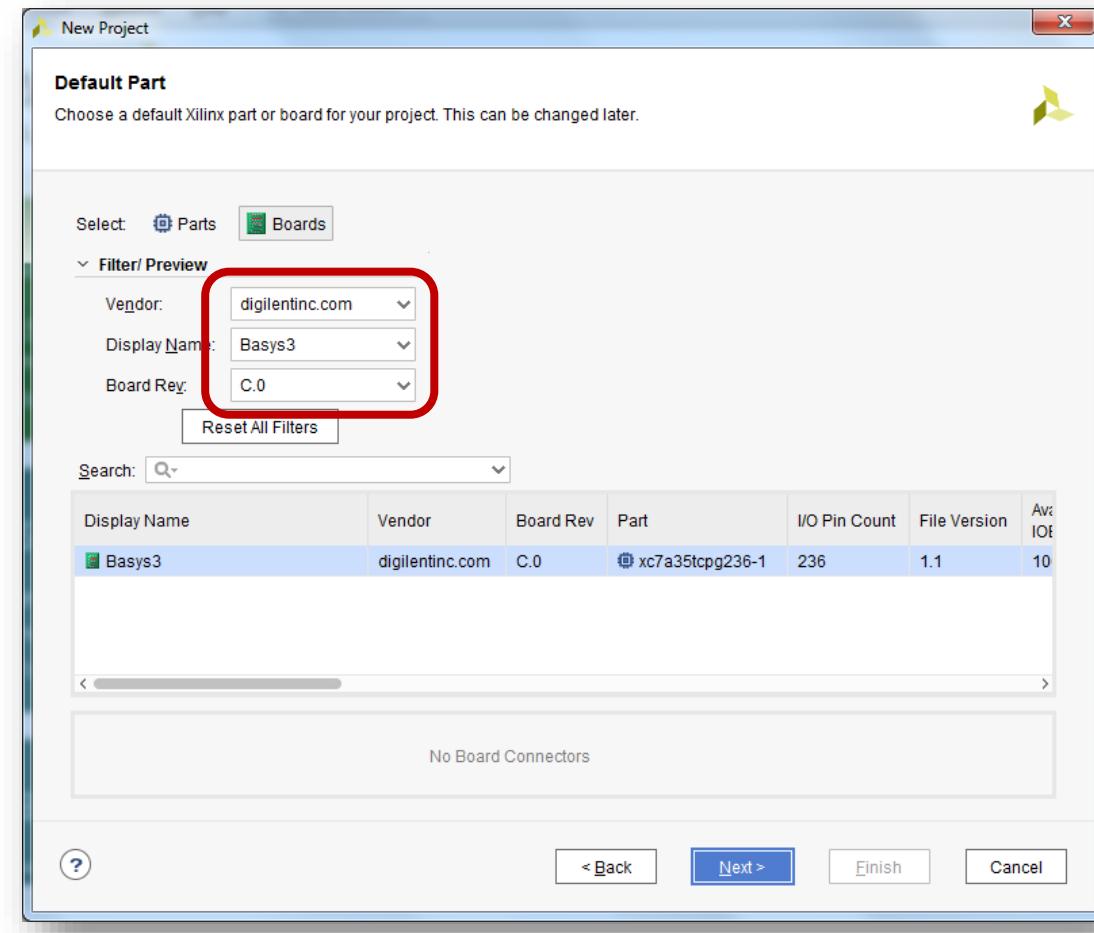
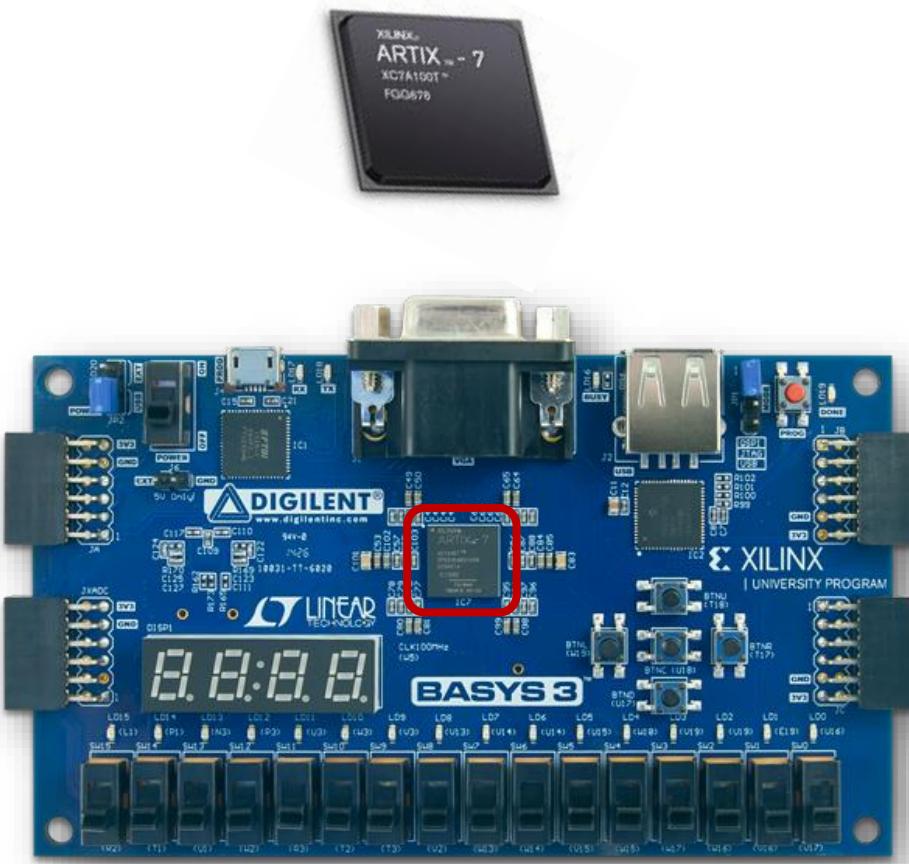
# Create a source file



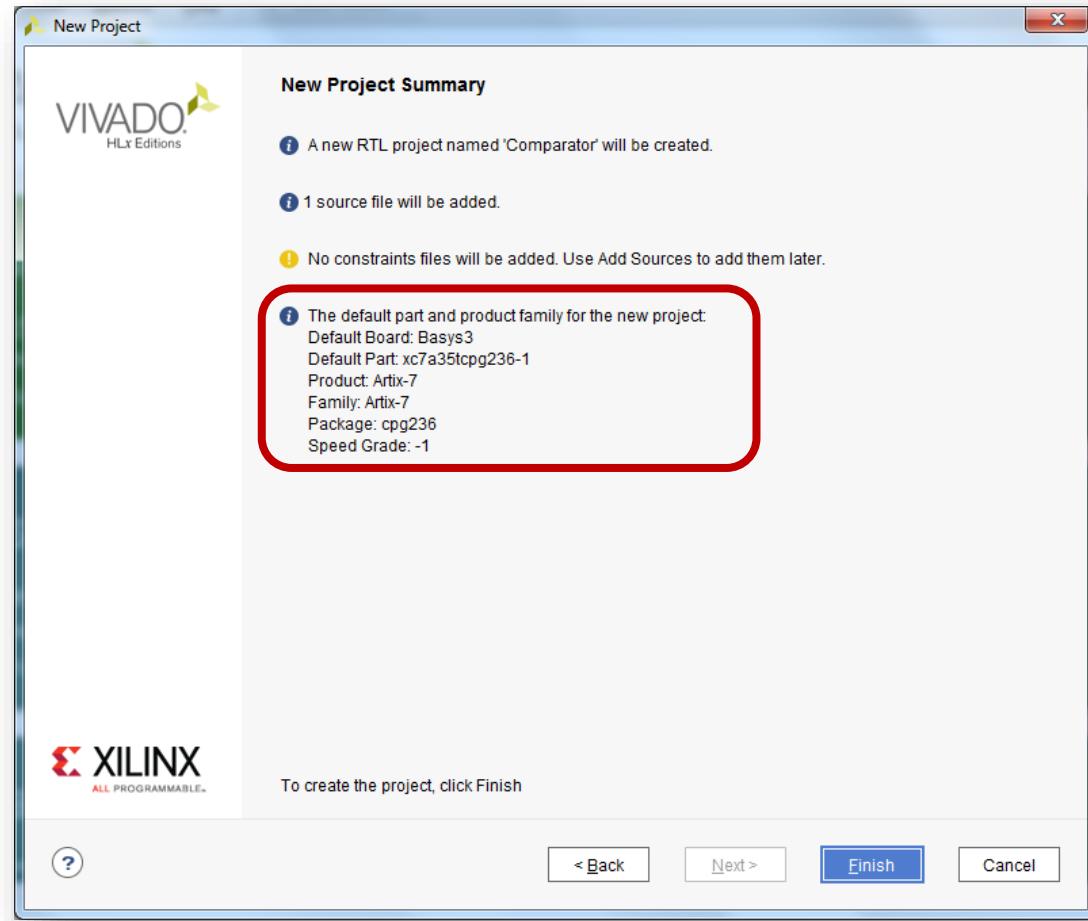
# A constraints (.cdx) file will be created later



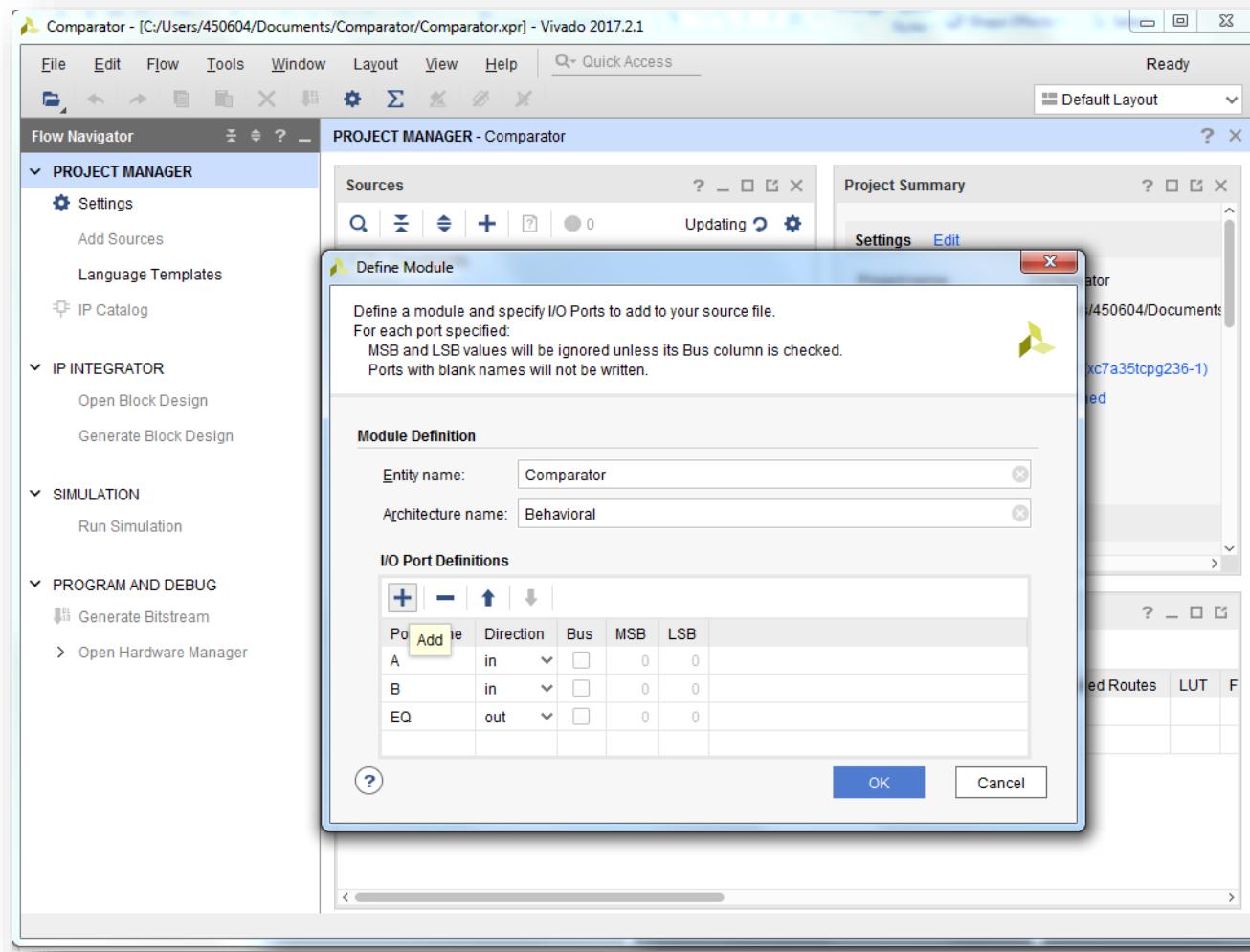
# Select the Basys3 board type



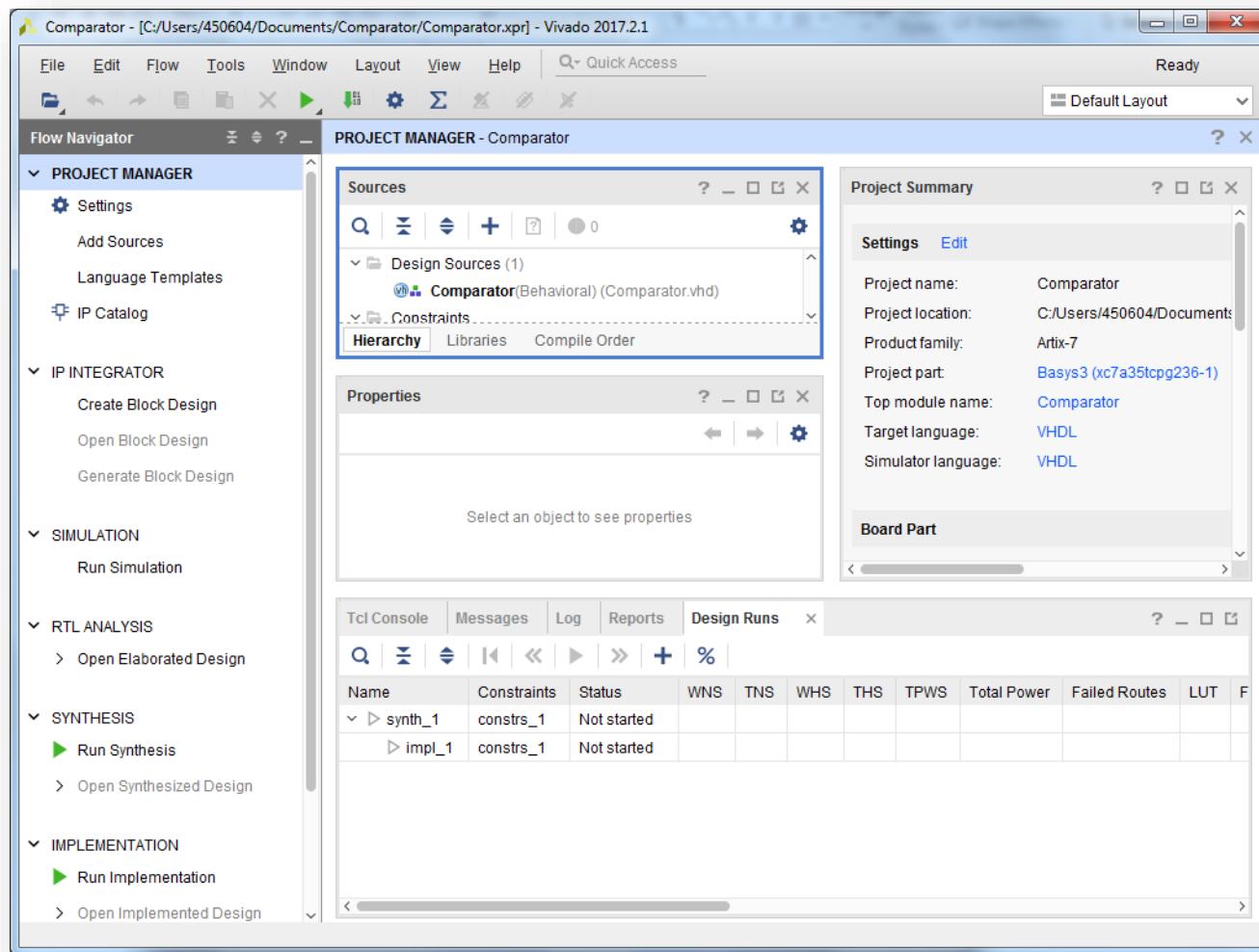
# Note project summary



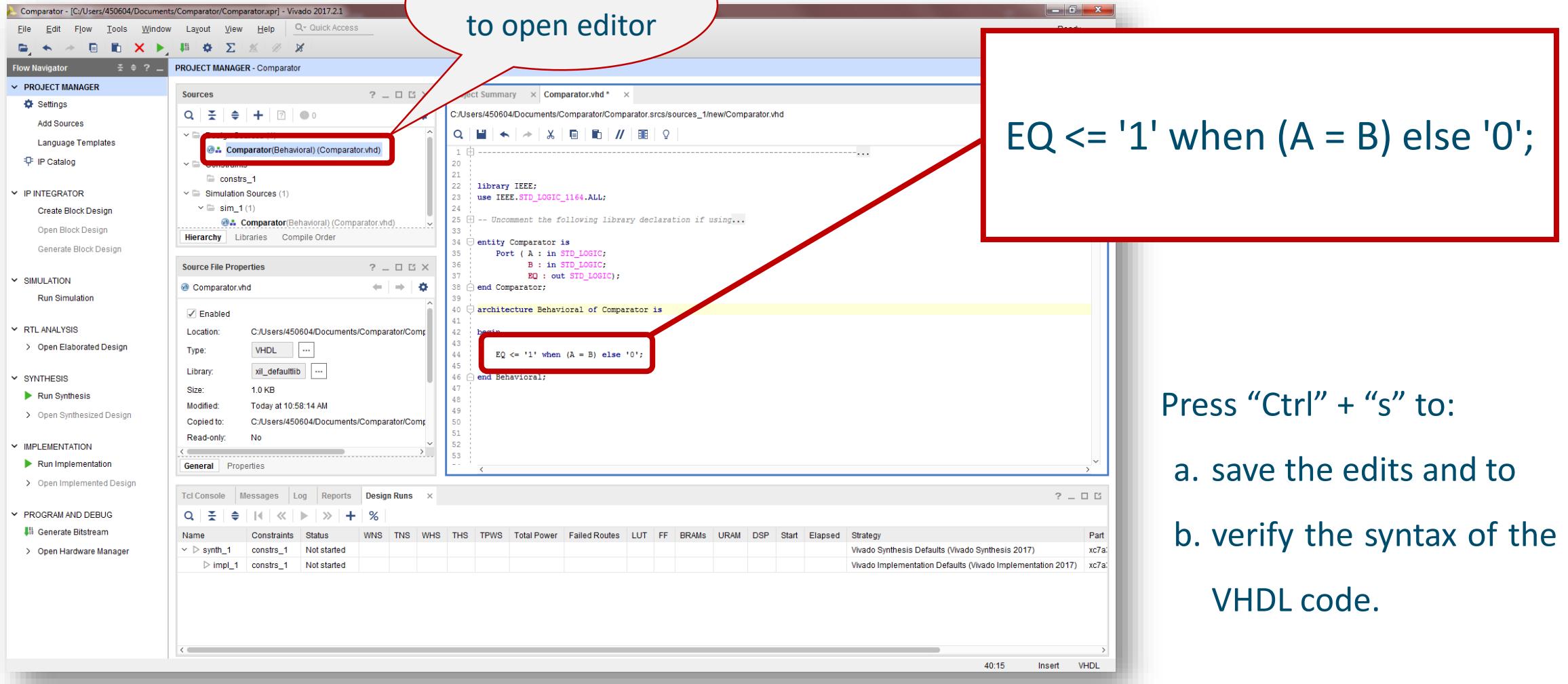
# Set the port names and directions



# Design environment



# Define comparator behaviour

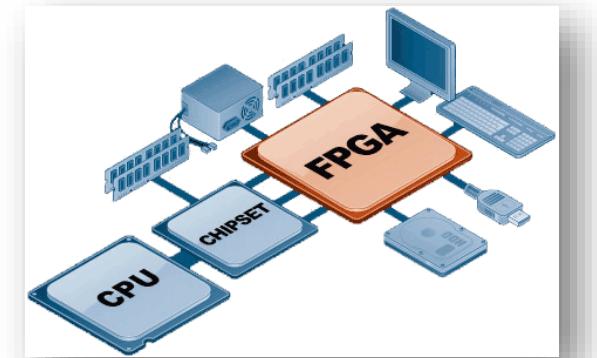
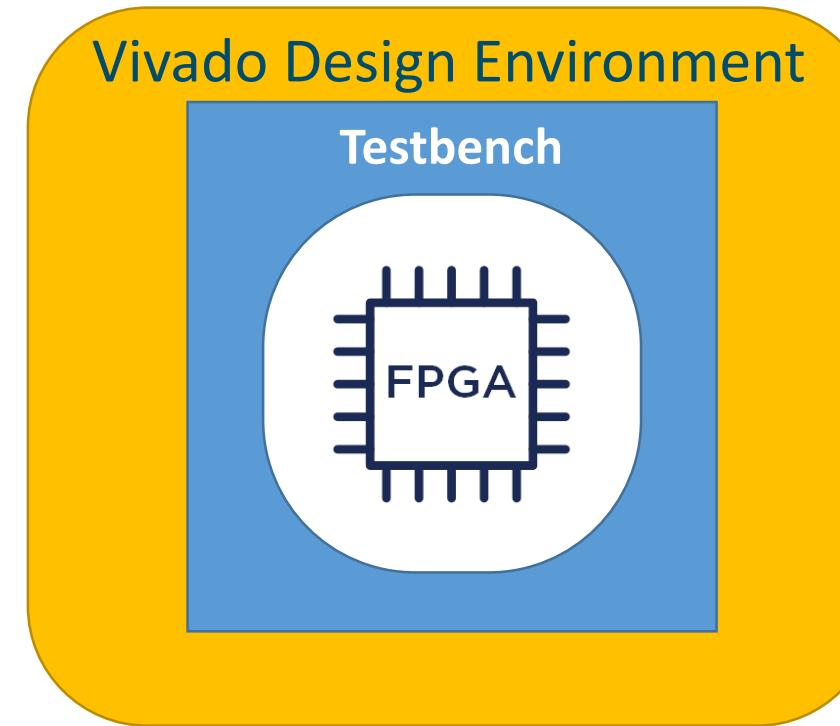


Press “Ctrl” + “s” to:

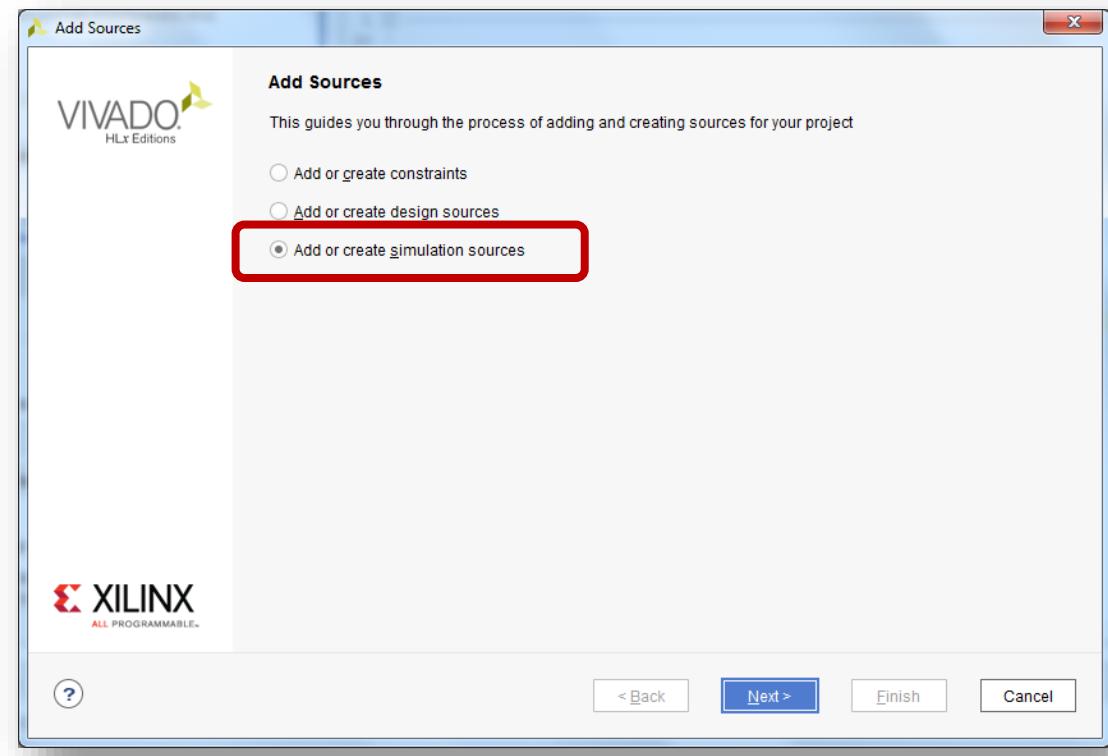
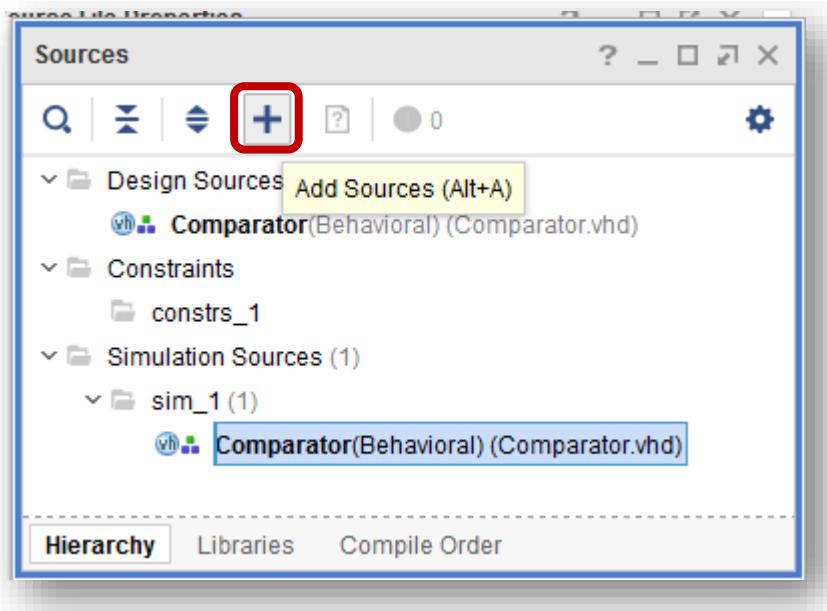
- save the edits and to
- verify the syntax of the VHDL code.

# A “Testbench”

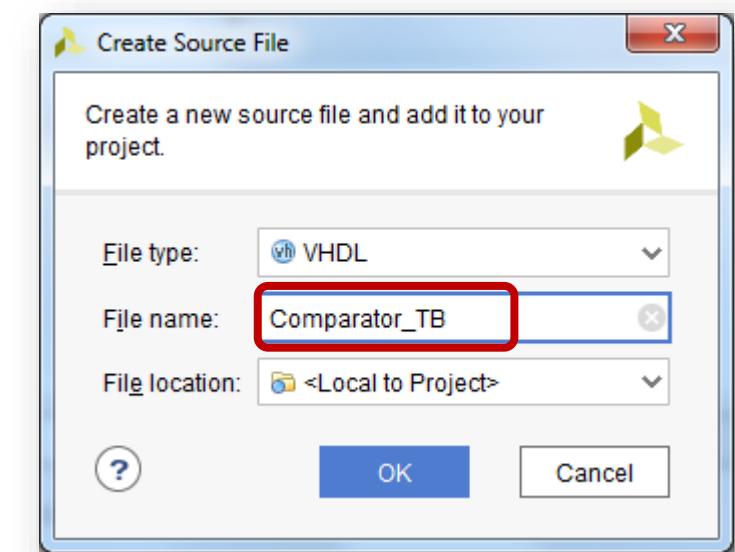
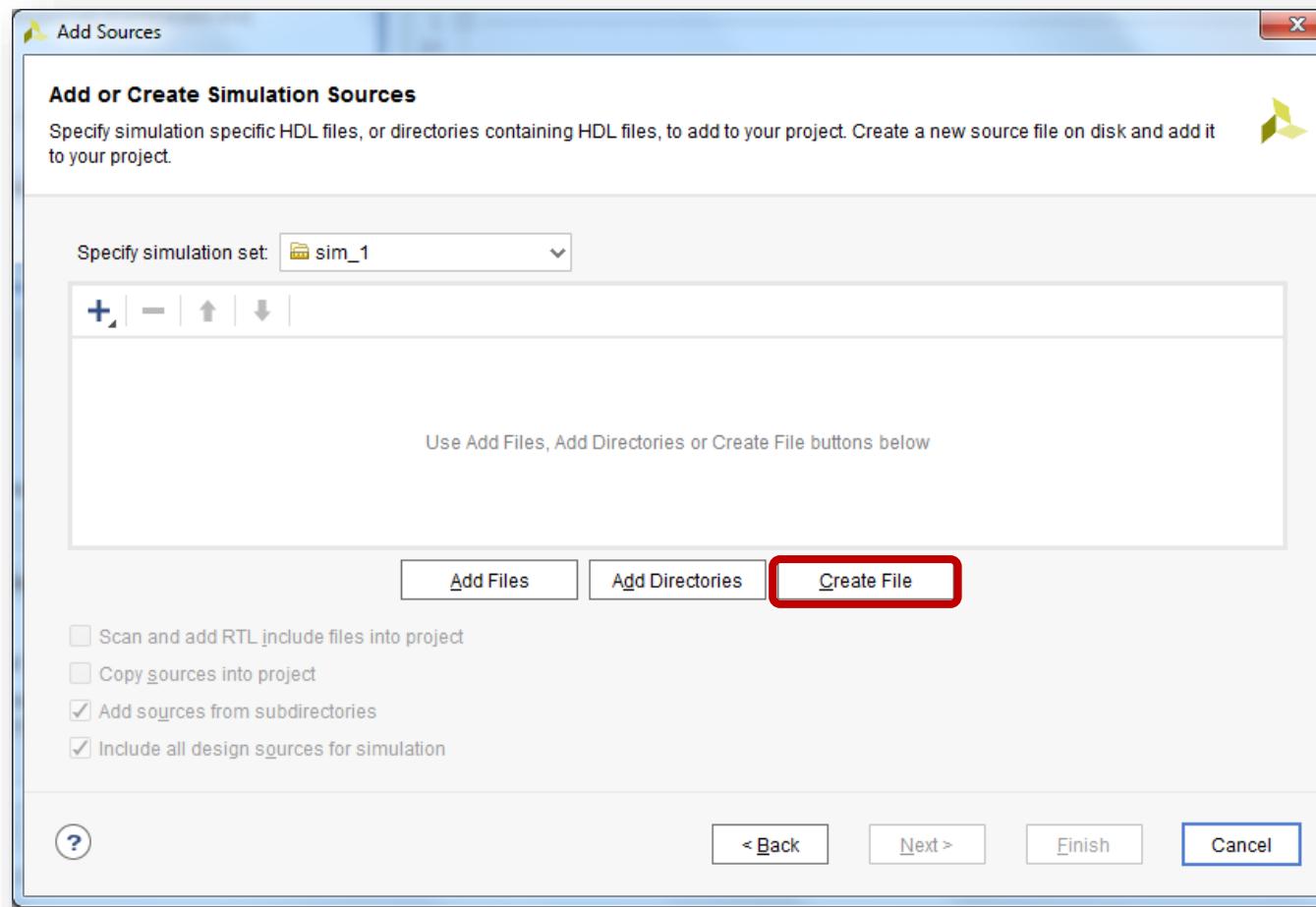
- A testbench is a construct that exists in the Vivado simulation environment
- The testbench simulation enables a unit under test (UUT) to connect to virtual components (memory, communication devices and/or CPUs) and be driven with a known set of stimuli
- This permits testing to happen before hardware is configured



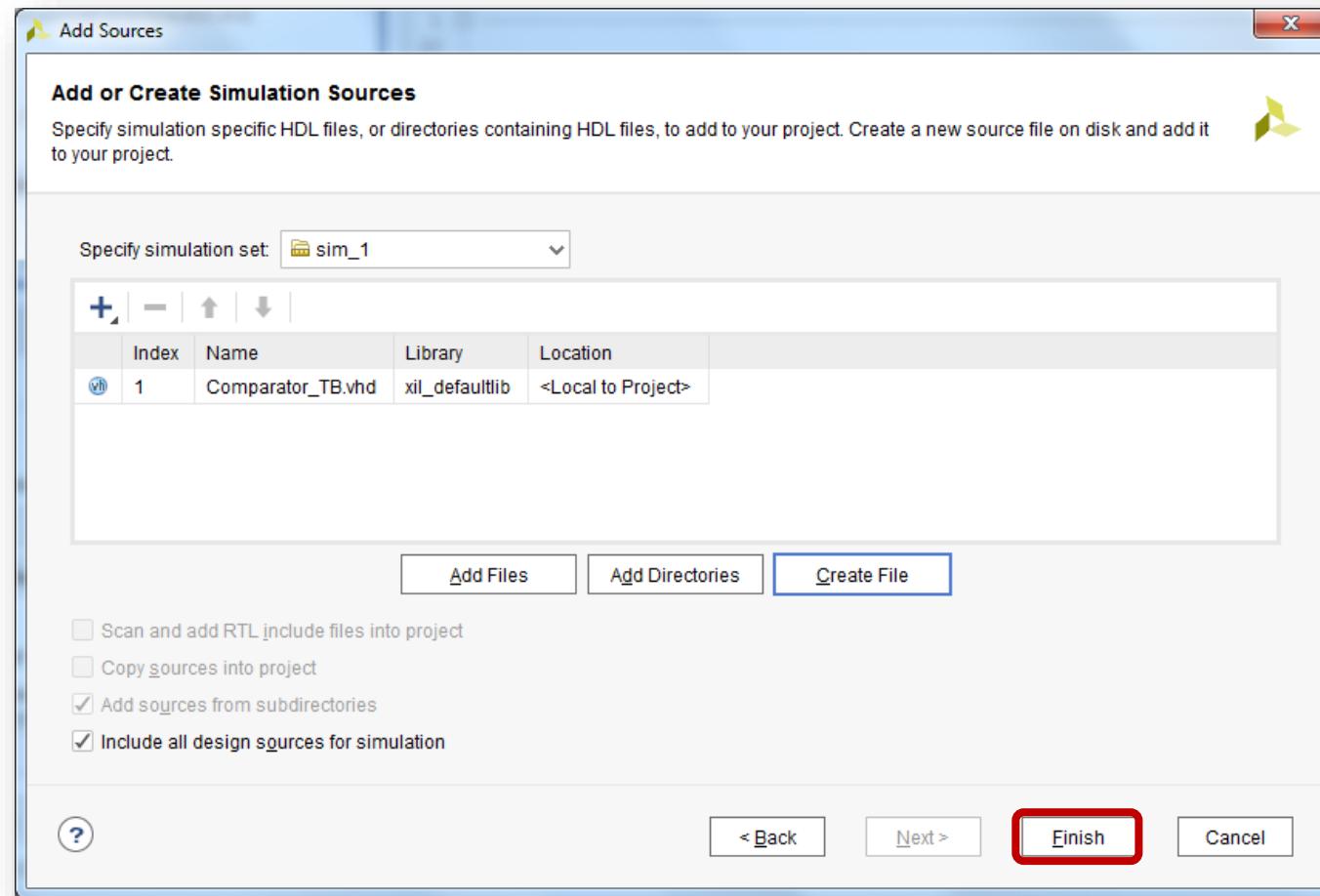
# Create a “Testbench” simulation source file



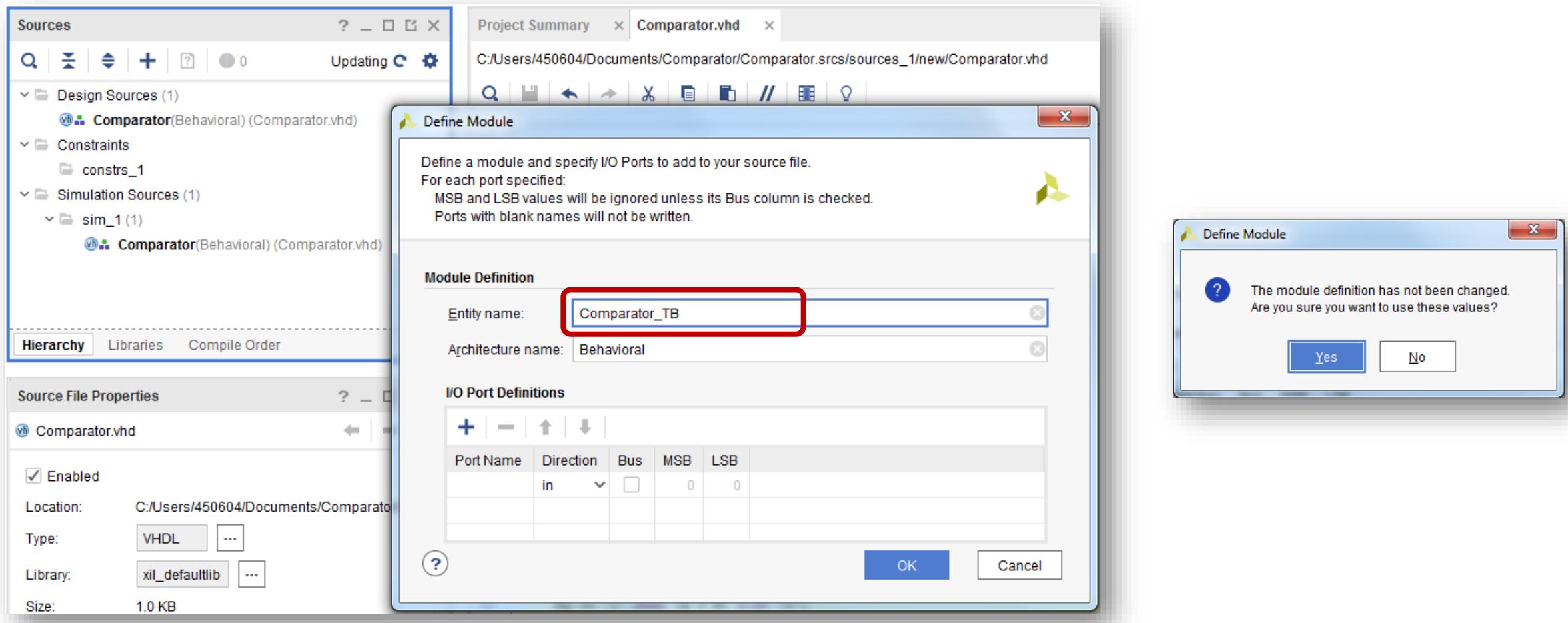
# Create Testbench source file: “Comparator\_TB”



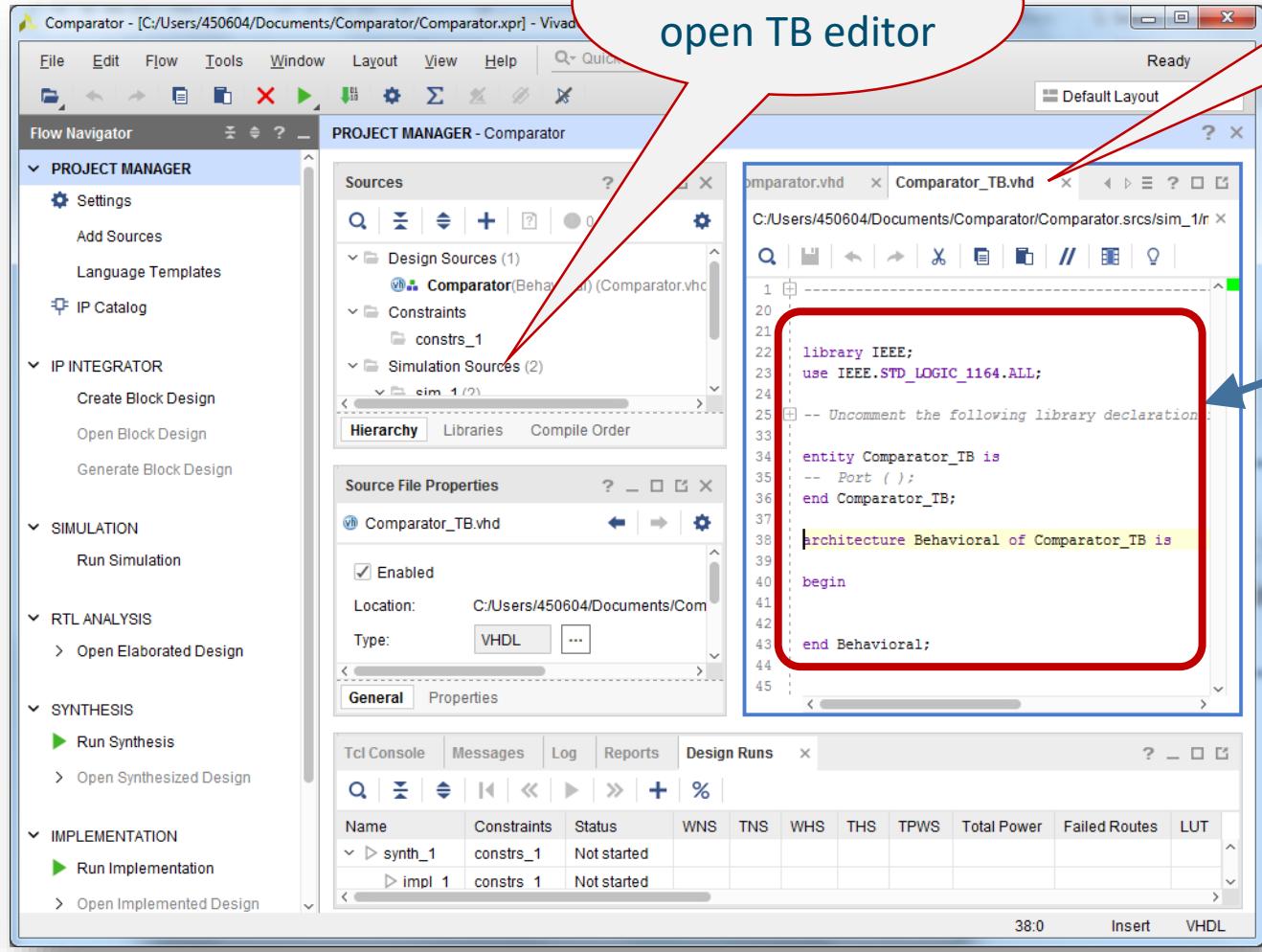
# Create Testbench source file: “Comparator\_TB”



# Set I/O Ports of the Testbench



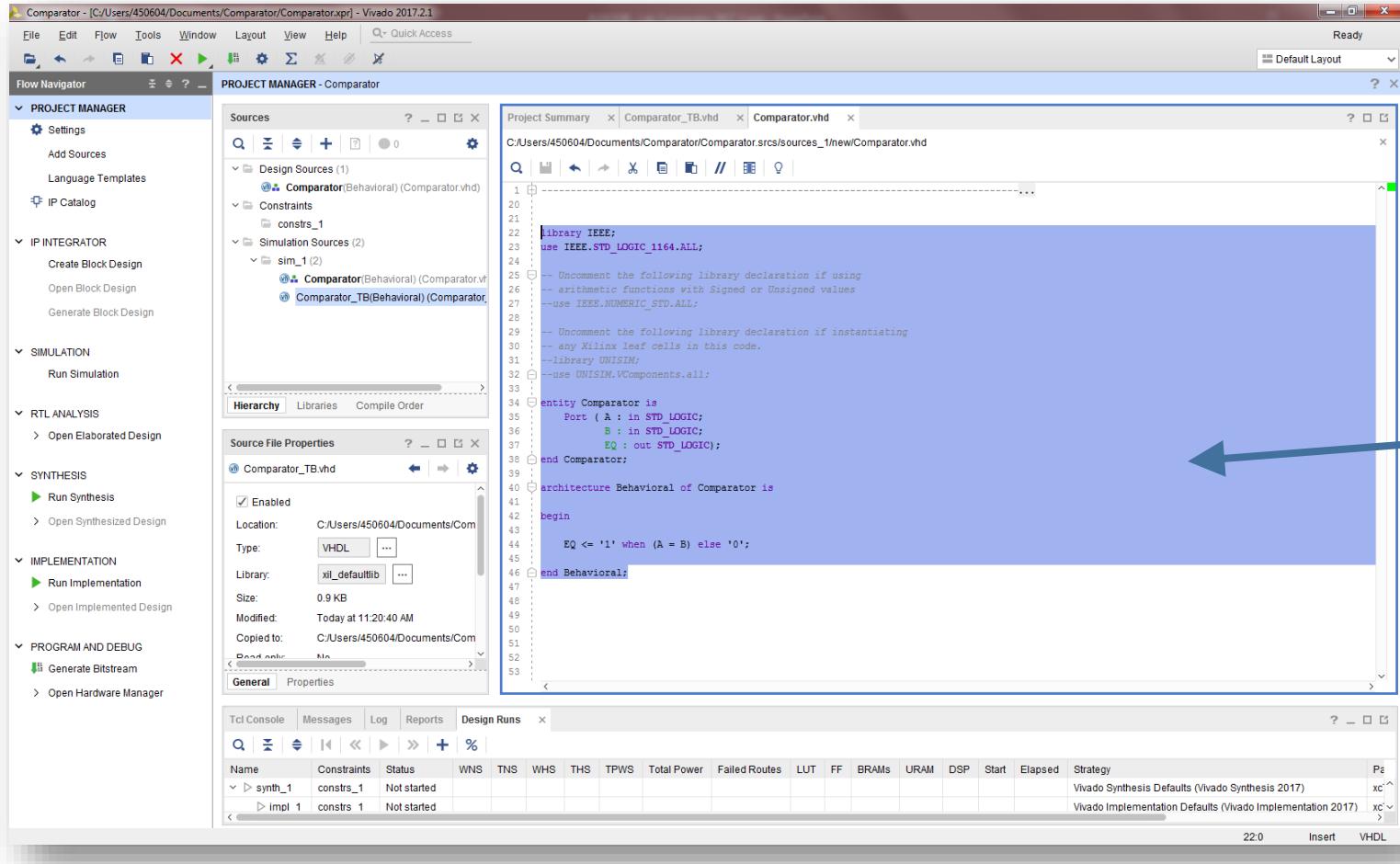
# Testbench code



Testbench  
editor

Use following steps to **replace**  
this VHDL simulation code

# Copy design source code to clipboard



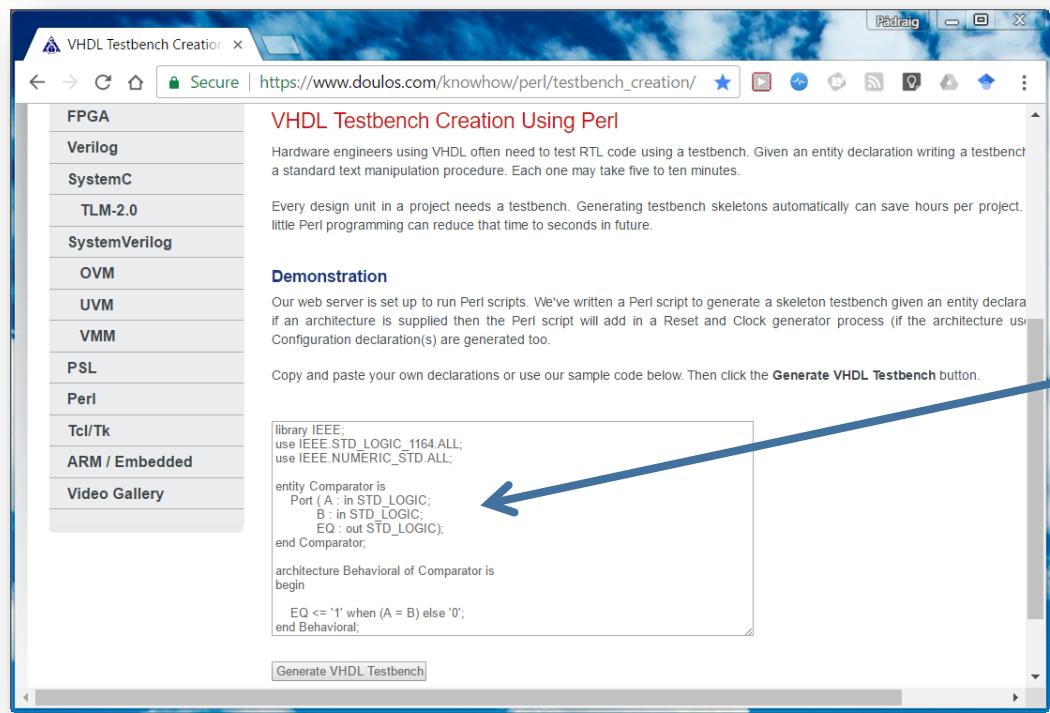
Copy design source code from Comparator.vhd file

Note: This is not the testbench editor

# Generate Testbench VHDL code

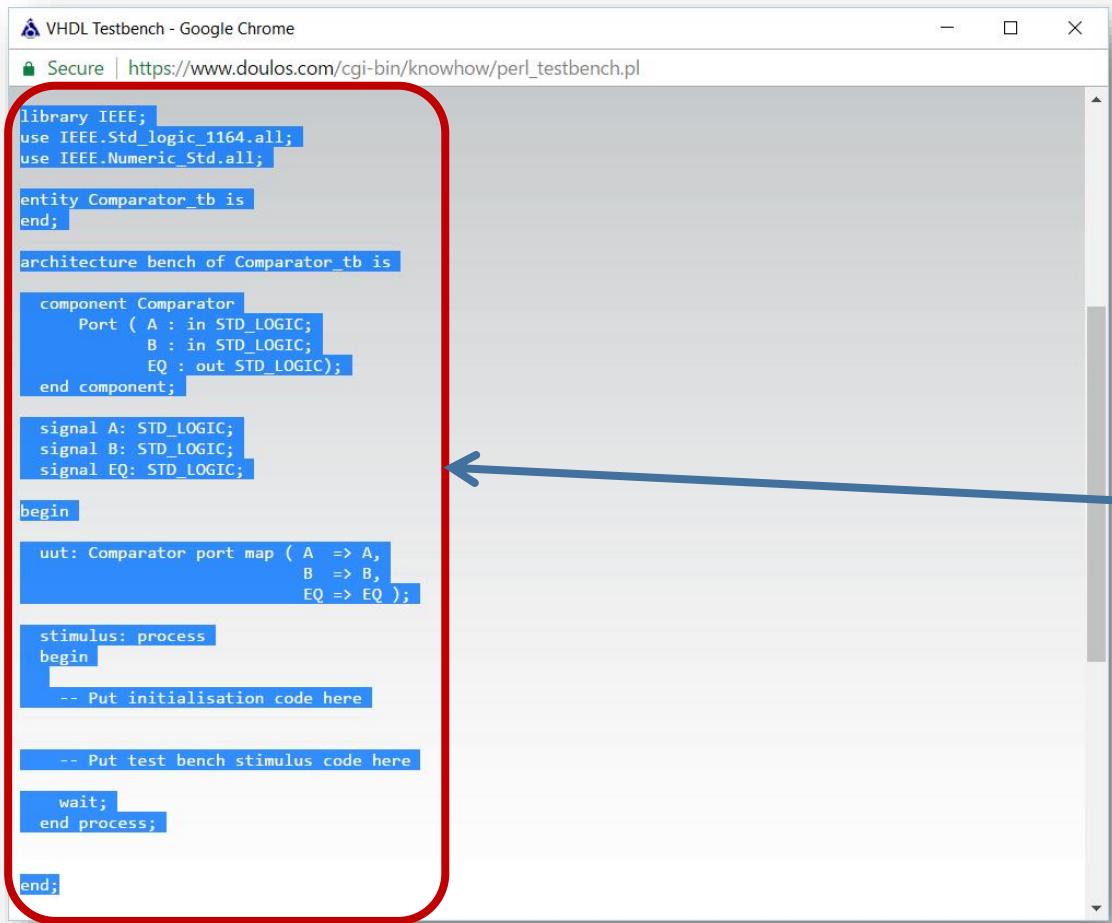
Use online VHDL Testbench Editor to generate code

- [https://www.doulos.com/knowhow/perl/Testbench\\_creation/](https://www.doulos.com/knowhow/perl/Testbench_creation/)



Paste the **design source code**  
(Comparator.vhd) into the online editor.  
Generate new Testbench simulator code  
for testing the design source code.

# Copy Testbench VHDL code



```
library IEEE;
use IEEE.Std_logic_1164.all;
use IEEE.Numeric_Signed.all;

entity Comparator_tb is
end;

architecture bench of Comparator_tb is

component Comparator
    Port ( A : in STD_LOGIC;
           B : in STD_LOGIC;
           EQ : out STD_LOGIC);
end component;

signal A: STD_LOGIC;
signal B: STD_LOGIC;
signal EQ: STD_LOGIC;

begin

    uut: Comparator port map ( A => A,
                                B => B,
                                EQ => EQ );

    stimulus: process
    begin
        -- Put initialisation code here

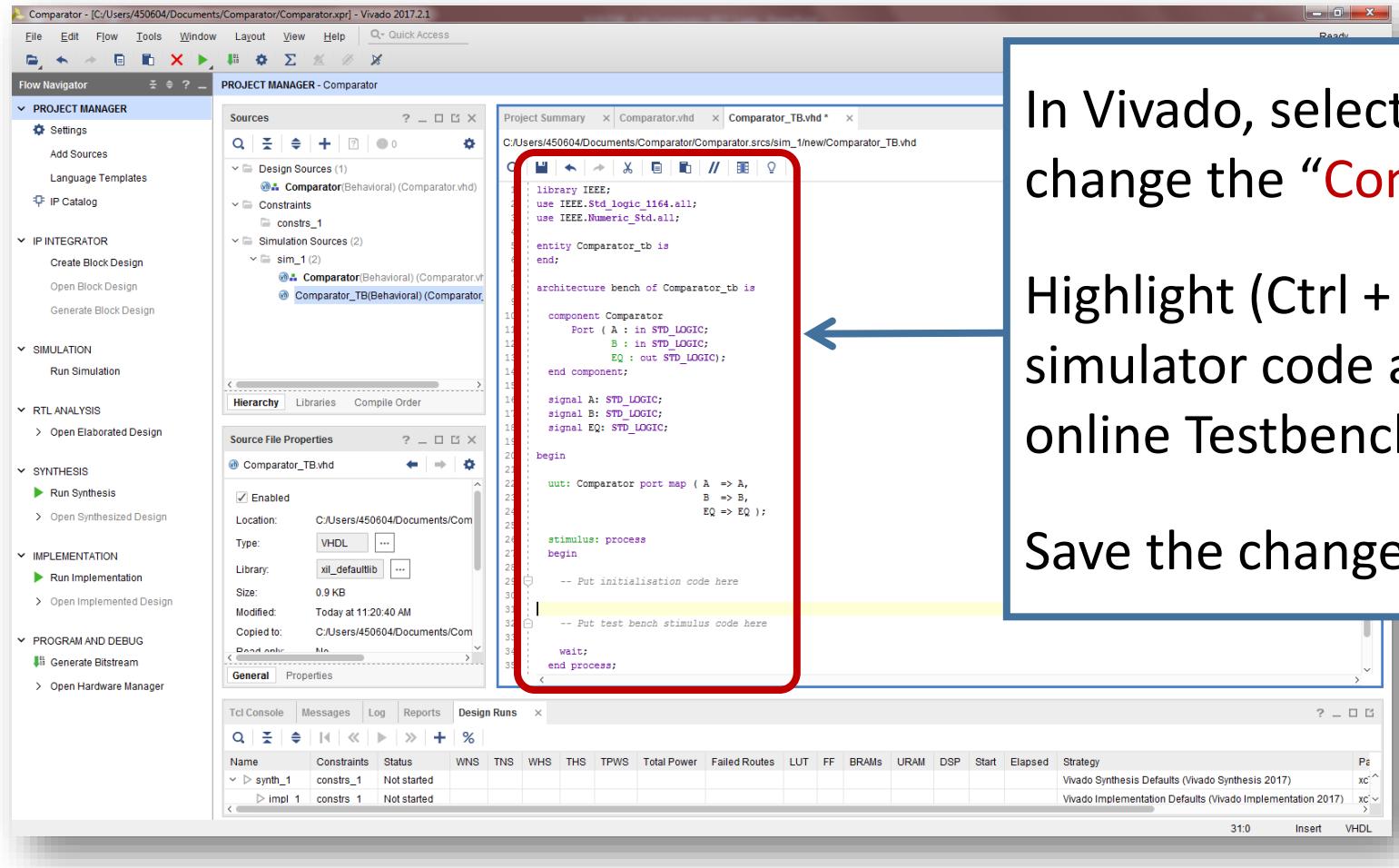
        -- Put test bench stimulus code here

        wait;
    end process;

end;
```

From the online Testbench generator,  
highlight and copy this sub-section of  
the testbench simulator code

# Paste Testbench VHDL code

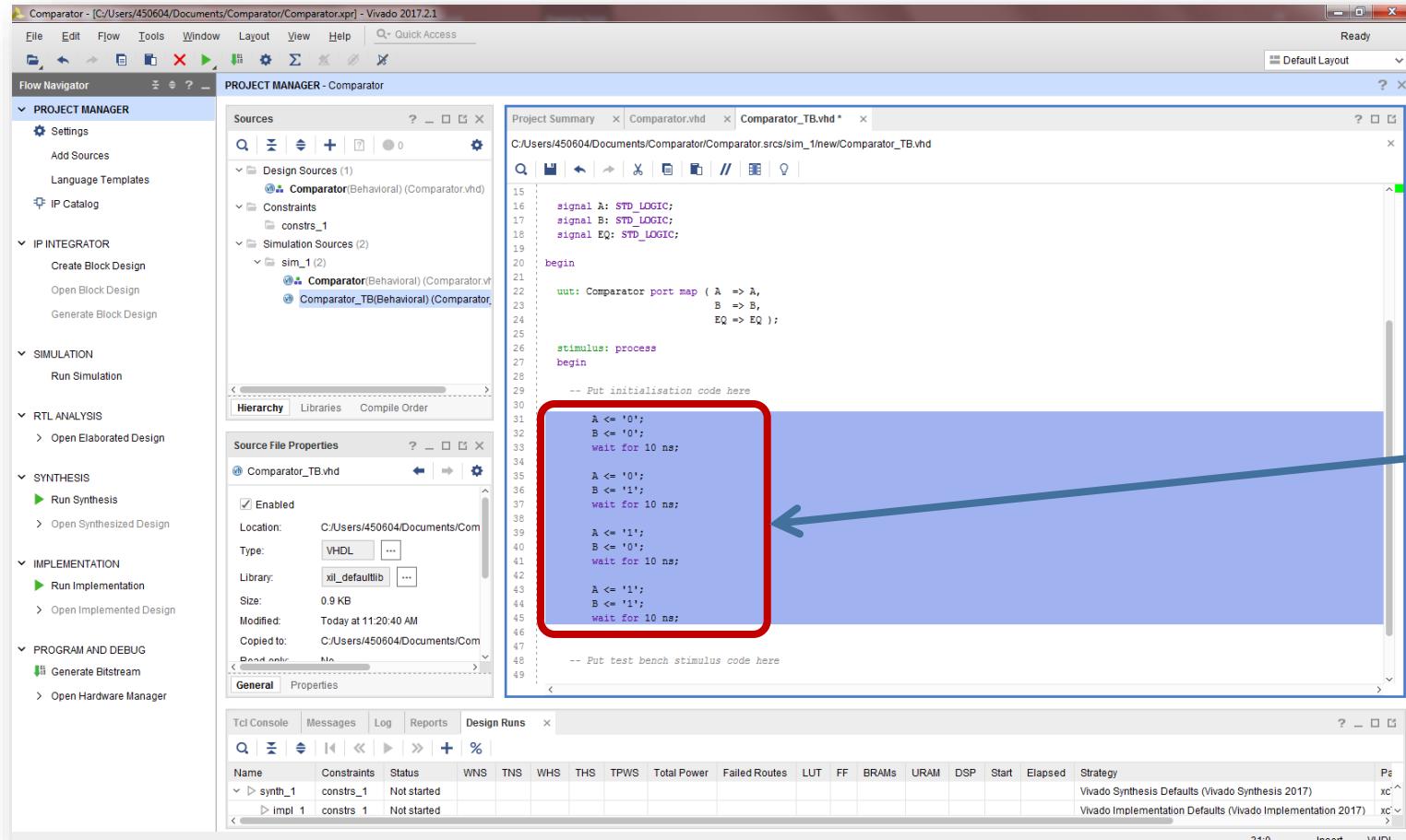


In Vivado, select the Testbench editor to change the “**Comparator\_TB.vhd**” file

Highlight (Ctrl + “a”) the initial VHDL simulator code and paste (Ctrl + “v”) the online Testbench generator code

Save the changes

# Edit Testbench VHDL code



Add the stimulus code into the **Comparator\_TB.vhd** file:

A <= '0';  
B <= '0';  
wait for 10 ns;

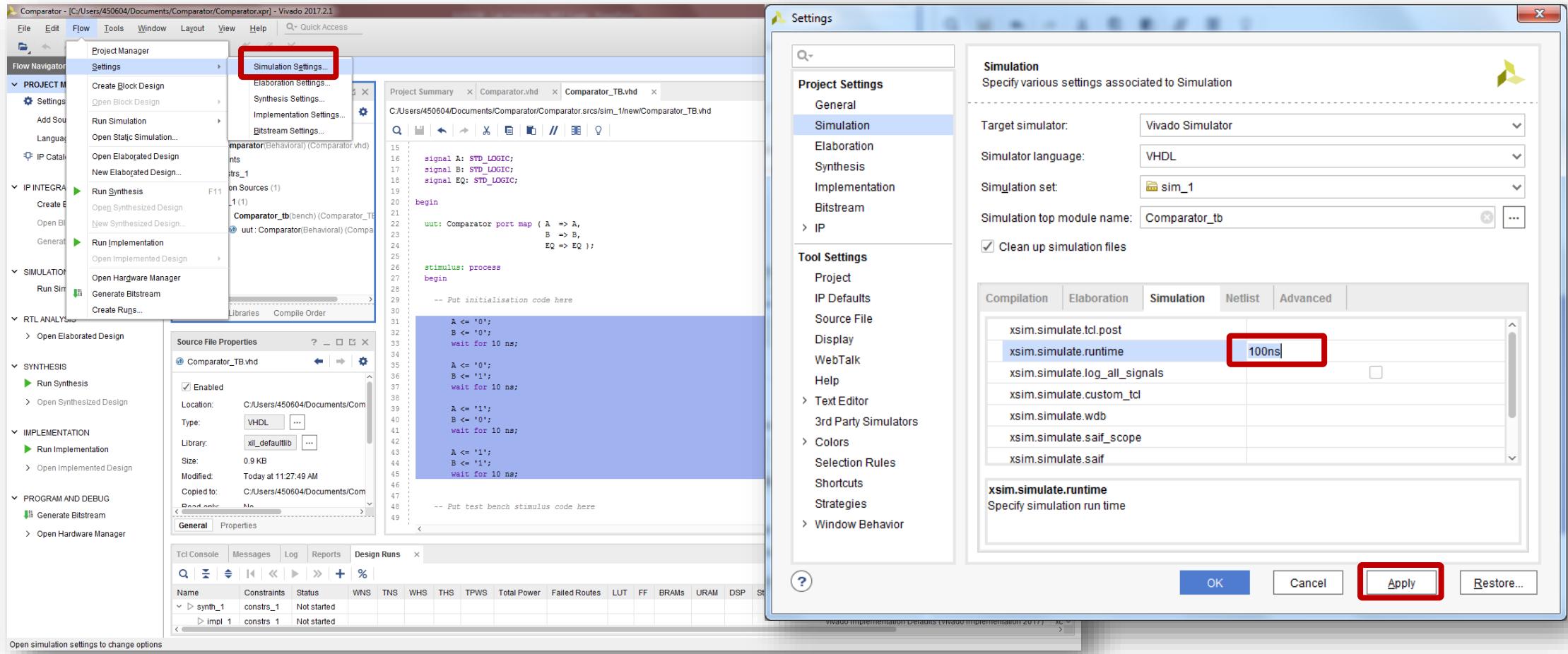
A <= '0';  
B <= '1';  
wait for 10 ns;

A <= '1';  
B <= '0';  
wait for 10 ns;

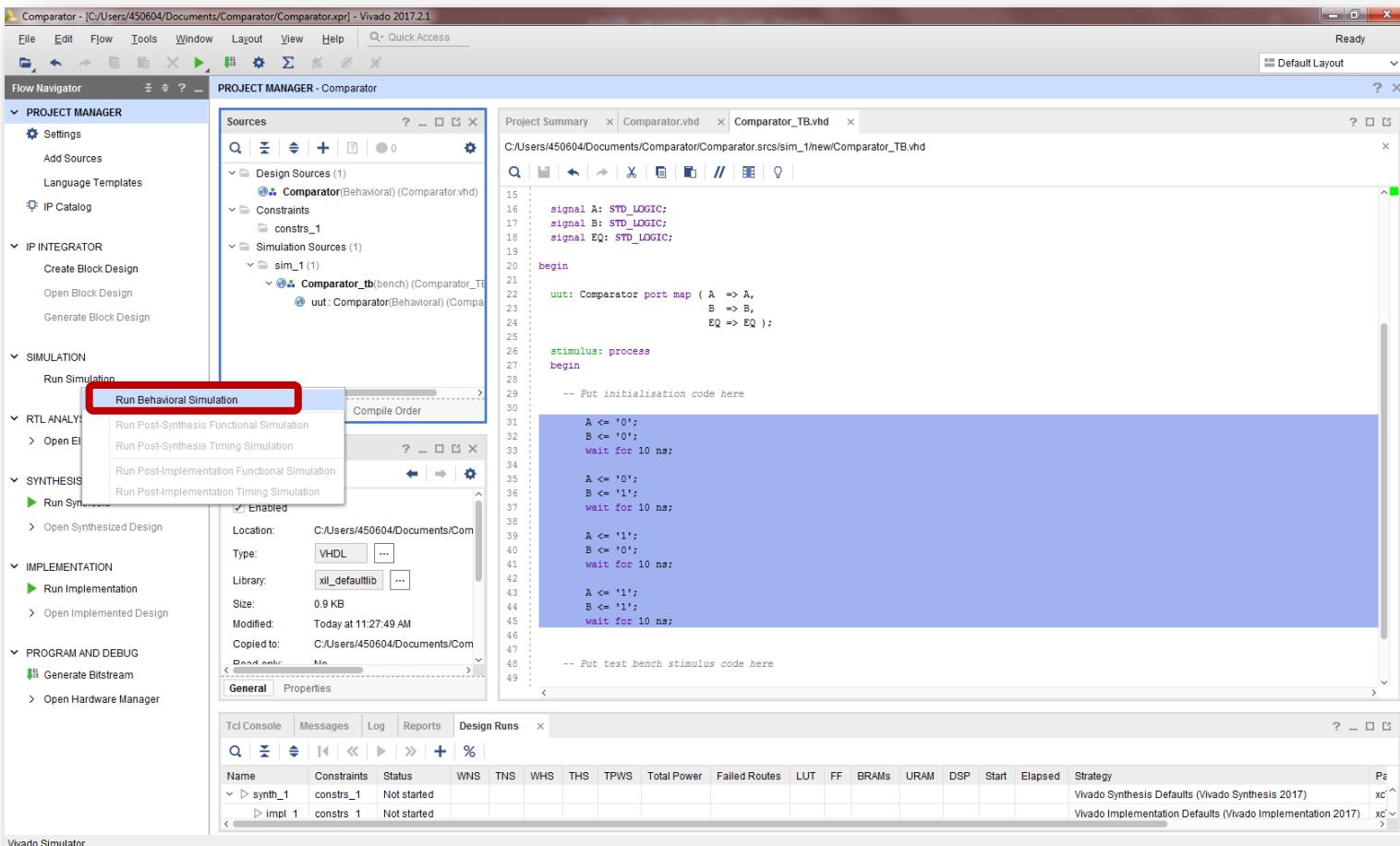
A <= '1';  
B <= '1';  
wait for 10 ns;

Remember to save the changes

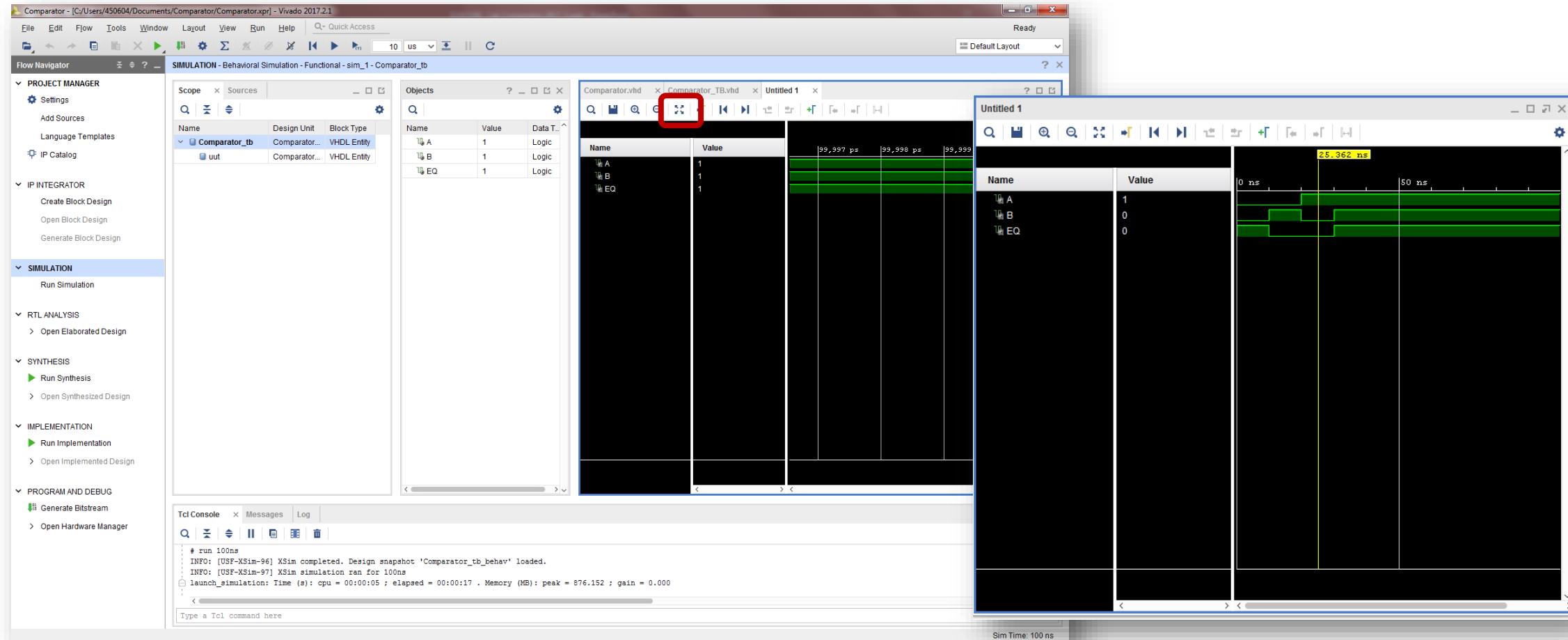
# Simulation settings: runtime 100 ns



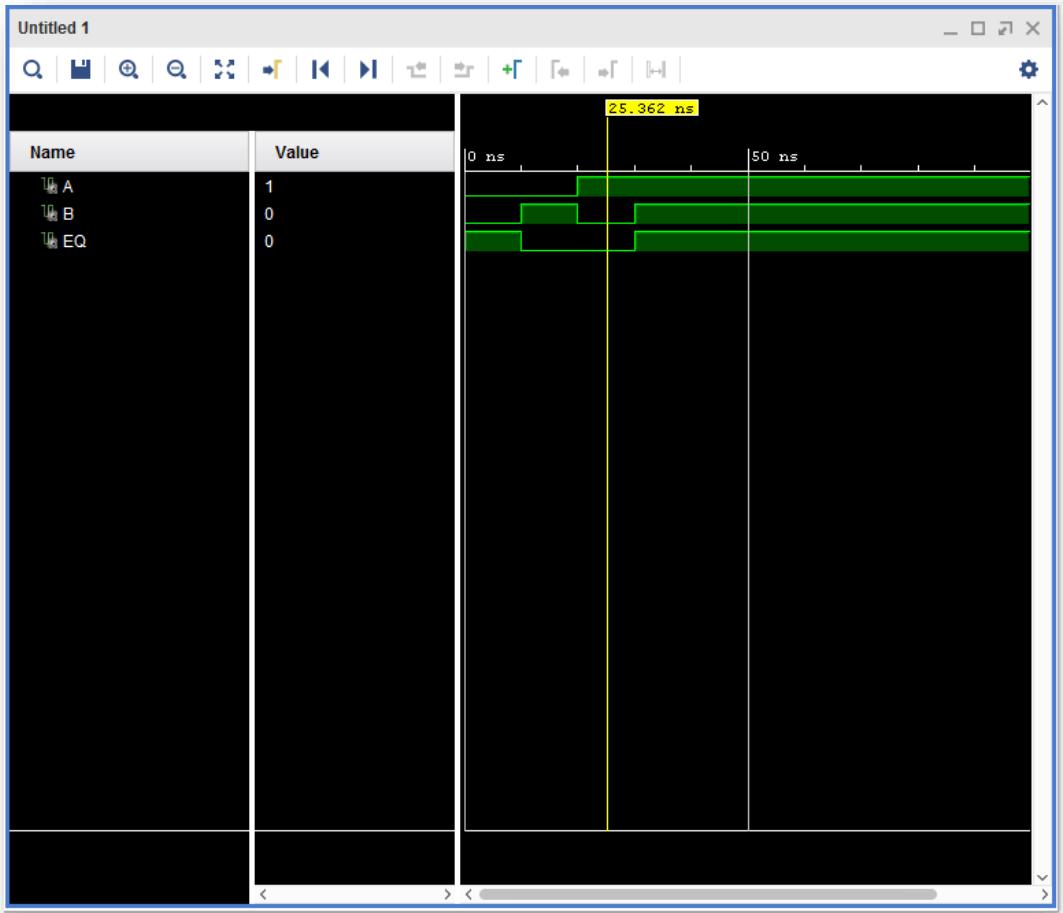
# Run behavioural simulation



# Zoom-fit the simulation results

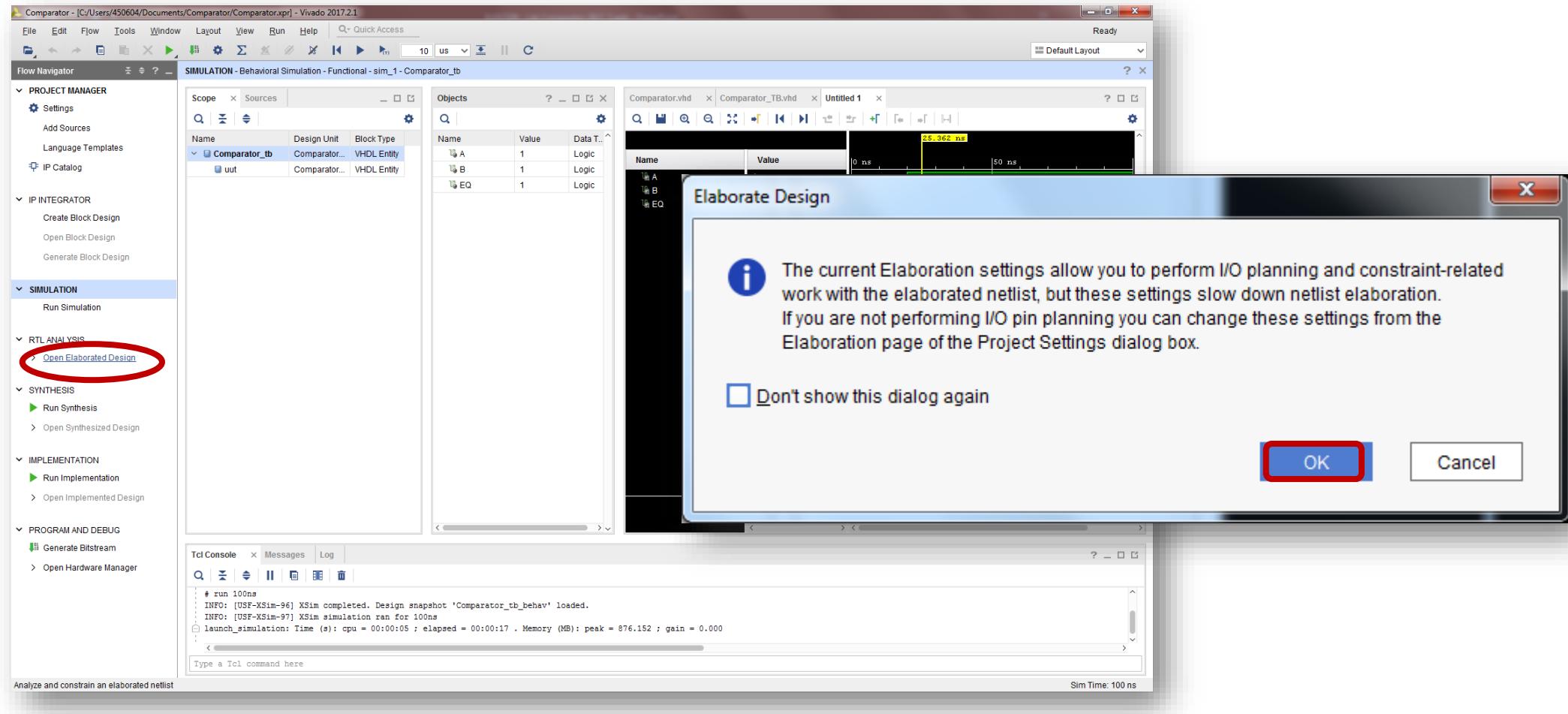


# Check the simulation with a marker

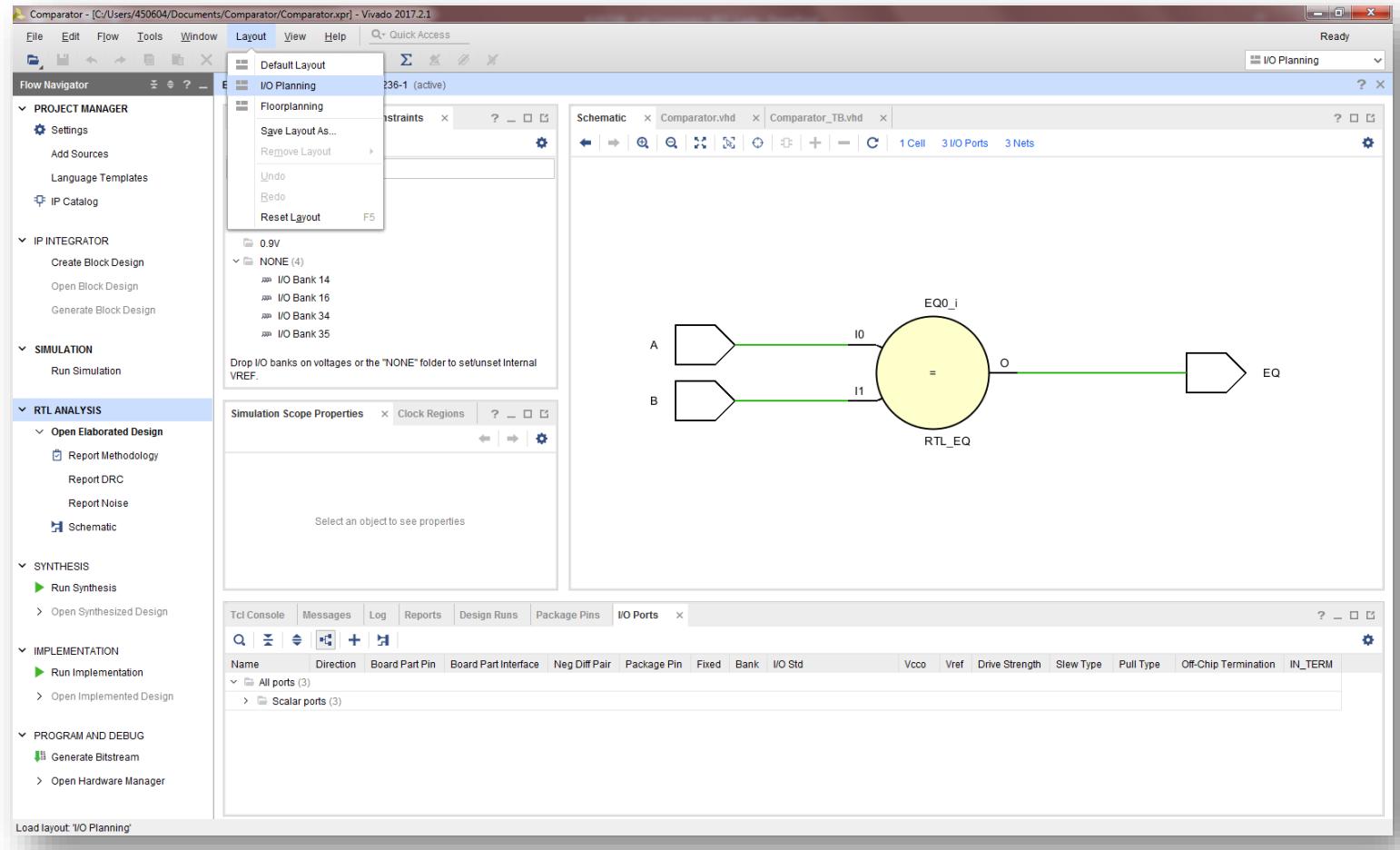


A	B	EQ
0	0	1
0	1	0
1	0	0
1	1	1

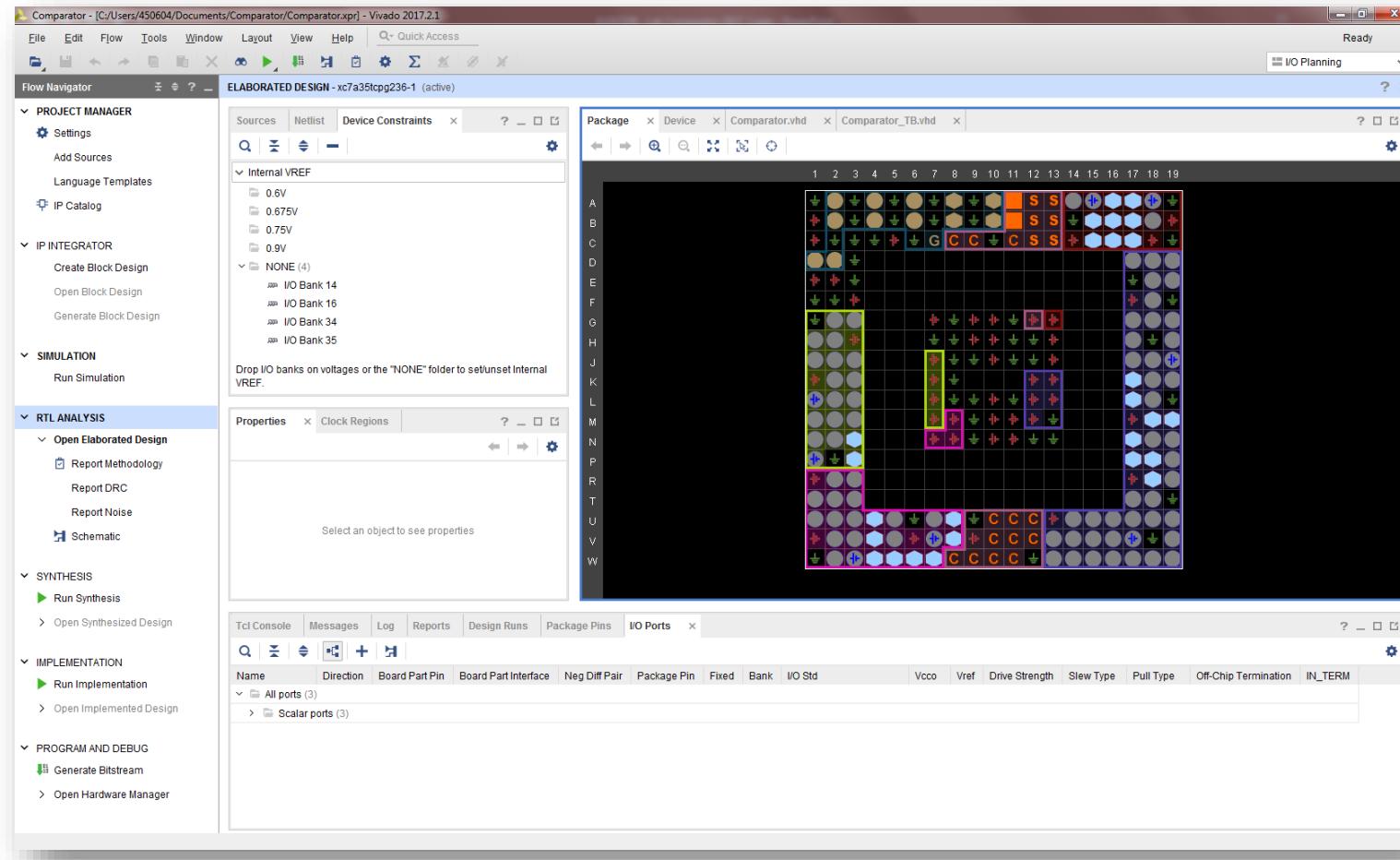
# Open elaborated design



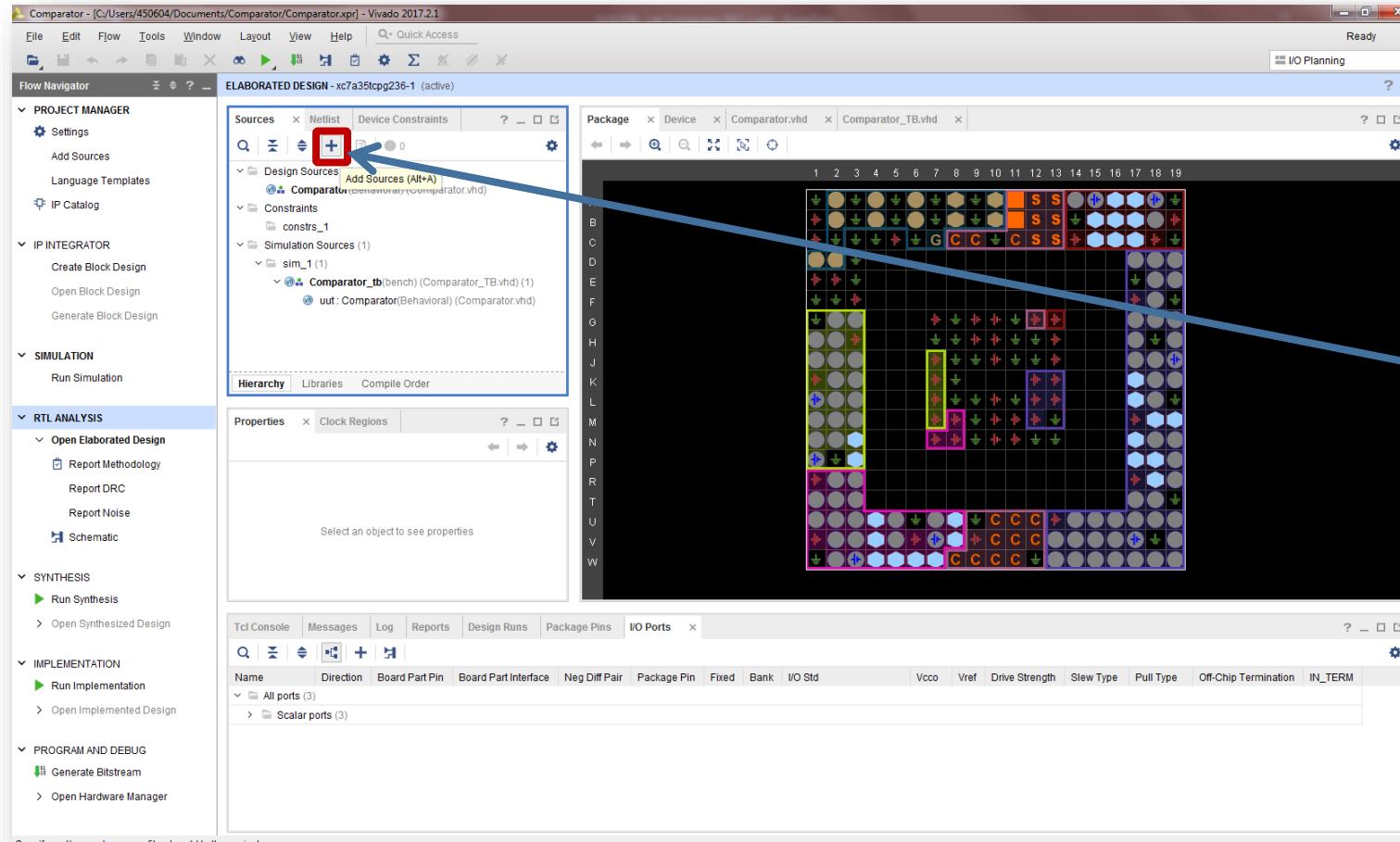
# Elaborated design – RTL view



# Elaborated design package

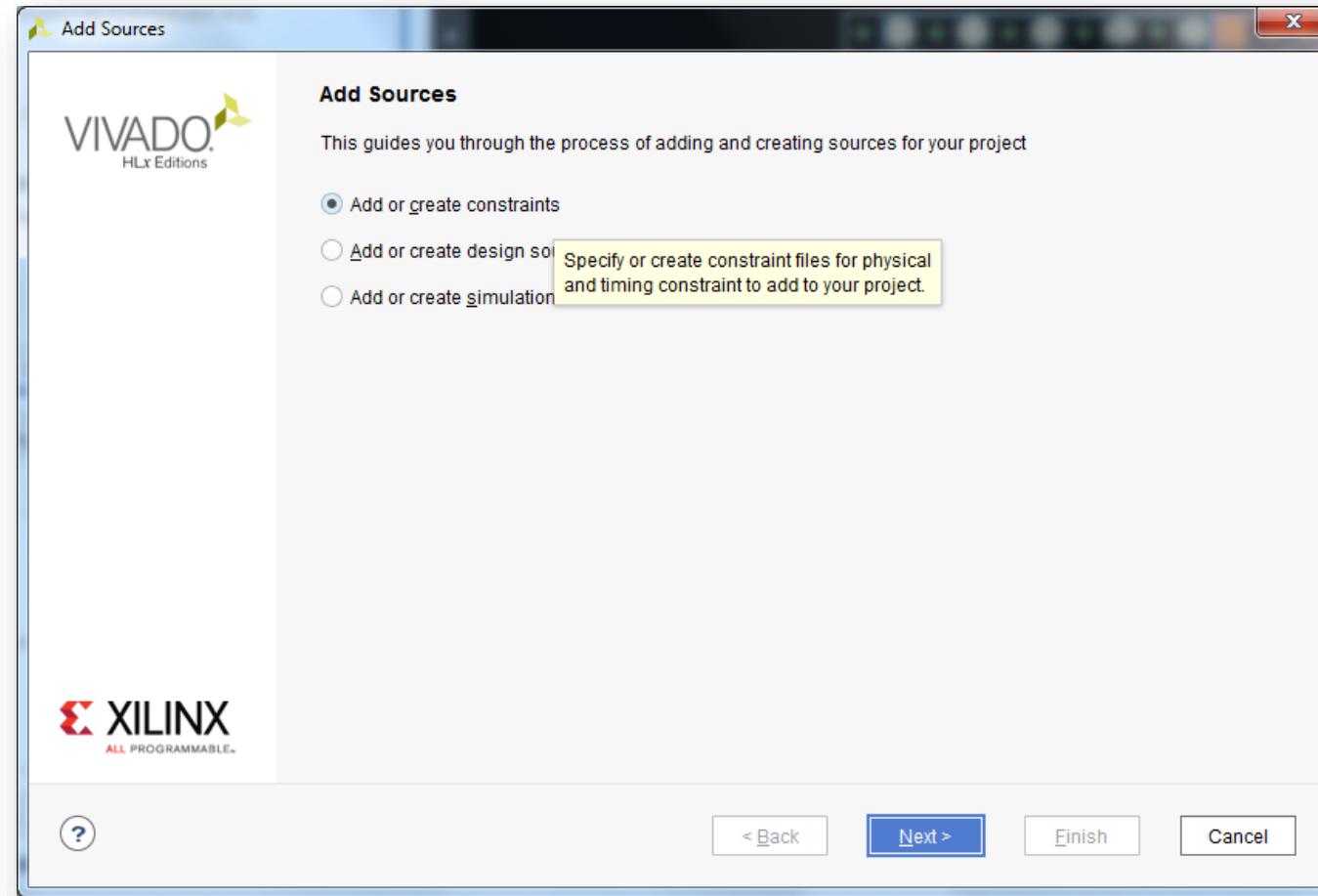


# Create a constraints file

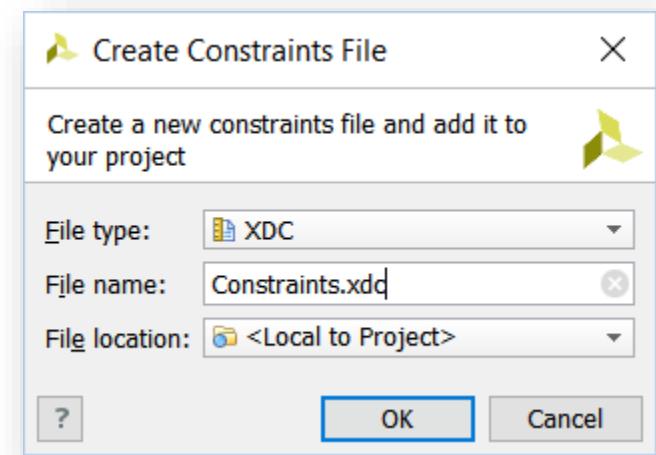
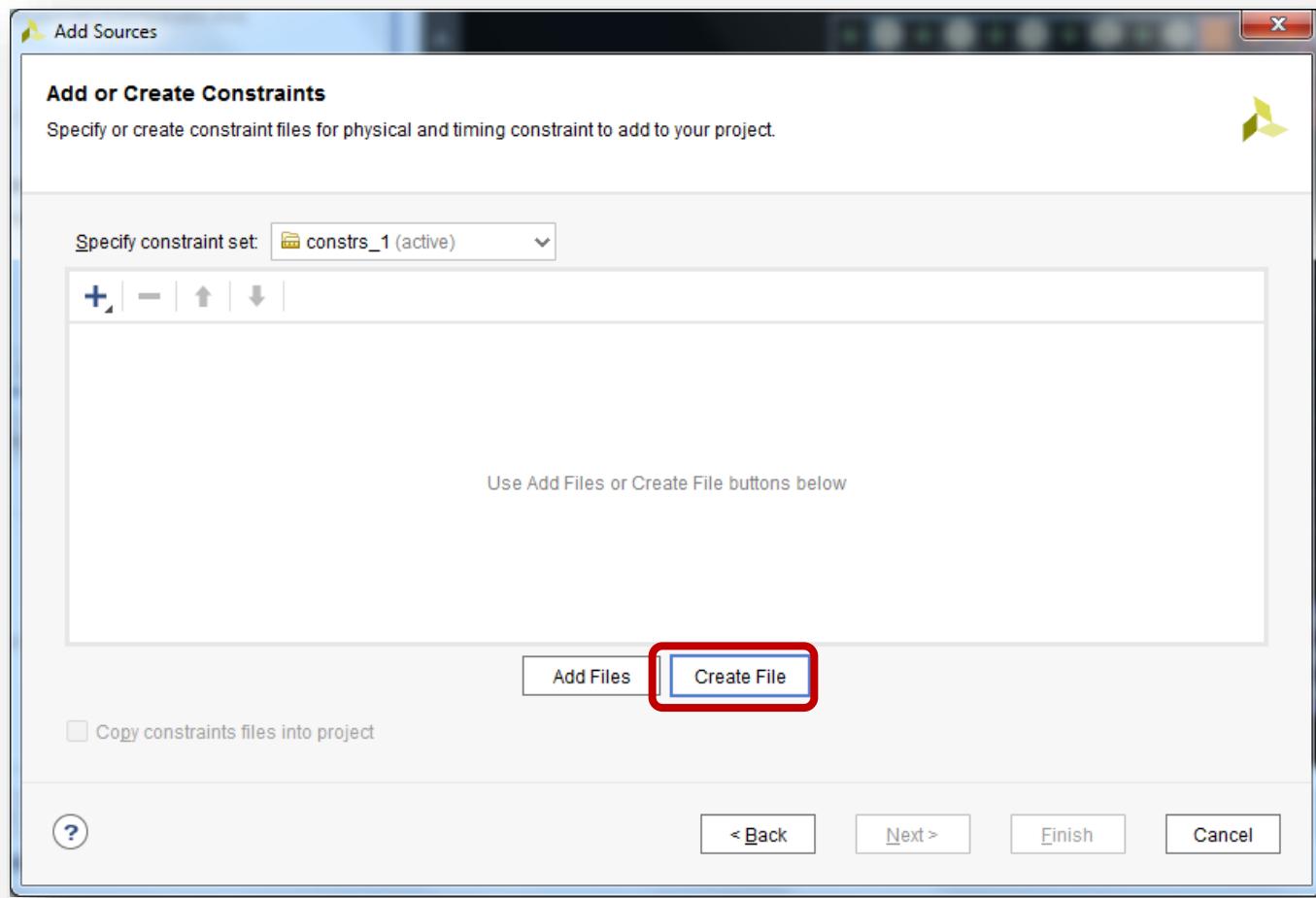


Add a source file

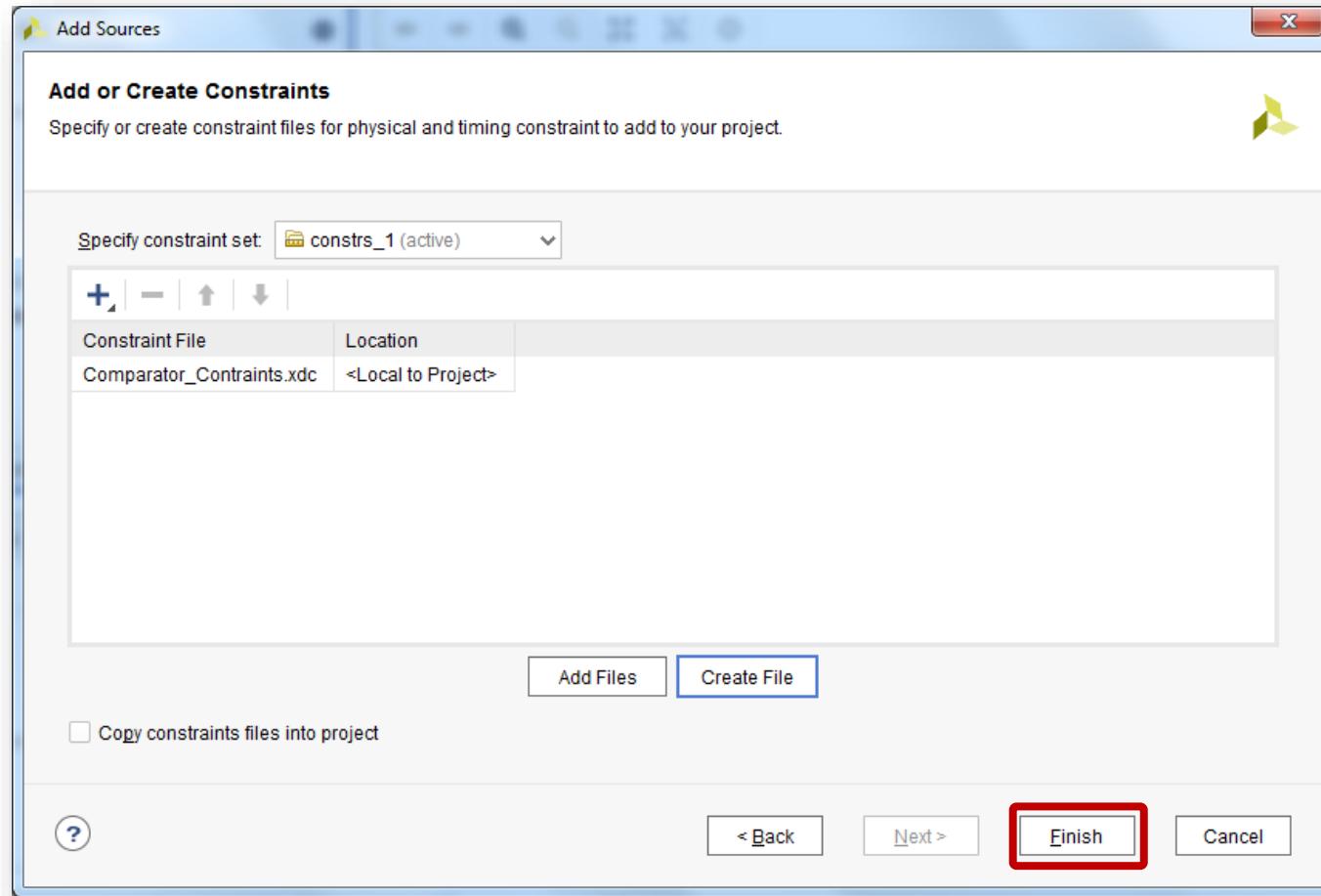
# Create a constraints file



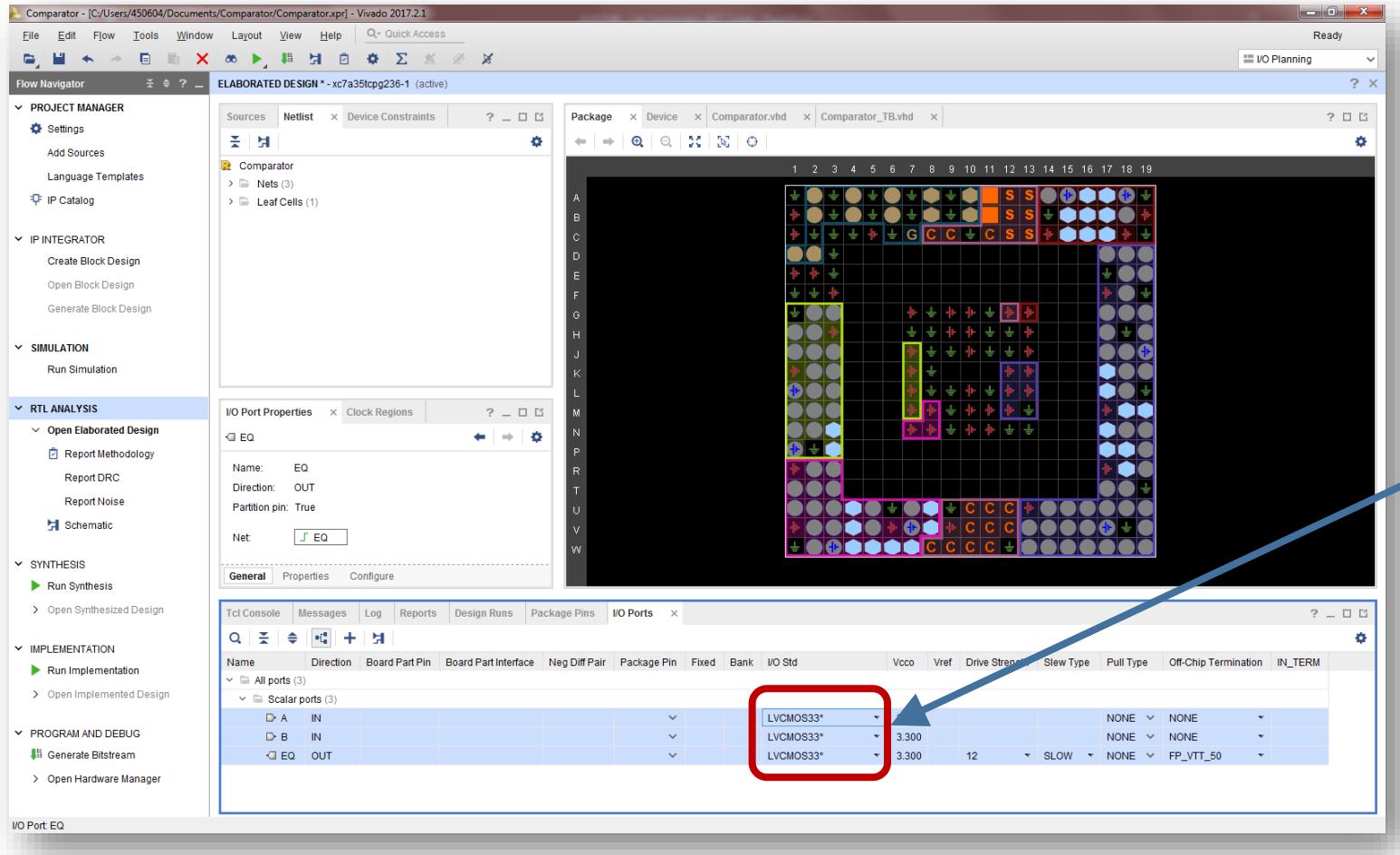
# Create a constraints file



# Create a constraints file

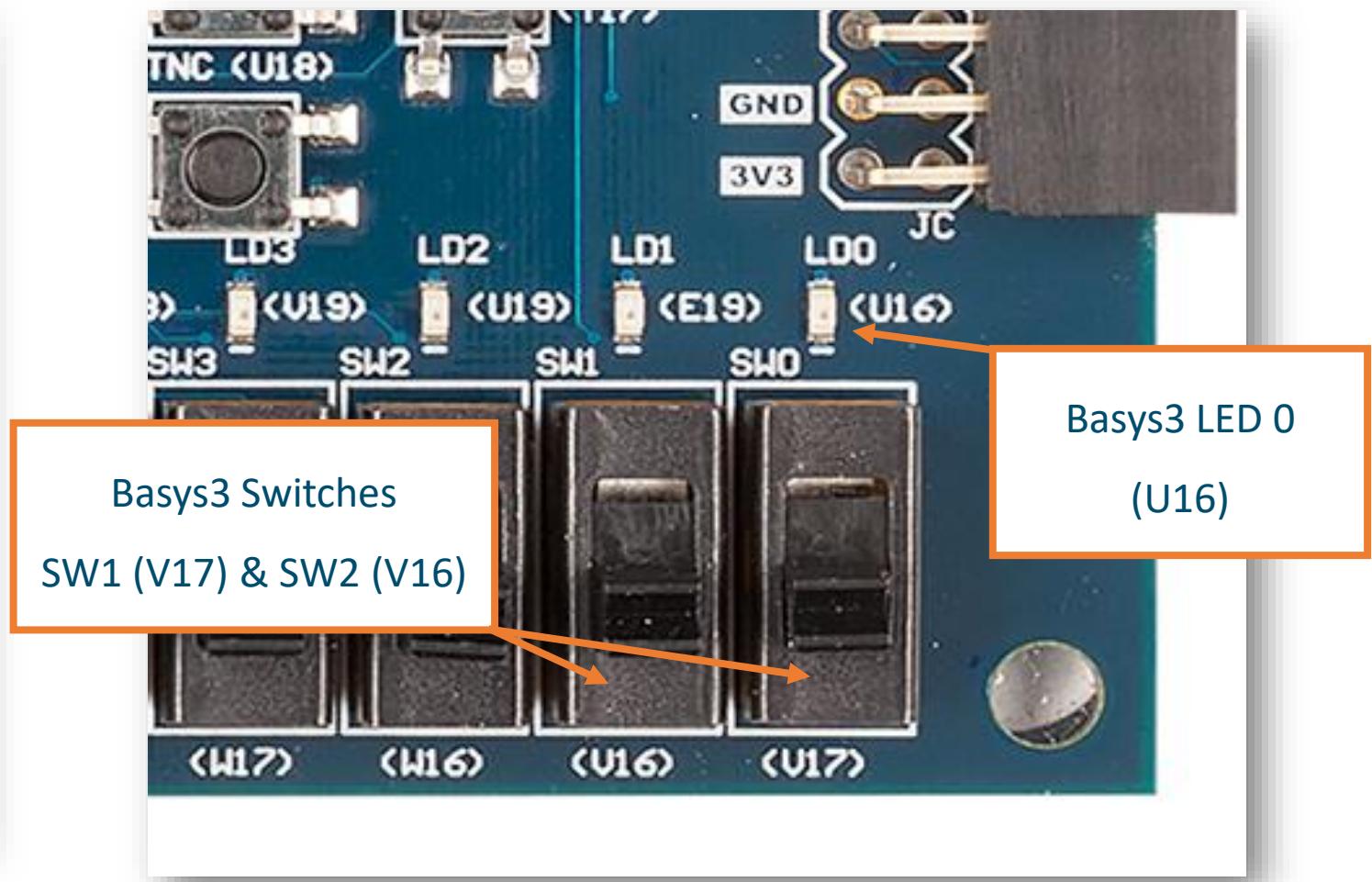
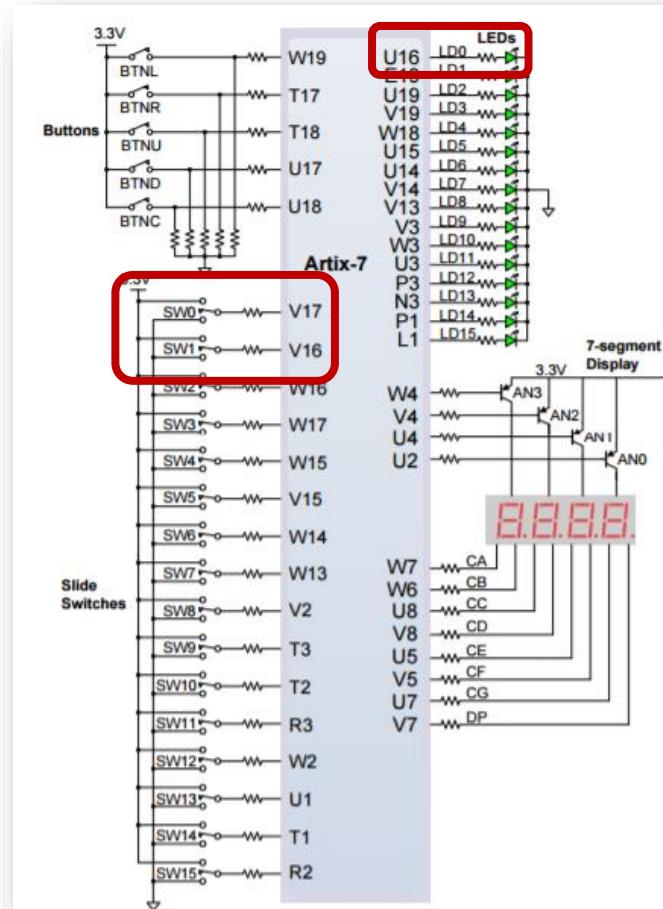


# Set IO pin voltage levels to LV CMOS 3.3 V

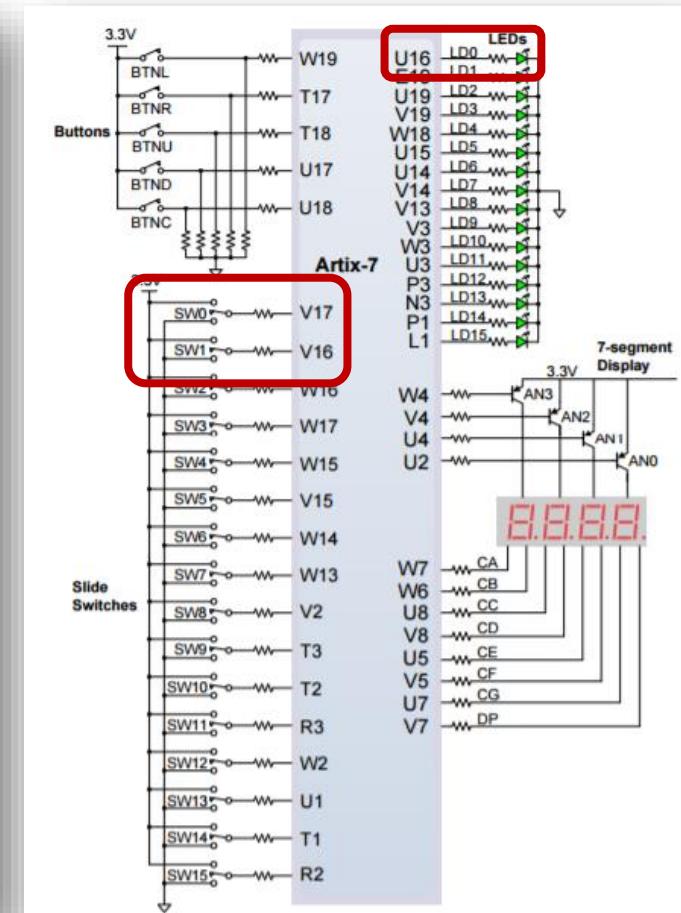
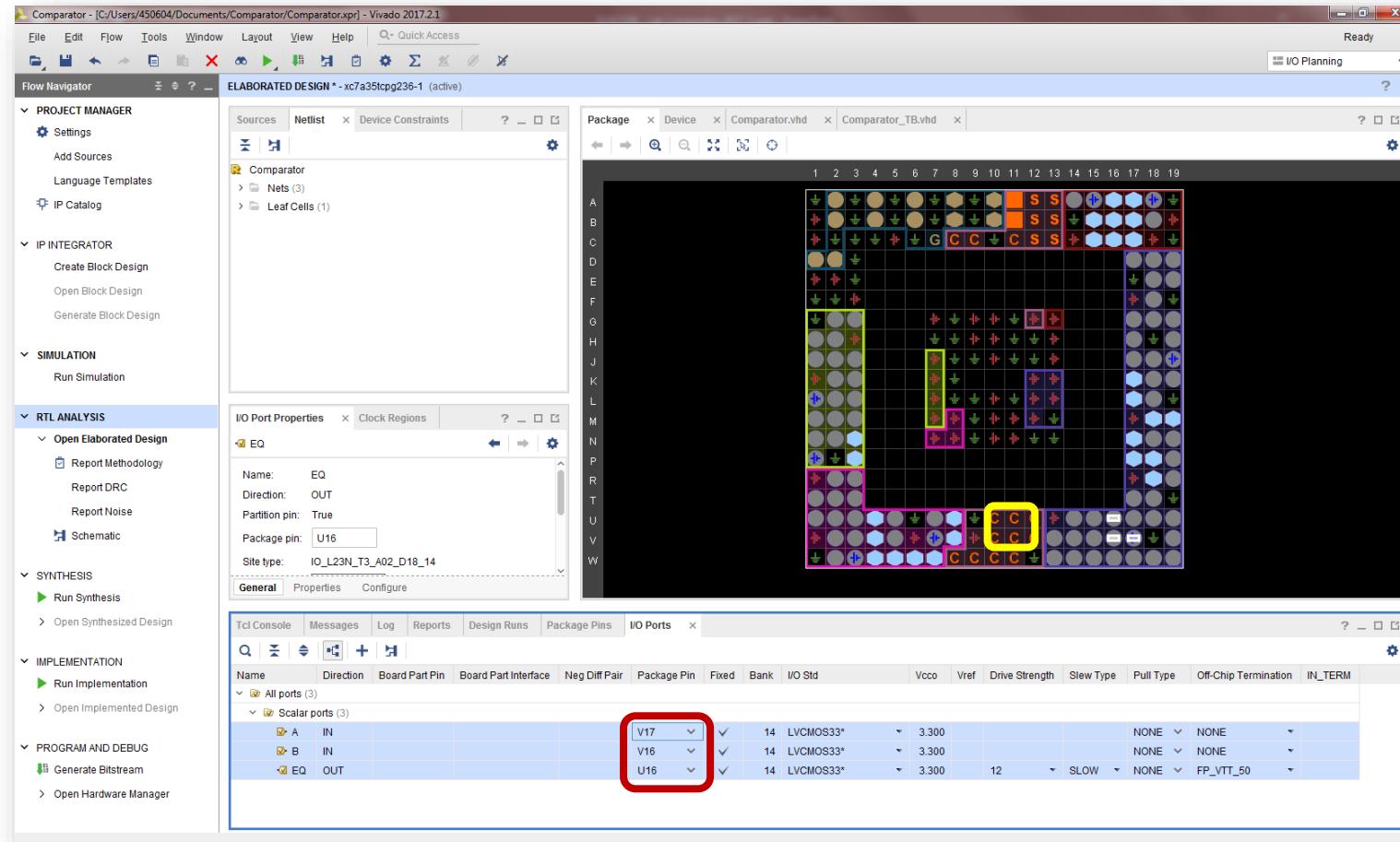


Select LVC MOS33  
for the 3.3 V level  
of the Basys3 IO Pins

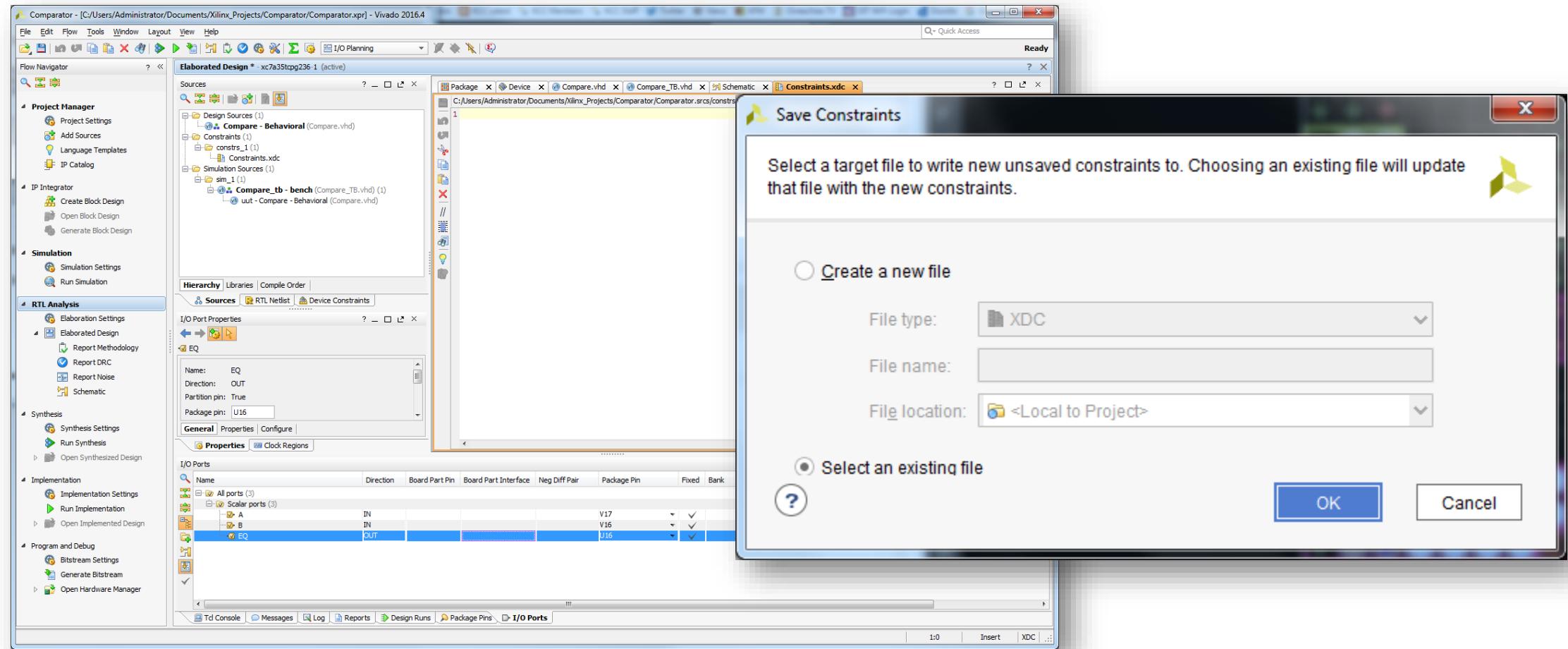
# Select I/O Planning view and I/O Pins



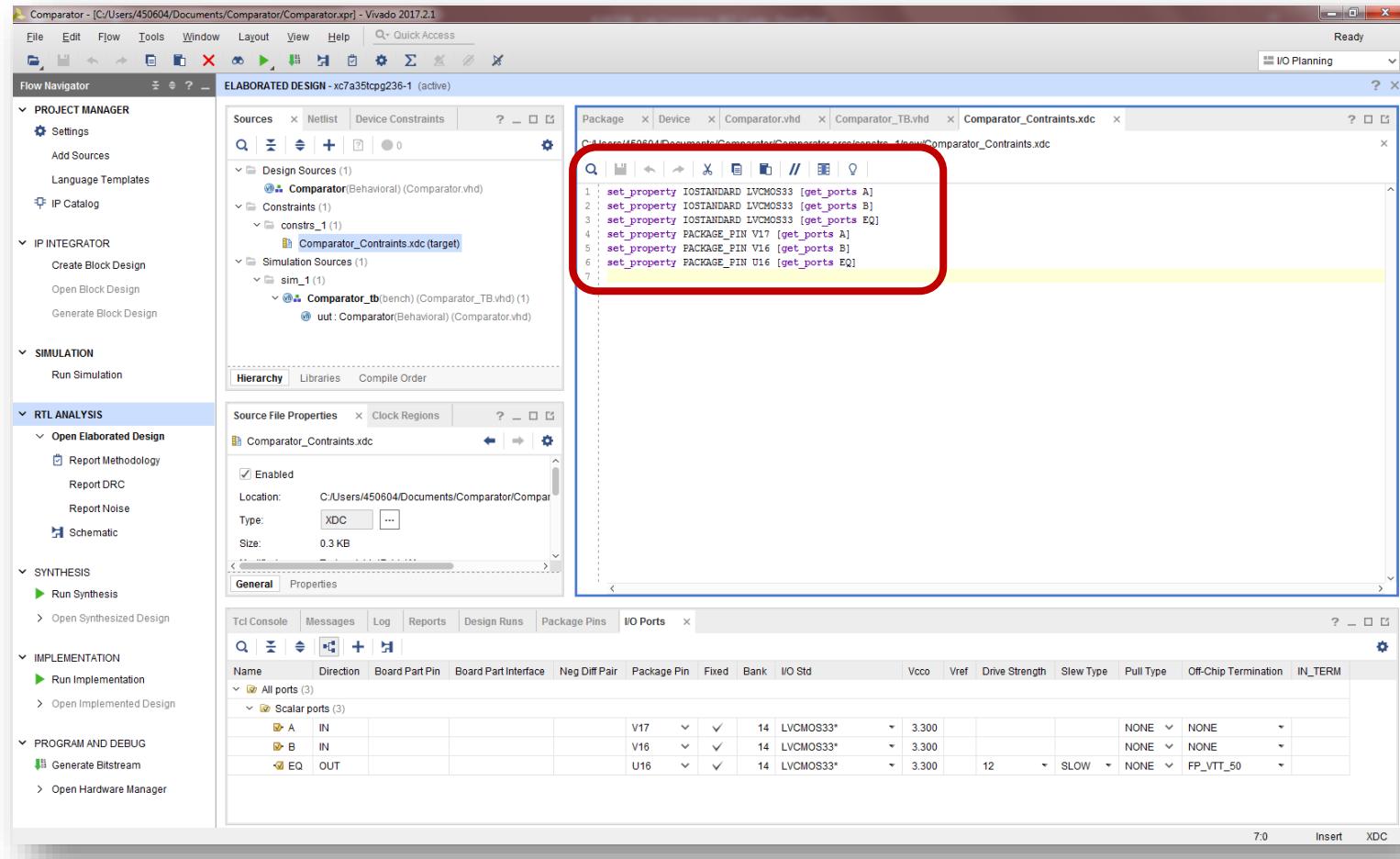
# Select I/O Planning view and I/O Pins



# Update Constraints.xdc file with Cntl + “s”



# Inspect the updated Constraints.xdc file



# Run synthesis



Screenshot of the Vivado 2017.2.1 interface showing the synthesis process.

The main window displays the project structure:

- PROJECT MANAGER**: Settings, Add Sources, Language Templates, IP Catalog.
- IP INTEGRATOR**: Create Block Design, Open Block Design, Generate Block Design.
- SIMULATION**: Run Simulation.
- RTL ANALYSIS**: Open Elaborated Design, Report Methodology, Report DRC, Report Noise, Schematic.
- SYNTHESIS**: **Run Synthesis** (highlighted with a red box).
- IMPLEMENTATION**: Run Implementation, Open Implemented Design.
- PROGRAM AND DEBUG**: Generate Bitstream, Open Hardware Manager.

A message at the bottom left says: "Run synthesis on your project source files".

The central workspace shows the elaborated design (xc7a35tcpg236-1) with tabs for Comparator.vhd, Comparator\_TB.vhd, and Comparator\_Constraints.xdc. The Constraints tab displays the following XDC code:

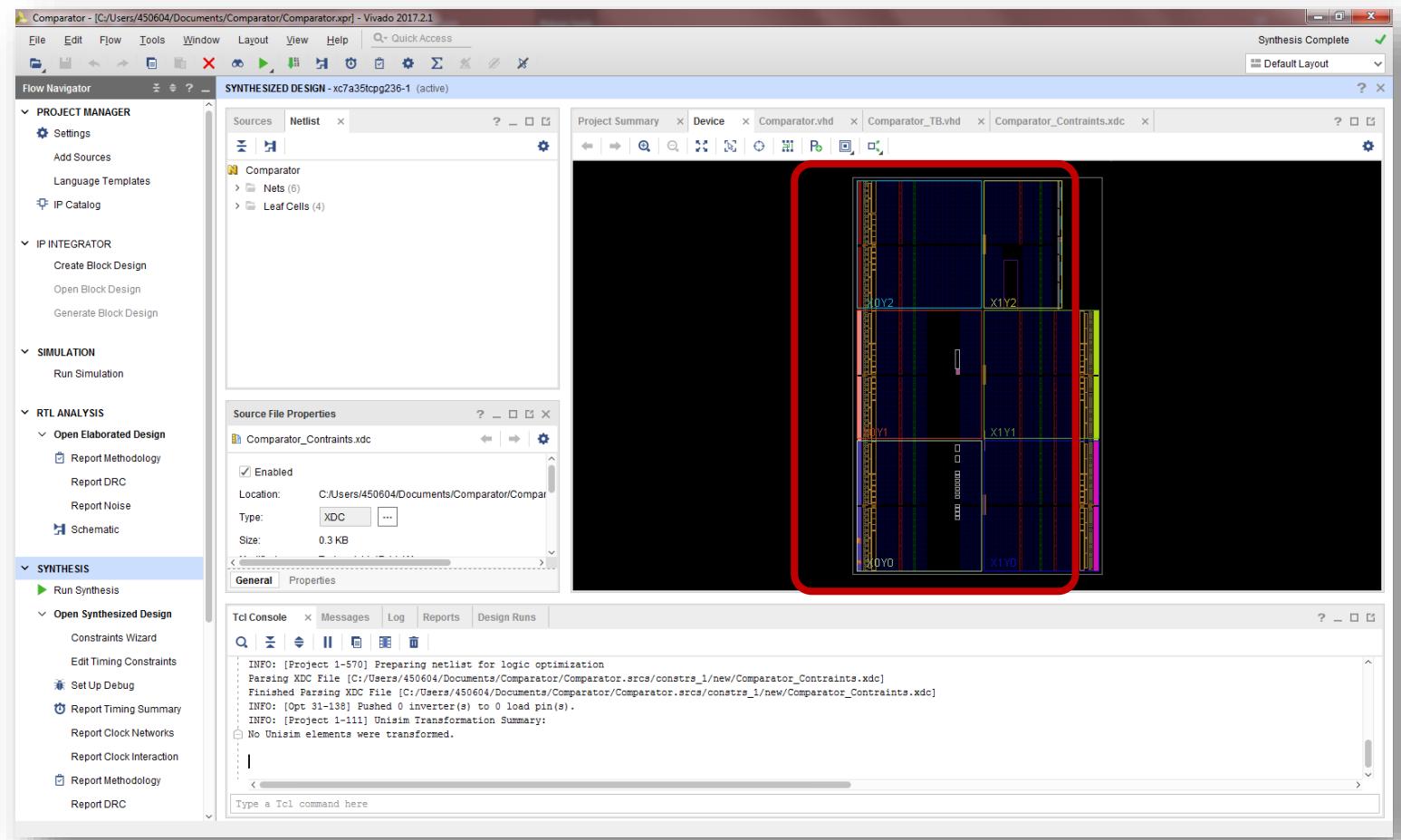
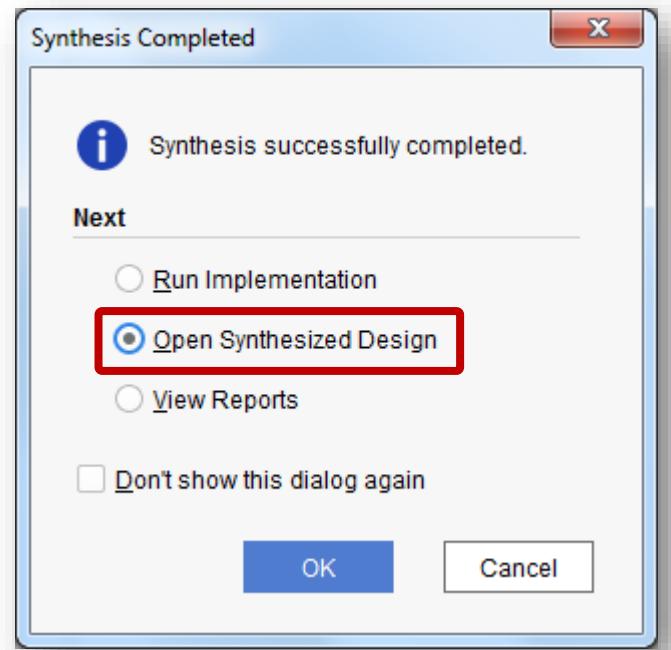
```
set_property IOSTANDARD LVCMOS33 [get_ports A]
set_property IOSTANDARD LVCMOS33 [get_ports B]
set_property IOSTANDARD LVCMOS33 [get_ports EQ]
set_property PACKAGE_PIN V17 [get_ports A]
set_property PACKAGE_PIN V16 [get_ports B]
set_property PACKAGE_PIN V16 [get_ports EQ]
```

To the right, a "Launch Runs" dialog box is open, prompting the user to "Launch the selected synthesis or implementation runs". It includes the following options:

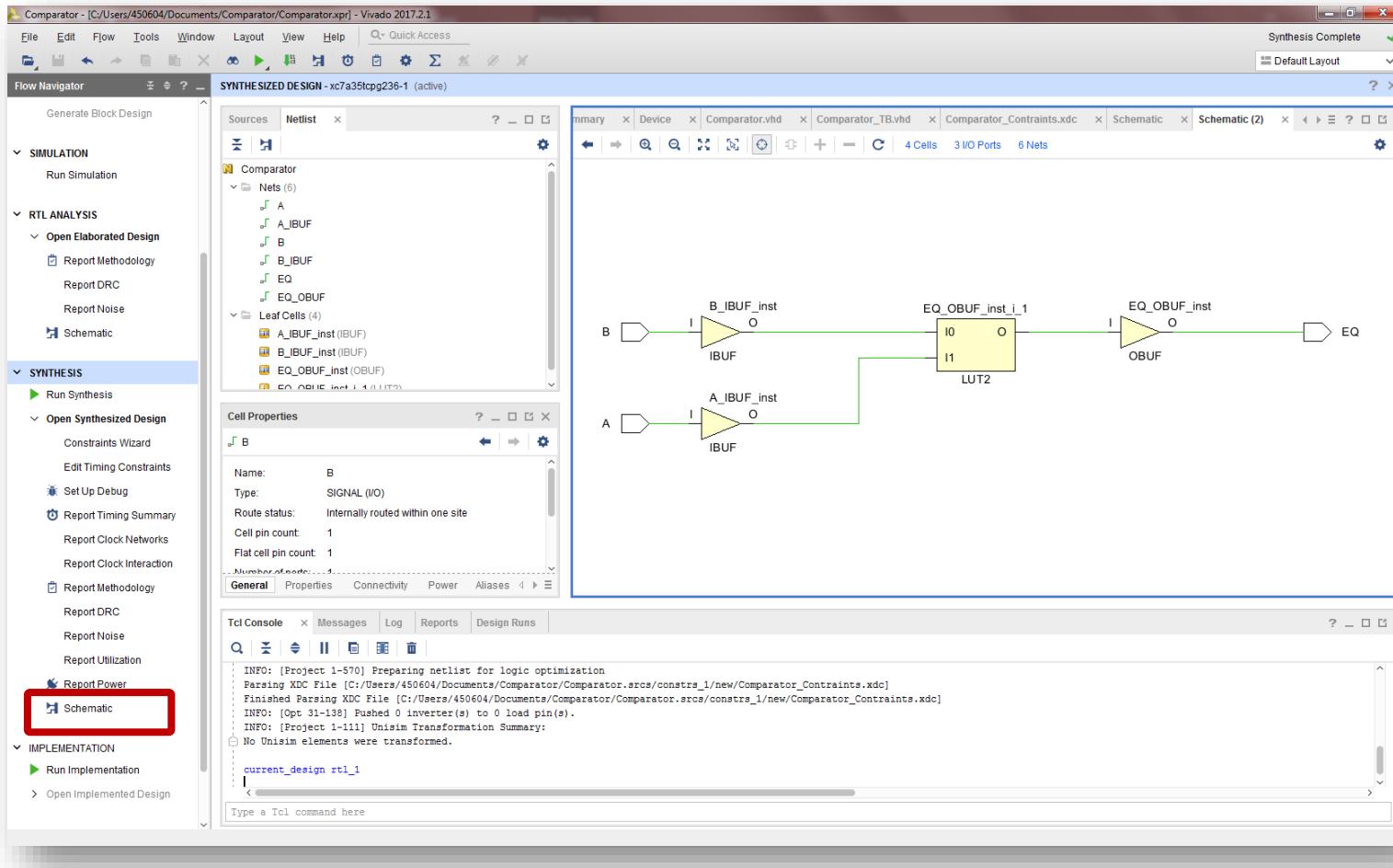
- Launch runs on local host: Number of jobs: 2
- Generate scripts only
- Don't show this dialog again

Buttons at the bottom are "OK" (highlighted with a red box) and "Cancel".

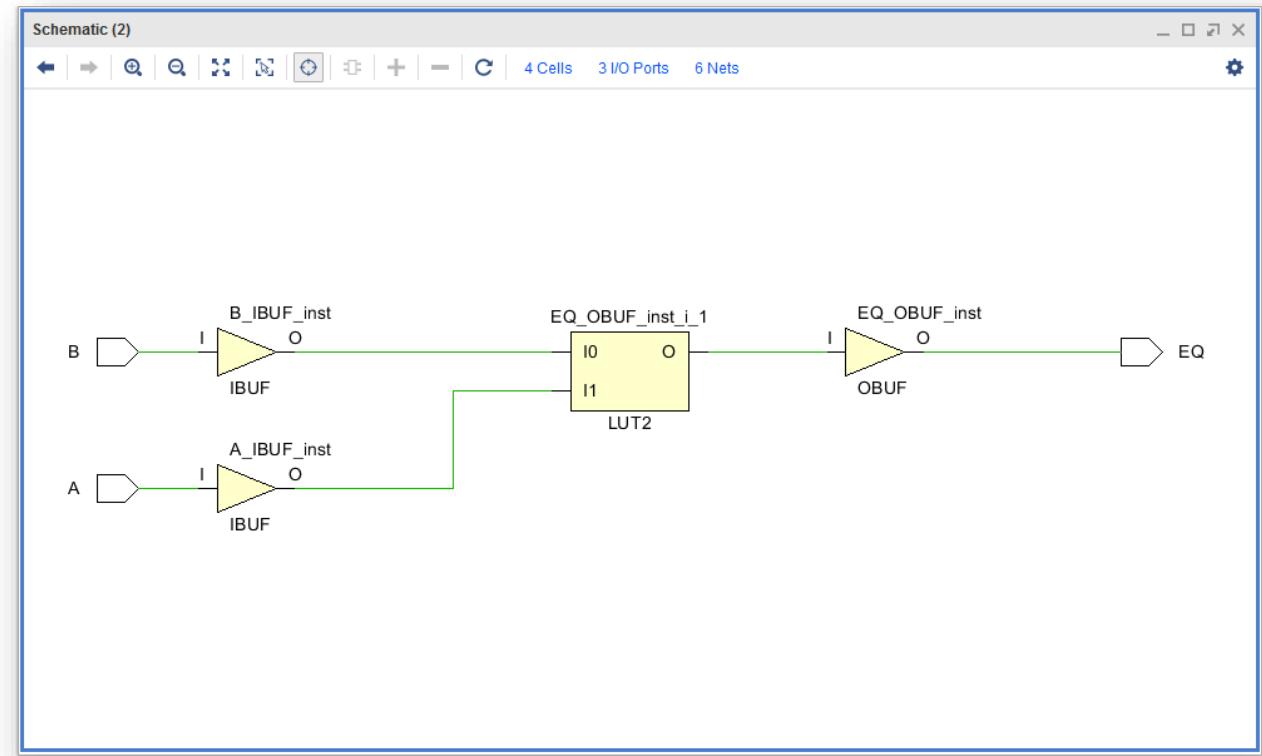
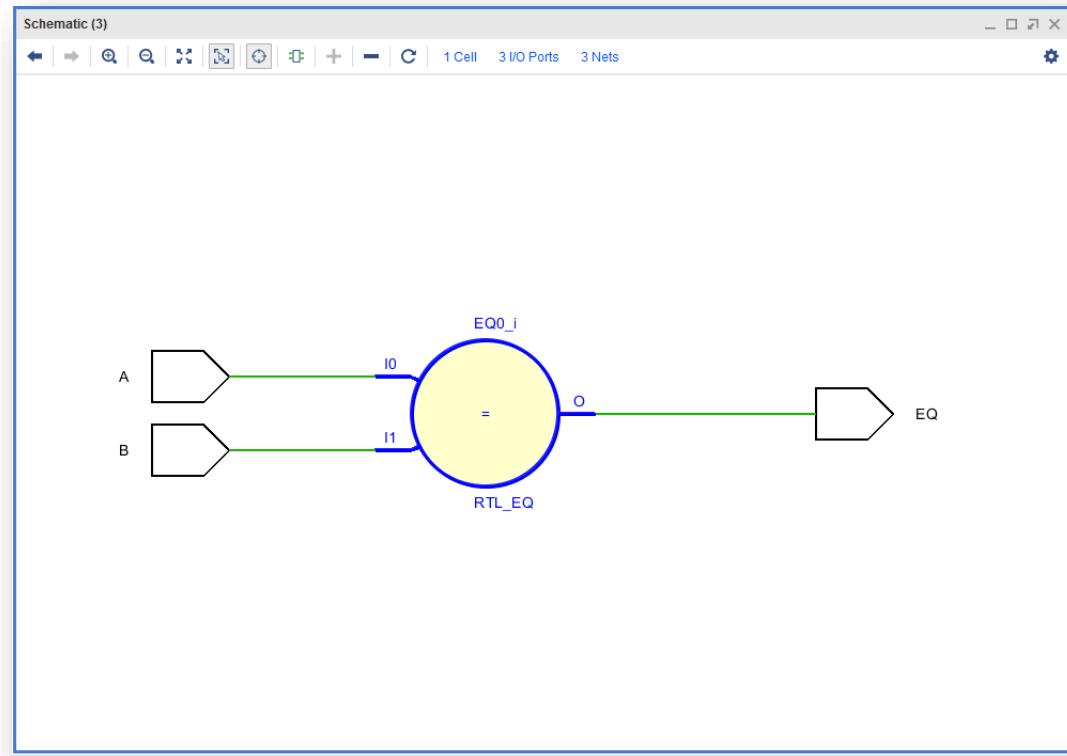
# Open synthesized design device layout



# Open Synthesized Design / Schematic



# Compare RTL and Synthesis Schematics

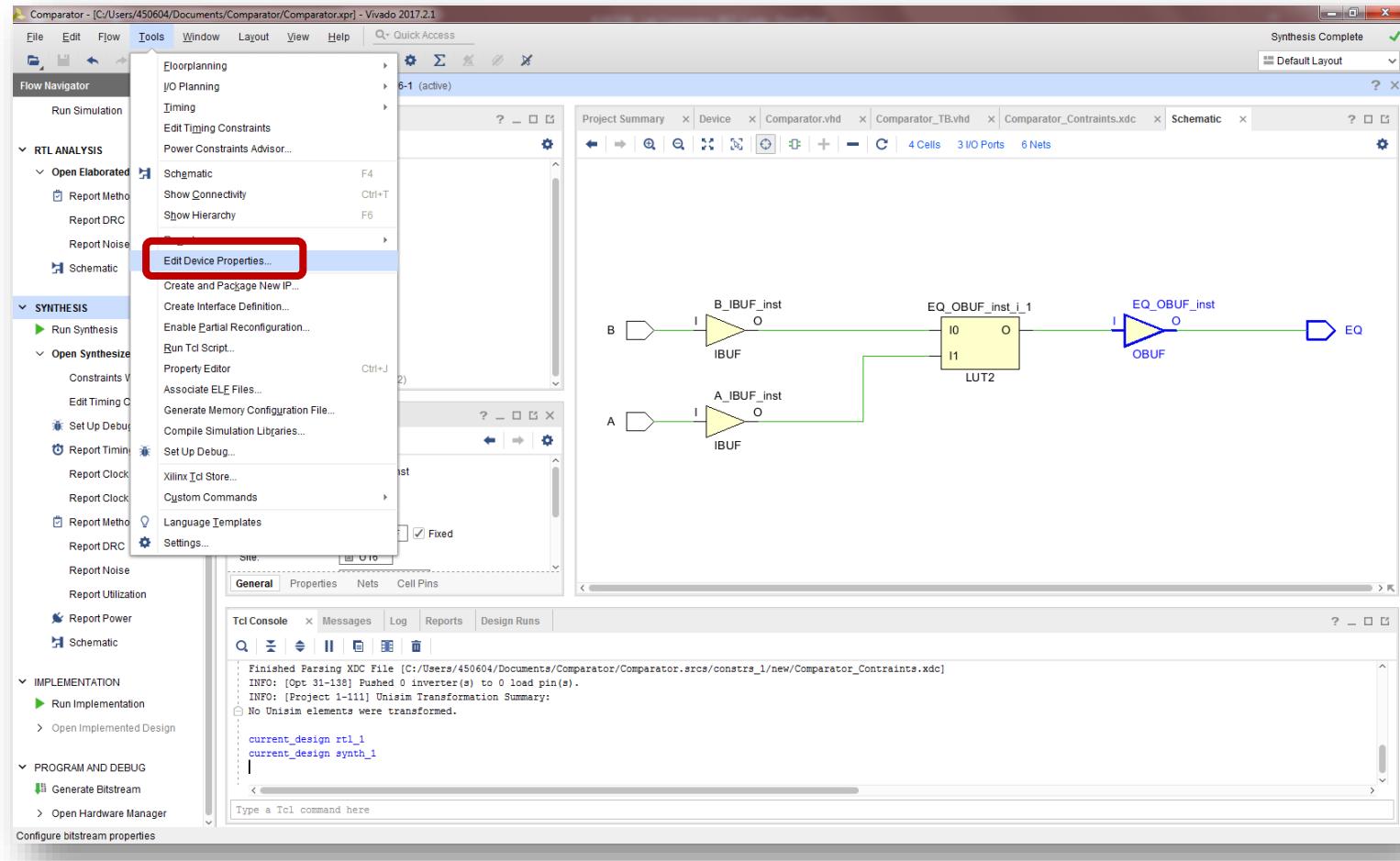


# Design reports

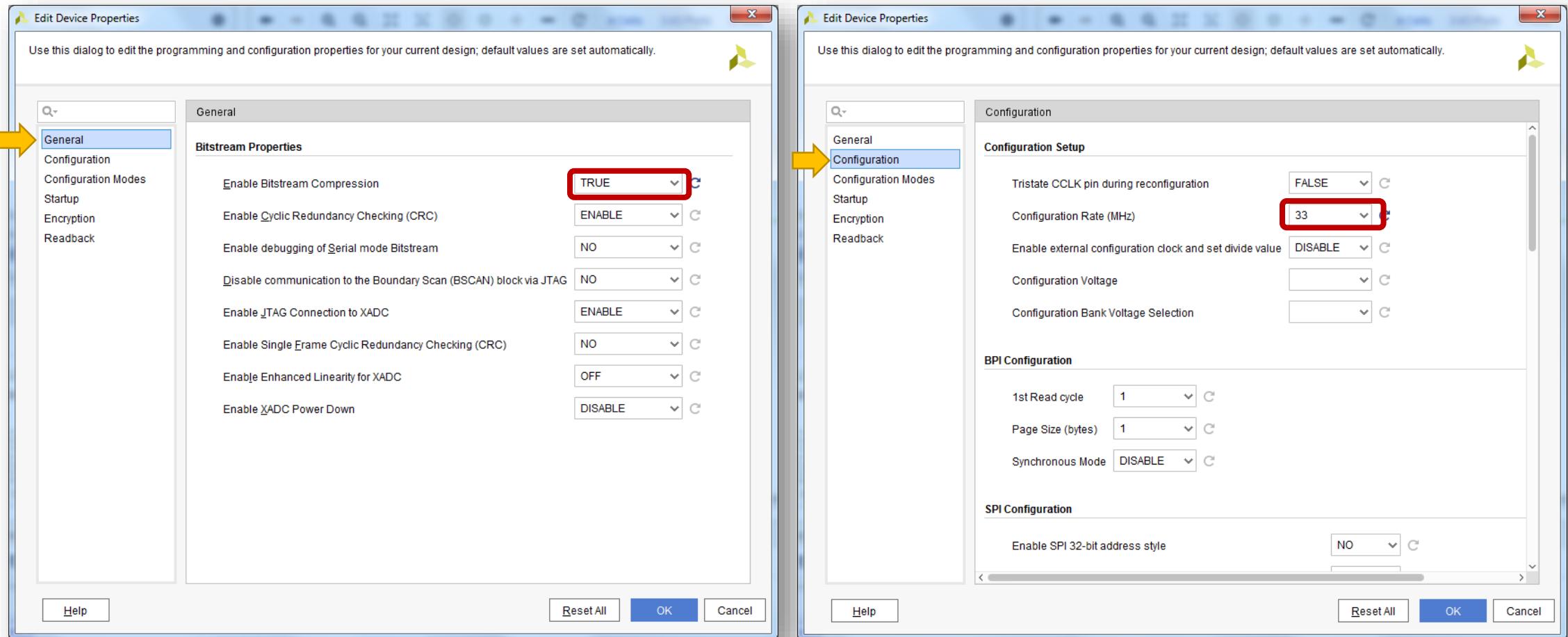
- Look at the Synthesis report:
  - Copy of the following table and complete it in your report

Cell Type Label	Functional Category	What is it used for?
LUT2		
IBUF		
OBUF		

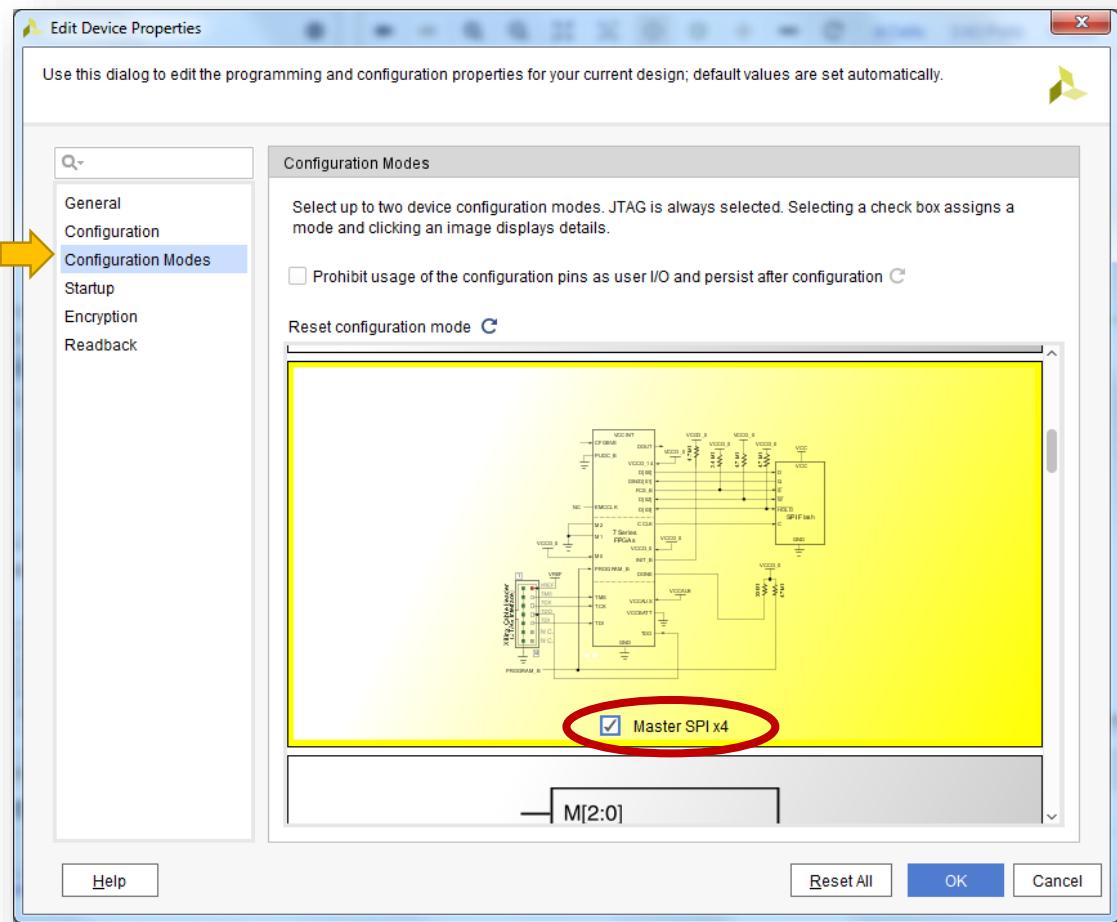
# Tools menu: Edit device properties



# Tools menu: Edit device properties



# Edit device properties



Select Serial Peripheral Interface BUS – **quad**

Examine the Master SPI x4 Configuration Mode Details by doubling clicking SPI image.

# Run implementation

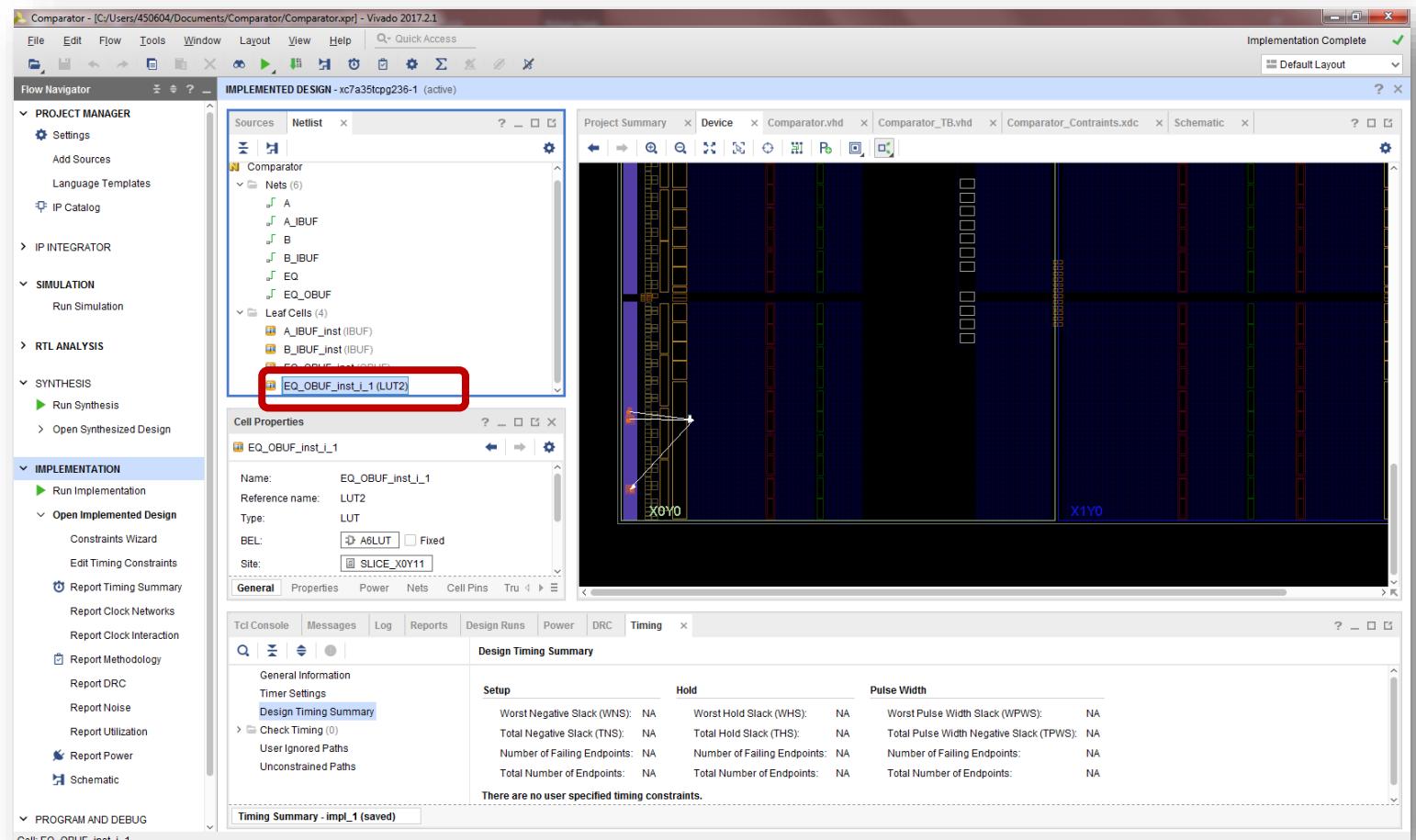
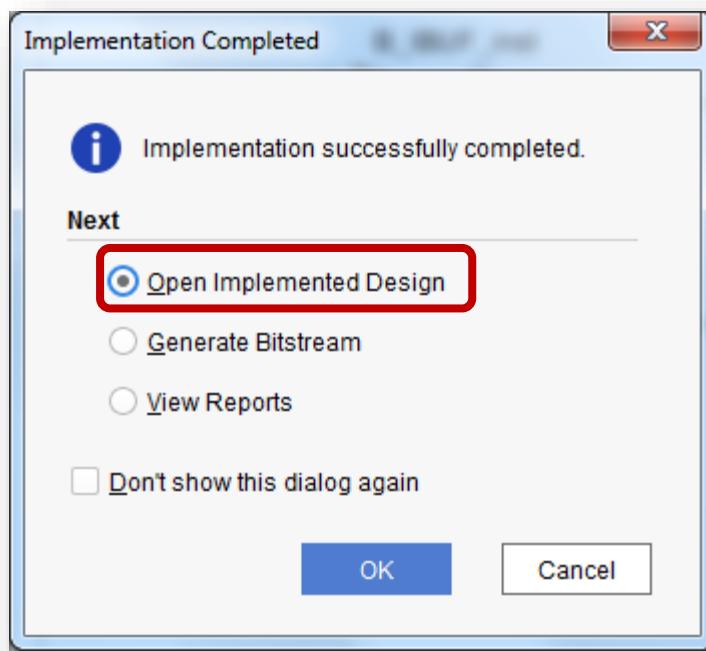


The screenshot shows the Vivado 2017.2 interface with the following components visible:

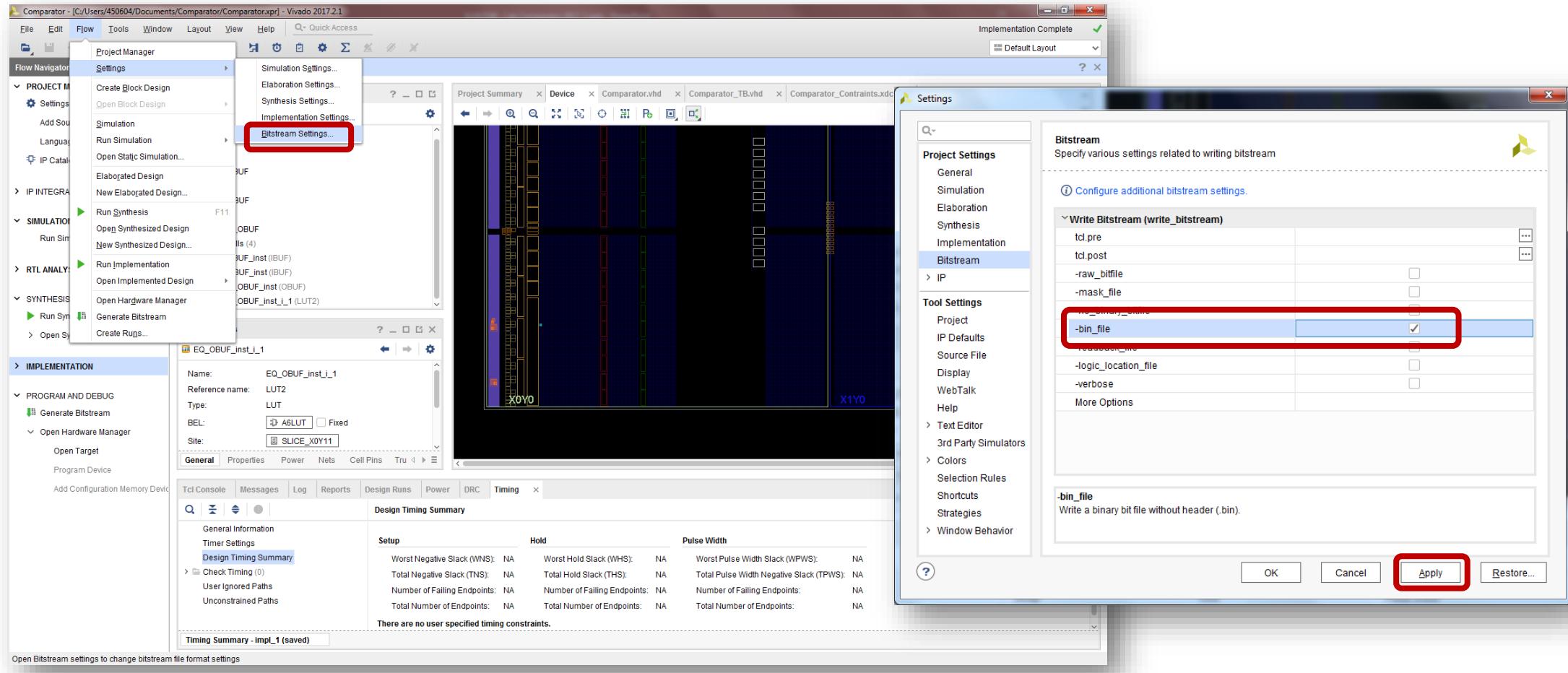
- Project Manager:** Shows the design is synthesized (Synthesis Complete).
- Synthesis:** The "Run Implementation" button is highlighted with a red box.
- Schematic View:** Displays the internal logic of the comparator. It includes two inputs (A and B) connected to IBUF buffers, which feed into a LUT2 block. The LUT2 has two outputs, I0 and I1, which are then connected to EQ\_OBUF buffers. The final output is another EQ\_OBUF buffer.
- Tcl Console:** Shows the command: 

```
current_design rtl_1  
current_design synth_1  
startgroup  
set_property BITSTREAM.GENERAL.COMPRESS TRUE [get_designs synth_1]  
set_property BITSTREAM.CONFIG.CONFIGRATE 33 [get_designs synth_1]  
set_property config_mode SPIx4 [current_design]  
endgroup
```
- Save Project Dialog:** A modal dialog asking "Save project before launching implementation?". It lists "Synthesized Design - constrs\_1 - Comparator\_Constraints.xdc" and provides "Save", "Don't Save", and "Cancel" buttons. The "Save" button is highlighted with a red box.
- Launch Runs Dialog:** A modal dialog for launching synthesis or implementation runs. It includes fields for "Launch directory" (set to "<Default Launch Directory>"), "Options" (radio buttons for "Launch runs on local host" (selected) and "Generate scripts only"), and a checkbox for "Don't show this dialog again". The "OK" button is highlighted with a red box.

# Examine leaf cells in the implemented design



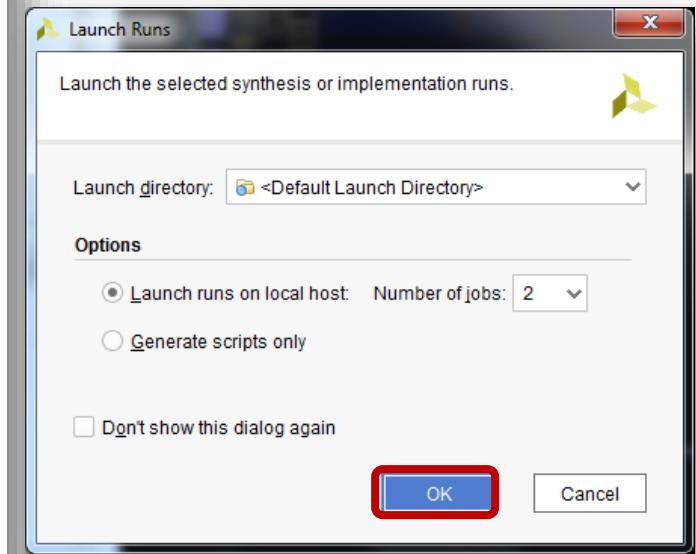
# Bitstream settings



# Generate bitstream

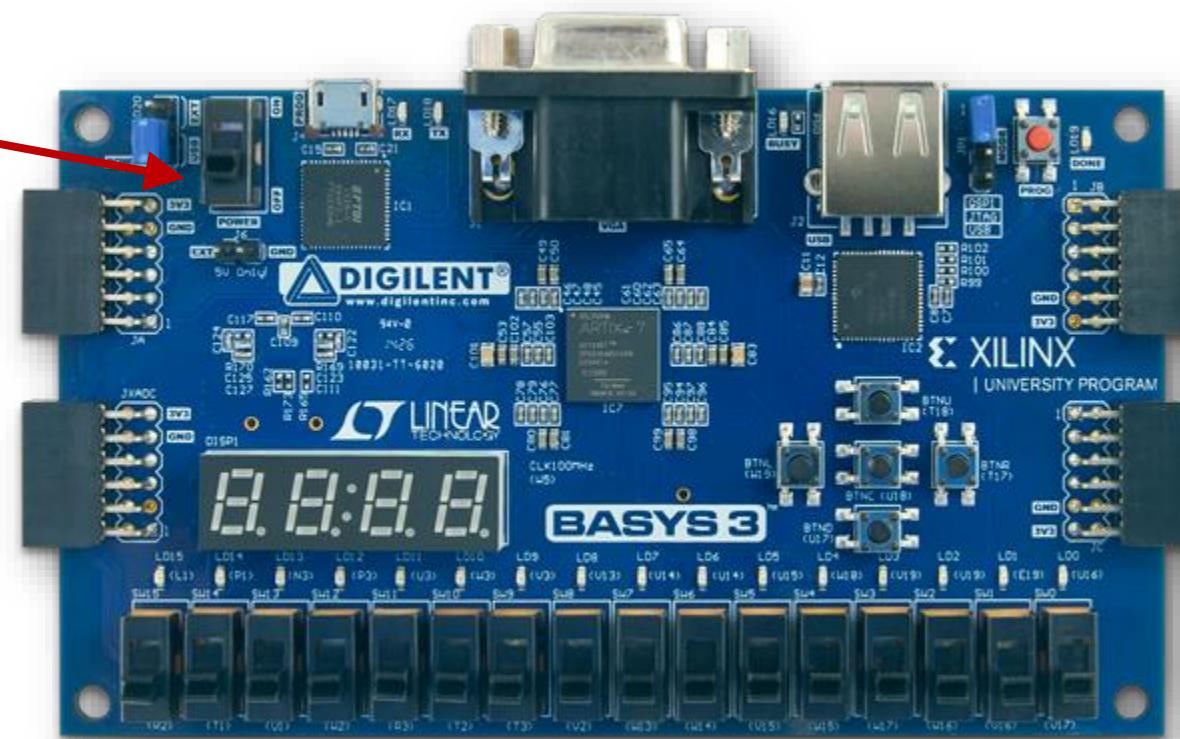


Screenshot of the Vivado 2017.2.1 software interface for a Comparator project. The main window shows the implemented design with various logic cells and nets. The left sidebar navigation pane highlights the 'IMPLEMENTATION' section, which is currently active. A red box highlights the 'Program Device' button in the 'IMPLEMENTATION' section. The bottom status bar indicates: 'Generate a programming file after implementation'.



# Connect Basys 3 to PC with USB cable

Connect with switch  
in **powered OFF**  
setting



# Basys3 board: configure pins

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod port(s)	10	Programming mode jumper
3	Analog signal Pmod port (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Table 1. Basys 3 Callouts and component descriptions.

# Basys3 board: Pin configurations

Figure 3 shows the different options available for configuring the FPGA. An on-board "mode" jumper (JP1) selects between the programming modes.

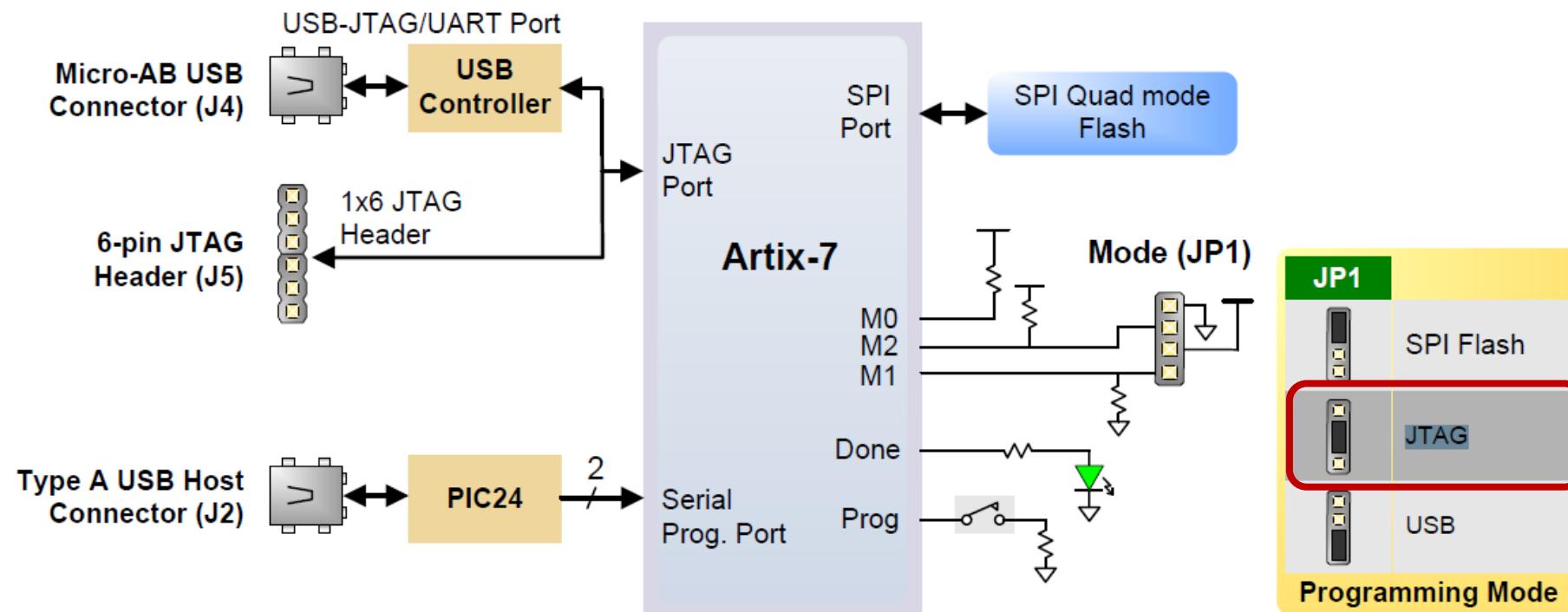
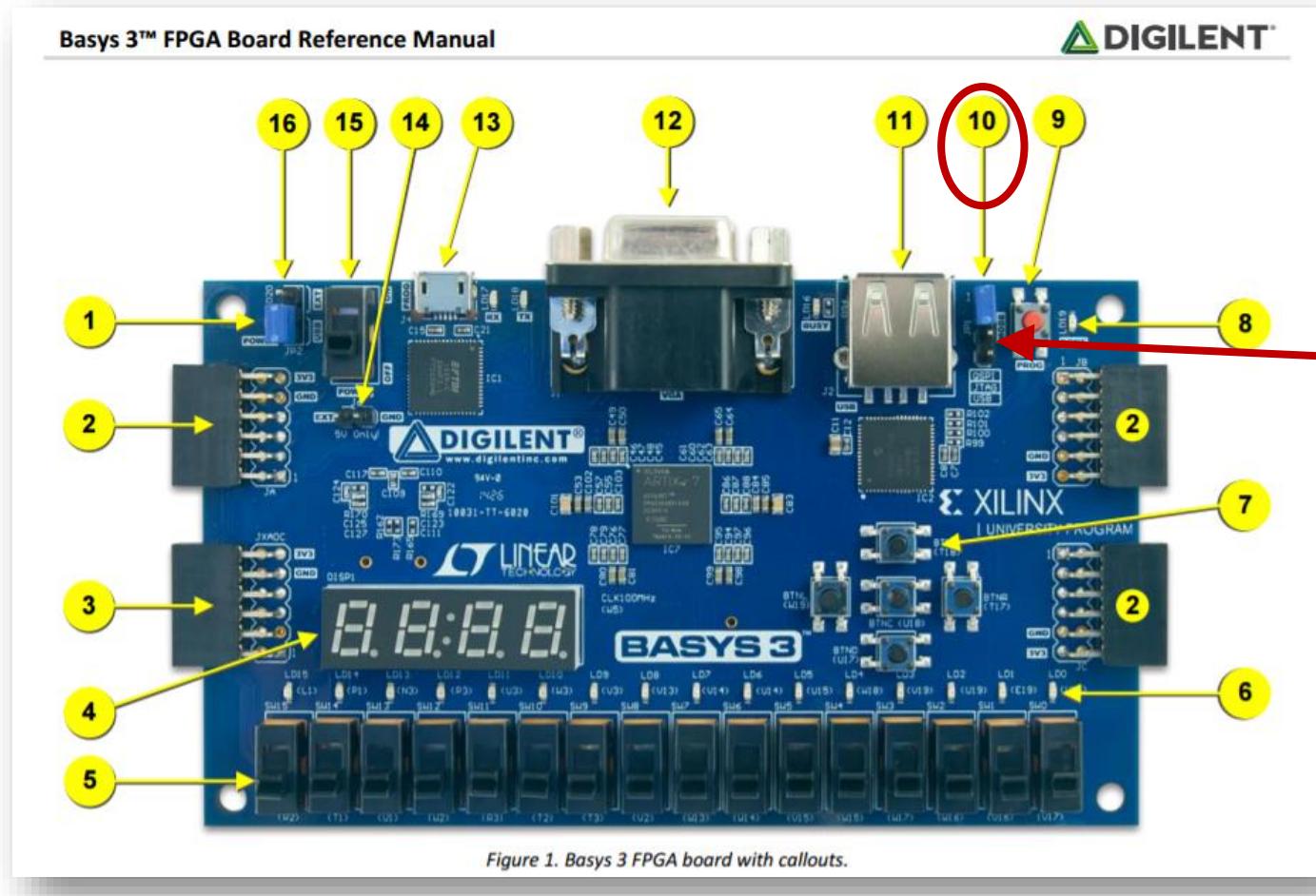


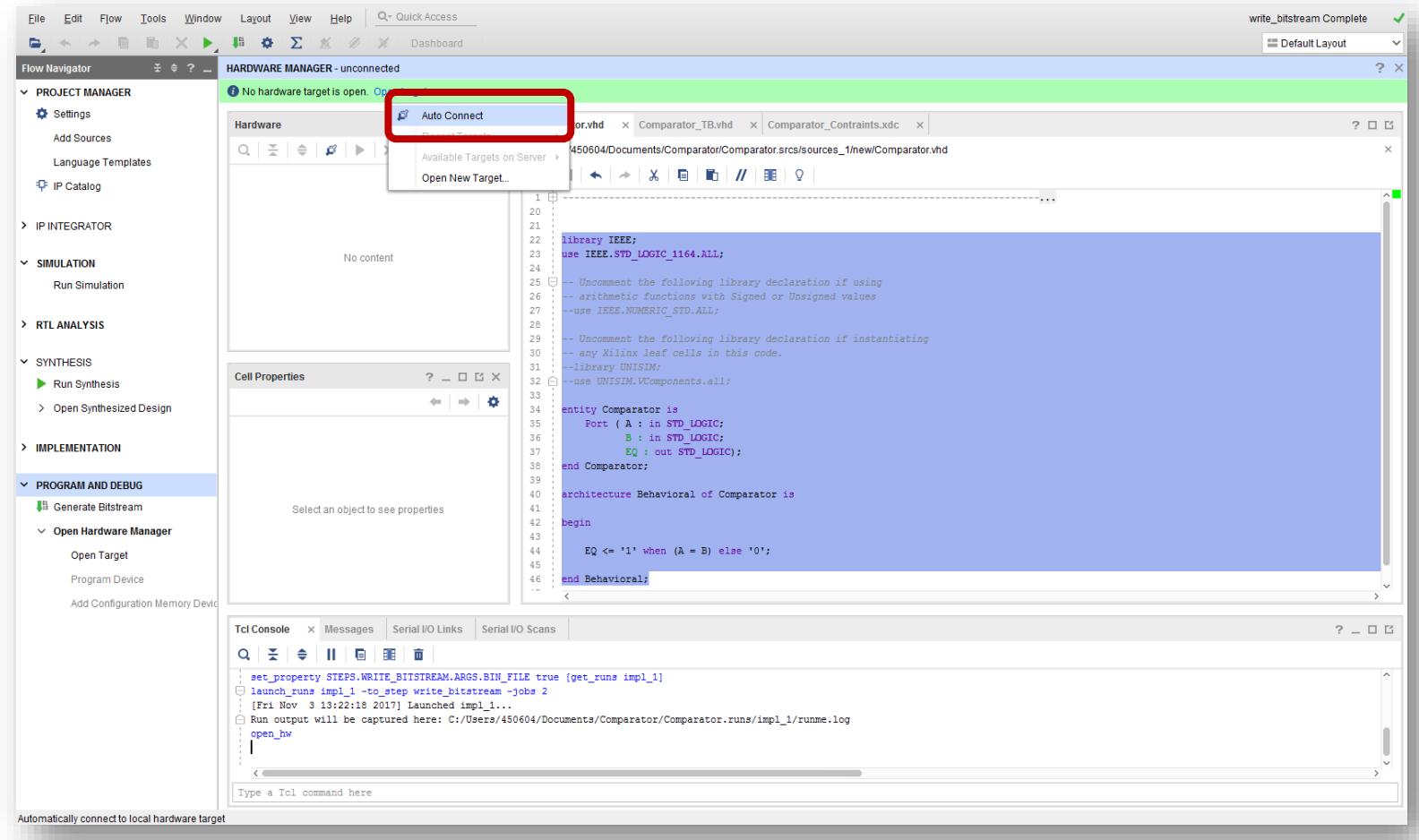
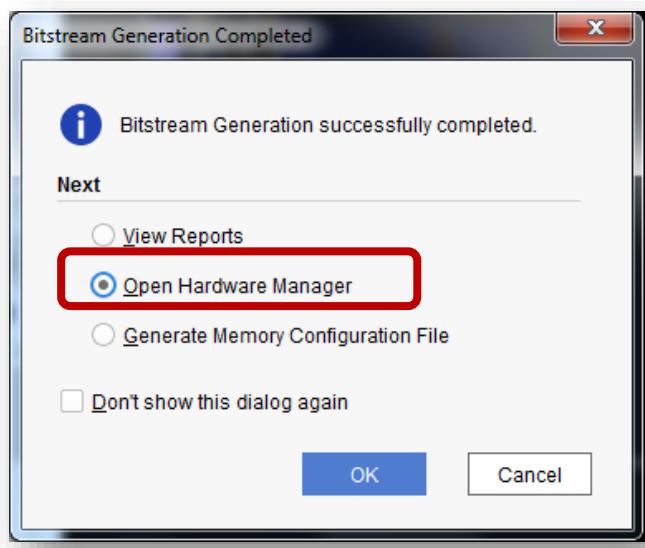
Figure 3. Basys 3 configuration options.

# Basys3 board: configure pin 10 and power on

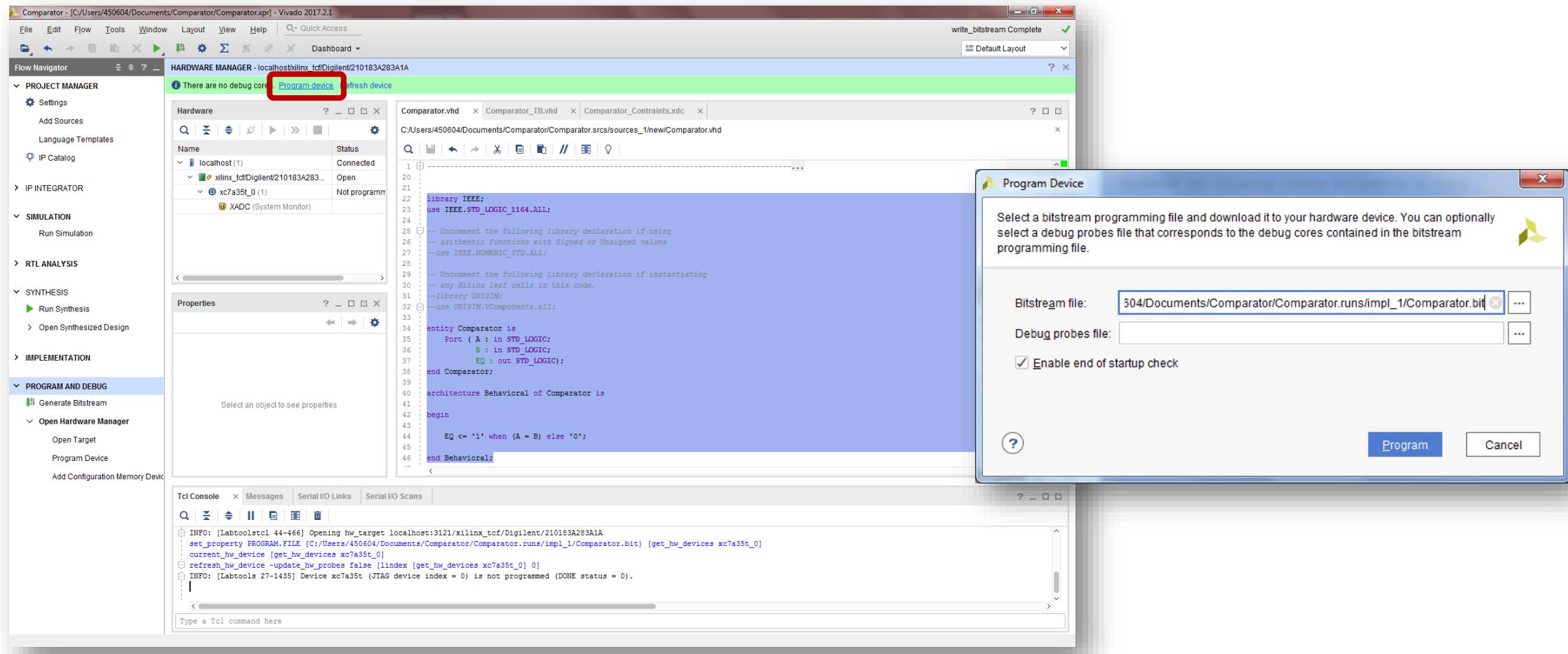


To select the JTAG setting,  
place jumper on middle  
pair of the four pins.  
(Note: QSPI setting is illustrated)

# Open hardware manager



# Program device



# Does the programme work properly?

Using the Laboratory Report Template:

- Apply signals to the board via the switches to confirm that the design works as planned
- Show that the programme worked using **photographs** of the Basys 3 to show the LEDs and the different switch states
- Remember to complete all the checks on your report before submitting

# YouTube - Advice

- Take a look at the following video to get oriented on the work flow.
- <https://www.youtube.com/watch?v=JtixlupNSOQ> (9:11)

