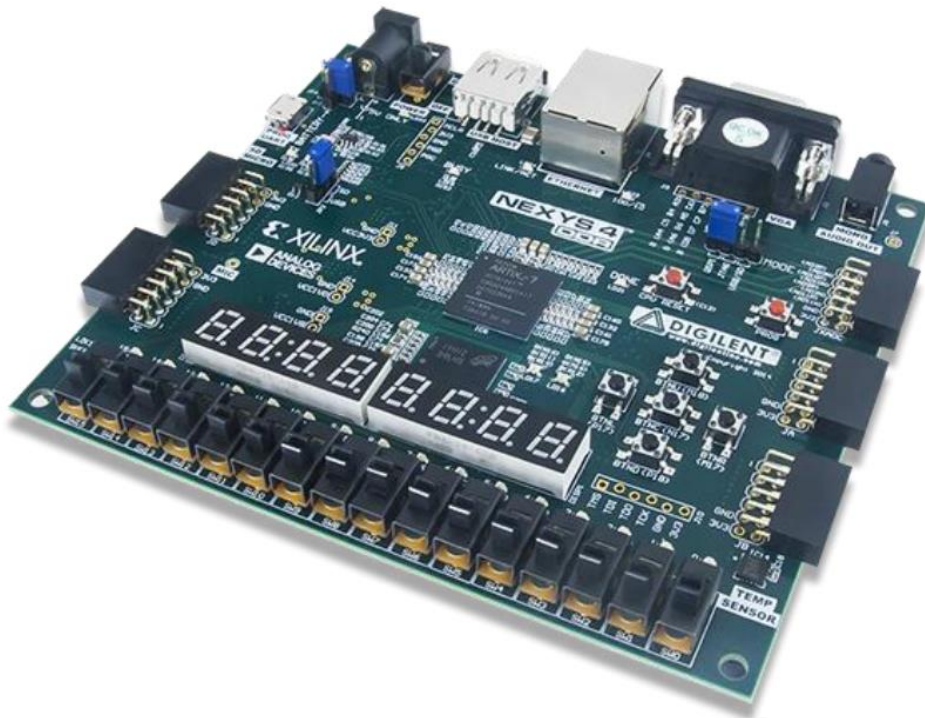


Nexys4 DDR & Vivado 2017.4

Getting Started



Department of Electrical and Electronic Engineering

Southern Univ. of Science & Technology

by Yu Yajun

Jan. 2019

Nexys4 DDR Board

The Nexys4 DDR board is a complete, ready-to-use digital circuit development platform based on the latest Artix-7™ Field Programmable Gate Array (FPGA) from Xilinx®.

The Artix-7 FPGA is optimized for high performance logic, and offers more capacity, higher performance, and more resources than earlier designs. Artix-7 100T features include:

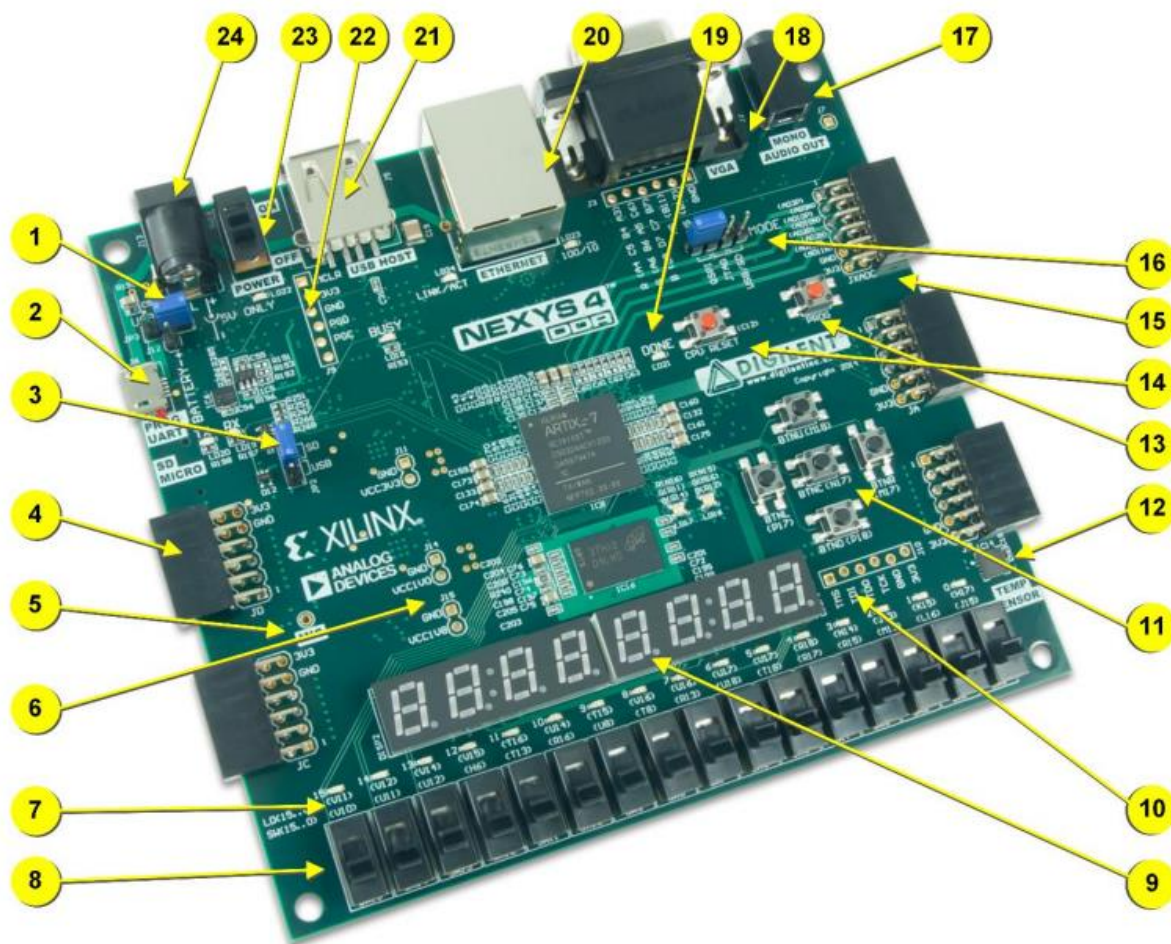
- 15,850 logic slices, each with four 6-input LUTs and 8 flip-flops
- 4,860 Kbits of fast block RAM
- Six clock management tiles, each with phase-locked loop (PLL)
- 240 DSP slices
- Internal clock speeds exceeding 450 MHz
- On-chip analog-to-digital converter (XADC)

The Nexys4 DDR also offers an improved collection of ports and peripherals, including:

- 16 user switches
- 16 user LEDs
- Two 4-digit 7-segment displays
- USB-UART Bridge
- Two tri-color LEDs
- Micro SD card connector
- 12-bit VGA output
- PWM audio output
- PDM microphone
- 3-axis accelerometer
- Temperature sensor
- 10/100 Ethernet PHY
- 128MiB DDR2
- Serial Flash
- Four Pmod ports
- Pmod for XADC signals
- USB HID Host for mice, keyboards and memory sticks
- Digilent USB-JTAG port for FPGA programming and communication

The Nexys4 DDR is compatible with Xilinx's new high-performance Vivado® Design Suite. Xilinx offers free WebPACK™ versions of Vivado Design Tools. Refer to **Vivado Installation Instruction** for Download and install the Vivado 2017.4 Webpack.

The features of Nexys4 DDR board is shown in Figure 1.



Callout	Component Description	Callout	Component Description
1	Power select jumper and battery header	13	FPGA configuration reset button
2	Shared UART/ JTAG USB port	14	CPU reset button (for soft cores)
3	External configuration jumper (SD / USB)	15	Analog signal Pmod connector (XADC)
4	Pmod connector(s)	16	Programming mode jumper
5	Microphone	17	Audio connector
6	Power supply test point(s)	18	VGA connector
7	LEDs (16)	19	FPGA programming done LED
8	Slide switches	20	Ethernet connector
9	Eight digit 7-seg display	21	USB host connector
10	JTAG port for (optional) external cable	22	PIC24 programming port (factory use)
11	Five pushbuttons	23	Power switch
12	Temperature sensor	24	Power jack

Figure 1. Nexys4 DDR board features.

Details configuration of the Nexys4 DDR board may be found from the **Nexys4 DDR™ FPGA Board Reference Manual**.

Overview of the design procedures:

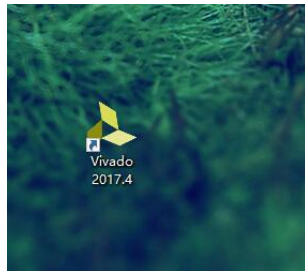
Step 1 – Creating a Project	5
Step 2 – Creating VHDL Source / Design Files	9
Step 3 – Simulating your design	15
Step 4 – Synthesizing Your Design	21
Step 5 – Creating constraint files	29
Step 6 – Implementing your Design and Verify	32
Step 7 – Configuring FPGA Board	32
Step 8 – Programming PROM of the FPGA Board	37
Appendix A – Source code of the design.....	37
Appendix B – Source code of the test bench	38
Appendix C – I/O devices on the Nexys4 DDR	39

Vivado Quick Start Tutorial

Creating a Project – Step 1

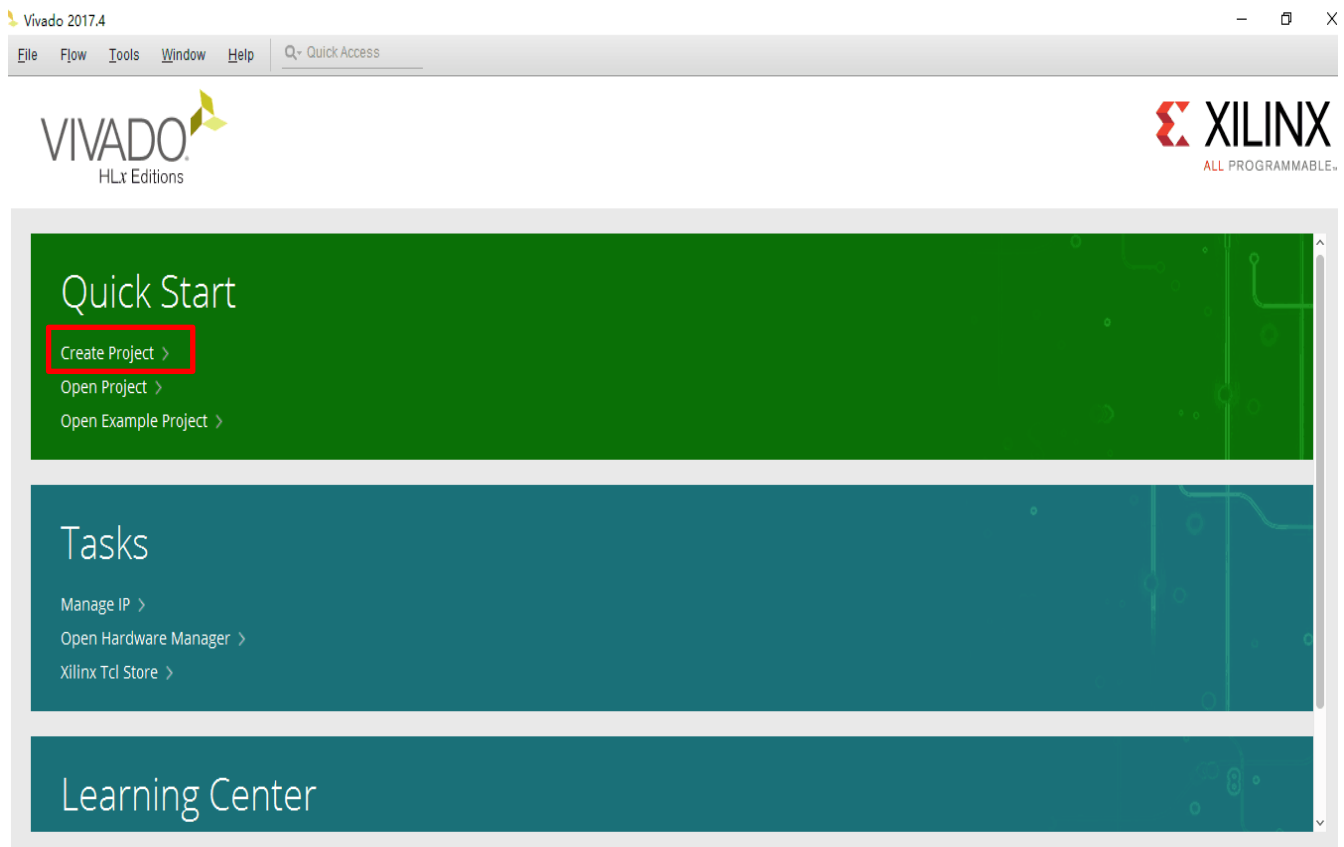
Starting the Vivado Software

Go to **Start > All Programs > Xilinx Design Tools > Vivado 2017.4 >> Vivado 2017.4** or double-click the following **desktop icon**.

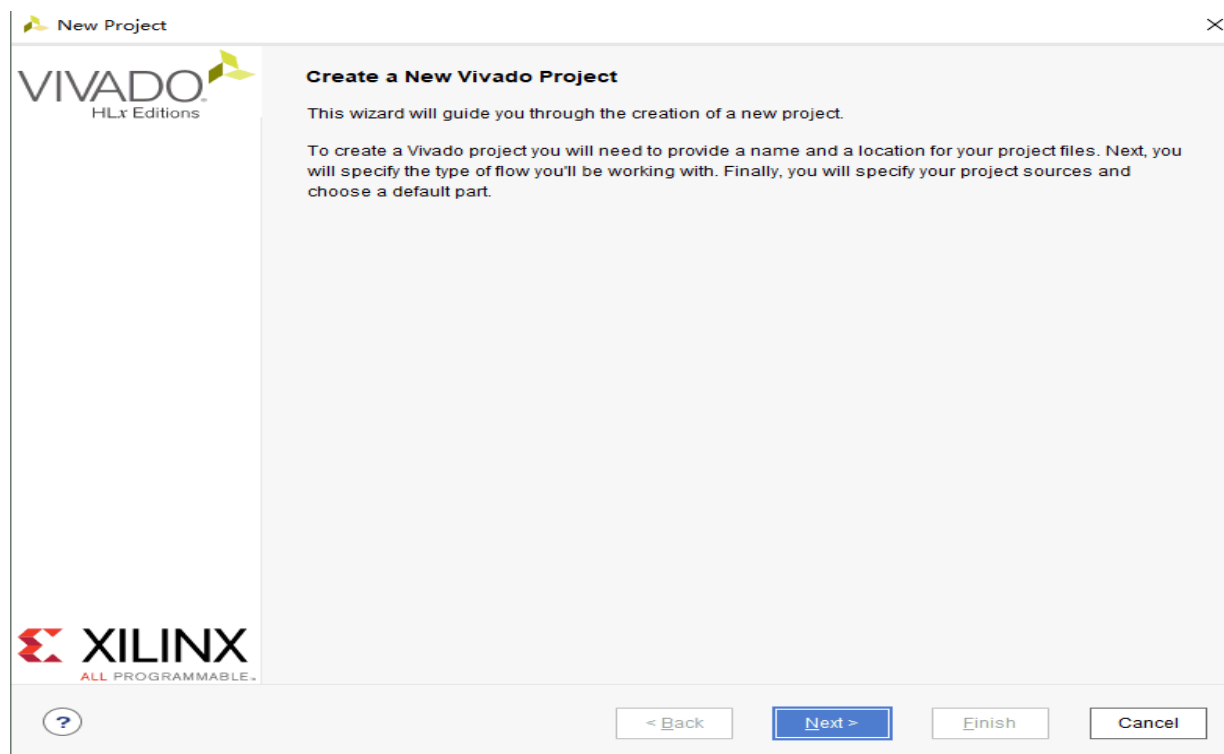


Create a New Project

In the main interface of Vivado, click **Create New Project** under **Quick Start**.

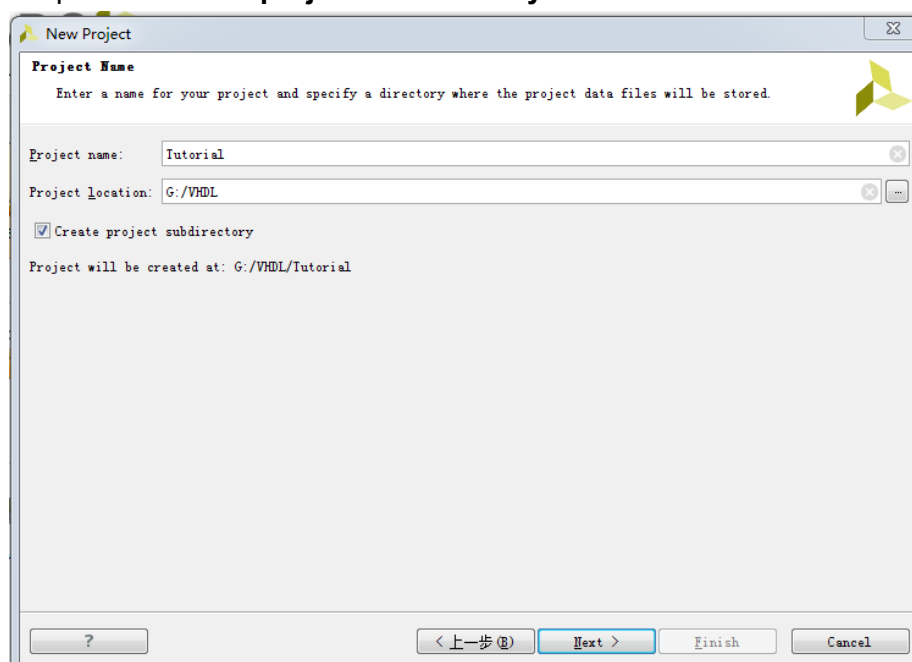


A new project wizard **Create a New Vivado Project** appears.



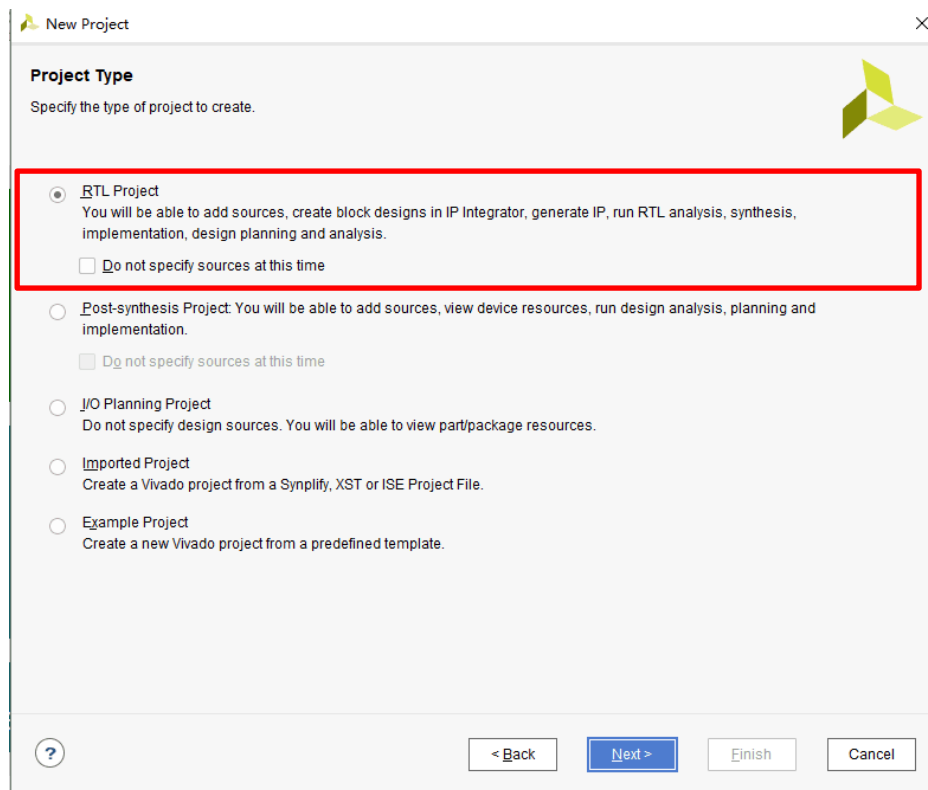
Click **Next** to continue. In the followed **Project Name** dialog, type in the following information:

- Project name: **Tutorial**
- Project location: **d:\ee332\gxyyz** (x refers to your **class number**; Monday class is 1 and Thursday class is 2. yy refers to your **group number**. z is 1 or 2, the **number in your group**). In the following, this project directory is referred to as **/project_dir**
- Tick the option of **Create project subdirectory**.

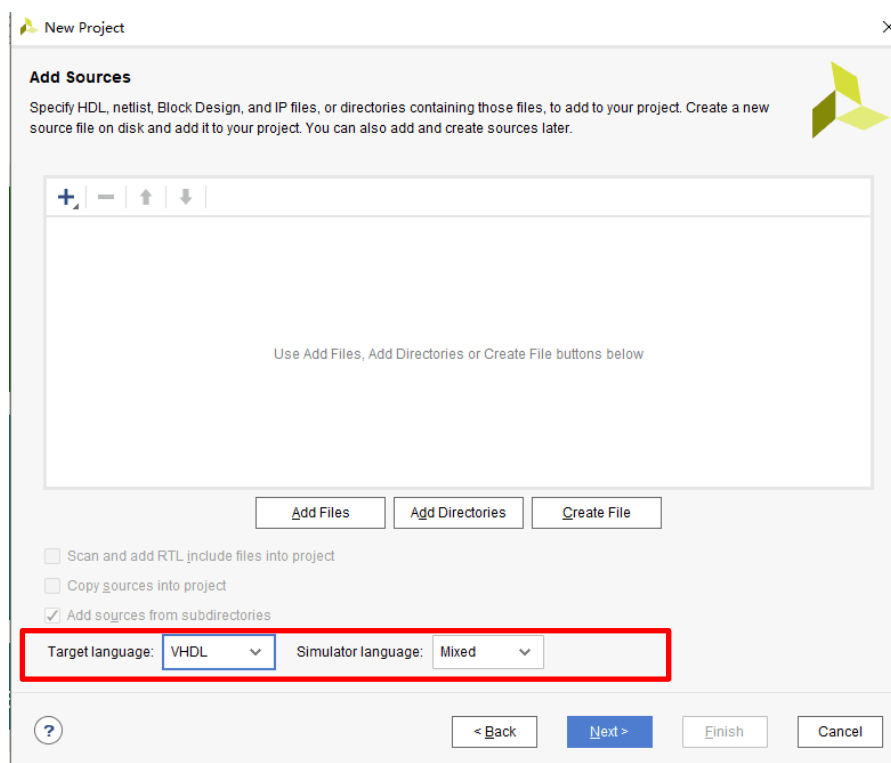


Click **Next**.

In the following appeared **Project Type** dialog, keep the default setting, i.e., select **RTL Project**, and click **next**.

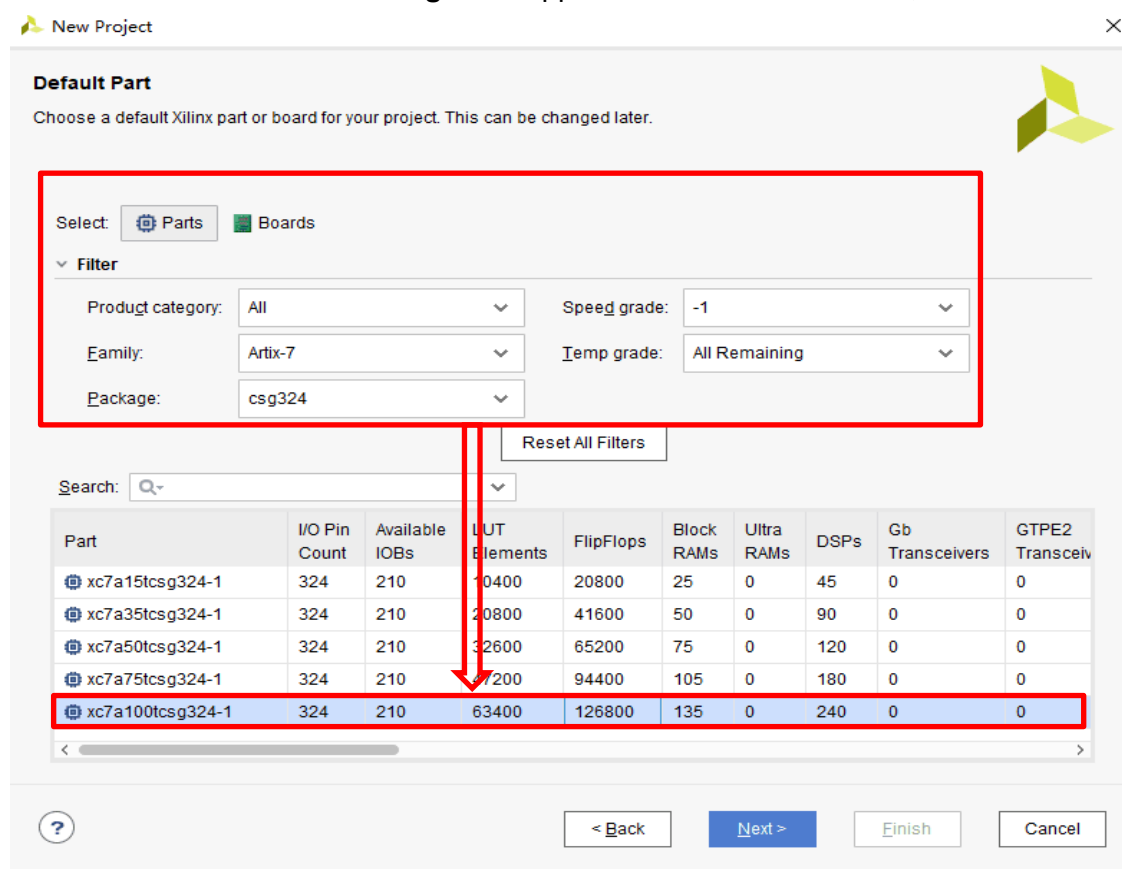


In the appeared **Add Source** dialog,



set the **Target Language** at the bottom to **VHDL**, and click **Next**.

Skip the next dialog by clicking **Next** until you see the following **Default Part** dialog. Under the **Parts**, select the following settings in the **Filter** pull down menu. And select the item **xc7a100tcsg324-1** appeared in the list as follows, and click **Next**.



Default Part

Choose a default Xilinx part or board for your project. This can be changed later.

Select: ☒ Parts ☐ Boards

Filter

Product category: All Speed grade: -1

Family: Artix-7 Temp grade: All Remaining

Package: csg324

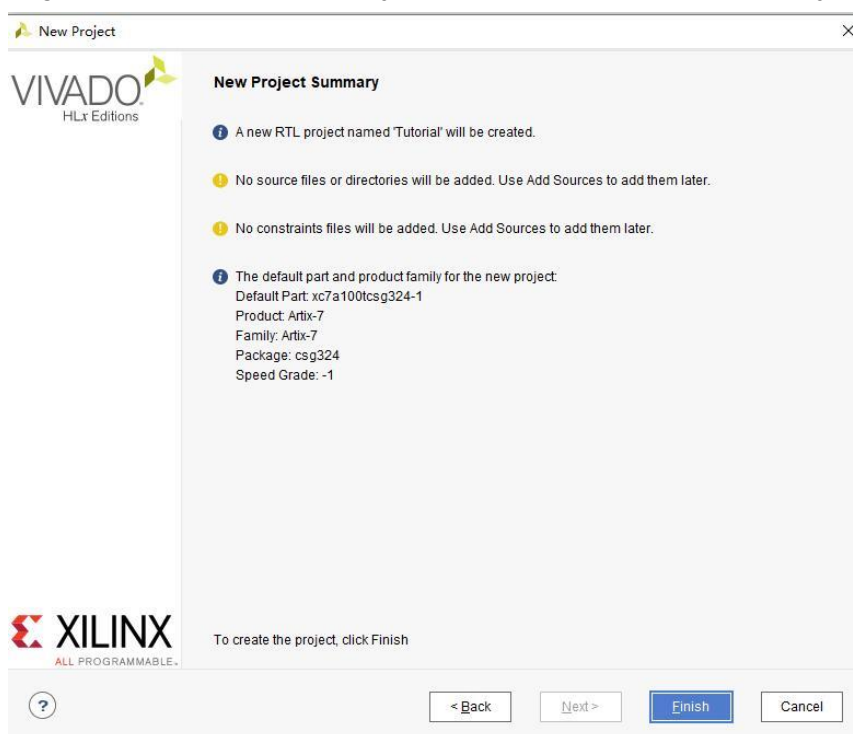
Reset All Filters

Search:

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 Transceiv
xc7a15tcsg324-1	324	210	10400	20800	25	0	45	0	0
xc7a35tcsg324-1	324	210	20800	41600	50	0	90	0	0
xc7a50tcsg324-1	324	210	32600	65200	75	0	120	0	0
xc7a75tcsg324-1	324	210	47200	94400	105	0	180	0	0
xc7a100tcsg324-1	324	210	63400	126800	135	0	240	0	0

< Back Next > Finish Cancel

The appeared dialog summarizes the new project. Click **Finish** and a new project is created.



New Project Summary

- A new RTL project named 'Tutorial' will be created.
- No source files or directories will be added. Use Add Sources to add them later.
- No constraints files will be added. Use Add Sources to add them later.
- The default part and product family for the new project:
 Default Part: xc7a100tcsg324-1
 Product: Artix-7
 Family: Artix-7
 Package: csg324
 Speed Grade: -1

To create the project, click Finish

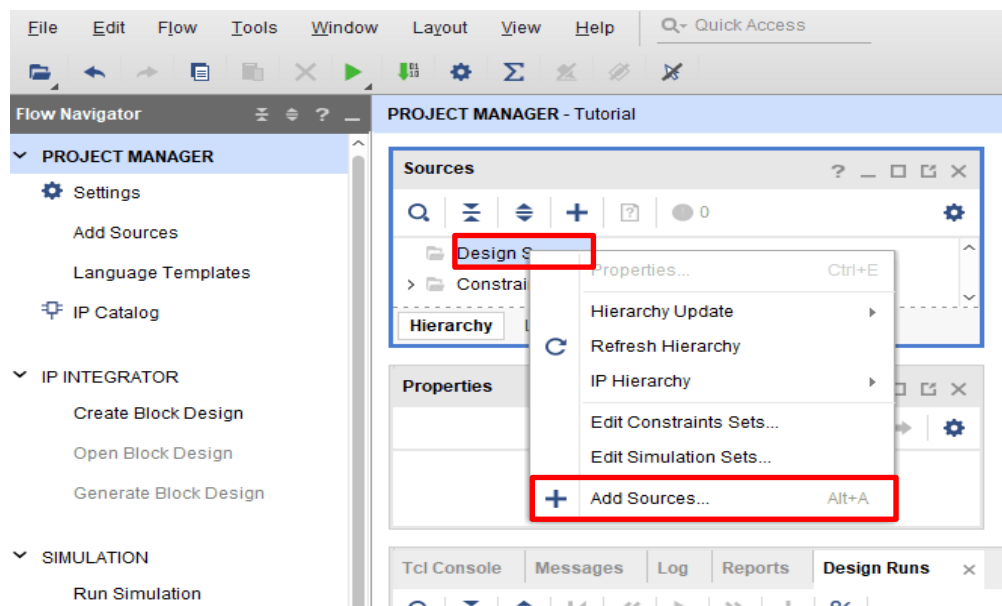
< Back Next > Finish Cancel

Creating VHDL Source / Design Files – Step 2

In this section, you will create the top HDL file for your design.

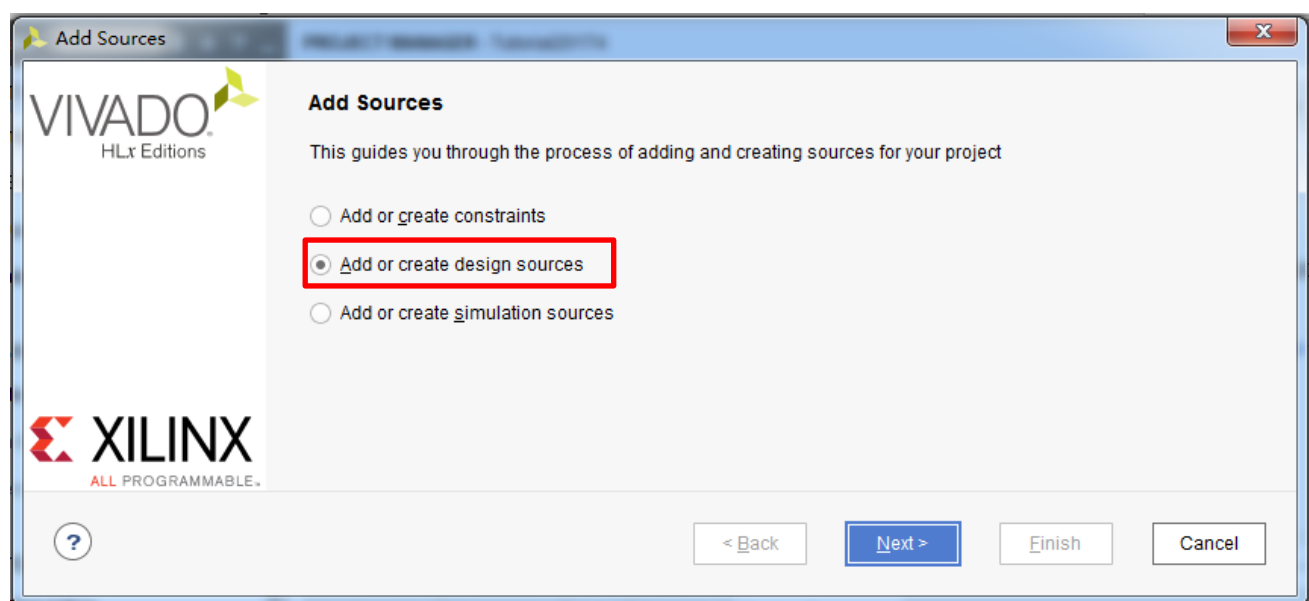
Creating a VHDL Source

Under Vivado main interface, in the region of **Project Manager**, right click **Design Sources** as follows,

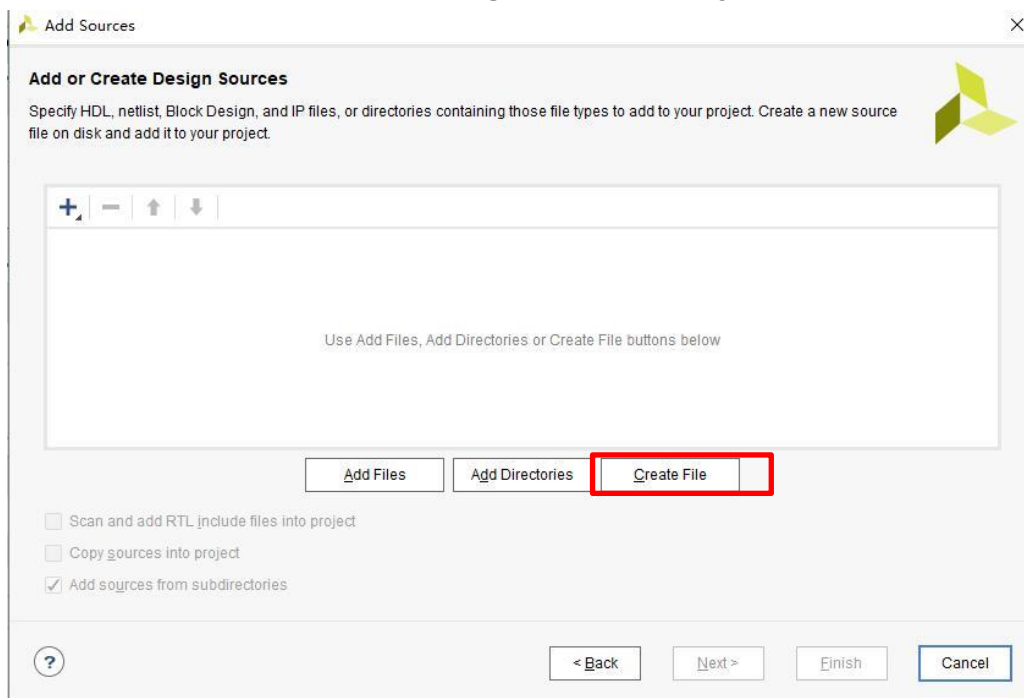


and select the **Add Sources ...** in the pull down menu.

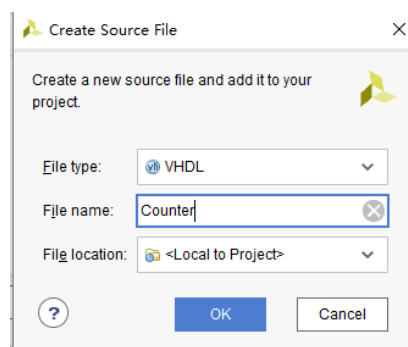
In the appeared **Add Sources** window, select **Add or create design sources**, and then click **Next** to continue.



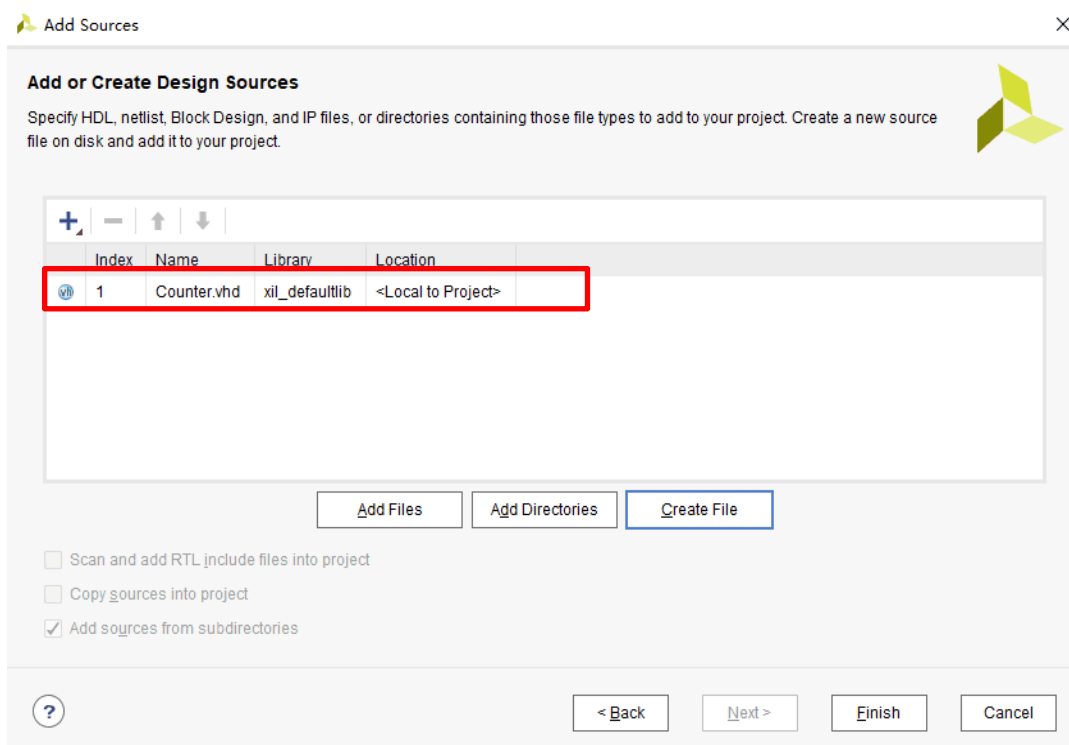
In the appeared **Add or Create Design Sources** dialog, click **Create File**.



In the popped up **Create Source File** dialog, select **VHDL** as the File type, and key in **Counter** in the **File name**.



Keep all the other settings, and click **Ok** to return to the **Add or Create Design Sources** as below.



Note that a source file **Counter.vhd** has been created. Keep all the other settings, and click **Finish** to complete the adding resource of **Counter**.

Defining input and output ports

A **Define Module** window is automatically popped out. In this window, key in the following port settings.

Port Name	Direction	Bus	MBS	LBS
CLOCK	In			
RESET	In			
DIRECTION	In			
COUNT_OUT	Out	Tick	3	0

Thus, we have the following window. Note the editable Entity name and Architecture name, and the newly set input/output ports.

Define Module

Define a module and specify I/O Ports to add to your source file.
 For each port specified:
 MSB and LSB values will be ignored unless its Bus column is checked.
 Ports with blank names will not be written.

Module Definition

Entity name: Counter

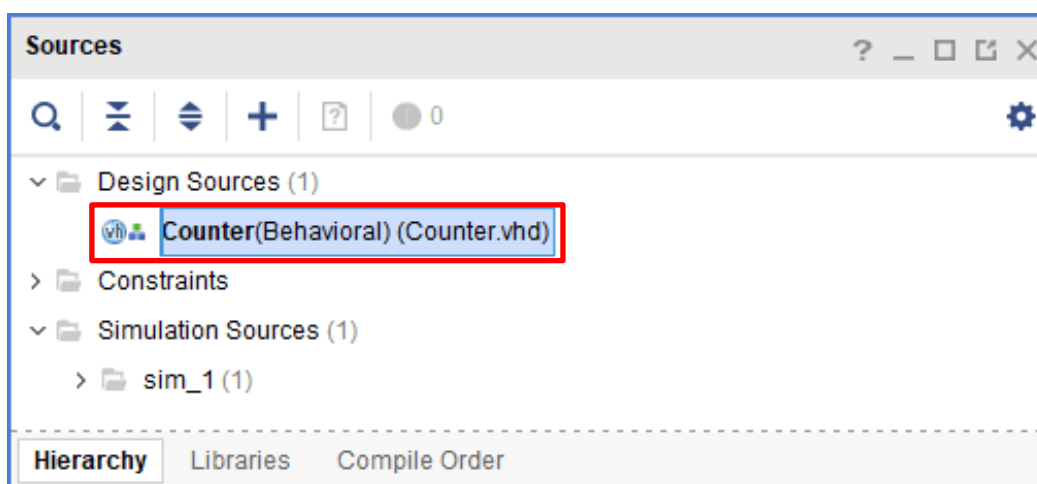
Architecture name: Behavioral

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
CLOCK	in	<input type="checkbox"/>	0	0
RESET	in	<input type="checkbox"/>	0	0
DIRECTION	in	<input type="checkbox"/>	0	0
COUNT_OUT	out	<input checked="" type="checkbox"/>	3	0
	in	<input type="checkbox"/>	0	0

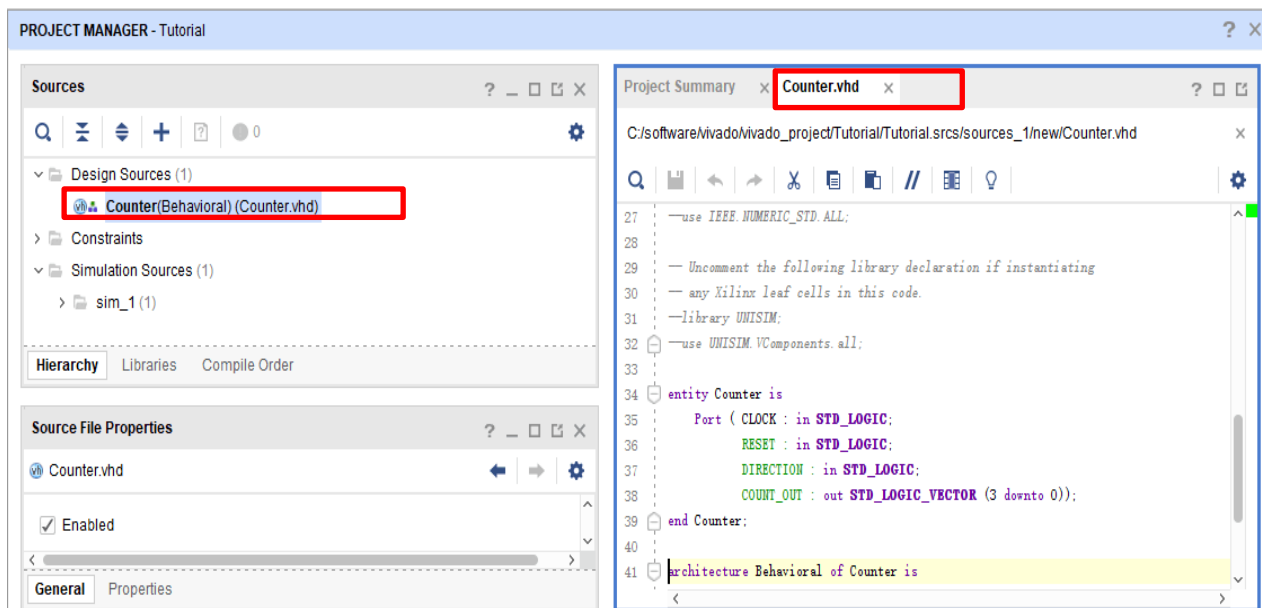
OK Cancel

Click **Ok** and you will see that a **Counter** module is created under the **Design Sources**.



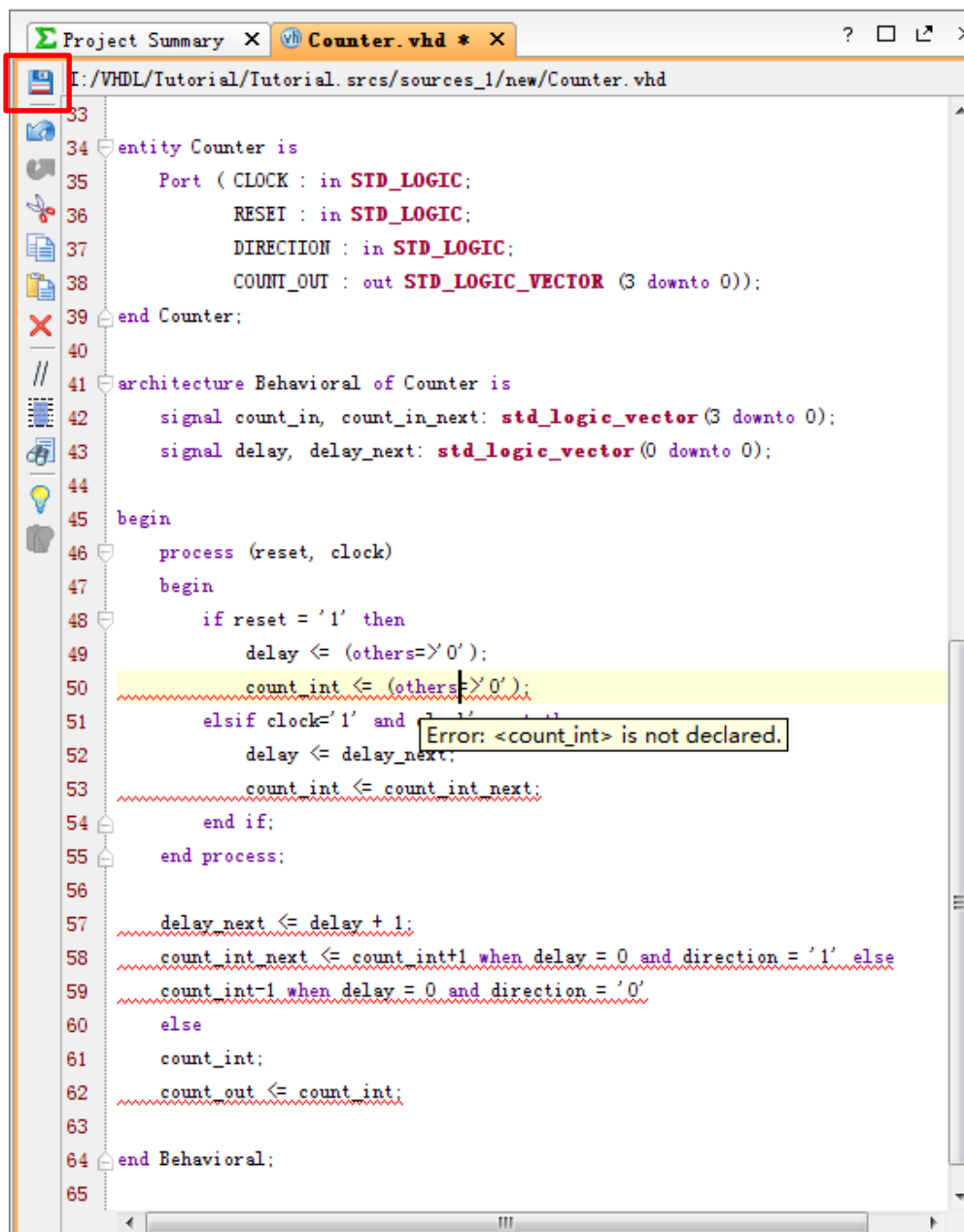
Editing of the VHDL Source

Double click the **Counter – Behavioral (Counter.vhd)** in the **Project Manager**, and the basic code of **Counter.vhd** is shown in the right window as follows.



Edit the source code as follows:

- Append the following signal declaration after the “architecture Behavioral of Counter is”
 signal count_in, count_in_next: std_logic_vector(3 downto 0);
 signal delay, delay_next: std_logic_vector(0 downto 0);
- Append the following process after the first “begin”
 process (reset, clock)
 begin
 if reset = '1' then
 delay <= (others=>'0');
 count_int <= (others=>'0');
 elsif clock='1' and clock'event then
 delay <= delay_next;
 count_int <= count_int_next;
 end if;
 end process;
- Append the following lines after the “end process”
 delay_next <= delay + 1;
 count_int_next <= count_int+1 when delay = 0 and direction = '1' else
 count_int-1 when delay = 0 and direction = '0'
 else
 count_int;
 count_out <= count_int;
- **Save** the file by click the **first icon** on the **left graphic menu bar**.
- You have now created the VHDL source for the tutorial project as follows.



Checking the Syntax of the New Counter Module

In the **Counter.vhd** source code, the lines with **red underline ripples** imply that there are syntax errors. Placing **cursor** onto the lines shows the hints of the errors. Examples can be seen in the above figure.

Since **count_in** is declared in the declaration session of architecture, but **count_int** is used in the body of architecture, so that **count_int** is not recognized. Change all **count_int** and **count_int_nex** in the body of architecture to **count_in** and **count_in_next**, respectively. In addition, arithmetic operations are applied to signals of type **std_logic** and **std_logic_vector**. Therefore, **IEEE.STD_LOGIC_ARITH.ALL** and **IEEE.STD_LOGIC_UNSIGNED.ALL** must be included, i.e., add

```
use IEEE.STD_LOGIC_ARITH.ALL;  
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

after

```
library IEEE;  
use IEEE.STD_LOGIC_1164.ALL;
```

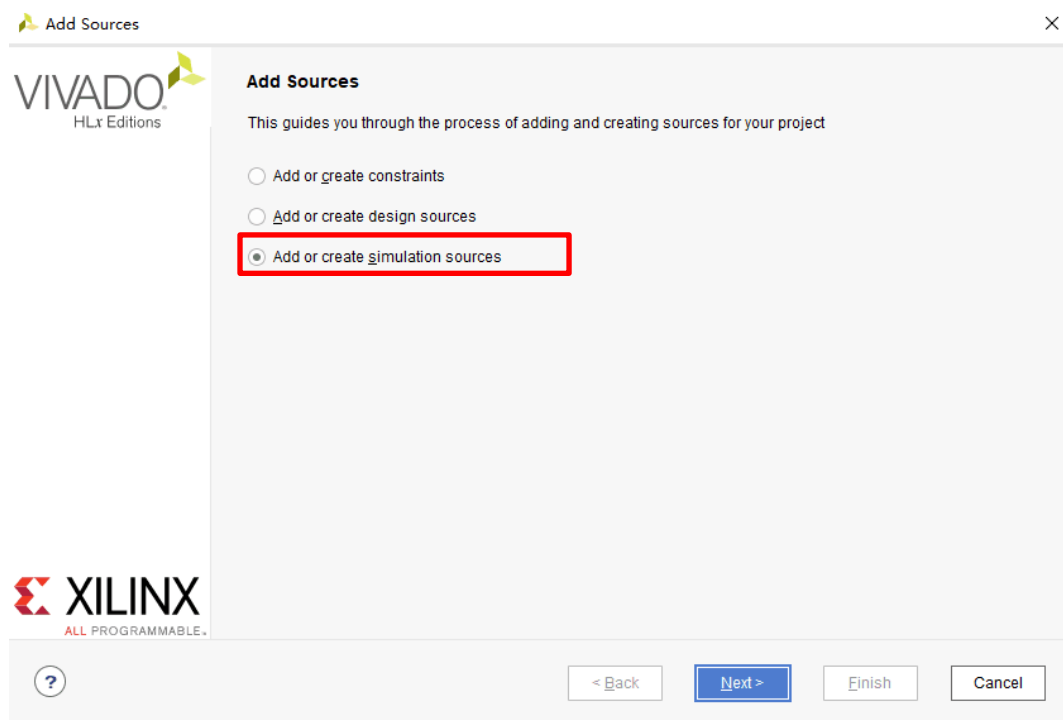
in the beginning.

- Save the file, and you will see that all syntax errors have been cleared..
- Refer to appendix A for the complete code.

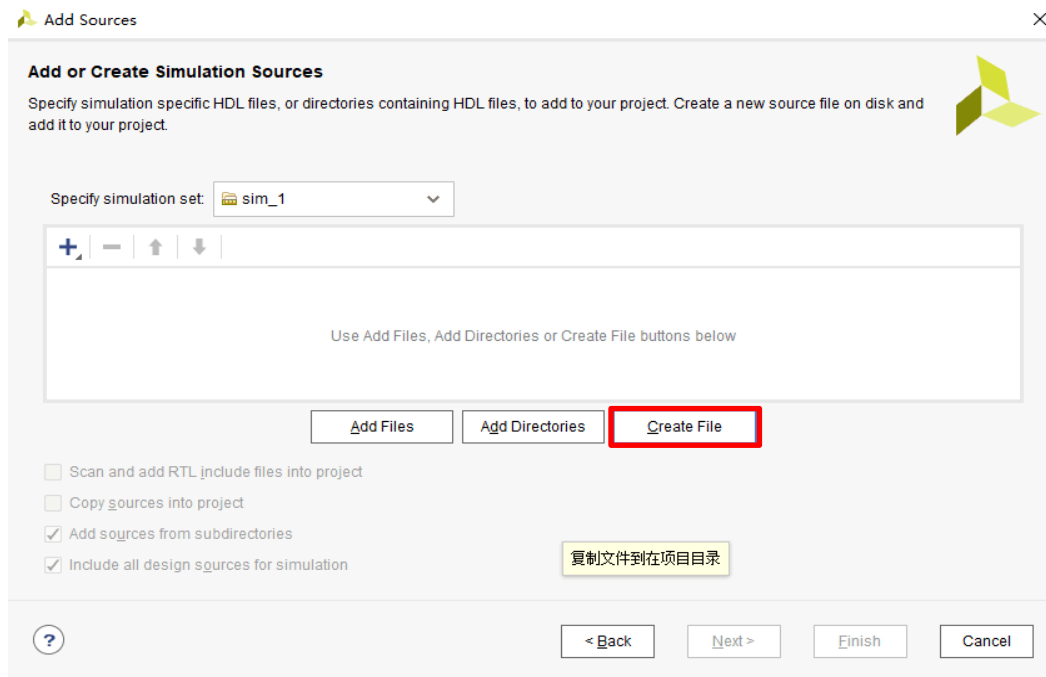
Simulating your design – Step 3

Creating a Testbench File

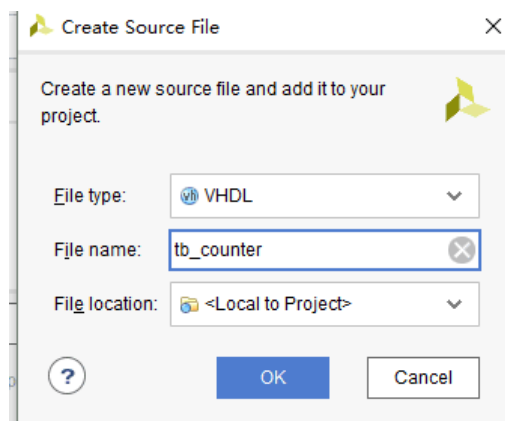
- Create a test bench containing input stimulus to verify the functionality of the counter module.
- Create a new test bench source by selecting **Add Source ...** (right click on **Design sources**).
- In the **Add Sources** Wizard, select **Add or create simulation sources**. Click **Next**



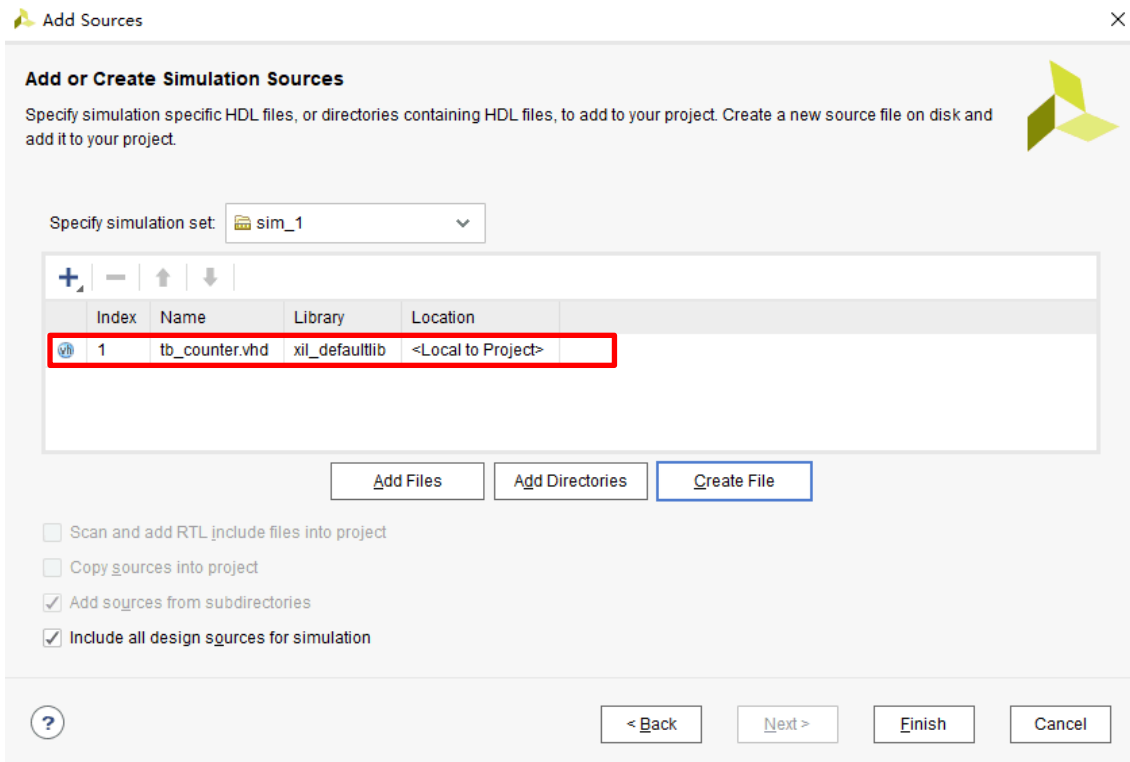
In the **Add or Create Simulation Sources** dialog, click **Create File**.



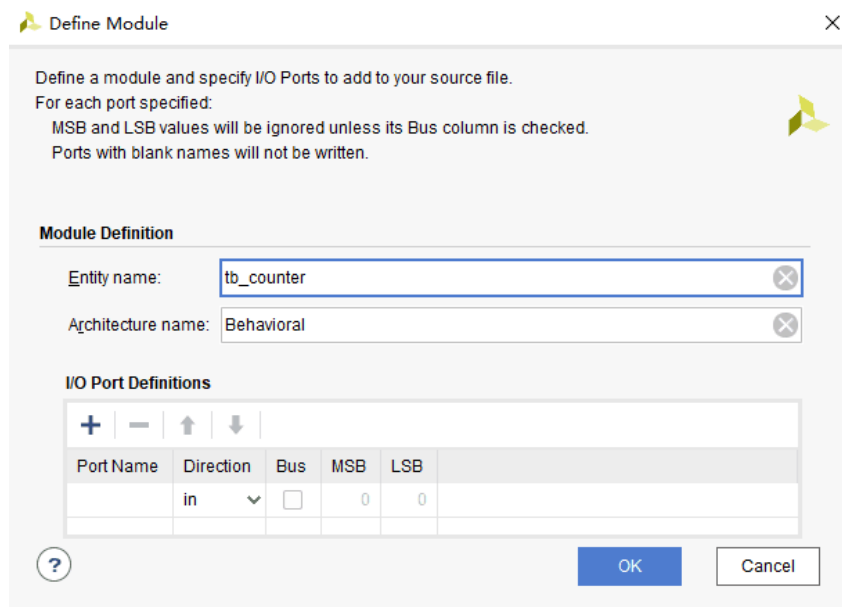
Key in **tb_counter** as the **Filter Name** in the followed **Create Source File** dialog as follows:



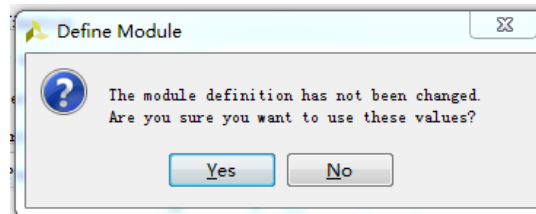
By clicking **OK**, it returns to the **Add or Create Simulation Sources**. Check the entry in the windows as follows and click **Finish**.



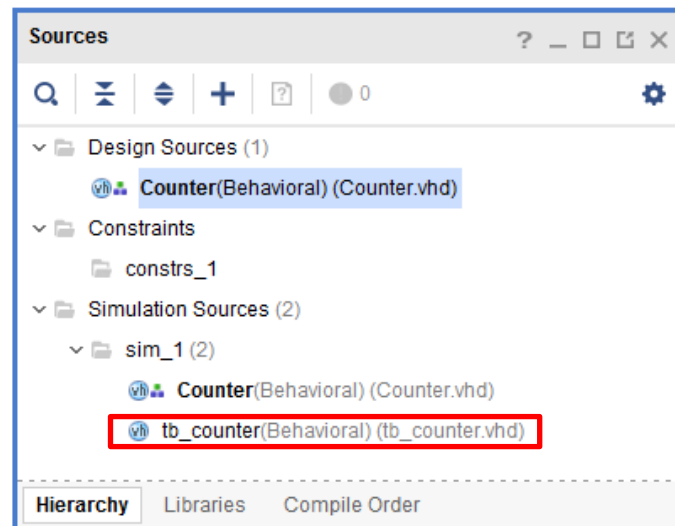
In the followed **Define Module** dialog, keep all the default settings, and click **OK**. Note that testbench is a design entity without input/output ports.



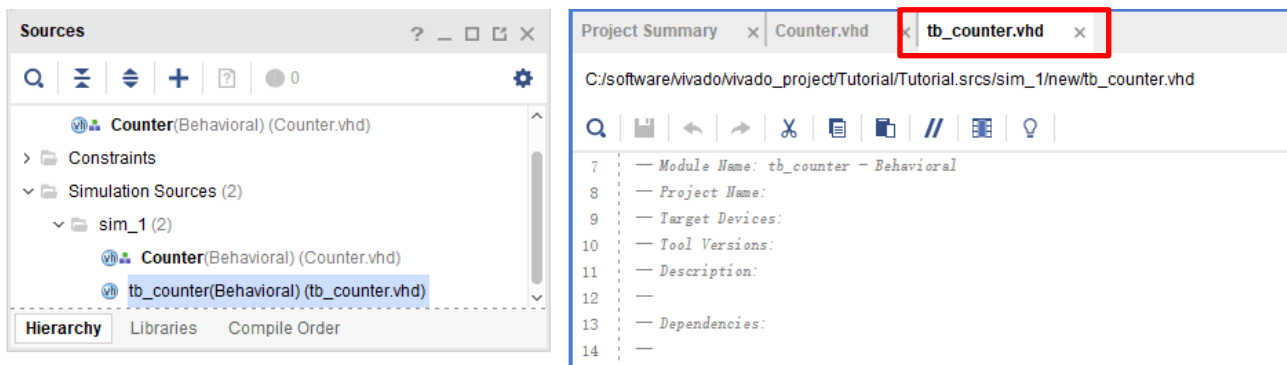
Click **Yes** in the followed popped up window.



You will see in the main interface of Vivado under **Sources**, a .vhd file **tb_counter** appears under the directory **Simulation Sources/Sim_1** as follows.



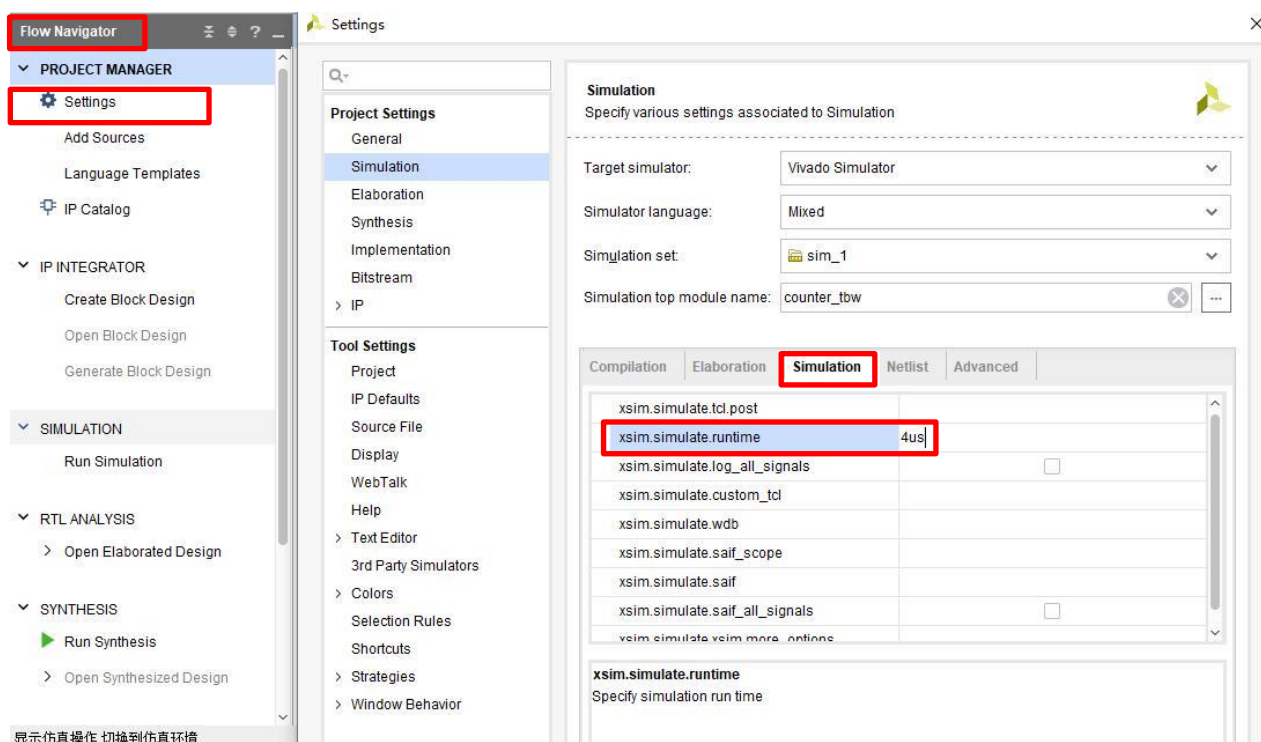
Double click **tb_counter**, and the basic code frame of the testbench file is shown in the window on the right as follows.



Replace the complete testbench code by the code in **Appendix B**.

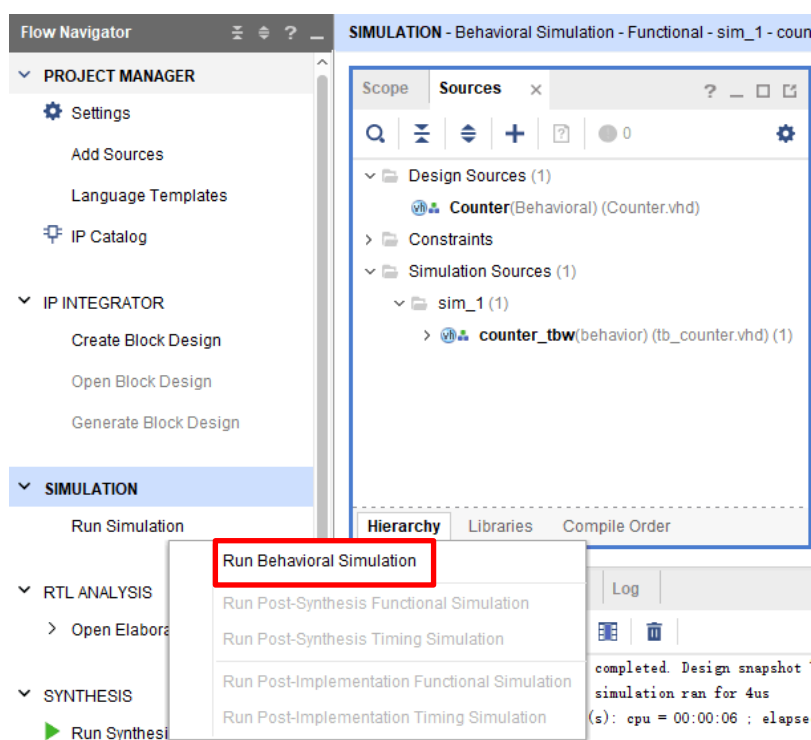
Verifying Functionality using Behavioral Simulation

Click **Settings** under **Project Manager** in the left **Flow Navigator** window. A **Settings** dialog appears. Select the **Simulation** under **Project Setting**, and set **xsim.simulate.runtime*** to **4us** as follows.

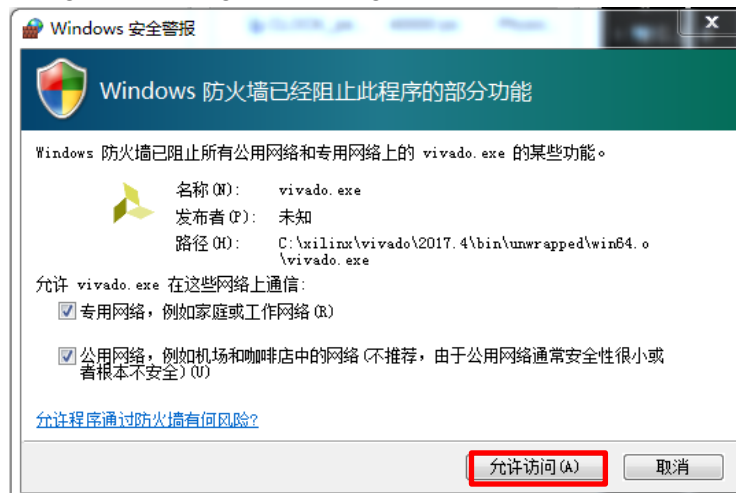


Click **OK** to return.

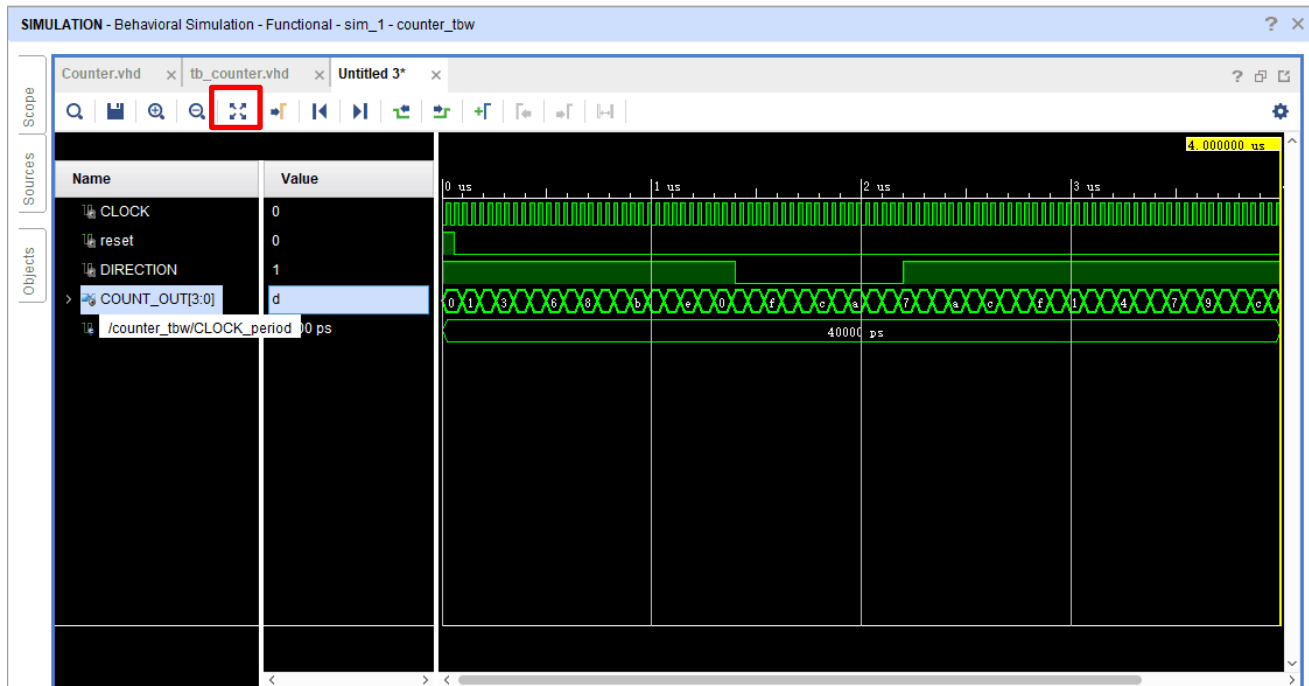
Click **Run Simulation** under the **Flow Navigator**, and select **Run Behavioral Simulation** in the floating menu.



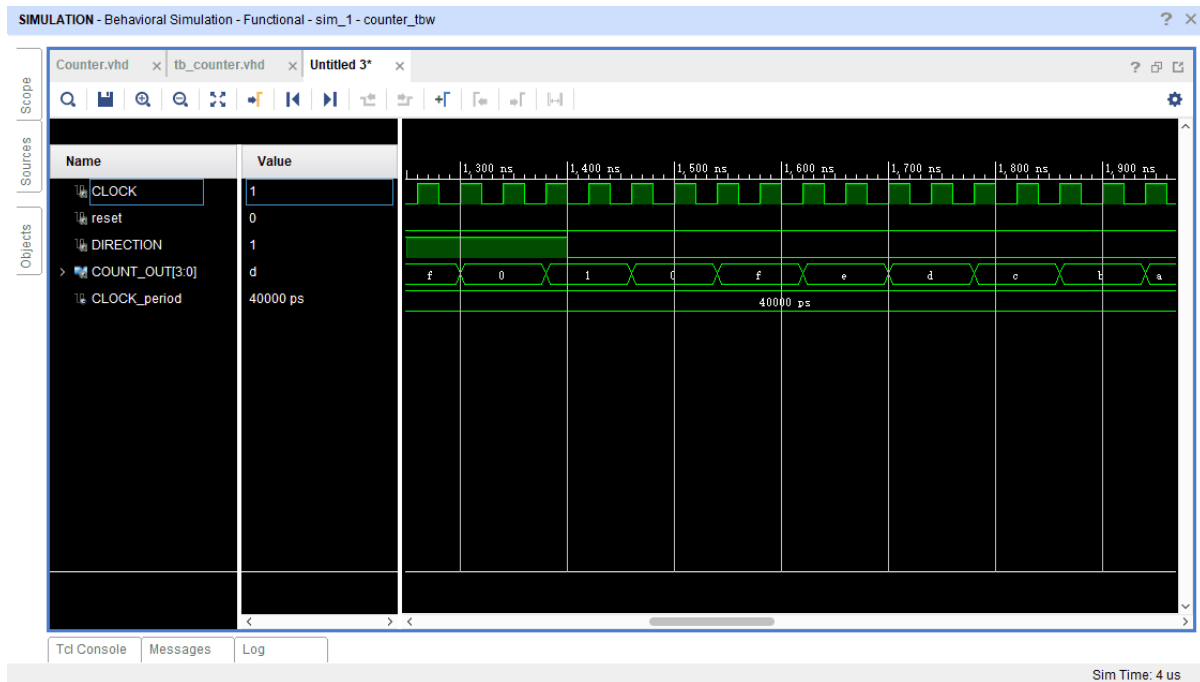
If a warning window asking for allowing visit through a firewall, allow the request.



Waveforms are shown in the right window. Click **Zoom Fit** at the left **graphic menu bar** to show the entire sets of waveforms.



You may also click **Zoom in** or **Zoom out** to view the waveform, for example to adjust the waveforms such that you can see the values of the **COUNT_OUT**, as in the following figure.

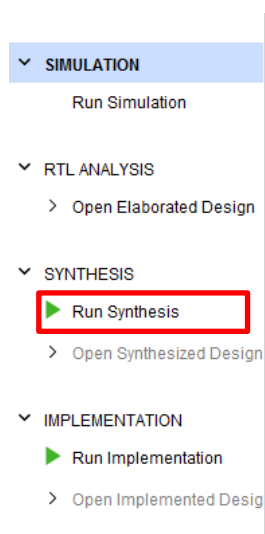


Verify the function of the Counter.

Synthesizing Your Design – Step 4

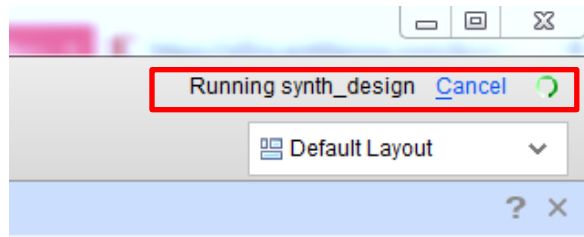
Synthesize the design

Once the behavior simulation shows the function of the circuit is correct, we can synthesis the design by click **Run Synthesis** under **Synthesis** in the left **Flow Navigator** window as follows.

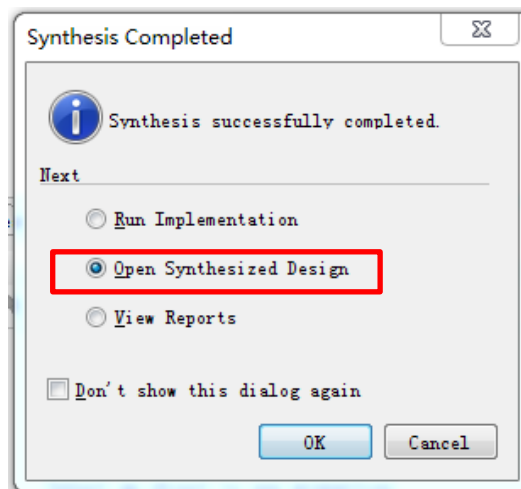


In the followed **Launch Run**, click **OK** to start the synthesis. You may **tick** the option **Don't show this dialog again**, so that this dialog will not show thereafter.

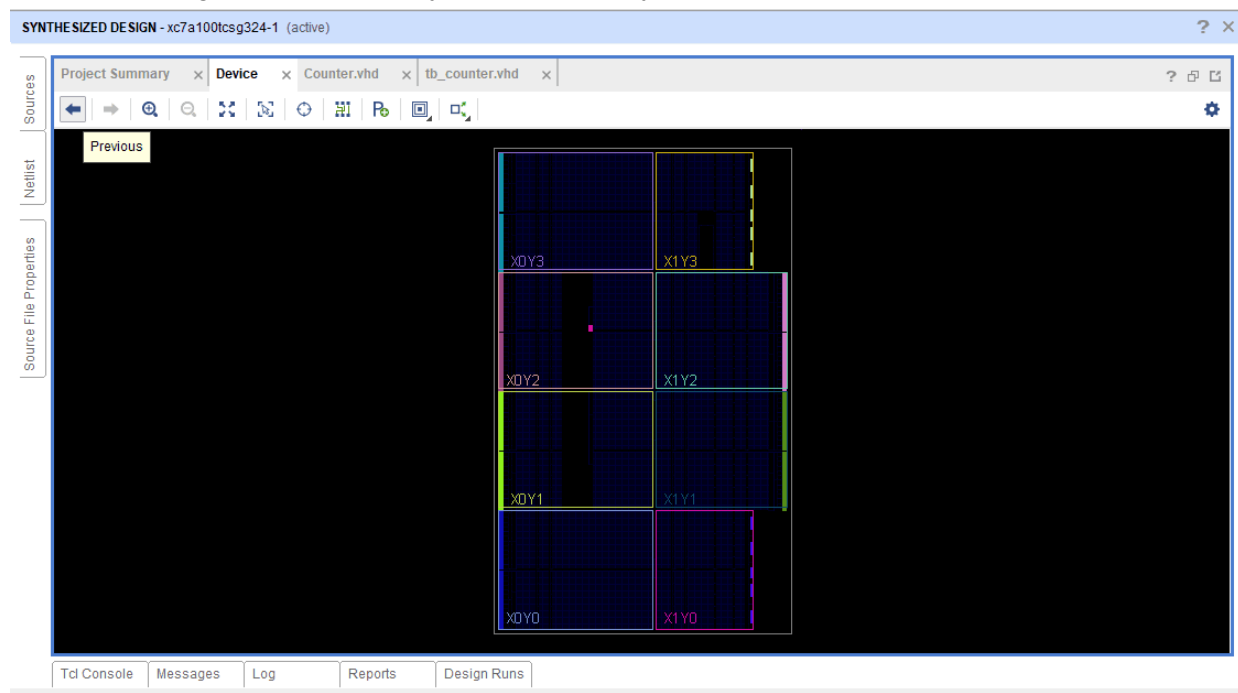
A **synthesis status bar** will be shown on the top right corner of the window.



Once it is completed, select **Open Synthesized Design** in the popped up dialog and click **OK**.

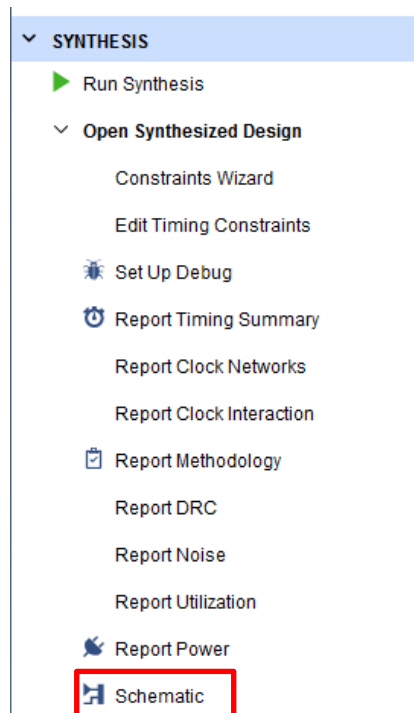


Since our design is simple, it only occupies a very small part of the FPGA chip.

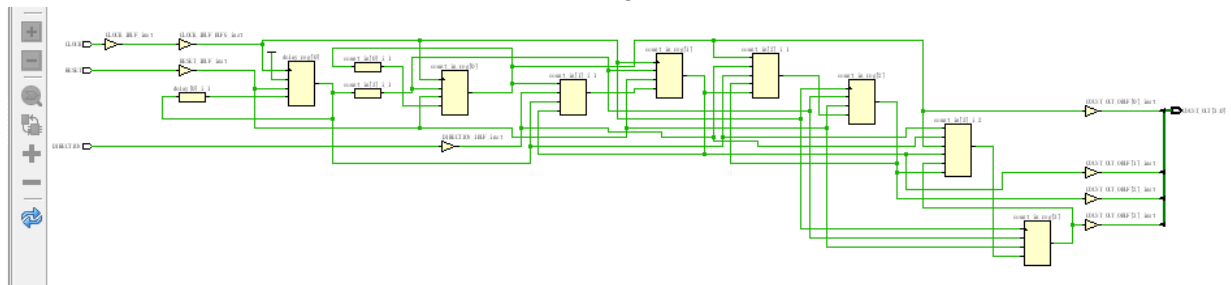


[View Schematic and Project Summary](#)

In addition, you can view the **schematic** of your design by click **Synthesis -> Synthesized Design -> Schematic** under **Flow Navigator**.

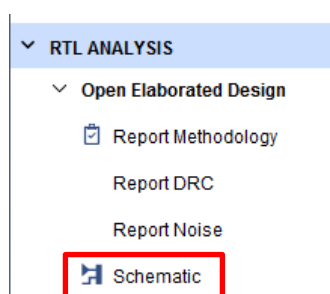


A schematic of the Counter will be shown on the right window.

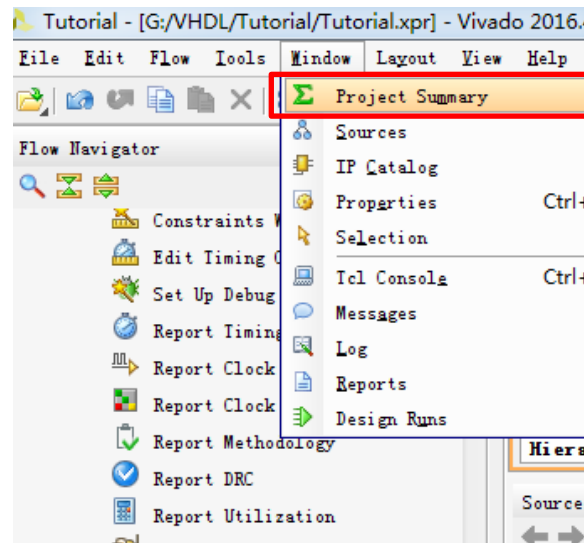


You can zoom to view the schematic in detail.

You may also view the RTL design under **Flow Navigator -> RTL analysis -> Open Elaborated Design -> Schematic**.



If the **Project Summary** window is not opened in the right part of the main interface, click **Window -> Project Summary** from the **top menu**.



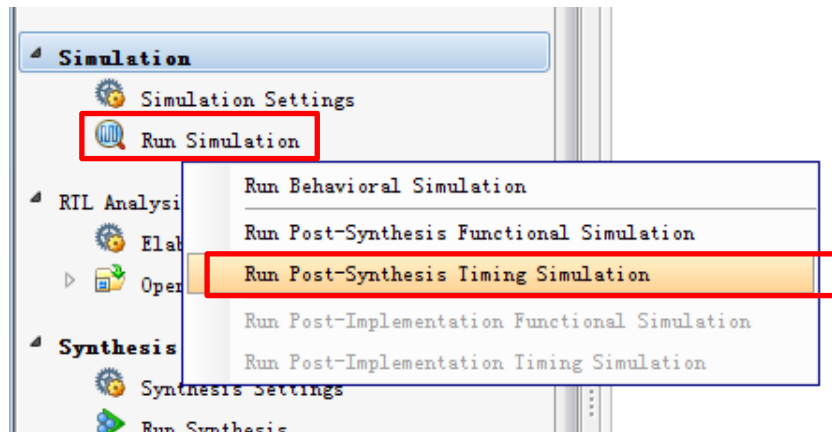
From the **Project Summary**, you can see the overview of the design. Note in particular the **Utilization**, in which the number of **LUT** and **FF**, both used and in total, are shown in graphic and table forms.

 A screenshot of the 'Project Summary' window in Vivado. The window title bar is highlighted with a red rectangle. The window displays project information, synthesis status, and timing data.

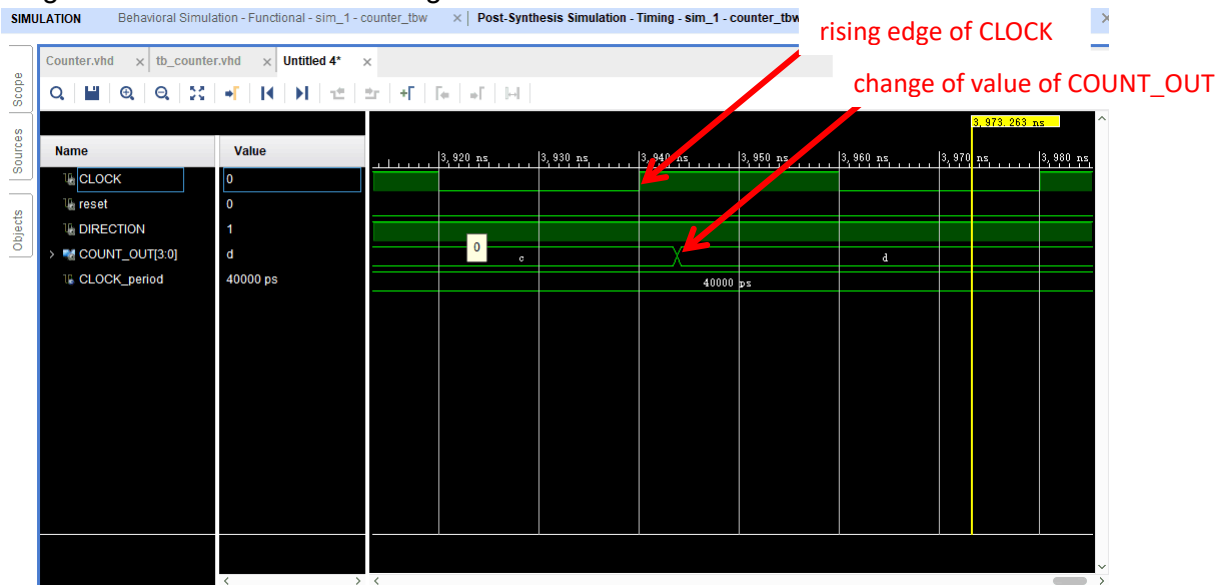
Project Information	
Project name:	Tutorial
Project location:	C:/software/Nivado/Nivado_project/Tutorial
Product family:	Artix-7
Project part:	xc7a100tcsq324-1
Top module name:	Counter
Target language:	VHDL
Simulator language:	Mixed

Synthesis	Implementation
Status: ✔ Complete	Status: ✔ Complete
Messages: ! 2 warnings	Messages: ! 2 warnings
Part: xc7a100tcsq324-1	Part: xc7a100tcsq324-1
Strategy: Vivado Synthesis Defaults	Strategy: Vivado Implementation Defaults
Report Strategy: Vivado Synthesis Default Reports	Report Strategy: Vivado Implementation Default Reports
	Incremental compile: None

DRC Violations	Timing
Summary: ! 2 critical warnings ! 1 warning Implemented DRC Report	<div>Setup Hold Pulse Width</div> <div>Worst Negative Slack (WNS): NA</div> <div>Total Negative Slack (TNS): NA</div> <div>Number of Failing Endpoints: NA</div> <div>Total Number of Endpoints: NA</div>



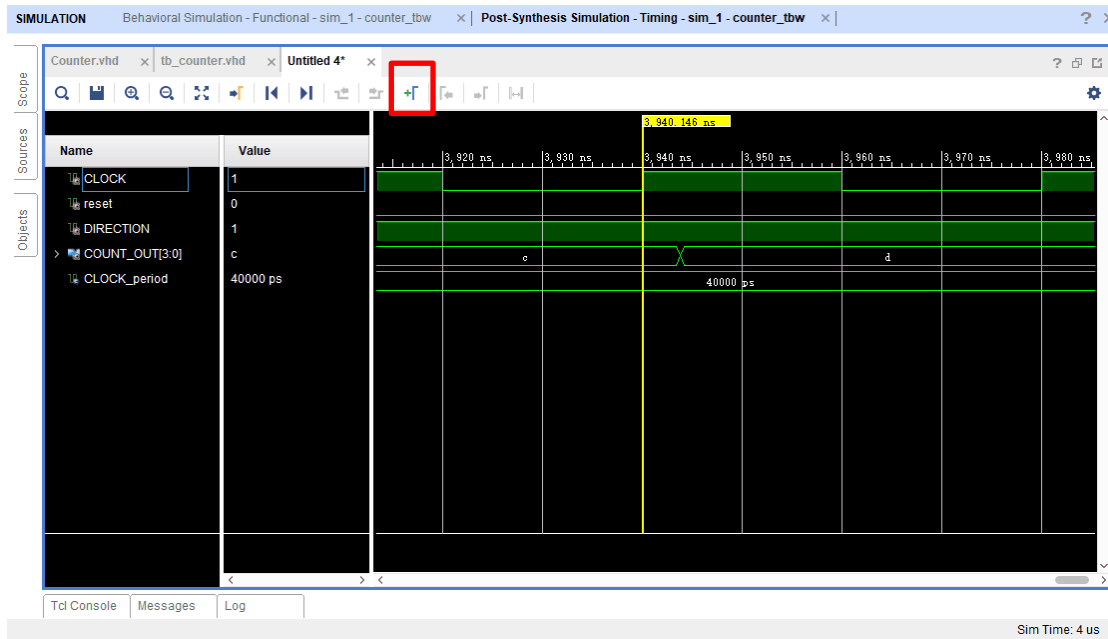
By zooming in the waveform of the timing simulation result as follows,



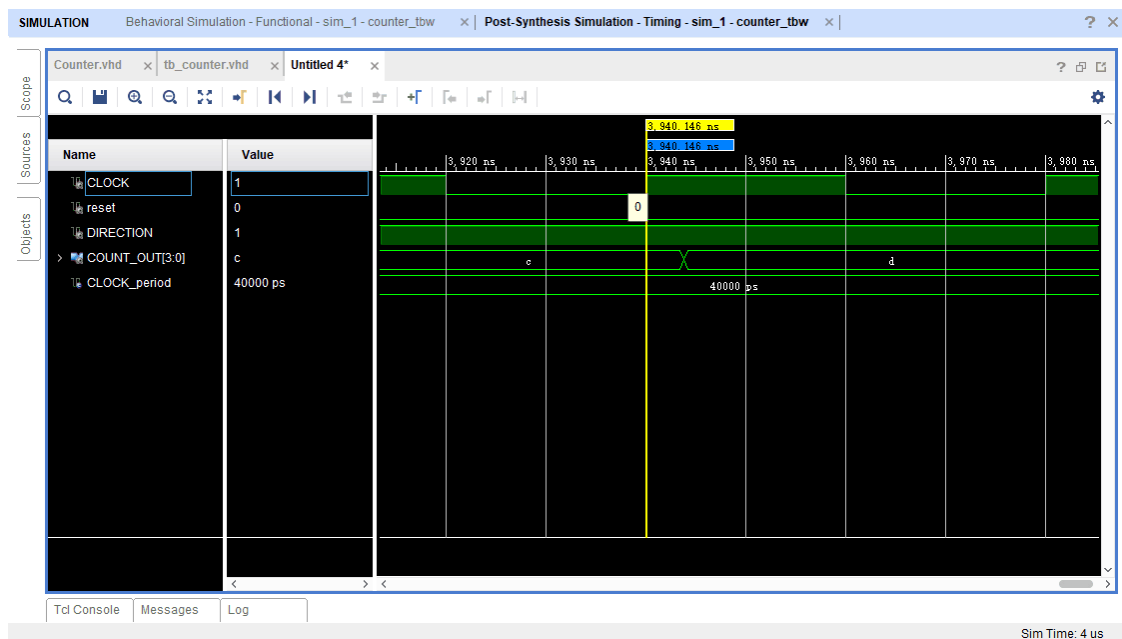
you will see that the **change of the value** of **COUNT_OUT** occurs after the rising edge of the **clock** by some delays.

Using Markers in Simulation Waveforms

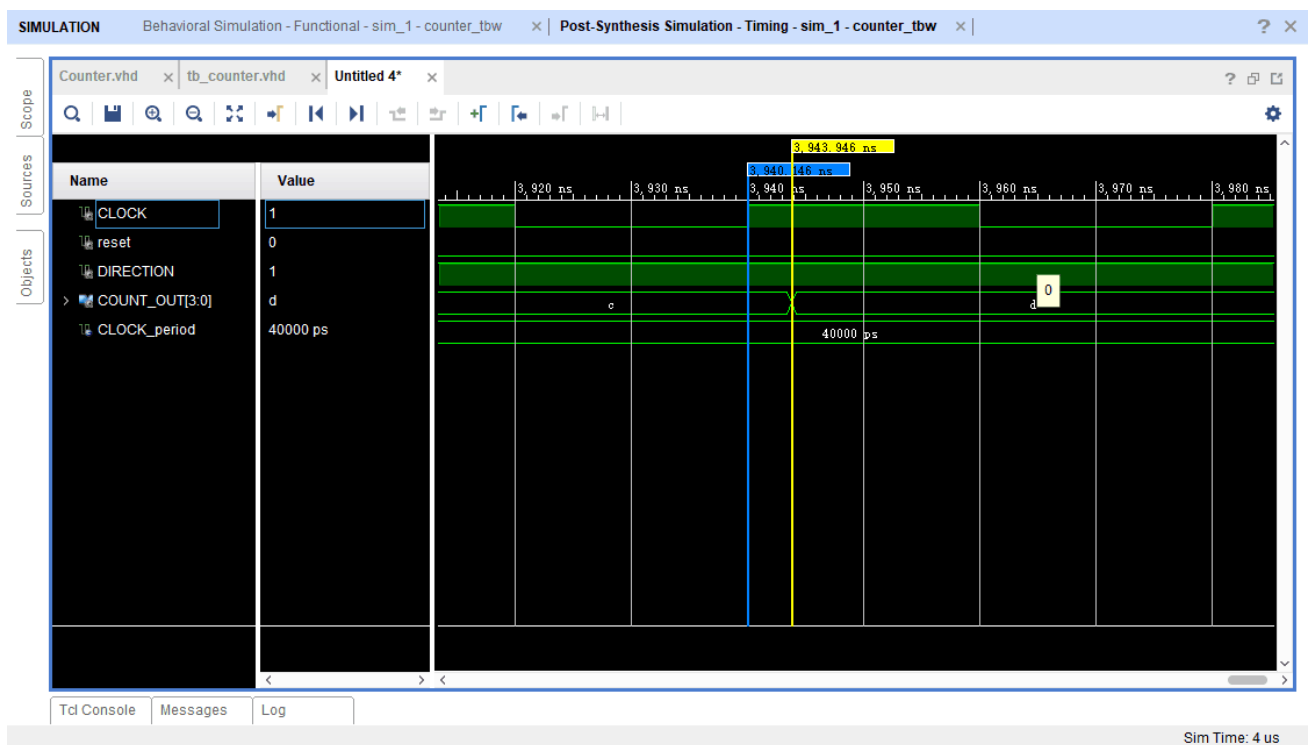
You may measure the delay by using **markers**. Click any of rising edge of the **CLOCK**, a yellow marker is shown.



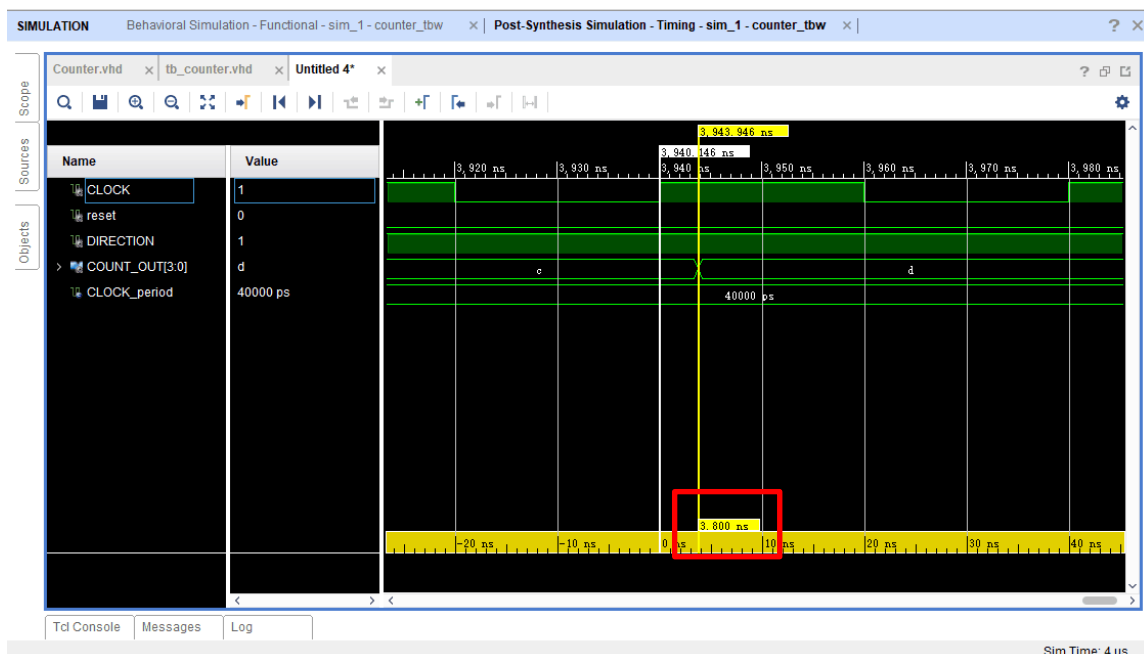
Click **Add Marker** on the left graphic menu bar, a new **blue marker** appears at the same location of the yellow marker.



By **clicking at the change of the value of COUNT_OUT** occurring right after the selected rising edge of the CLOCK, the **yellow marker** is moved to the change of **COUNT_OUT** as follows:



Now, you may click the **blue marker** to **toggle the display of a floating ruler** as follows:



Note that when the ruler is displayed, the last icon of the **graphic menu bar** is highlighted. Using the ruler, you can easily **measure the delay** between the rising edge of the CLOCK and the change of the counting values.

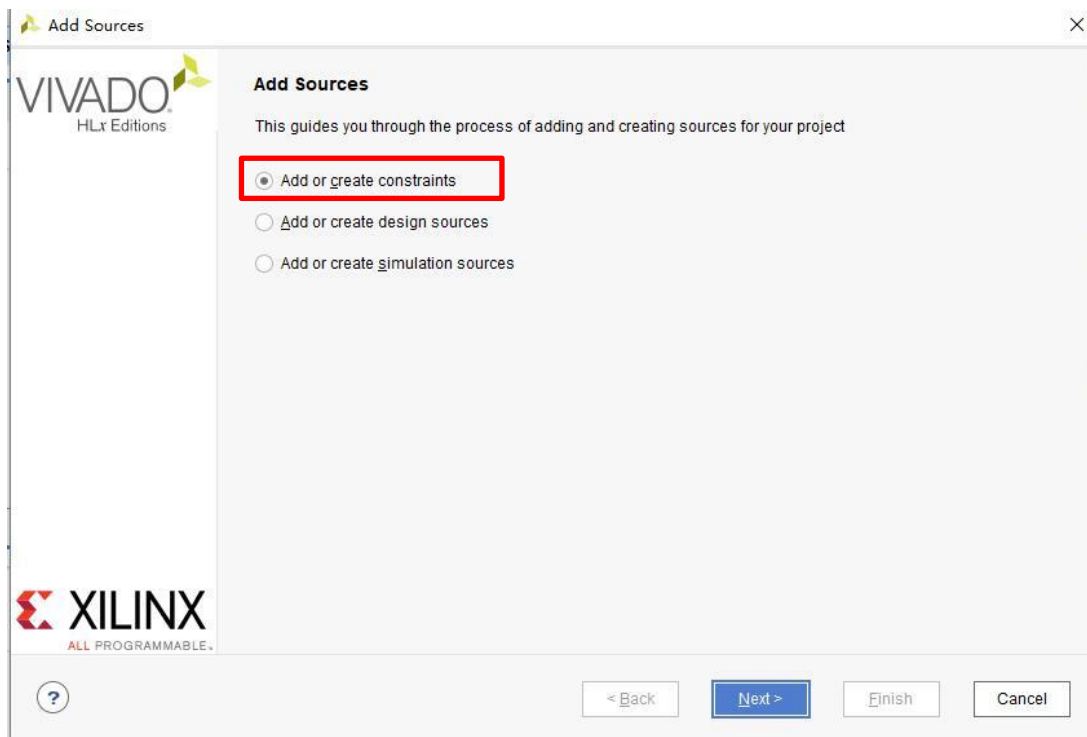
Creating constraint files – Step 5

Assigning Pin Location Constraint

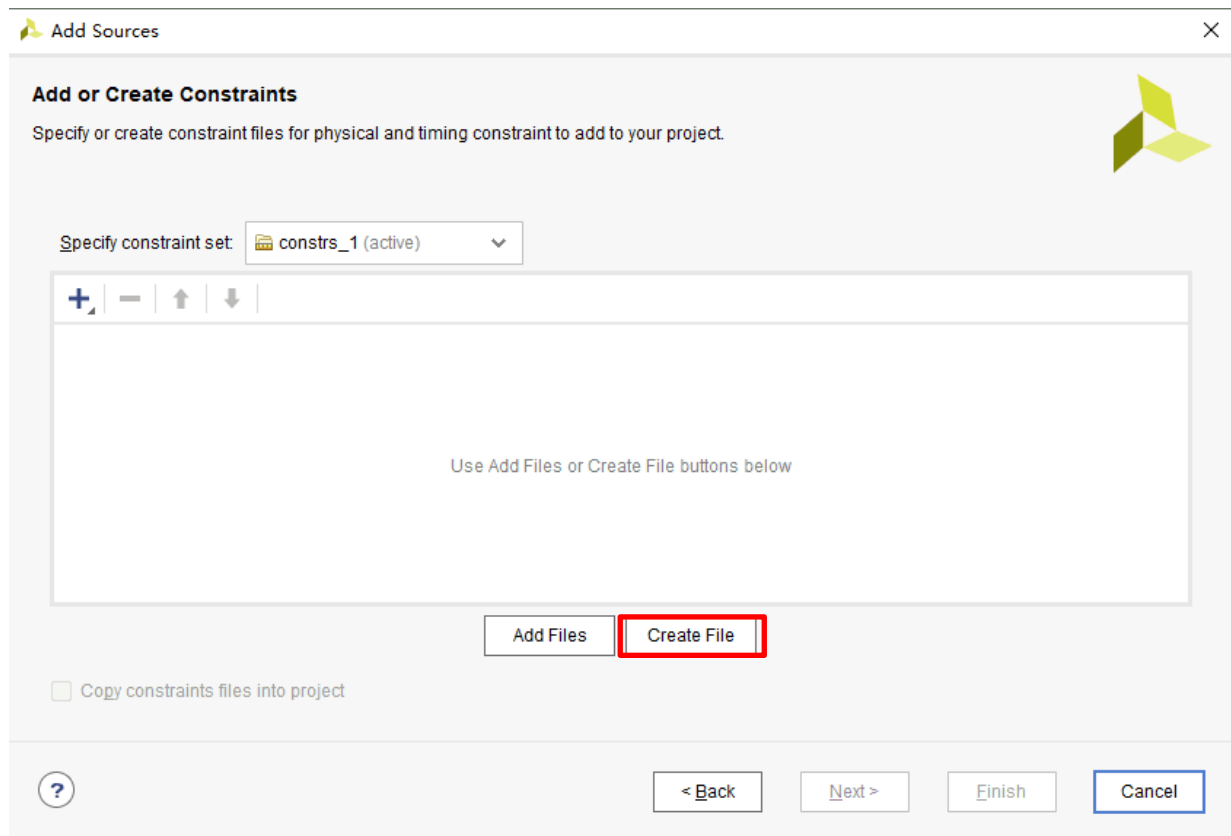
Now we are creating a constraint file into the project to associate physical pins of Nexy4 DDR to each input and output port. Constraint files are added as **sources**, so that the step is similar to that in Step 2.

In the project Manager of Vivado, right click **Design Sources**, and select **Add Sources** in the popped up dialog.

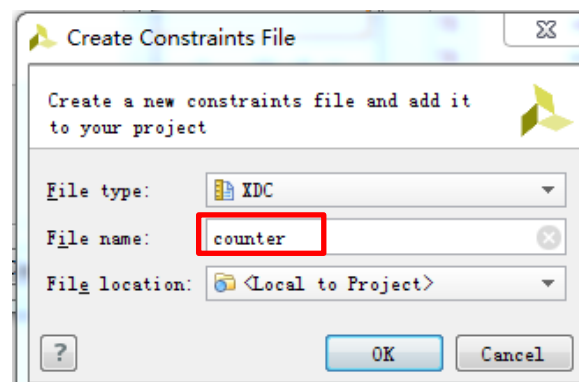
In the **Add Sources** wizard, select **Add or Create Constraints**, and click **Next**.



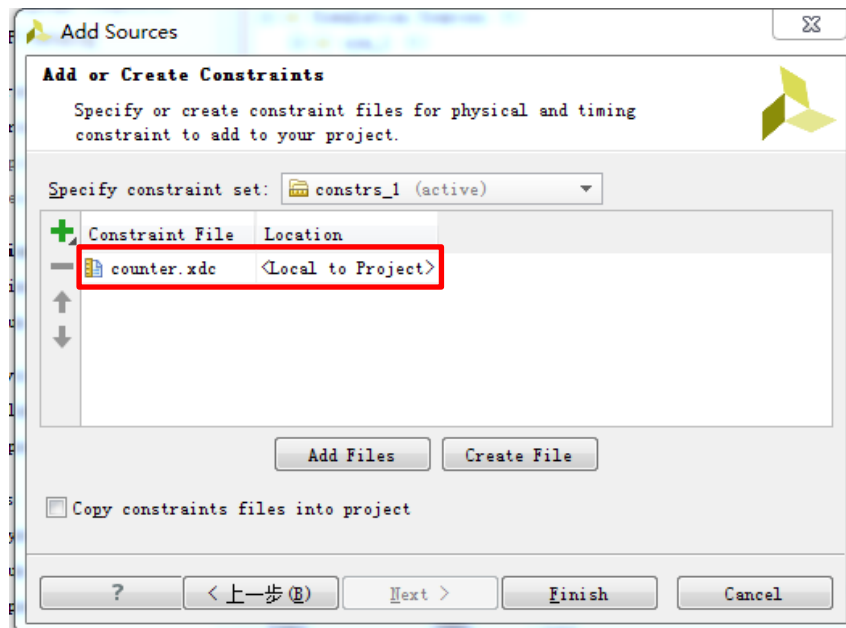
In the followed **Add or Create Constraints** dialog, click **Create File**.



In the newly popped up window, key in **counter** in the **File Name** item, keep all the other settings, and click **Ok**.

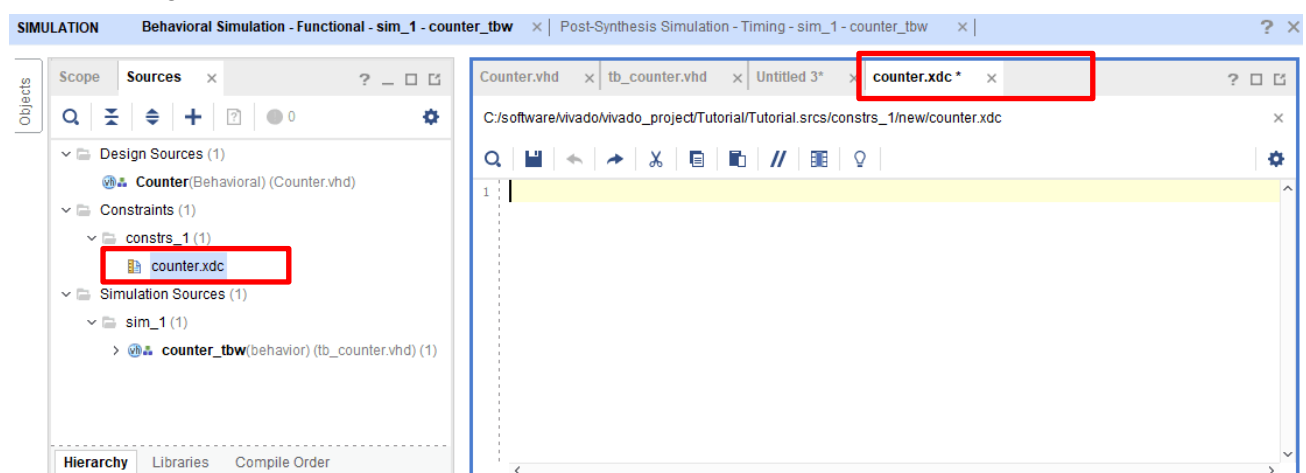


It returns to the previous **Add or Create Constraints** window as follows.



Click **Finish**.

In the **Project Manager**, expand **Constrains** until you see the created file **counter.xdc**. Double click the file to edit it. Since the file is just created, and it is empty as shown in the window at right.



We shall use a push button (BTNR, Pin M17) to reset the circuit, a switch (SW0, Pin J15) to control the counting direction, and 4 LEDs (LD0~LD3, Pins H17, K15, J13, N14) to illustrate the counting. Thus, we add in the following sentences to the constraint file, and save the file.

```
set_property PACKAGE_PIN E3 [get_ports CLOCK]
set_property PACKAGE_PIN M17 [get_ports RESET]
set_property PACKAGE_PIN J15 [get_ports DIRECTION]
set_property PACKAGE_PIN H17 [get_ports COUNT_OUT[0]]
set_property PACKAGE_PIN K15 [get_ports COUNT_OUT[1]]
set_property PACKAGE_PIN J13 [get_ports COUNT_OUT[2]]
set_property PACKAGE_PIN N14 [get_ports COUNT_OUT[3]]
set_property IOSTANDARD LVCMOS33 [get_ports CLOCK]
set_property IOSTANDARD LVCMOS33 [get_ports RESET]
set_property IOSTANDARD LVCMOS33 [get_ports DIRECTION]
```

```
set_property IOSTANDARD LVCMOS33 [get_ports COUNT_OUT[0]]
set_property IOSTANDARD LVCMOS33 [get_ports COUNT_OUT[1]]
set_property IOSTANDARD LVCMOS33 [get_ports COUNT_OUT[2]]
set_property IOSTANDARD LVCMOS33 [get_ports COUNT_OUT[3]]
```

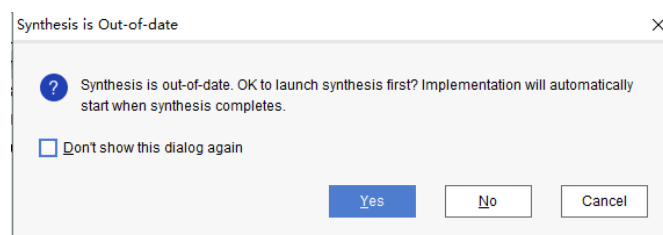
Refer to **Appendix C** for all IO (pins) of Nexys4-DDR.

Implementing your Design and Verify Step – 6

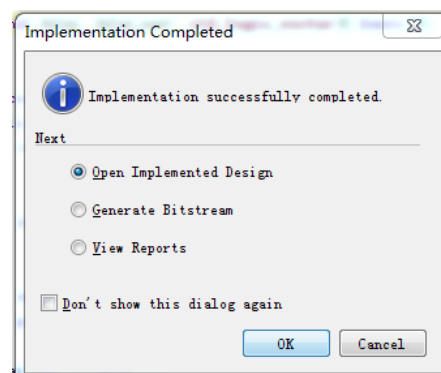
Implementing the Design

Click **Flow Navigator -> Implementation -> Run Implementation**.

If a dialog stating that the **Synthesis is out-of-date** is popped out, click **Yes** to launch synthesis first and then implementation is automatically followed.



Once the implementation is completed, you may open the implemented design by click **OK** in the followed dialog.



Verify Design using Functional and Timing Simulation

You may run the **post-implementation functional simulation** and **timing simulation** to verify the counter design after it has been completed implemented, and to compare the delay time with that of **post-synthesis timing simulation**.

Configuring FPGA Board – Step 7

You can configure your design onto the Nexys4 DDR FPGA board.

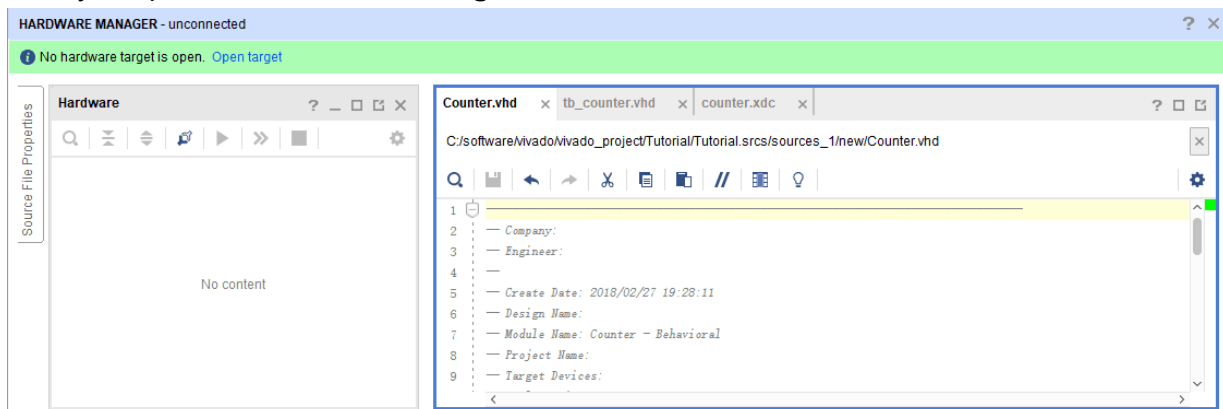
Generating FPGA Configuration File

Click **Flow Navigator -> Program and Debug -> Generate Bitstream**.

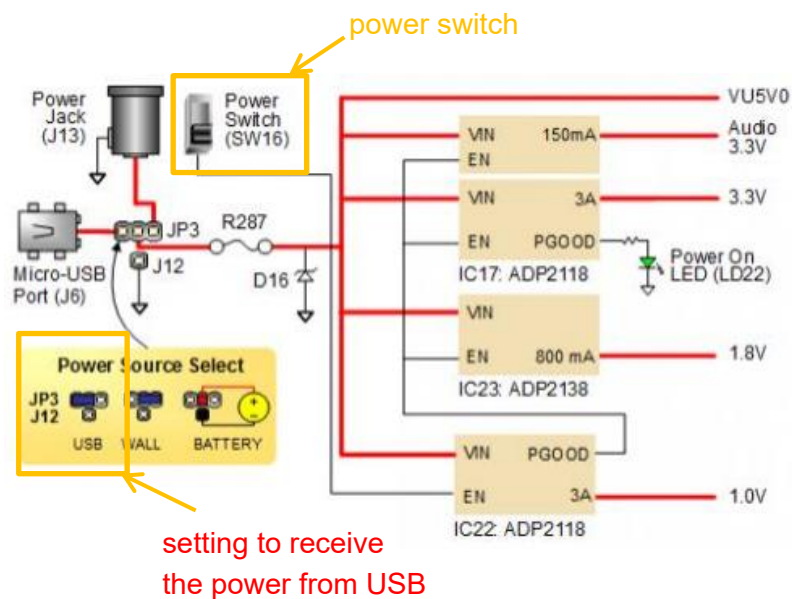
When the bitstream is generated, click **Open Hardware Manager** from the **Flow Navigator -> Program and Debug -> Open Hardware Manager**).

Download FPGA configuration file into Nexys4 DDR board via JTAG.

After you open the **Hardware Manager**, the interface is shown as follows:

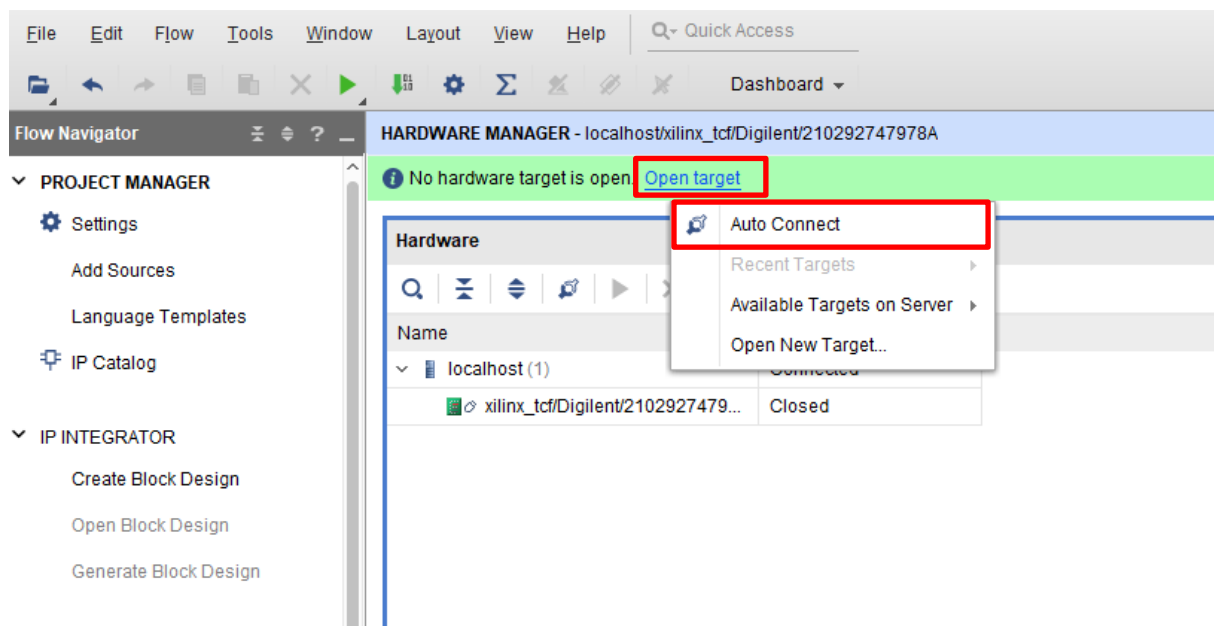


Connect **Nexys4 DDR** with computer using the provided **USB cable**. Before powered on, confirm that jumper **JP3** is set to receive the power from USB. Refer to the power circuit below, where Jumper JP3 is near the power jack.

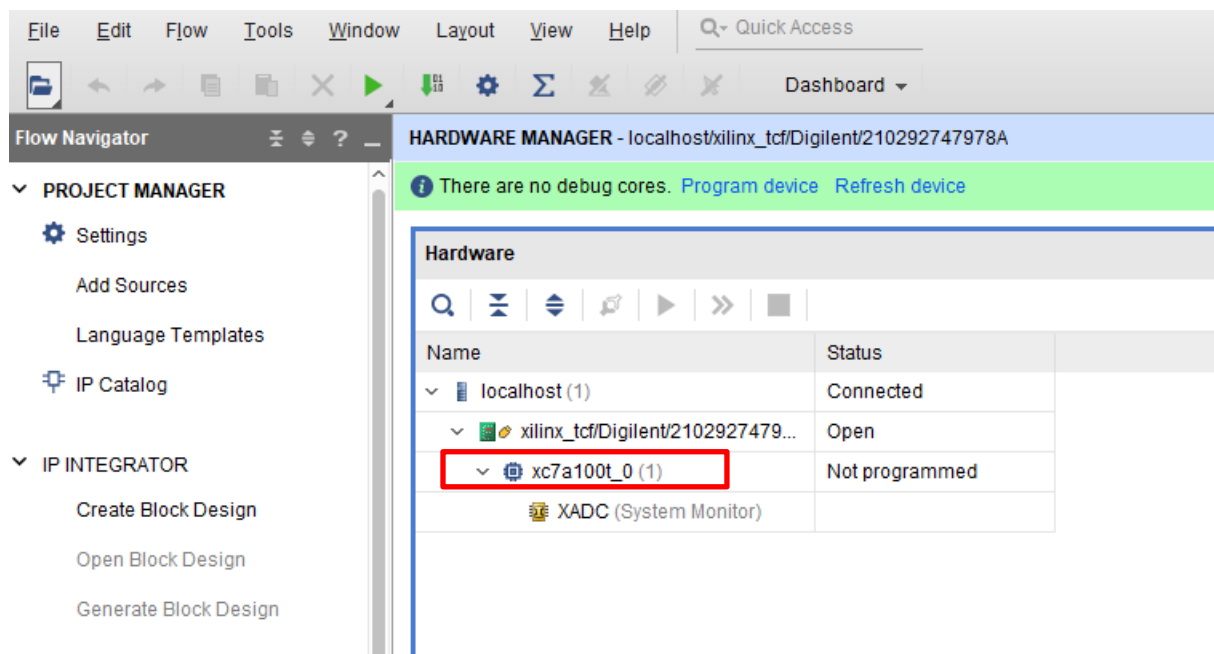


Turn on the **power switch**, which is a **single logic-level power switch** (SW16).

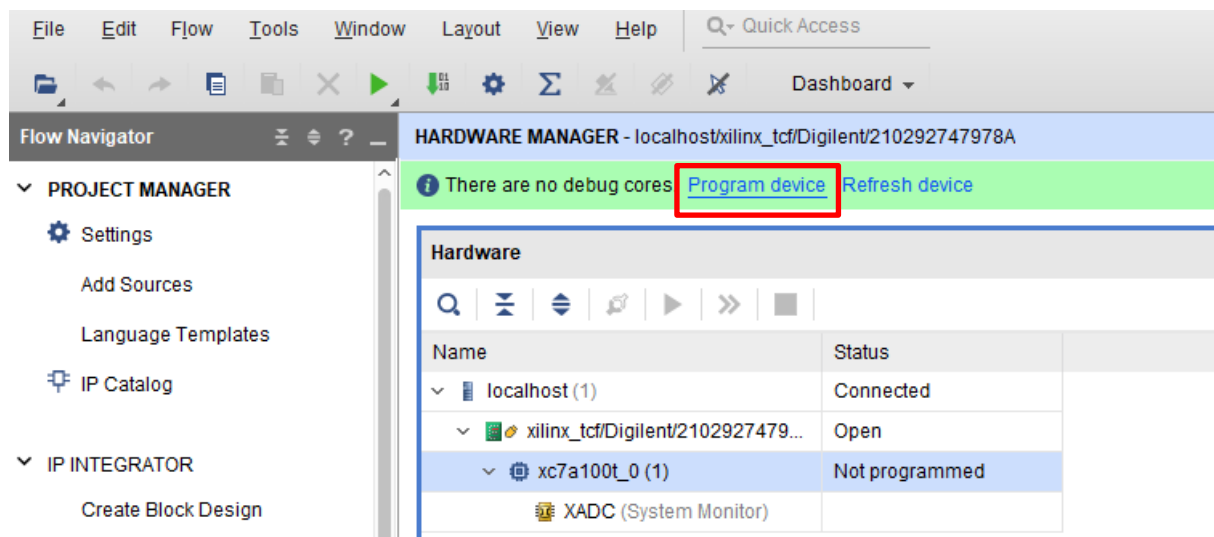
Click **Open target**, and select **Auto Connect**.



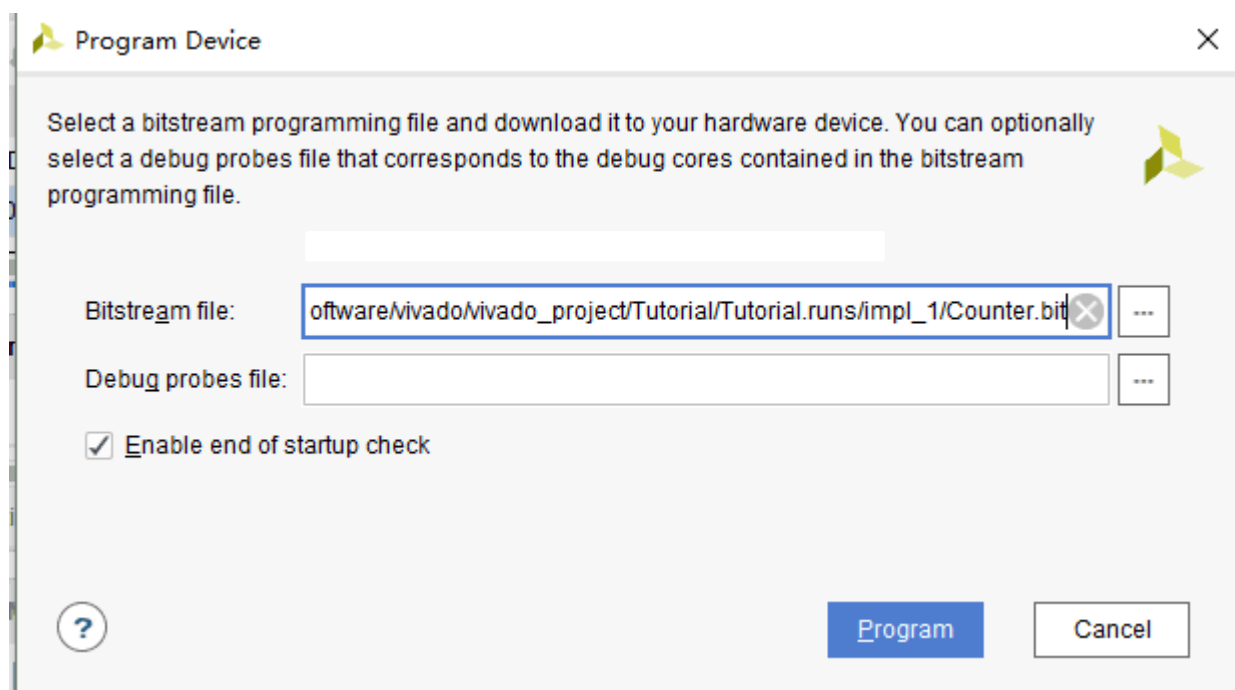
In the **Hardware manage** dialog, you can see that Vivado has detected the **Artix-7 100T FPGA** (shown as **xc7a100t_0**) on the scan chain.



Click **Program device**.



The following dialog is popped out. Select the **bitstream file** from the `/project_dir/Tutorial/Tutorial.runs/impl_1/Counter.bit`, and click **Program**.



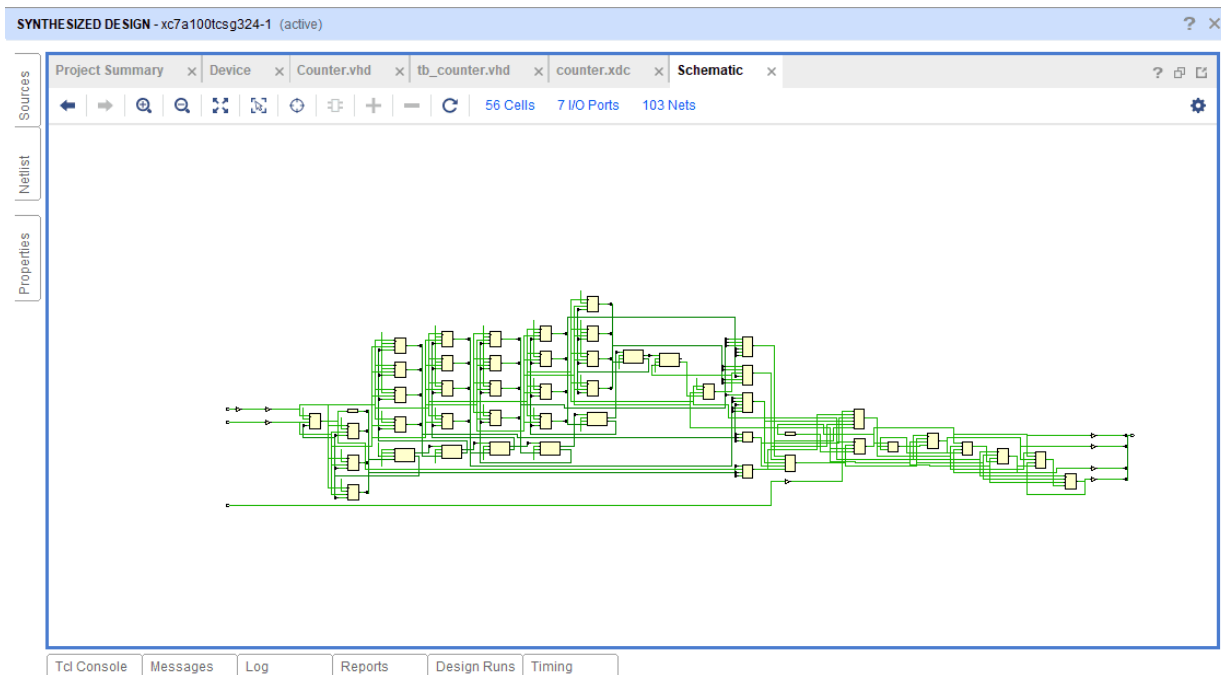
- When programming is completed, On the board, LEDs 0, 1, 2 and 3 are lit, indicating that the counter is running.
- Note that all the LEDs seem to be on all the time, instead of counting on the desired sequence. ***Can you explain why it is so?***

Return to the source code window by clicking **Flow Navigator -> Project Manag.** Double click the source file **Counter.vhd** to show the source code in the right window.

- Modify the VHDL source code to slow down the counting as follow:
signal delay, delay_next: std_logic_vector(24 downto 0);
- Save the file

Re-check the **syntax**, re-run the **synthesis**, **implementation**, **generate bitstream**, and **program the device** in Steps 2, 4, 6, and 7.

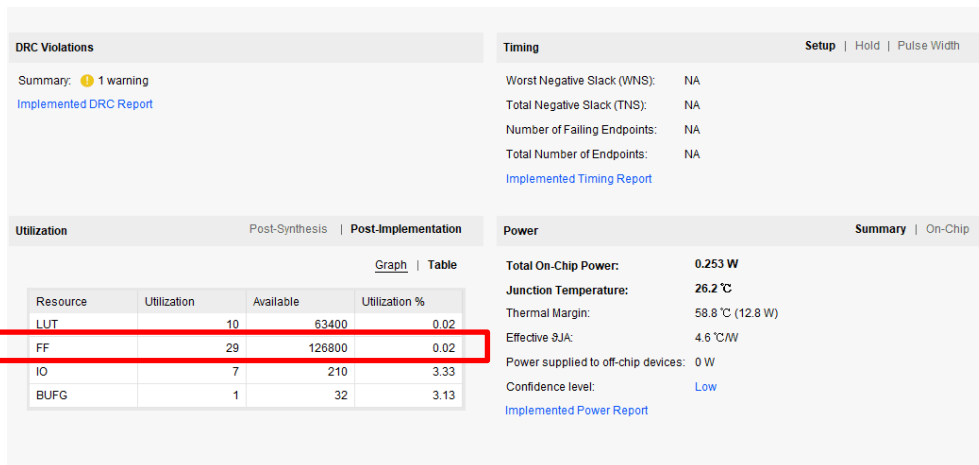
Now open the **schematic** again and compare the difference with the previous one before the code is modified. You can zoom in to view the details of the schematic.



View again the **RTL schematic** and **Project Summary**, and compare with the previous ones before the code is modified

The screenshot displays the Vivado IDE's Project Summary view for the same project. The 'Settings' section provides project metadata, and the 'Synthesis' and 'Implementation' sections show the progress and status of the design process.

Section	Property	Value
Settings	Project name:	Tutorial
	Project location:	C:/software/vivado/vivado_project/Tutorial
	Product family:	Artix-7
	Project part:	xc7a100tcs324-1
	Top module name:	Counter
	Target language:	VHDL
	Simulator language:	Mixed
Synthesis	Status:	Complete
	Messages:	4 warnings
	Part:	xc7a100tcs324-1
	Report Strategy:	Vivado Synthesis Default Reports
Implementation	Status:	Complete
	Messages:	2 warnings
	Part:	xc7a100tcs324-1
	Report Strategy:	Vivado Implementation Default Reports
	Incremental compile:	None



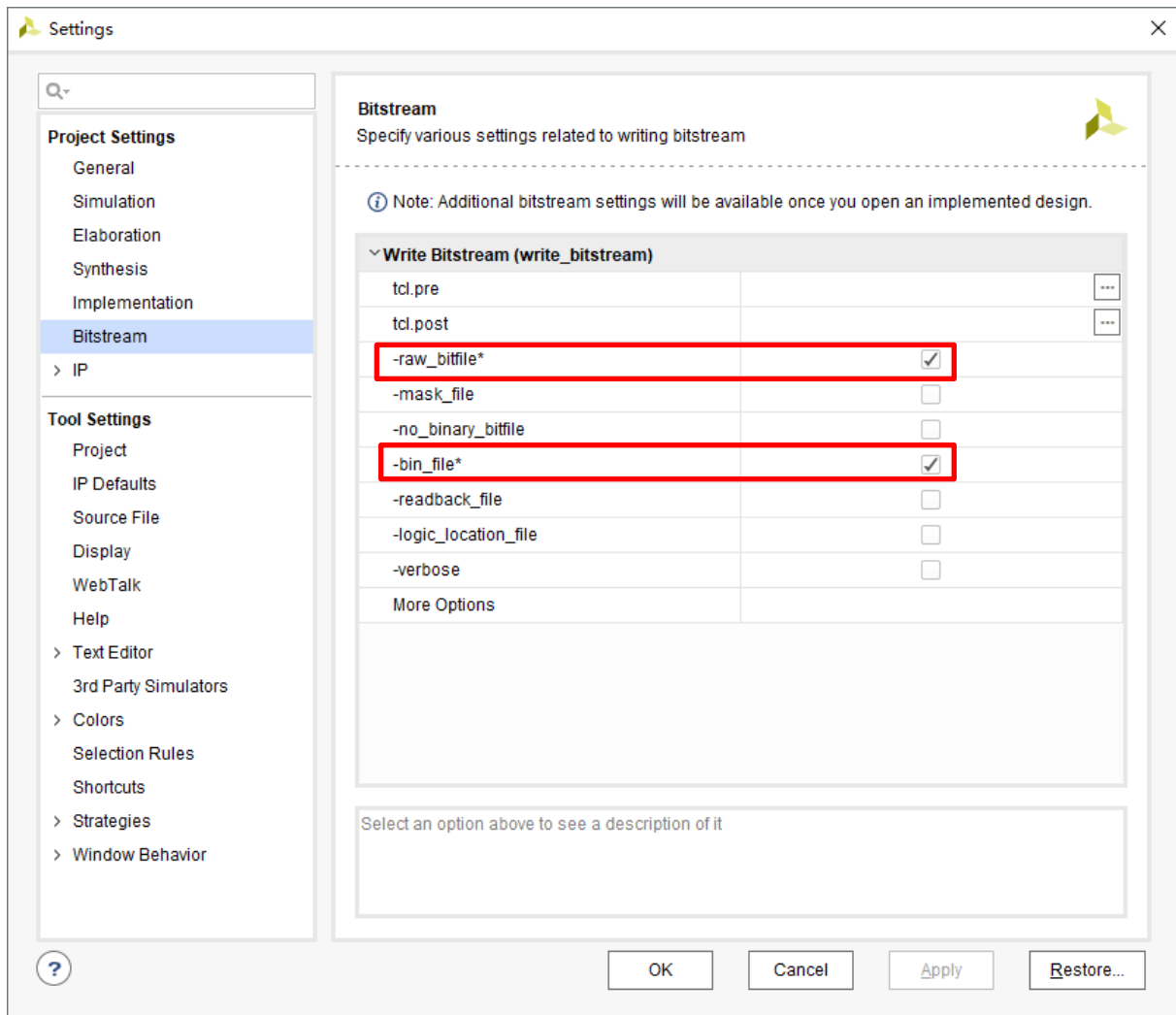
Look at the change in number of the Flip Flops (why is it 29 instead of 5?)

Programing PROM of the FPGA Board – Step 8

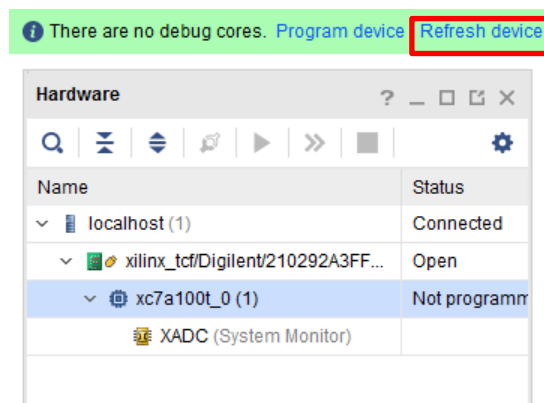
As you have seen now, once the FPGA is powered off and powered on again, it has to be re-configured again. Therefore, upon completion of the design, user will have to prepare a PROM file. This Prom will automatically program the FPGA with the designated design file upon power-up without downloading the design via JTAG.

The following steps demonstrate on the Prom file preparation and download it to the Prom.

- Click **Flow Navigator -> Project Manager -> Setting**, under the **Project Setting**, select **Bitstream**, tick – **bin_file** and tick-**raw_bitfile**, and then **OK**.

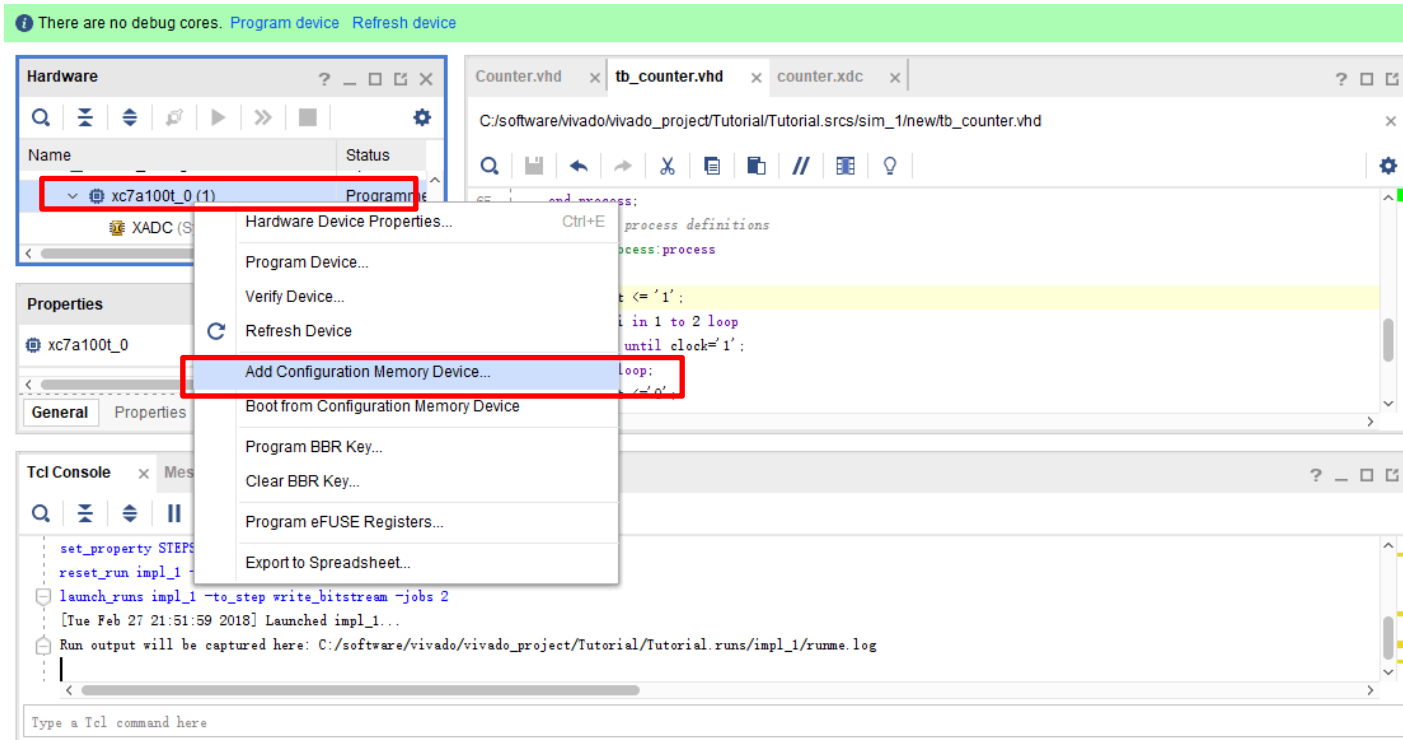


If it shows error that the Target has been shut down, you may refresh the device

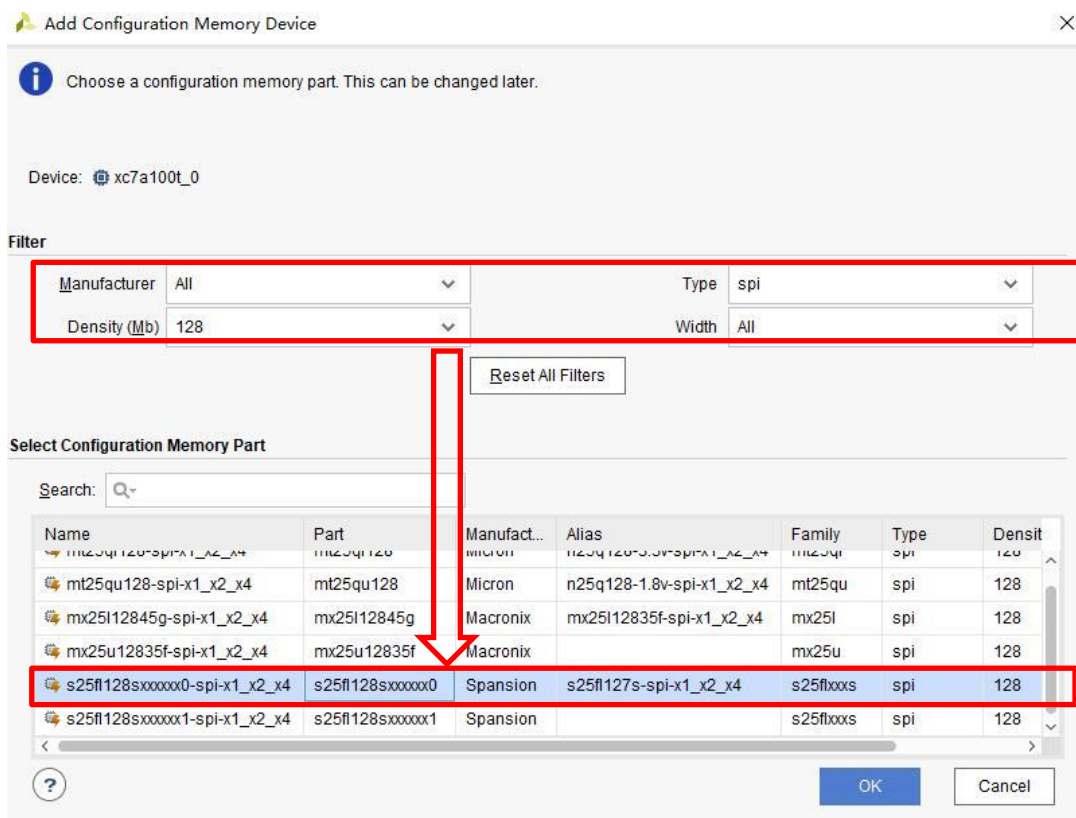


- Generate bitstream again by clicking **Flow Navigator -> Program and Debug -> Generate Bitstream**.
- Open the **Hardware Manage** once the bitstream is generated by clicking **Flow Navigator -> Program and Debug -> Hardware Manager**.

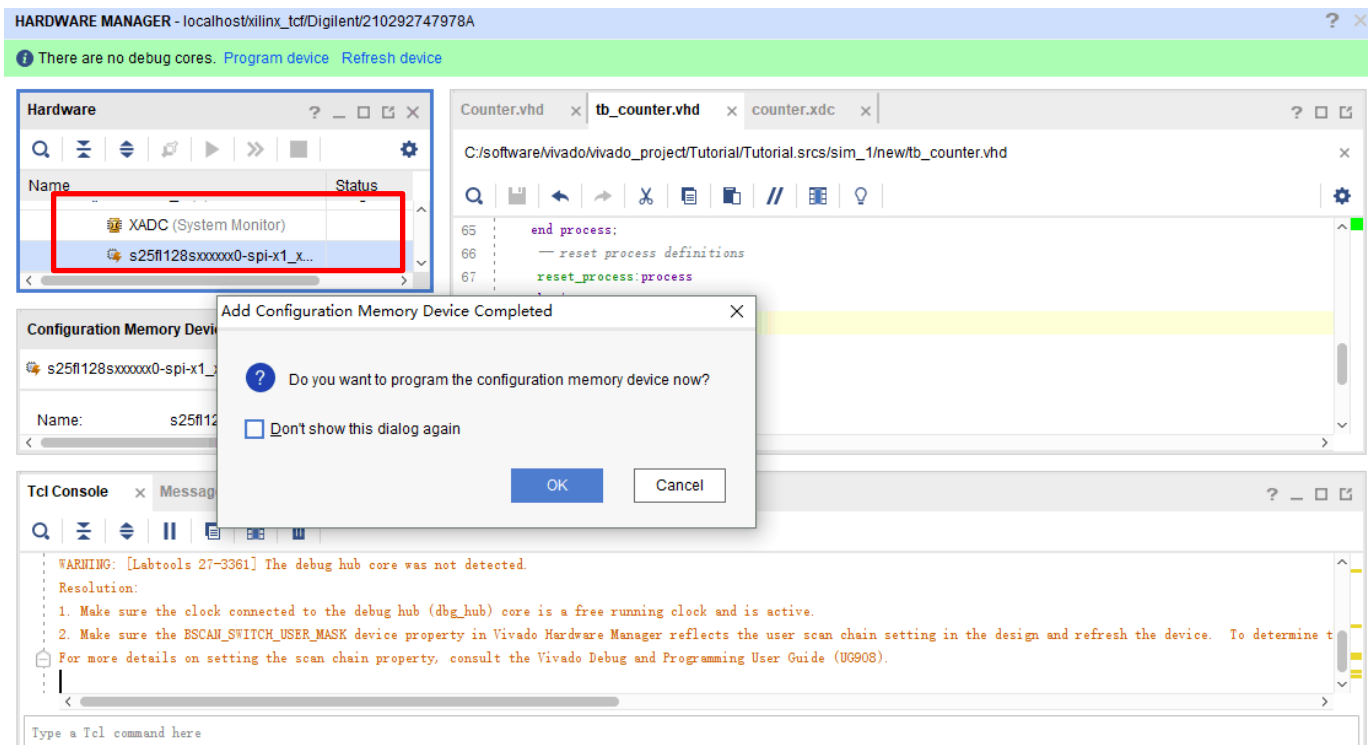
- **Open Target** if it is not opened by clicking **Flow Navigator -> Program and Debug -> Open Hardware Manager -> Open Target -> Auto Connect**.
- Right click **xc7a100t_0**, and click **Add Configuration Memory Device** in the floated menu.



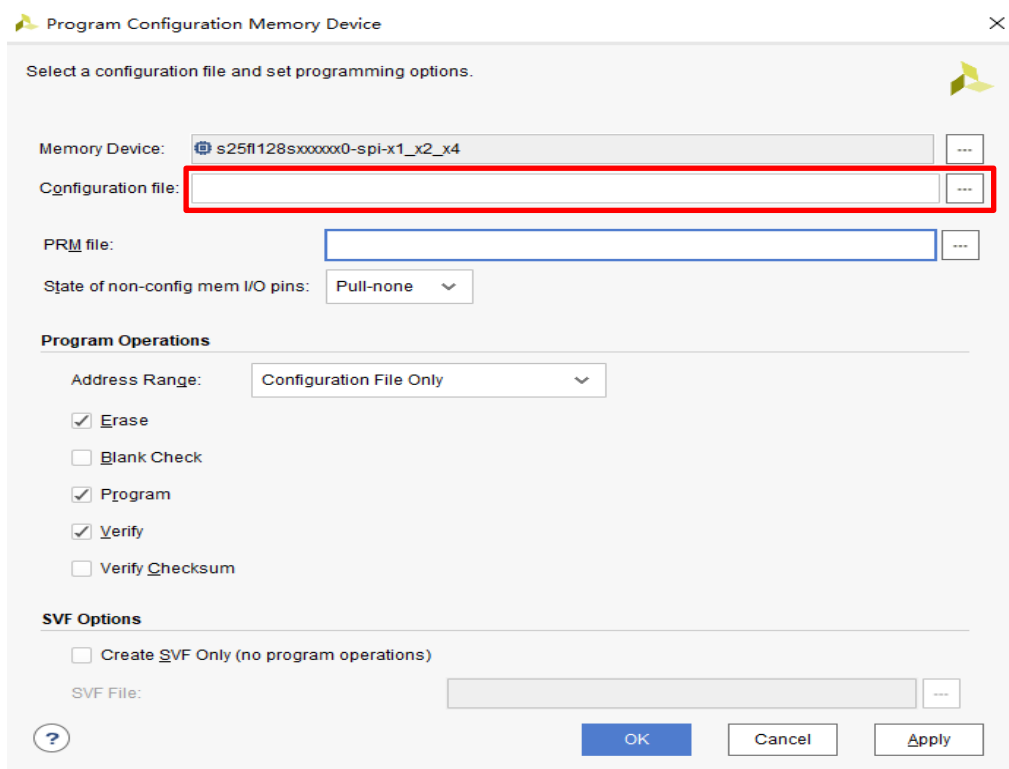
- Select **Density 128**, **Type spi** as follows, and select **s25fl128xxxxx0** in the appeared list. Then **OK**.



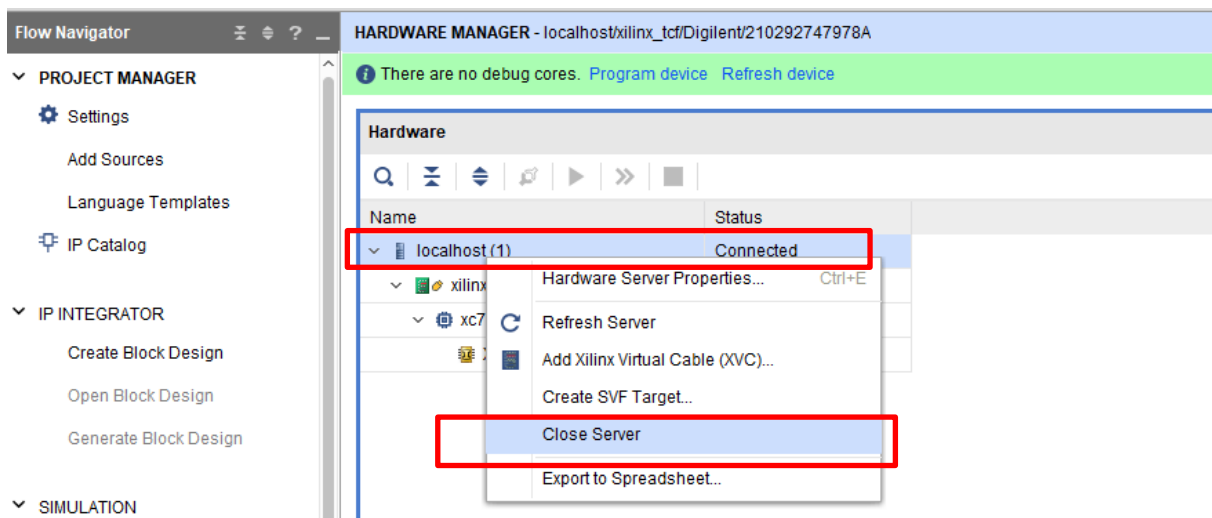
- You will see that **s25f1128xxxxxx0** has been attached to **xc7a100t_0**. In the popped up dialog, click **OK** to program the configuration memory device.



- In the popped up dialog, select **Project_dir/Tutorial/Tutorial.runs/impl_1/Counter.bin** as configuration file. Keep all the other settings as follows, and click **OK**.



- Once the memory is programmed successfully, the written program is not run immediately on the board. You have to first **disconnect the board** by right clicking **Local Host**, and then click **Close Server** in the **Project Manager** as follows:



- Switch the **power off**, and turn the **power on** again. After waiting for several seconds, you will see that the LEDs are counting on the desired sequence.

Appendix A:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity Counter is
    Port ( CLOCK : in STD_LOGIC;
          RESET : in STD_LOGIC;
          DIRECTION : in STD_LOGIC;
          COUNT_OUT : out STD_LOGIC_VECTOR (3 downto 0));
end Counter;

architecture Behavioral of Counter is
    signal count_in, count_in_next: std_logic_vector(3 downto 0);
    signal delay, delay_next: std_logic_vector(0 downto 0);

begin
    process (reset, clock)
    begin
        if reset = '1' then
            delay <= (others=>'0');
            count_in <= (others=>'0');
        elsif clock='1' and clock'event then
            delay <= delay_next;
            count_in <= count_in_next;
        end if;
    end process;

    delay_next <= delay + 1;
    count_in_next <= count_in+1 when delay = 0 and direction = '1' else
    count_in-1 when delay = 0 and direction = '0'
    else
    count_in;
    count_out <= count_in;
end Behavioral;
```

Appendix B:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.std_logic_unsigned.all;
USE ieee.numeric_std.ALL;

ENTITY counter_tbw IS
END counter_tbw;

ARCHITECTURE behavior OF counter_tbw IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT counter
    PORT(
        CLOCK : IN  std_logic;
        reset : IN  std_logic;
        DIRECTION : IN  std_logic;
        COUNT_OUT : OUT  std_logic_vector(3 downto 0)
    );
    END COMPONENT;
    --Inputs
    signal CLOCK : std_logic := '0';
    signal reset : std_logic := '0';
    signal DIRECTION : std_logic := '0';

    --Outputs
    signal COUNT_OUT : std_logic_vector(3 downto 0);

    -- Clock period definitions
    constant CLOCK_period : time := 40ns;

BEGIN
    -- Instantiate the Unit Under Test (UUT)
    uut: counter PORT MAP (
        CLOCK => CLOCK,
        reset => reset,
        DIRECTION => DIRECTION,
        COUNT_OUT => COUNT_OUT
    );

    -- Clock process definitions
    CLOCK_process :process
    begin
        CLOCK <= '0';
        wait for CLOCK_period/2;
        CLOCK <= '1';
        wait for CLOCK_period/2;
    end process;

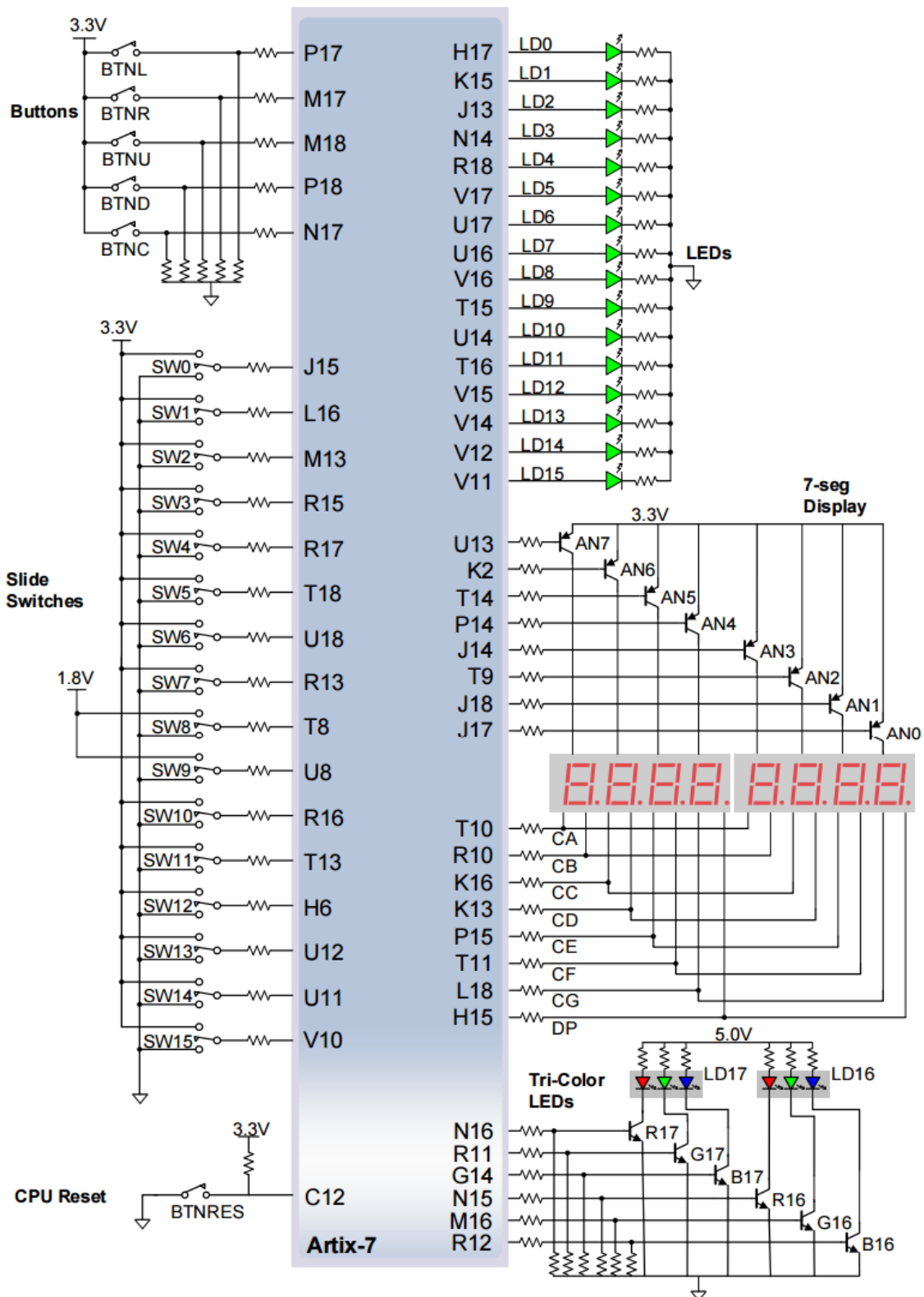
    -- reset process definitions
    reset_process:process
    begin
        reset <= '1';
        for i in 1 to 2 loop
            wait until clock='1';
        end loop;
        reset <= '0';
        wait;
    end process;

    -- Stimulus process
    stim_proc: process
    begin

        DIRECTION <= '1';
        wait for 1400ns;           --current Time:1400ns

        DIRECTION <= '0';
        wait for 800ns;           -- current Time:2200ns
        DIRECTION <= '1';
        wait;
    end process;
end;
```

Appendix C:



General Purpose I/O devices on the Nexys4 DDR.