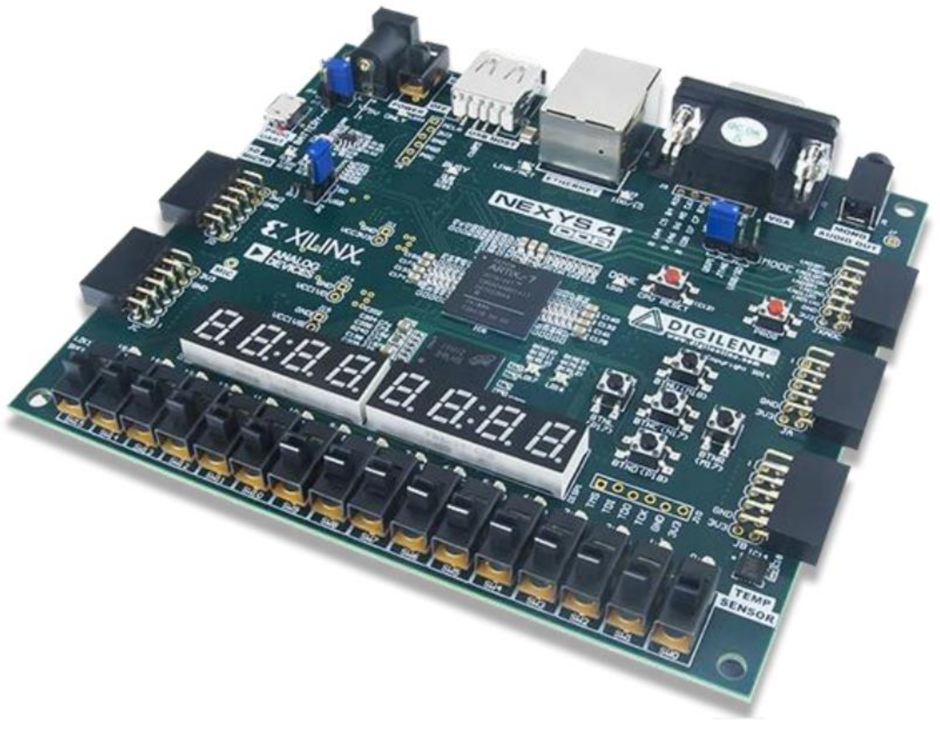
Nexys4 DDR & Vivado 2017.4 Getting Started v1.3



Department of Electrical and Electronic Engineering

Southern Univ. of Science & Technology

by Yu Yajun

May 2022

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:

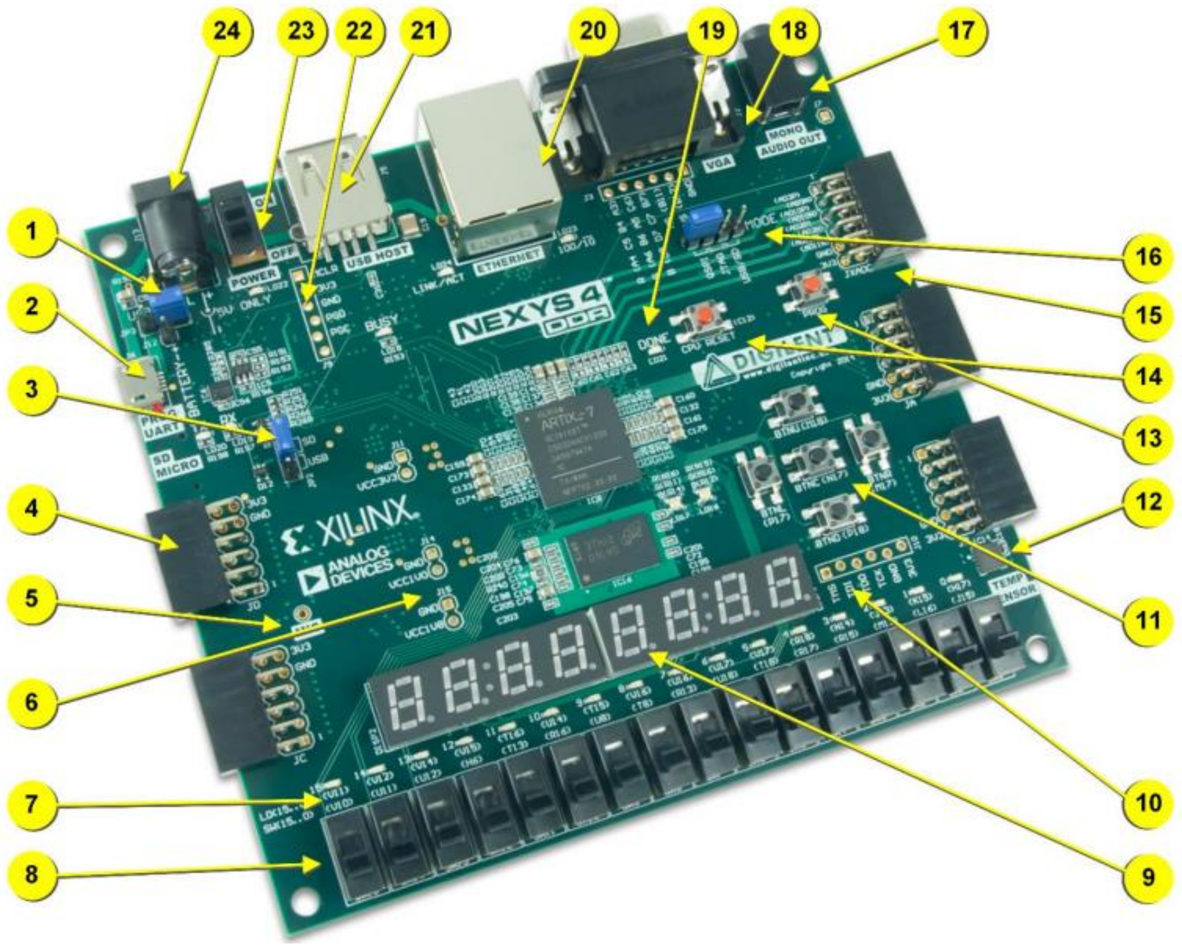
- 15,850 logic slices, each with four 6-input LUTs and 8 flip-flops
- 4,860 Kbits of fast block RAM
- Six clock management tiles, each with phase-locked loop (PLL)
- 240 DSP slices
- Internal clock speeds exceeding 450 MHz
- On-chip analog-to-digital converter (XADC)

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:

Vivado Quick Start Tutorial

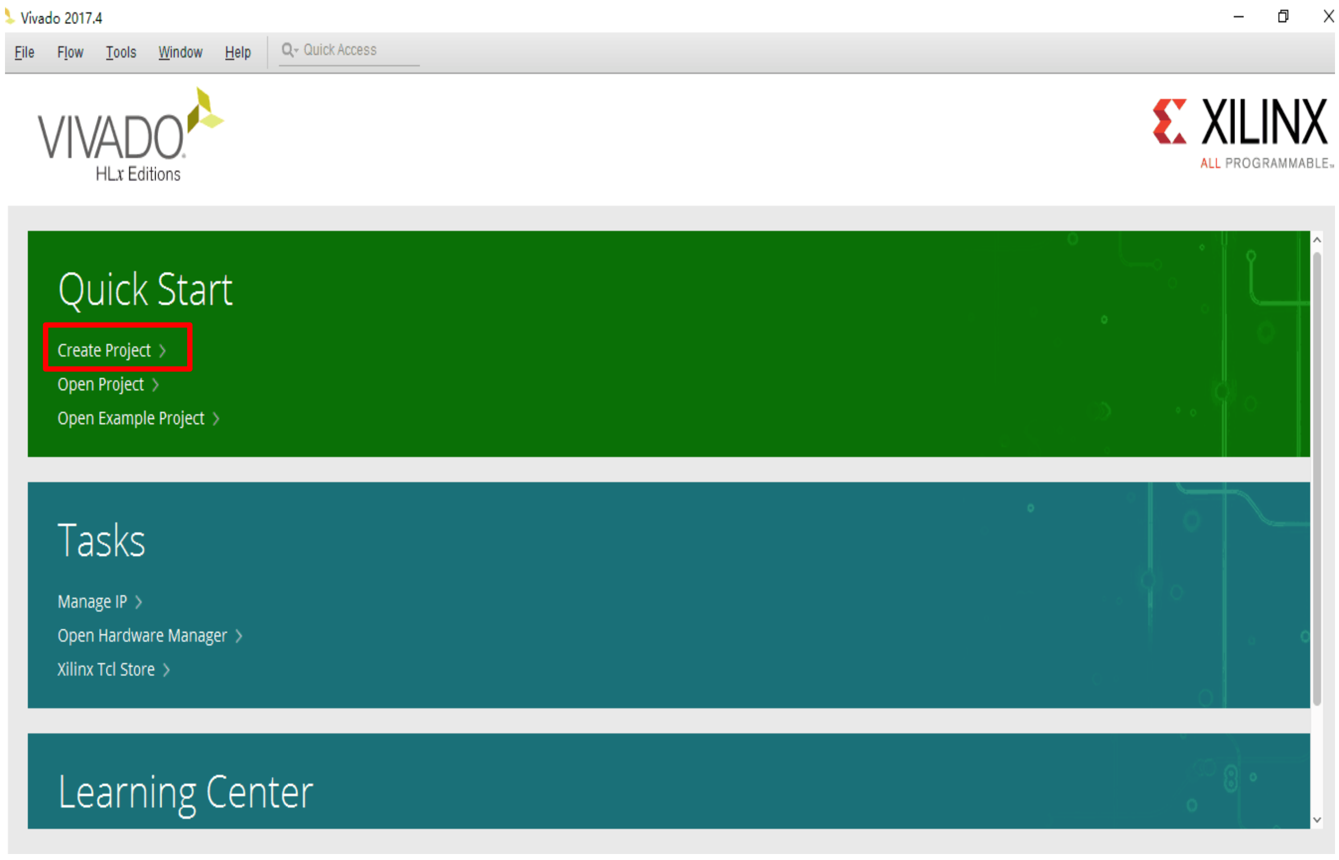
Creating a Project – Step 1

Starting the Vivado Software

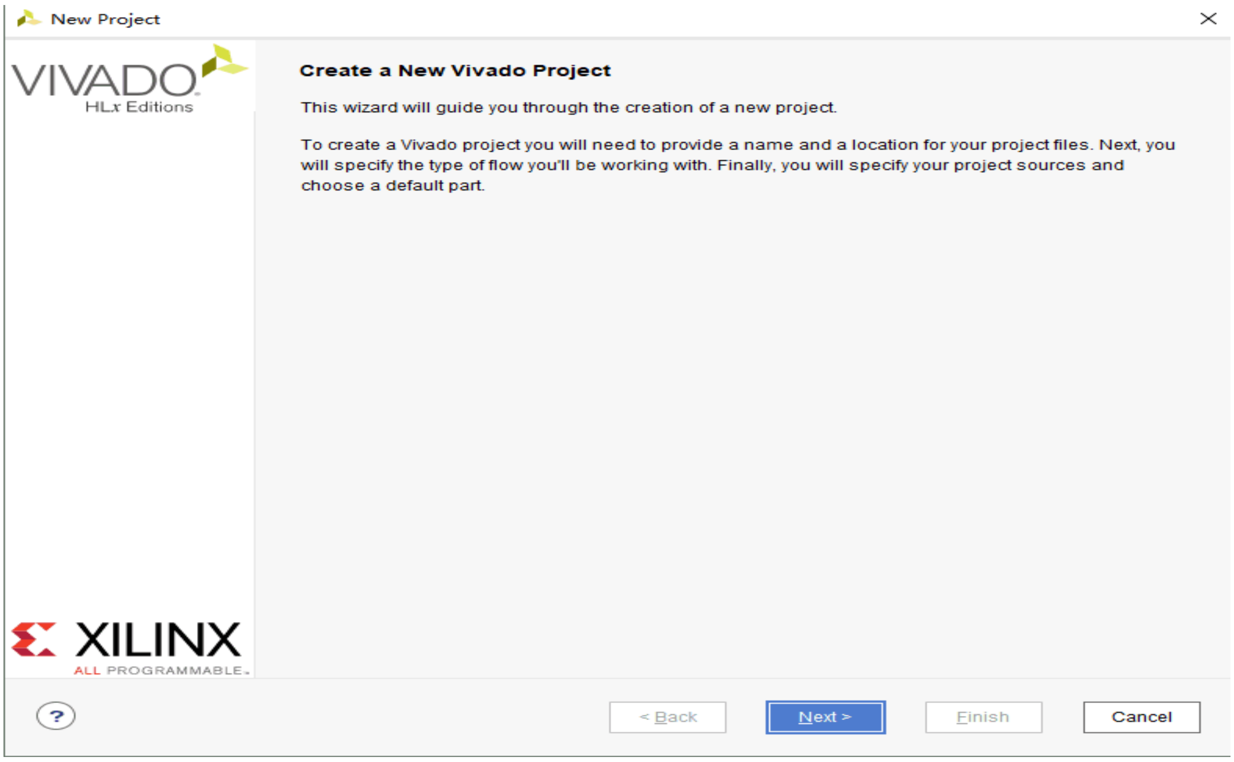
Go to **Start > All Programs > Xilinx Design Tools > Vivado 2017.4 >> Vivado 2017.4** or double-click the following **desktop icon**.

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.


New Project

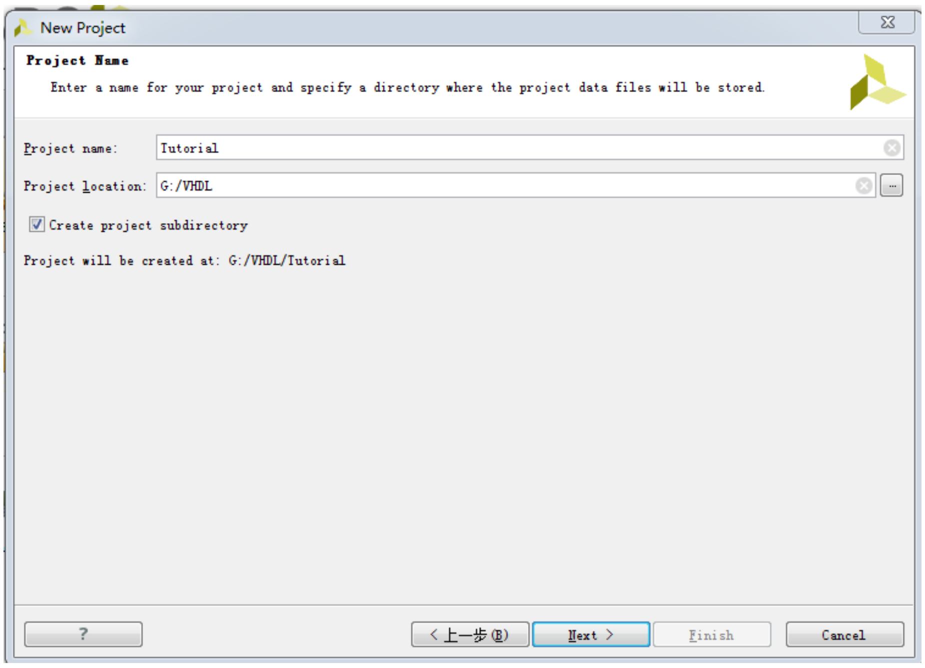
Create a New Vivado Project

This wizard will guide you through the creation of a new project.

To create a Vivado project you will need to provide a name and a location for your project files. Next, you will specify the type of flow you'll be working with. Finally, you will specify your project sources and choose a default part.

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**.


New Project

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.

Project name: Tutorial

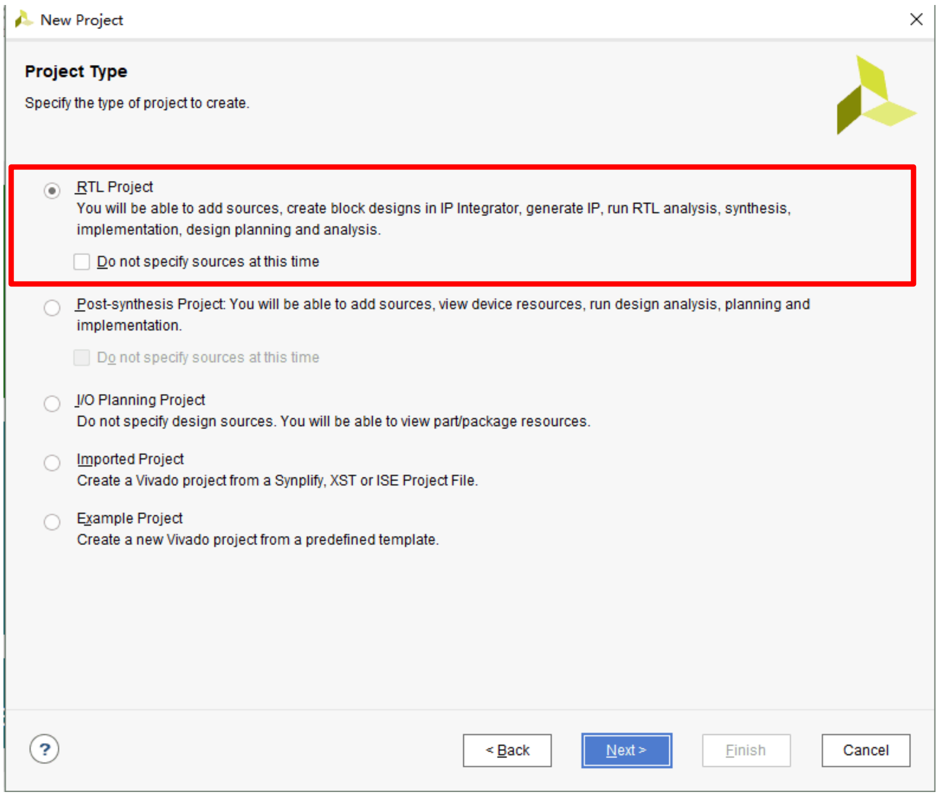
Project location: G:/VHDL

☑ Create project subdirectory

Project will be created at: G:/VHDL/Tutorial

Click **Next**.

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


New Project

Project Type

Specify the type of project to create.

(●) RTL Project
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
☐ Do not specify sources at this time

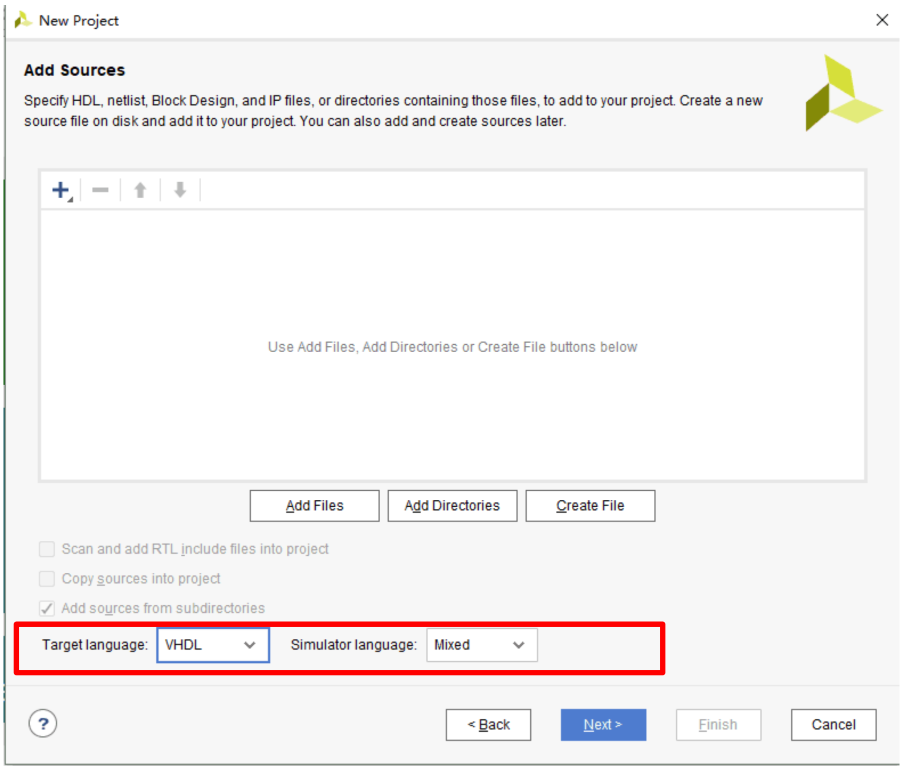
( ) Post-synthesis Project: You will be able to add sources, view device resources, run design analysis, planning and implementation.
☐ Do not specify sources at this time

( ) I/O Planning Project
Do not specify design sources. You will be able to view part/package resources.

( ) Imported Project
Create a Vivado project from a Synplify, XST or ISE Project File.

( ) Example Project
Create a new Vivado project from a predefined template.

In the appeared **Add Source** dialog,


New Project

Add Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

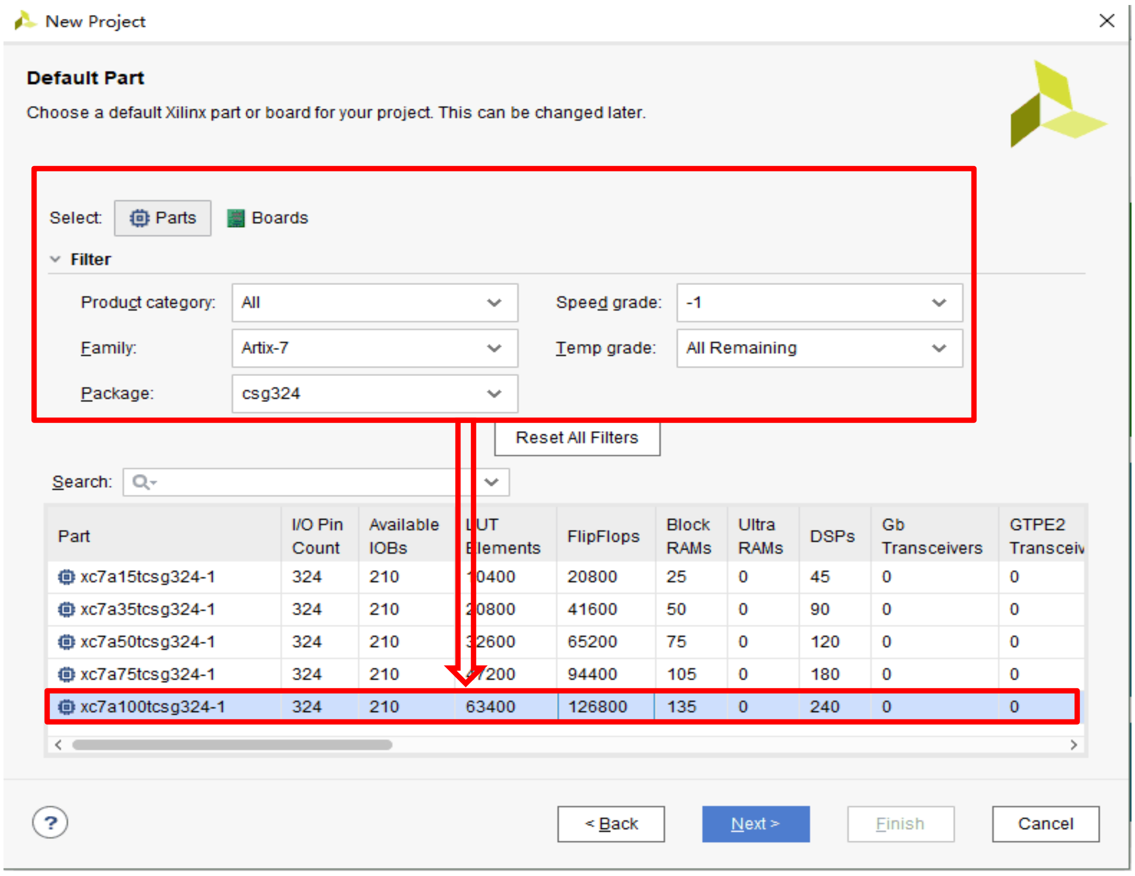
Use Add Files, Add Directories or Create File buttons below

☐ Scan and add RTL include files into project
☐ Copy sources into project
☑ Add sources from subdirectories

Target language: VHDL   Simulator language: Mixed

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**.

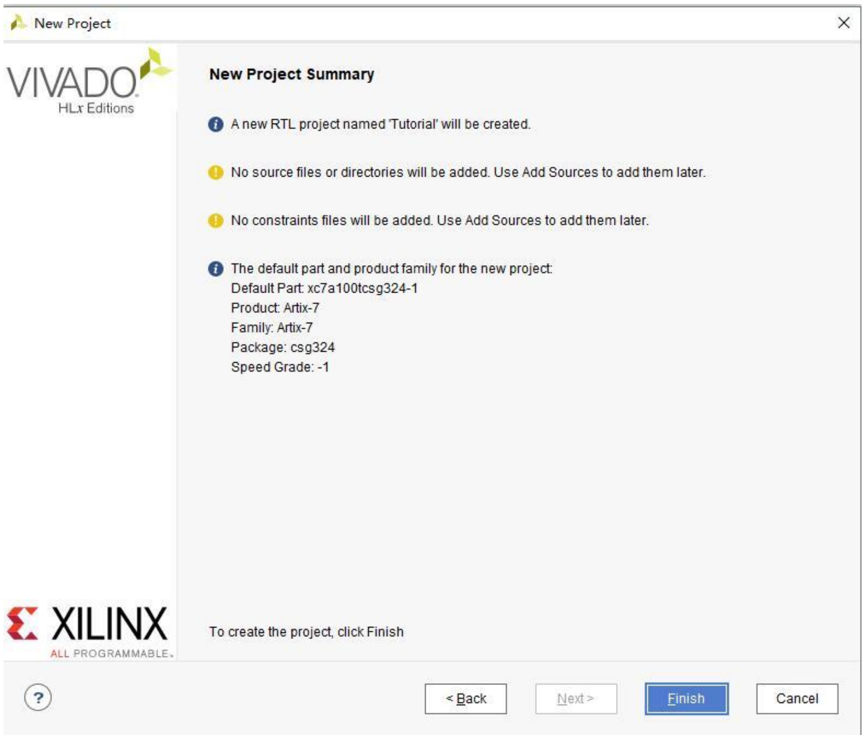

New Project
Default Part
Choose a default Xilinx part or board for your project. This can be changed later.
Select: Parts Boards
Filter

Product category: All
Family: Artix-7
Package: csg324
Speed grade: -1
Temp grade: All Remaining

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

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

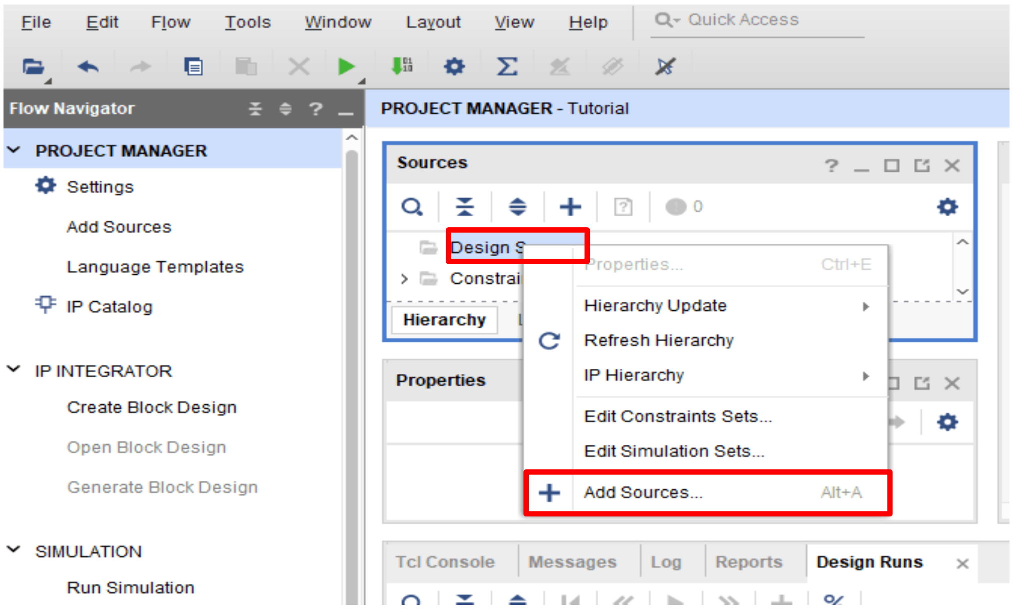
To create the project, click Finish

Creating VHDL Source / Design Files – Step 2

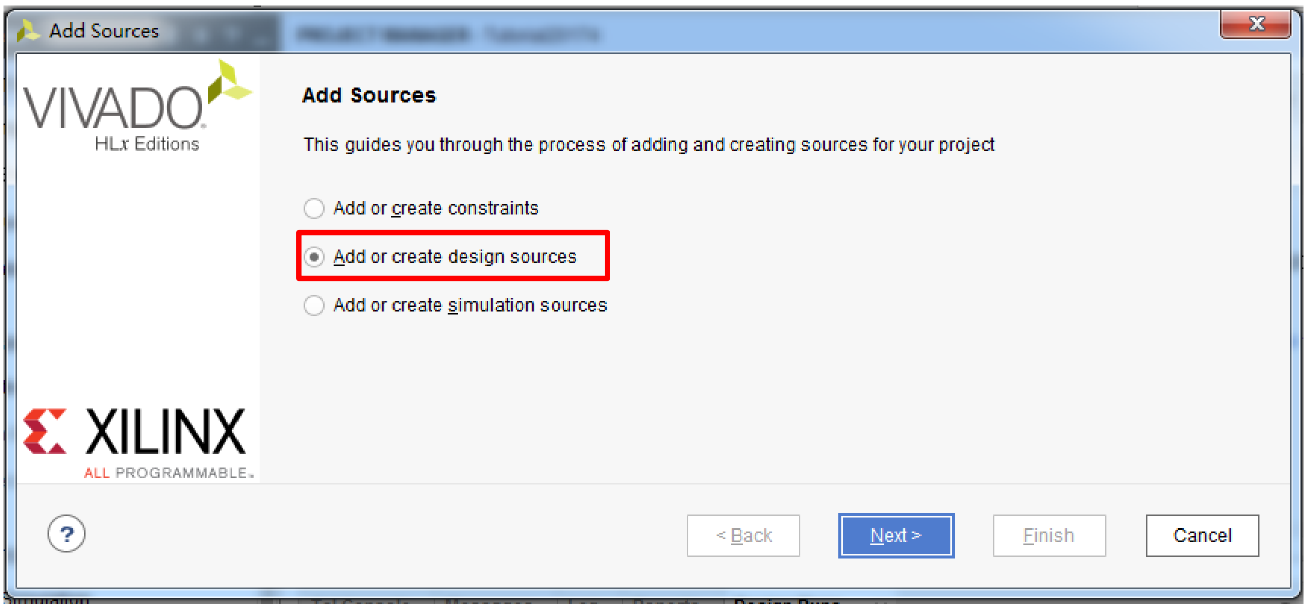
In this section, you will create the top HDL file for your design.

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.

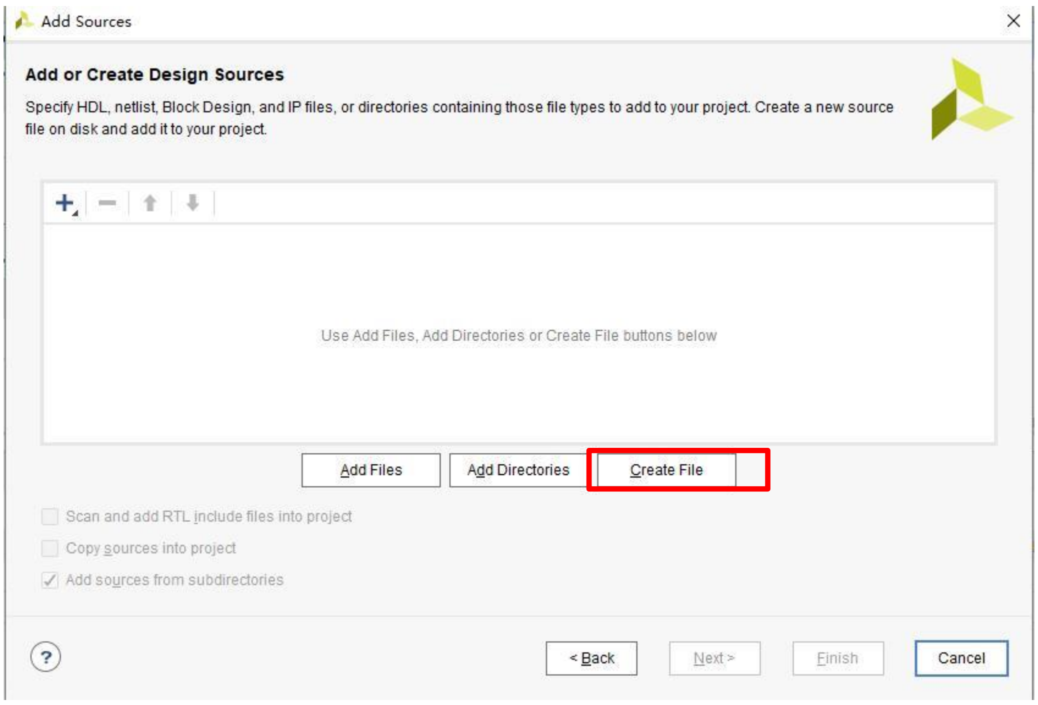


Add Sources

This guides you through the process of adding and creating sources for your project

Add or create constraints
[✓ Add or create design sources]
Add or create simulation sources

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


Add Sources

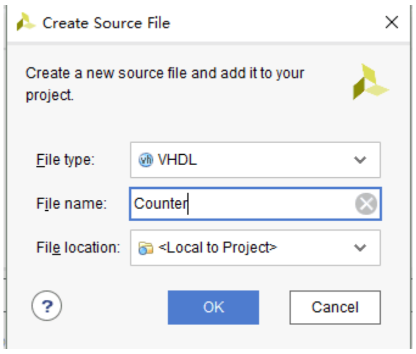
Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

Use Add Files, Add Directories or Create File buttons below

Add Files
Add Directories
Create File
Scan and add RTL include files into project
Copy sources into project
Add sources from subdirectories
< Back
Next >
Finish
Cancel

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


Create Source File

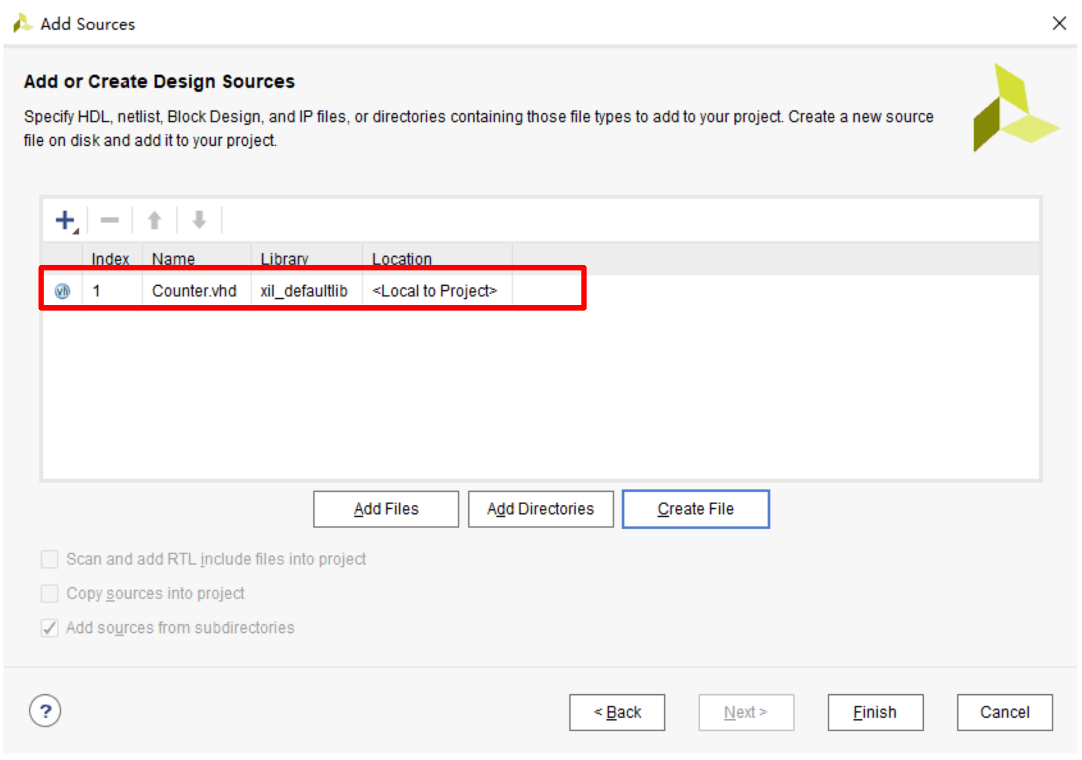
Create a new source file and add it to your project.

File type: VHDL

File name: Counter

File location: <Local to Project>

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


Add Sources

Add or Create Design Sources

Specify HDL, netlist, Block Design, and IP files, or directories containing those file types to add to your project. Create a new source file on disk and add it to your project.

Index	Name	Library	Location
1	Counter.vhd	xil_defaultlib	<Local to Project>

☐ Scan and add RTL include files into project
☐ Copy sources into project
☑ Add sources from subdirectories

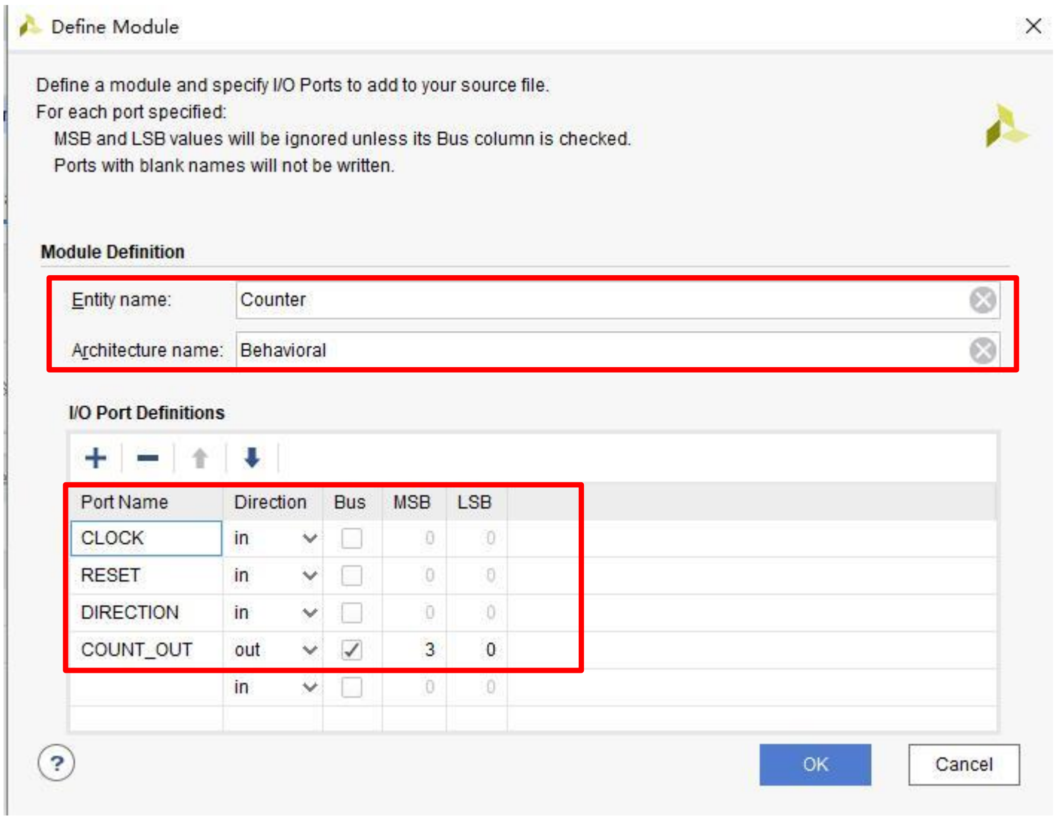
Note that a source file **Counter.vhd** has been created. Keep all the other settings, and click **Finish** to complete the adding resource of **Counter**.

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

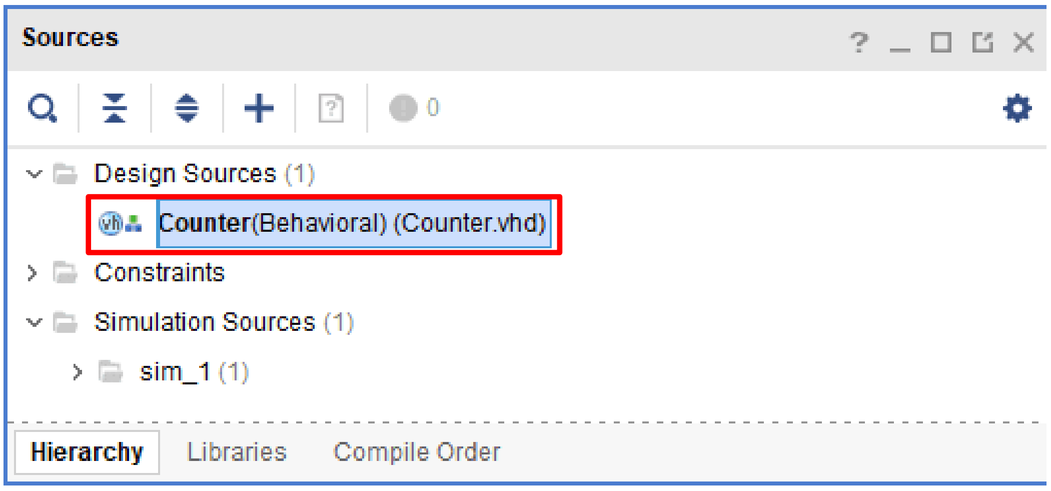
Entity name: Counter

Architecture name: Behavioral

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
CLOCK	in	☐	0	0
RESET	in	☐	0	0
DIRECTION	in	☐	0	0
COUNT_OUT	out	✓	3	0
	in	☐	0	0

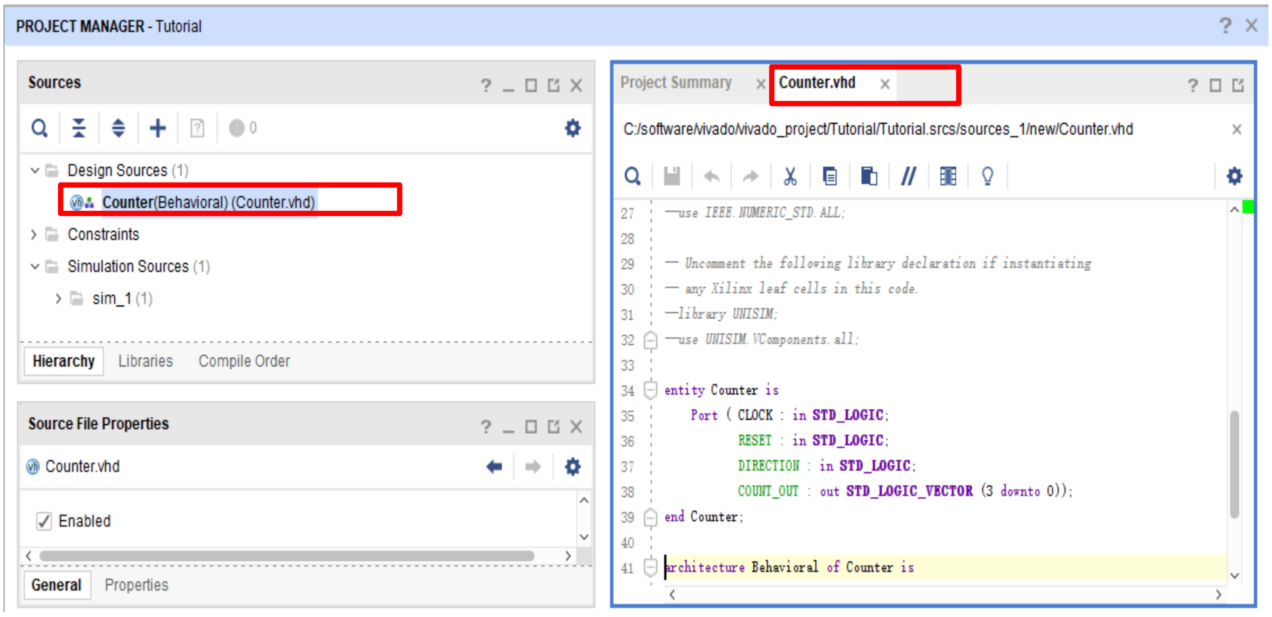
Click **Ok** and you will see that a **Counter** module is created under the **Design Sources**.


Sources

- Design Sources (1)
  - Counter(Behavioral) (Counter.vhd)
- Constraints
- Simulation Sources (1)
  - sim_1 (1)

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.



C:/software/vivado/vivado_project/Tutorial/Tutorial.srcs/sources_1/new/Counter.vhd

```
--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
```

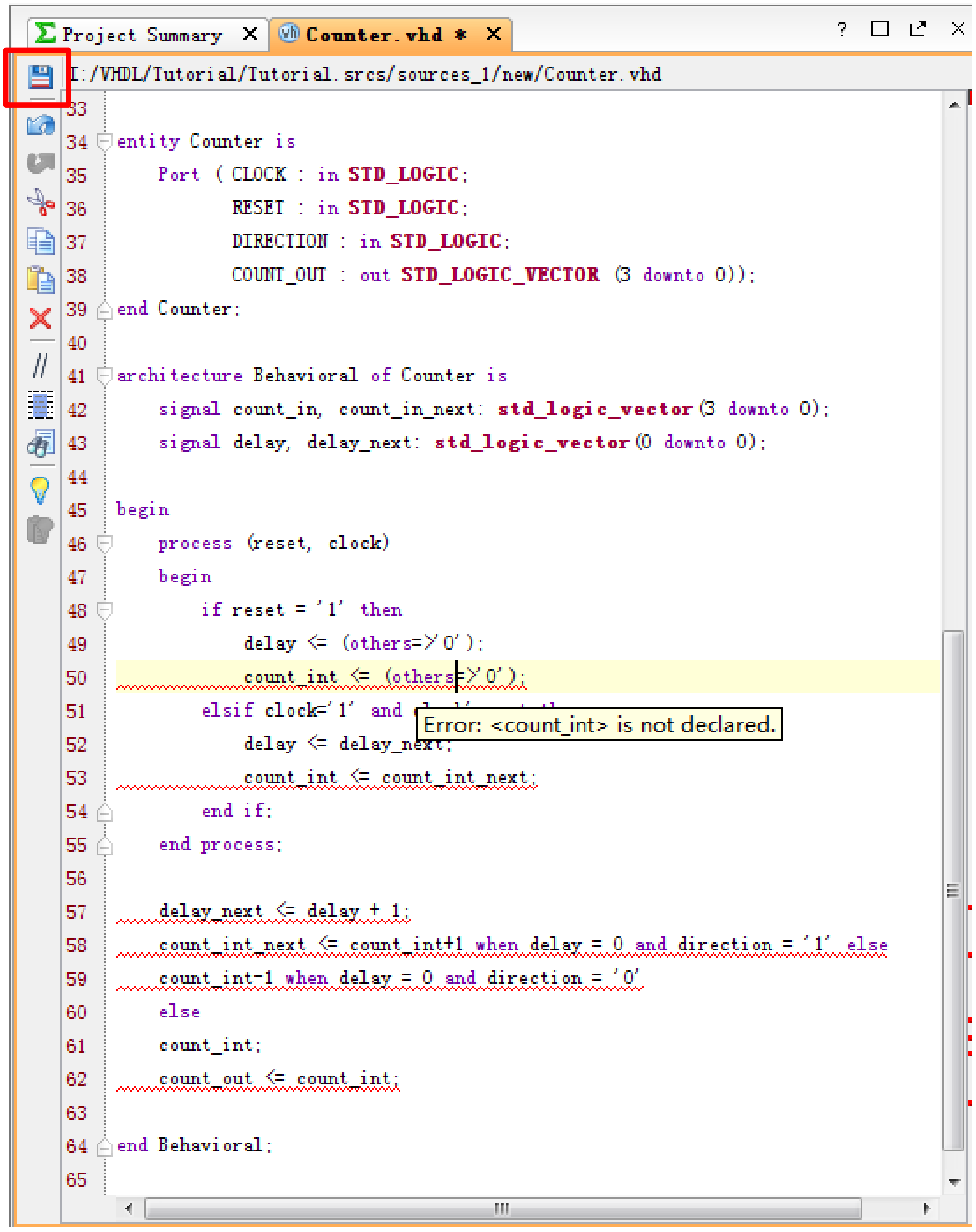
Edit the source code as follows:

- 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.



```
I:/VHDL/Tutorial/Tutorial.srcs/sources_1/new/Counter.vhd

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_int <= (others=>'0');
        elsif clock='1' and
            delay <= delay_next;
            count_int <= count_int_next;
        end if;
    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;

end Behavioral;
```

Error: <count_int> is not declared.

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;
```

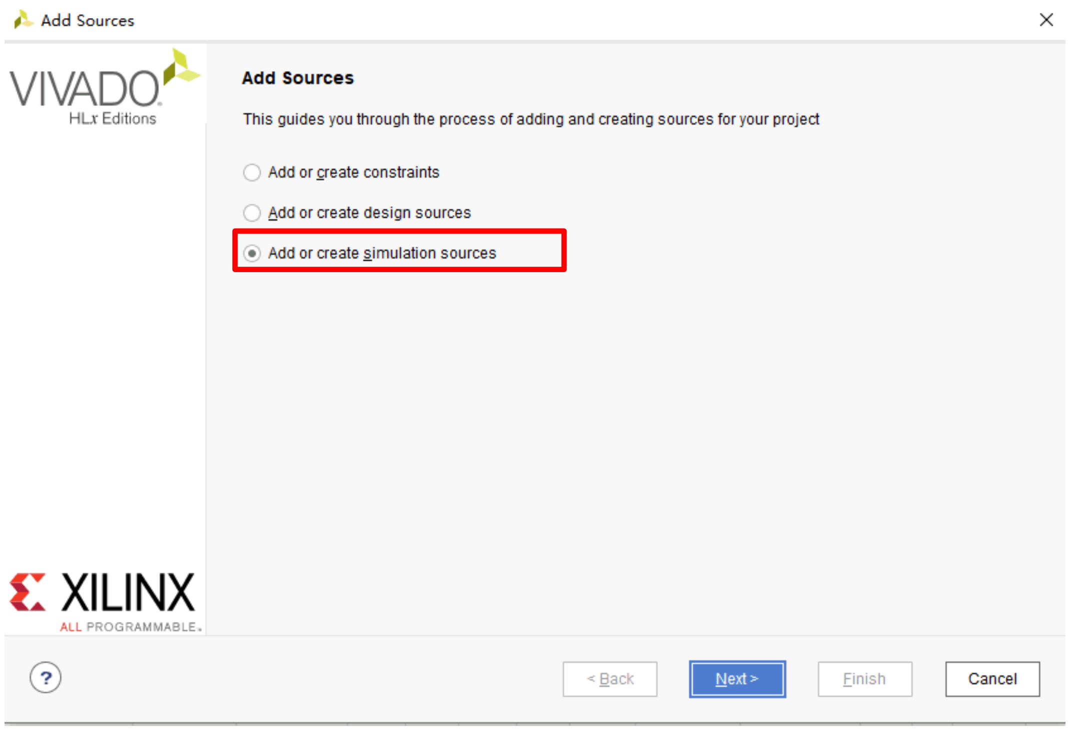
in the beginning.

- 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**

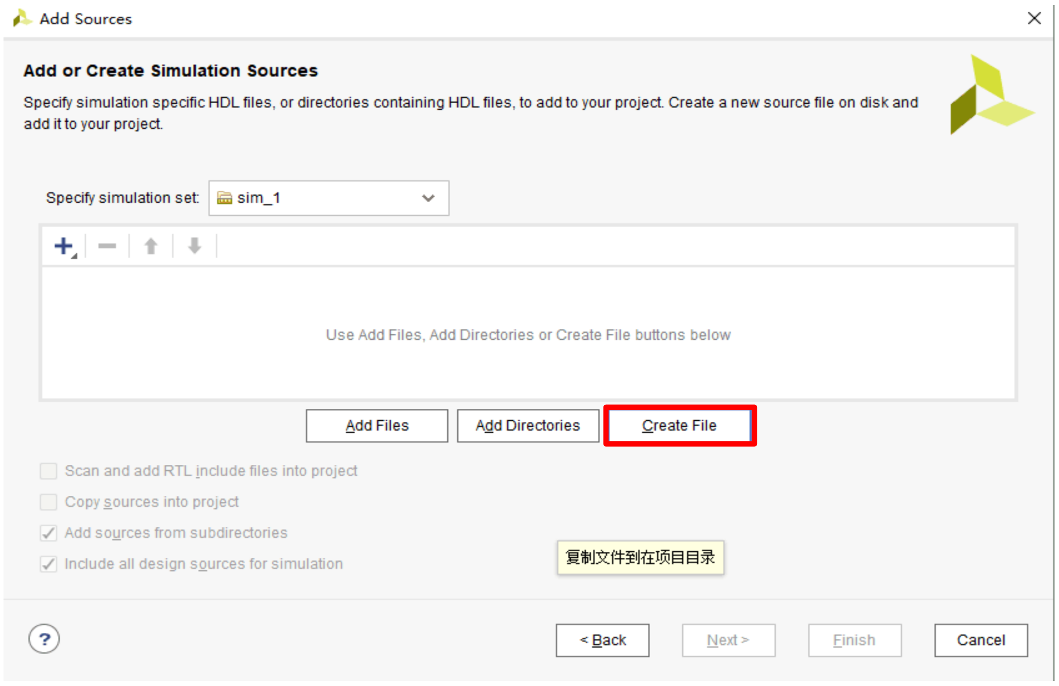


Add Sources

This guides you through the process of adding and creating sources for your project

- Add or create constraints
- Add or create design sources
- [✓ Add or create simulation sources]

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


Add Sources

Add or Create Simulation Sources

Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.

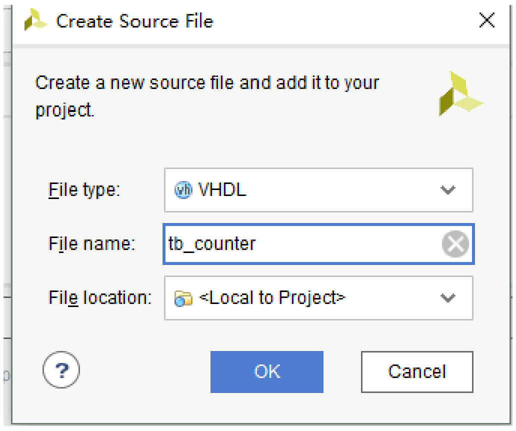
Specify simulation set: sim_1

Use Add Files, Add Directories or Create File buttons below

- [ ] Scan and add RTL include files into project
- [ ] Copy sources into project
- [x] Add sources from subdirectories
- [x] Include all design sources for simulation

复制文件到在项目目录

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


Create Source File

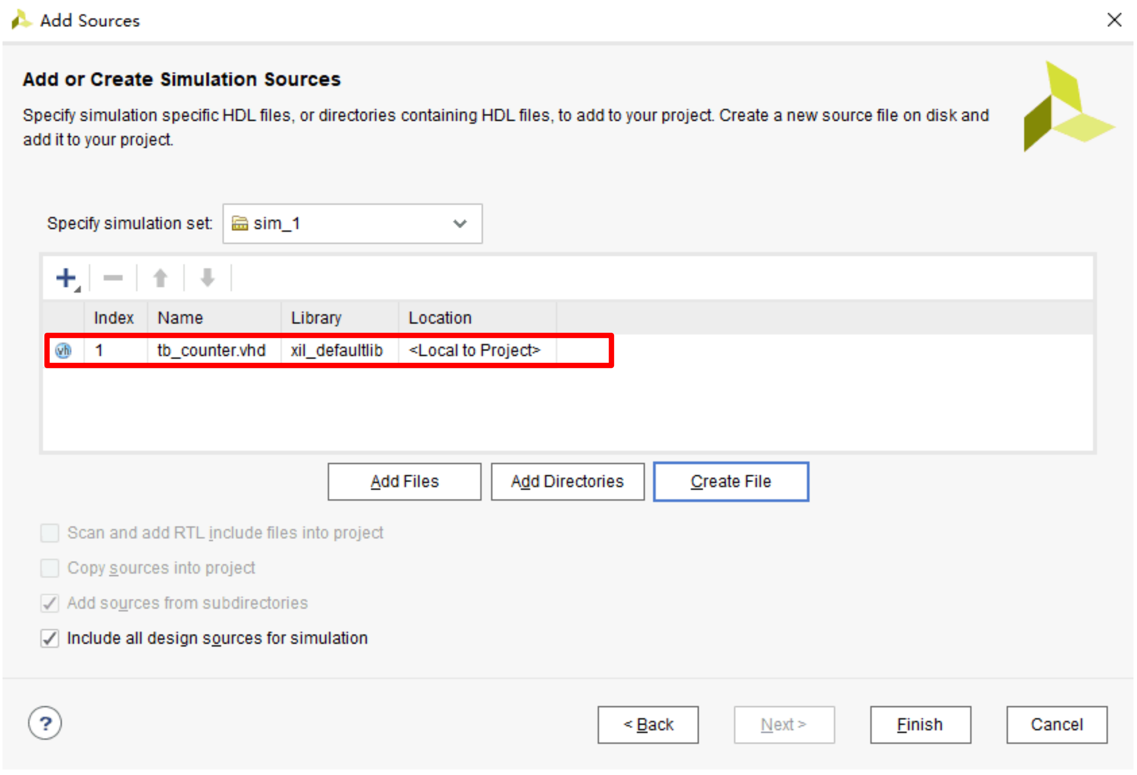
Create a new source file and add it to your project.

File type: VHDL

File name: tb_counter

File location: <Local to Project>

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


Add Sources

Add or Create Simulation Sources

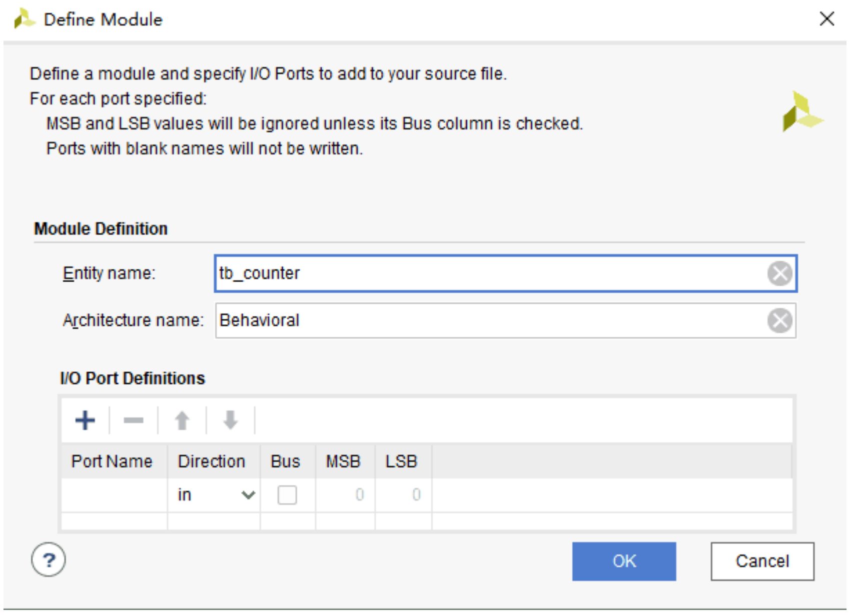
Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.

Specify simulation set: sim_1

Index	Name	Library	Location
1	tb_counter.vhd	xil_defaultlib	<Local to Project>

- [ ] Scan and add RTL include files into project
- [ ] Copy sources into project
- [x] Add sources from subdirectories
- [x] Include all design sources for simulation

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.


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

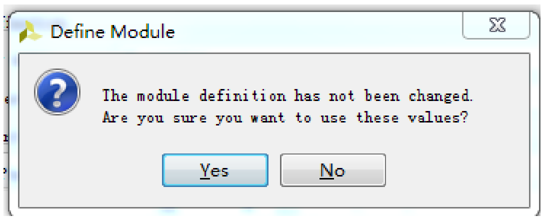
Entity name: tb_counter

Architecture name: Behavioral

I/O Port Definitions

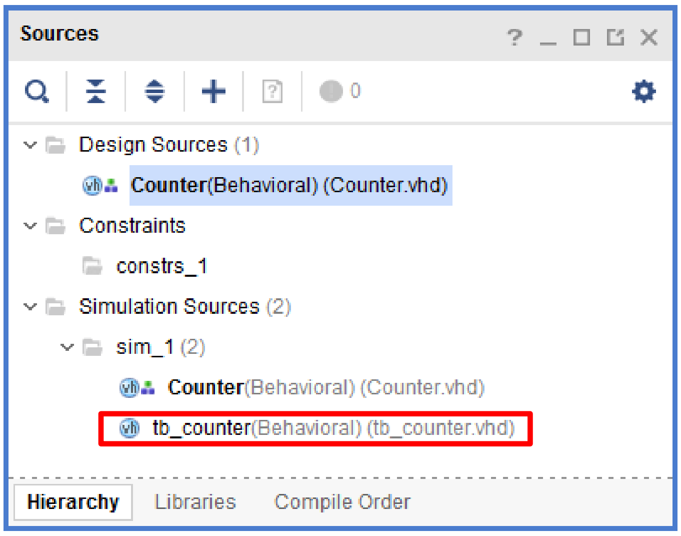
Port Name	Direction	Bus	MSB	LSB
	in	☐	0	0

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

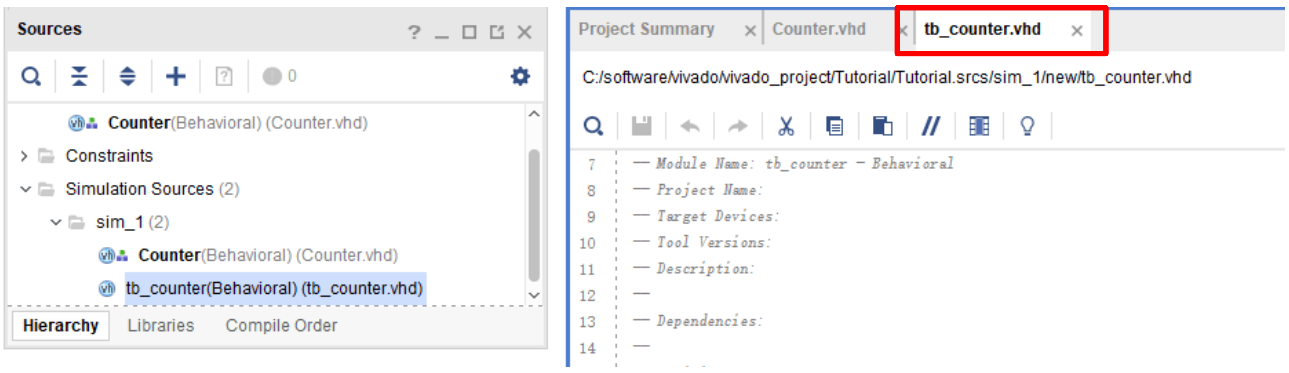


The module definition has not been changed.
Are you sure you want to use these values?

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.



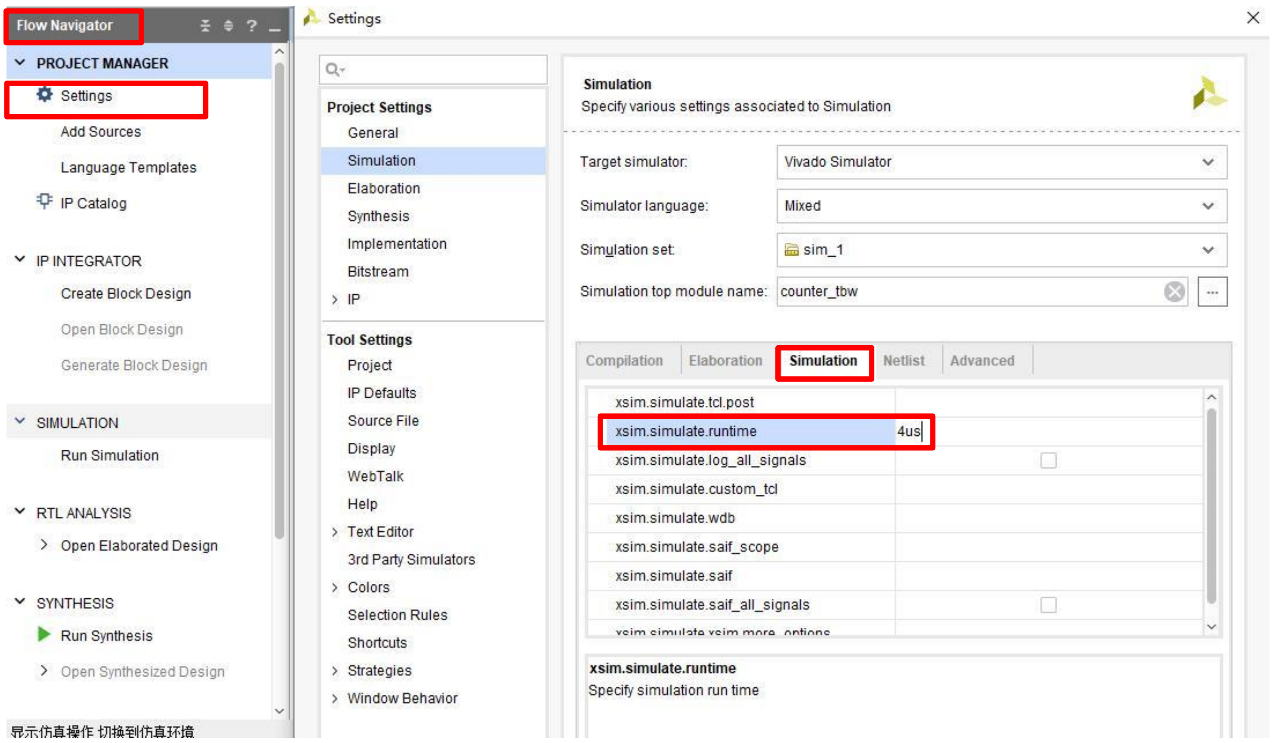
C:/software/vivado/vivado_project/Tutorial/Tutorial.srcs/sim_1/new/tb_counter.vhd

```
-- Module Name: tb_counter - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
```

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.



Simulation
Specify various settings associated to Simulation

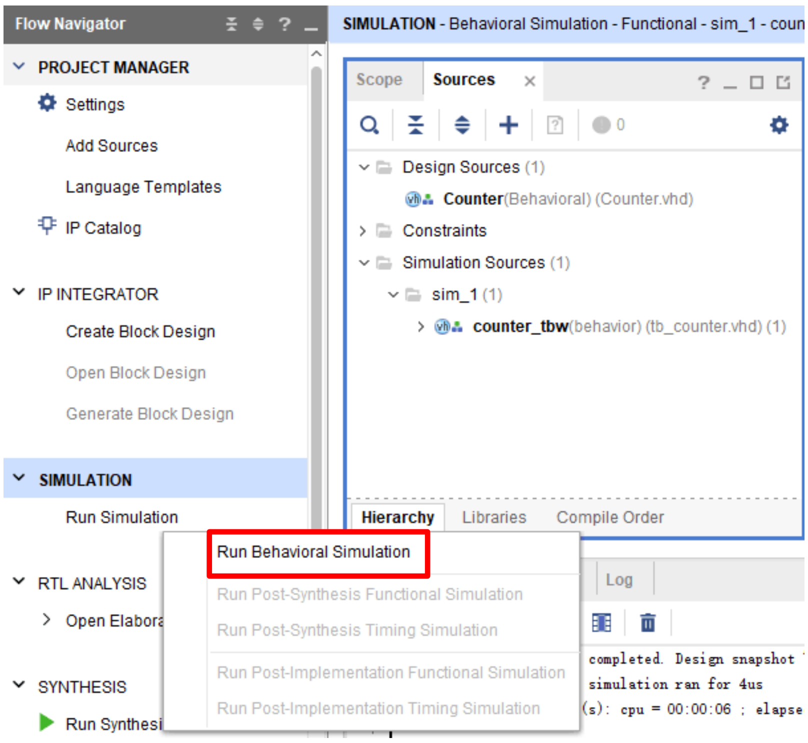
Target simulator: Vivado Simulator
Simulator language: Mixed
Simulation set: sim_1
Simulation top module name: counter_tbw

xsim.simulate.tcl.post	
xsim.simulate.runtime	4us
xsim.simulate.log_all_signals	☐
xsim.simulate.custom_tcl	
xsim.simulate.wdb	
xsim.simulate.saif_scope	
xsim.simulate.saif	
xsim.simulate.saif_all_signals	☐
xsim.simulate.xsim.more_options	

xsim.simulate.runtime
Specify simulation run time

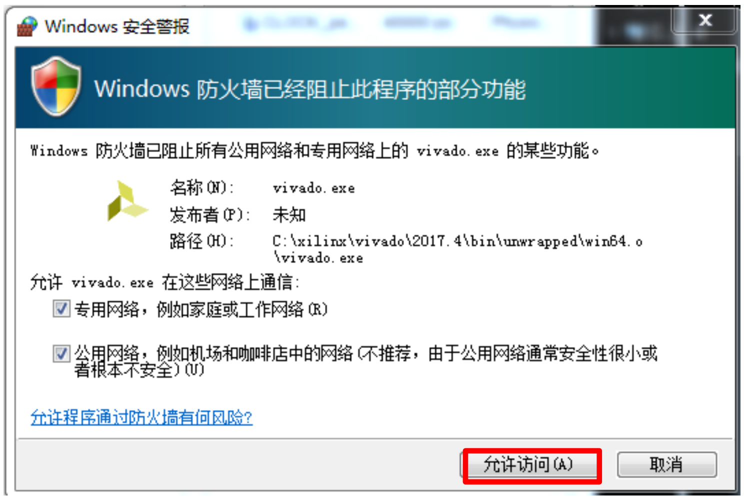
Click **OK** to return.

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


Flow Navigator
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 Elabora
SYNTHESIS
Run Synthesi
SIMULATION - Behavioral Simulation - Functional - sim_1 - coun
Scope
Sources
Design Sources (1)
Counter(Behavioral) (Counter.vhd)
Constraints
Simulation Sources (1)
sim_1 (1)
counter_tbw(behavior) (tb_counter.vhd) (1)
Hierarchy
Libraries
Compile Order
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
Run Post-Implementation Timing Simulation
Log

```
completed. Design snapshot
simulation ran for 4us
(s): cpu = 00:00:06 ; elapse
```

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



Windows 安全警报

Windows 防火墙已经阻止此程序的部分功能

Windows 防火墙已阻止所有公用网络和专用网络上的 vivado.exe 的某些功能。

名称(N):	vivado.exe
发布者(P):	未知
路径(H):	C:\xilinx\vivado\2017.4\bin\unwrapped\win64.o\vivado.exe

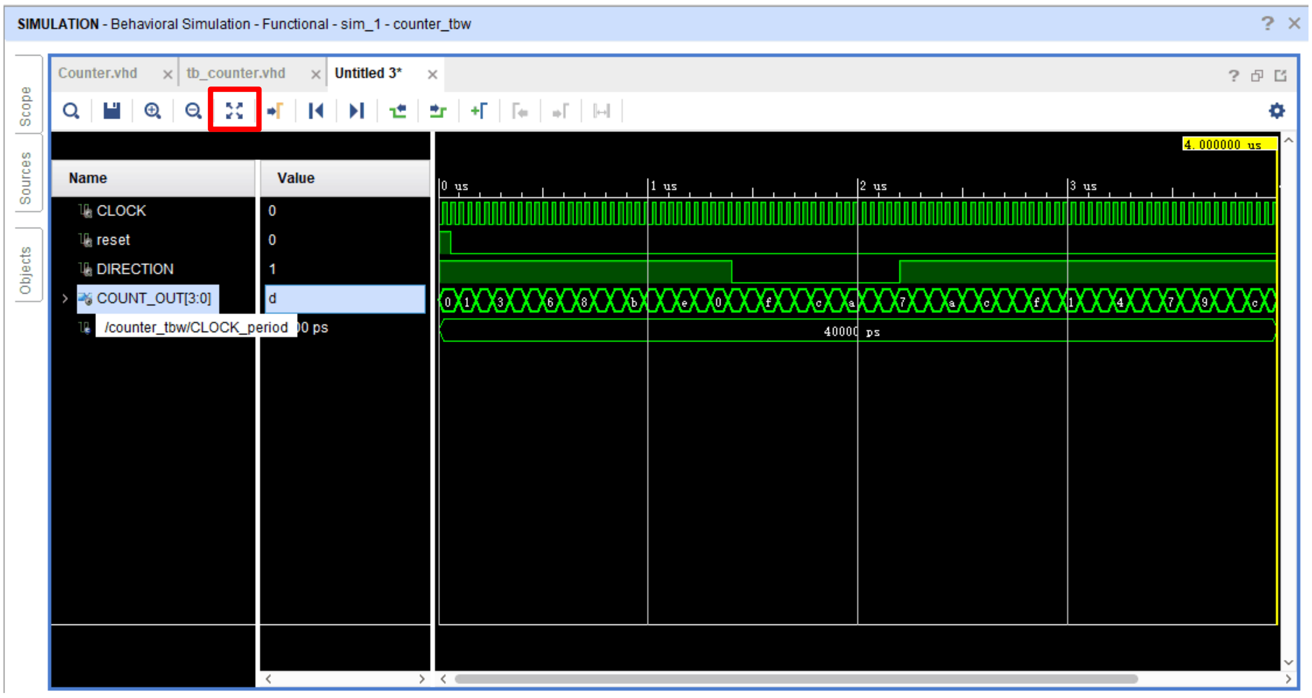
允许 vivado.exe 在这些网络上通信:

☑ 专用网络，例如家庭或工作网络(R)

☑ 公用网络，例如机场和咖啡店中的网络(不推荐，由于公用网络通常安全性很小或者根本不安全)(U)

允许程序通过防火墙有何风险?

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

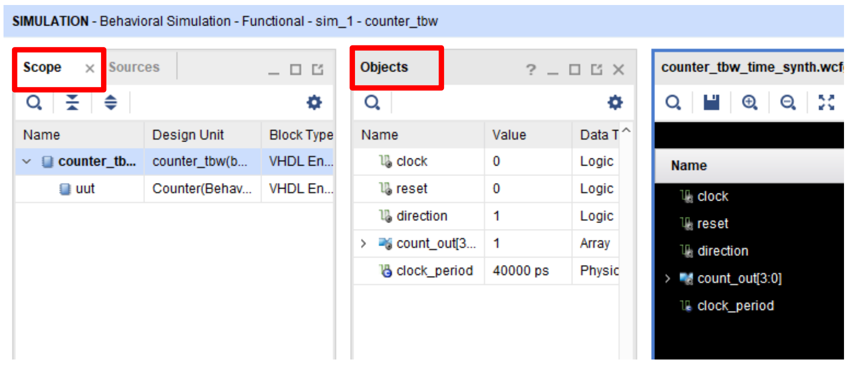


SIMULATION - Behavioral Simulation - Functional - sim_1 - counter_tbw

Name	Value
CLOCK	0
reset	0
DIRECTION	1
COUNT_OUT[3:0]	d
/counter_tbw/CLOCK_period	0 ps

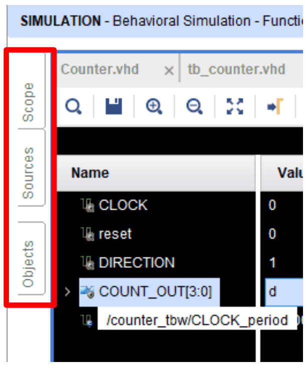
Sometimes we may add some internal signals to the simulation waveforms to facilitate debugging.

First you need to make sure that the **Scope** and **Objects** columns are visible as shown below. If not, please click on the sidebar to make it visible.


SIMULATION - Behavioral Simulation - Functional - sim_1 - counter_tbw

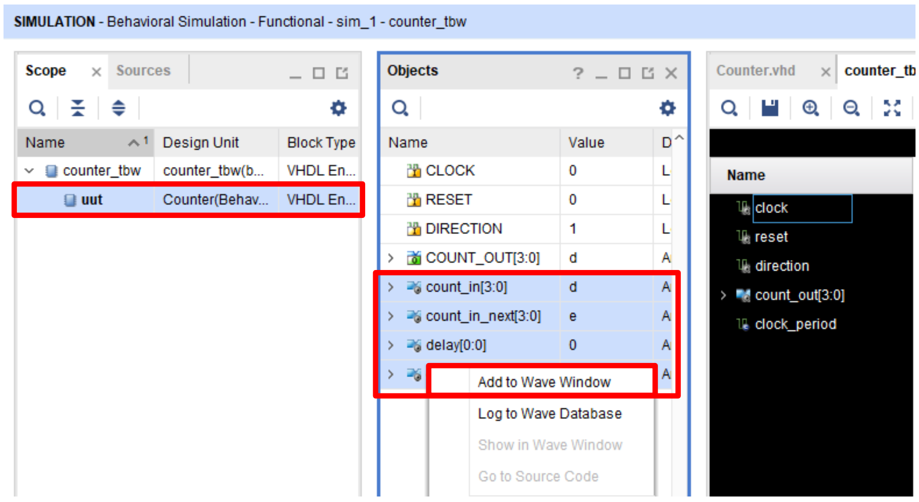
Name	Design Unit	Block Type
counter_tb...	counter_tbw(b...	VHDL En...
uut	Counter(Behav...	VHDL En...

Name	Value	Data T
clock	0	Logic
reset	0	Logic
direction	1	Logic
count_out[3...	1	Array
clock_period	40000 ps	Physic


SIMULATION - Behavioral Simulation - Functi

Name	Valu
CLOCK	0
reset	0
DIRECTION	1
COUNT_OUT[3:0]	d
/counter_tbw/CLOCK_period	

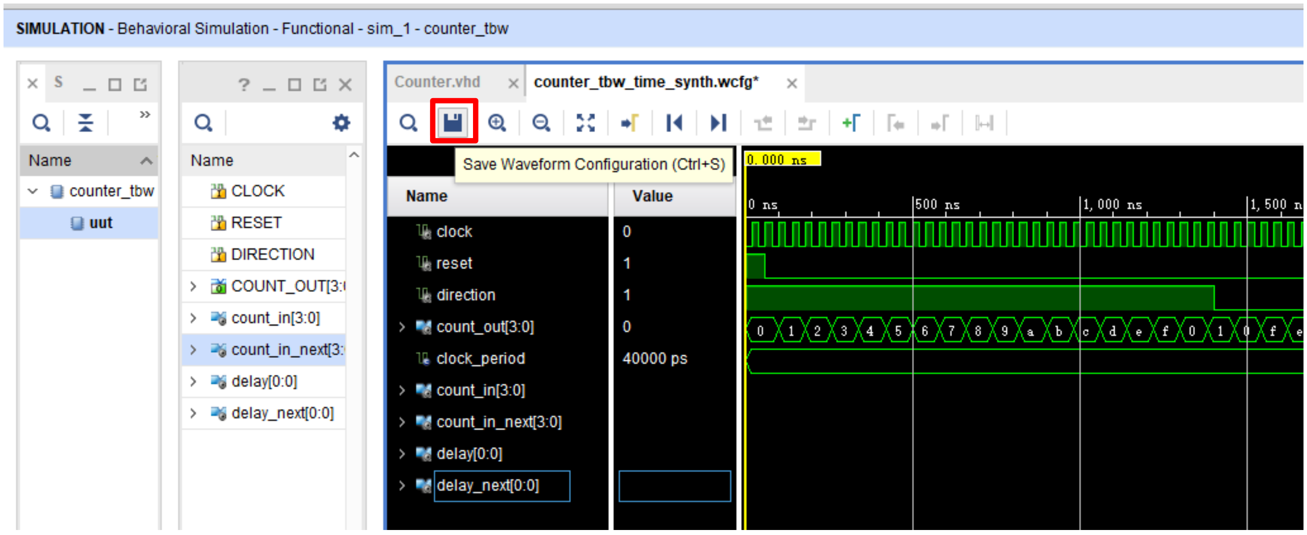
Then click the component **uut** instantiated in the **Scope** column, and select the internal signals of which the waveforms are to be displayed in the **Objects** column such as **count_in**, **count_in_next**, **delay** and **delay_next** (use Ctrl and the left-click to select multiple signals). Then right-click and select **Add to Wave Windows**.


SIMULATION - Behavioral Simulation - Functional - sim_1 - counter_tbw

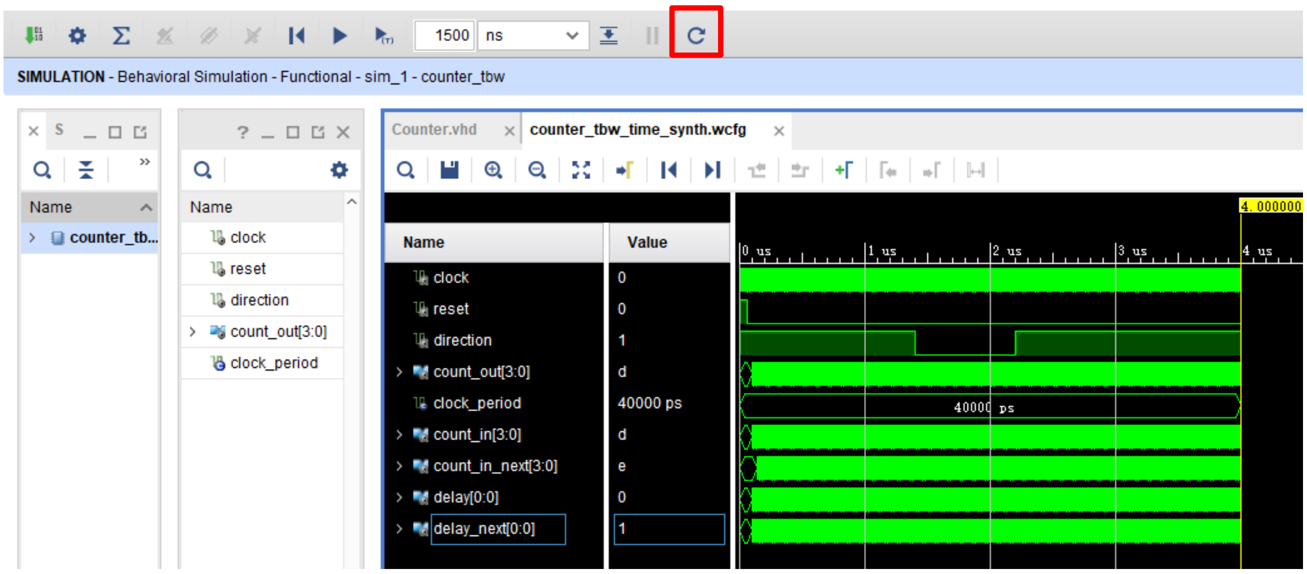
Name	Design Unit	Block Type
counter_tbw	counter_tbw(b...	VHDL En...
uut	Counter(Behav...	VHDL En...

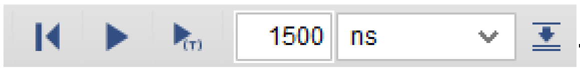
Name	Value	D
CLOCK	0	L
RESET	0	L
DIRECTION	1	L
COUNT_OUT[3:0]	d	A
count_in[3:0]	d	A
count_in_next[3:0]	e	A
delay[0:0]	0	A
		A

Now the internal signals have been added to the waveform file. To avoid repeating the adding of internal signals in the subsequent simulations, you may save the waveform configuration file by clicking the **save button** and then click ok to save the file as a .wcfg file.

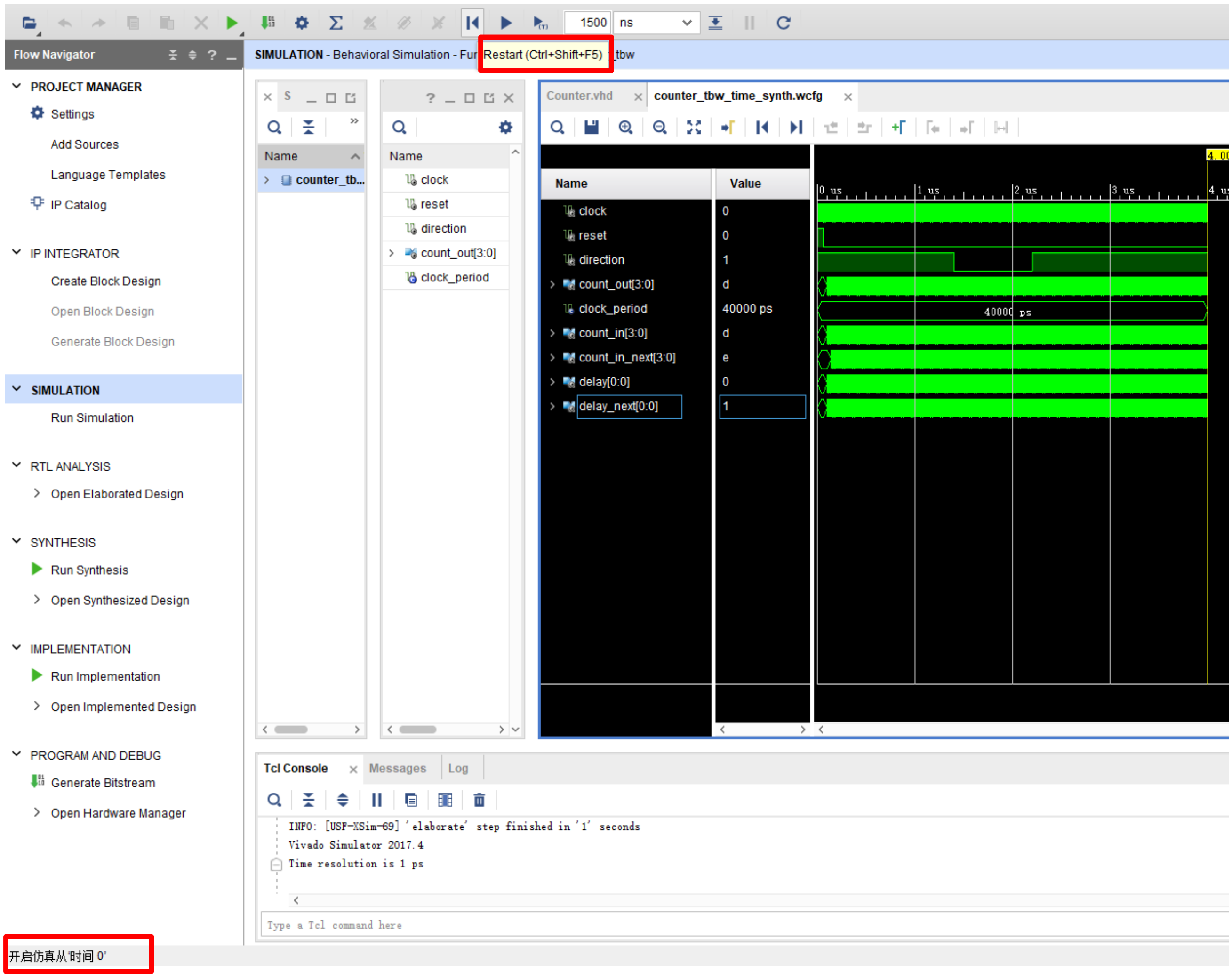


In order to see their waveforms, there are several ways to re-run the simulation starting at 0ns. First, you may use (Relaunch Simulation) directly. The simulation time (xsim.simulate.runtime) was set previously to 4us in the Project Setting. So the simulation stops after 4us.



In addition, we can use below the combined buttons to implement waveform running: 

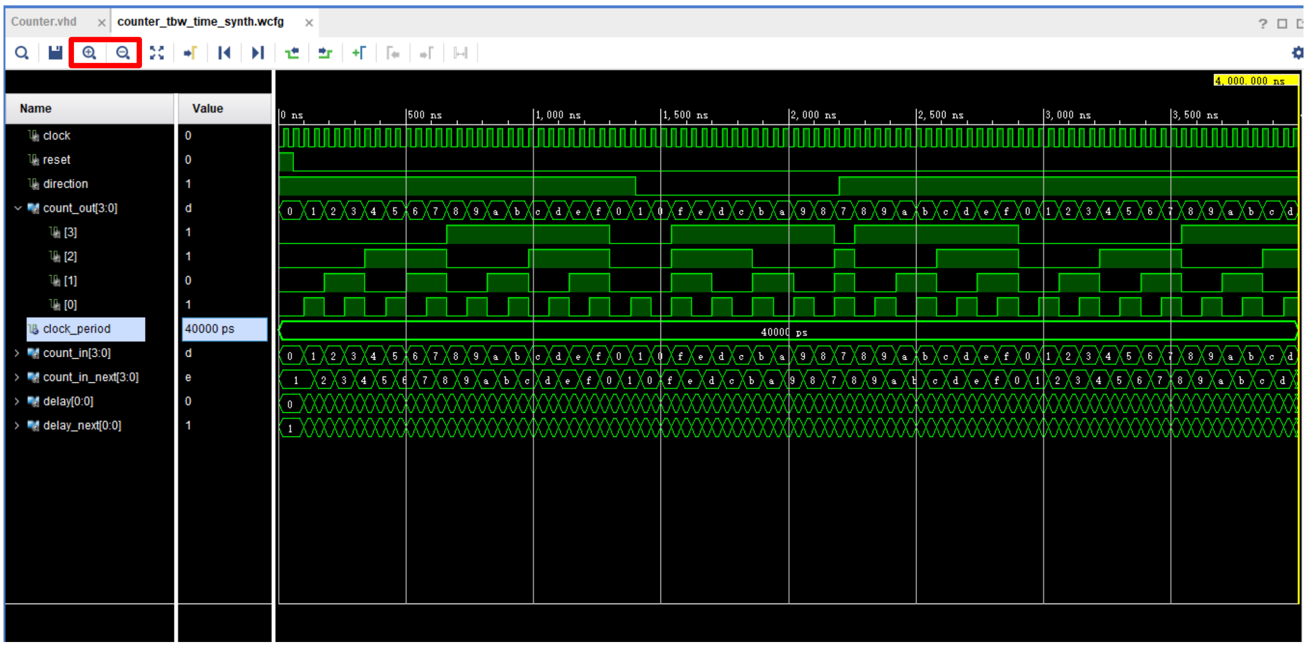
. The functions of these buttons may be viewed by putting cursor on top of these buttons, for example as showing in the following screen capture, and they are useful for future simulations.



```
INFO: [USF-XSim-69] 'elaborate' step finished in '1' seconds
Vivado Simulator 2017.4
Time resolution is 1 ps
```

Name	Value
clock	0
reset	0
direction	1
count_out[3:0]	d
clock_period	40000 ps
count_in[3:0]	d
count_in_next[3:0]	e
delay[0:0]	0
delay_next[0:0]	1

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.



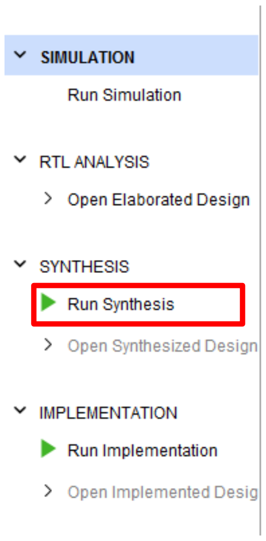
Name	Value
clock	0
reset	0
direction	1
count_out[3:0]	d
[3]	1
[2]	1
[1]	0
[0]	1
clock_period	40000 ps
count_in[3:0]	d
count_in_next[3:0]	e
delay[0:0]	0
delay_next[0:0]	1

Finally, we can observe the output of the signals to verify the function of the counter.

Synthesizing Your Design – Step 4

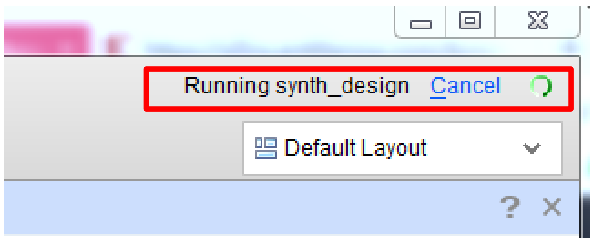
Synthesize the design

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.

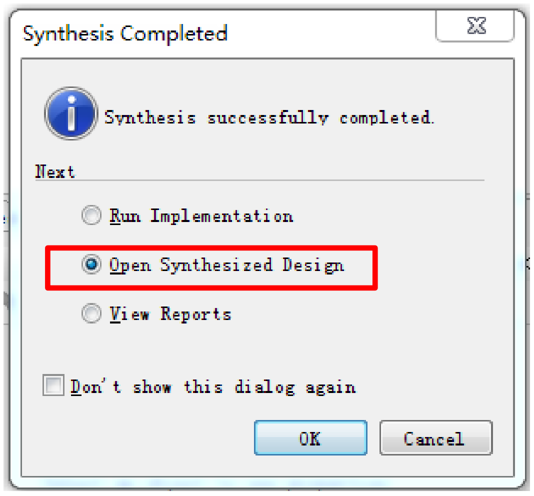


In the followed **Launch Run**, click **OK** to start the synthesis. You may **tick** the option **Don't show this dialog again**, so that this dialog will not show thereafter.

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**.


Synthesis Completed

Synthesis successfully completed.

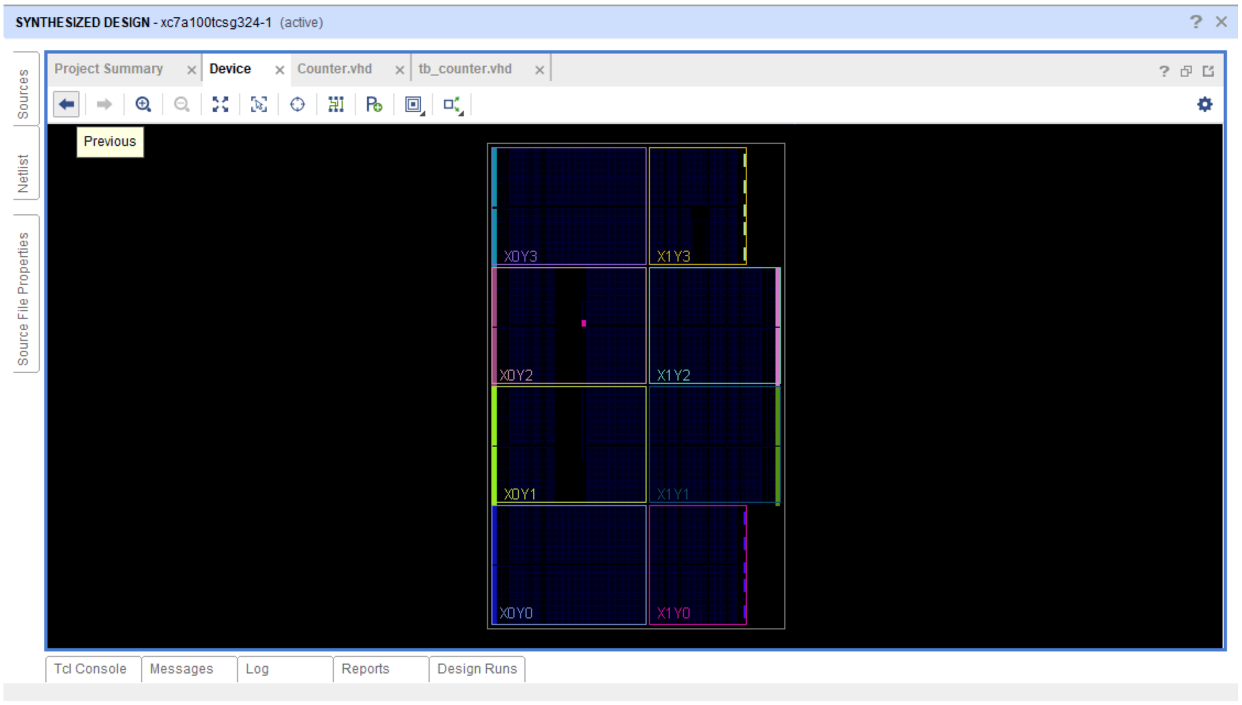
Next

Run Implementation [✓ Open Synthesized Design] View Reports

Don't show this dialog again

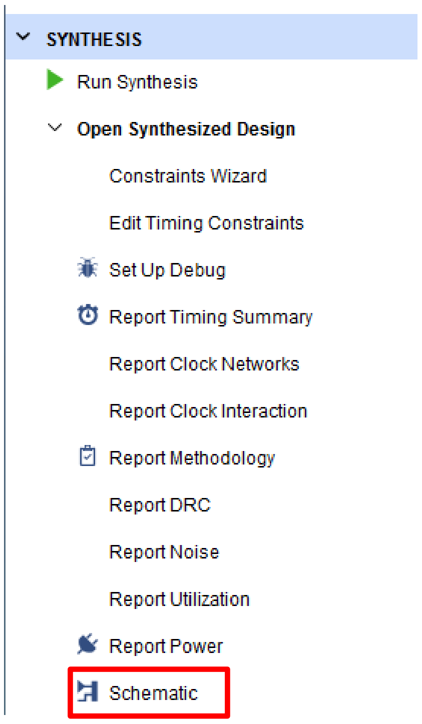
OK Cancel

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

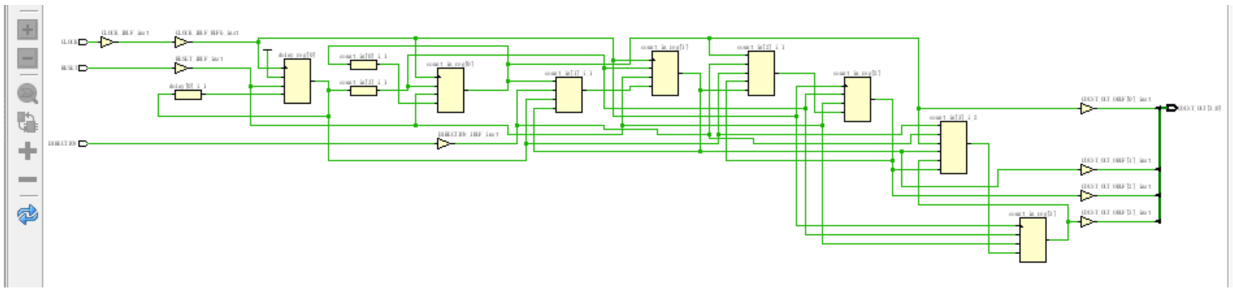

SYNTHESIZED DESIGN - xc7a100tcsg324-1 (active)

View Schematic and Project Summary

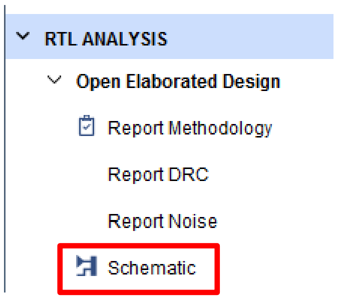
In addition, you can view the **schematic** of your design by click **Synthesis -> Synthesized Design -> Schematic** under **Flow Navigator**.



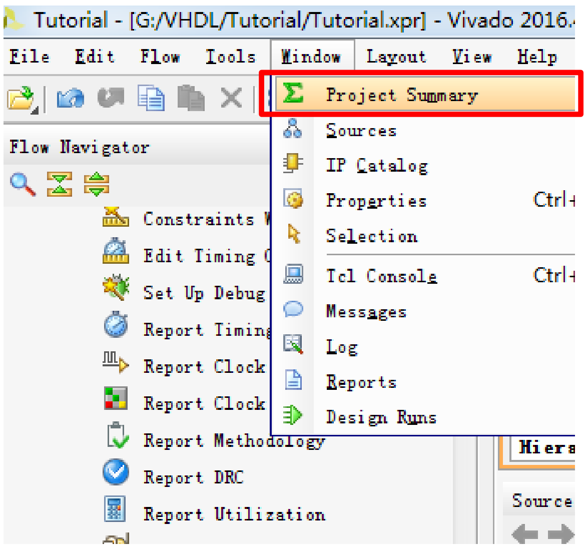
A schematic of the Counter will be shown on the right window.



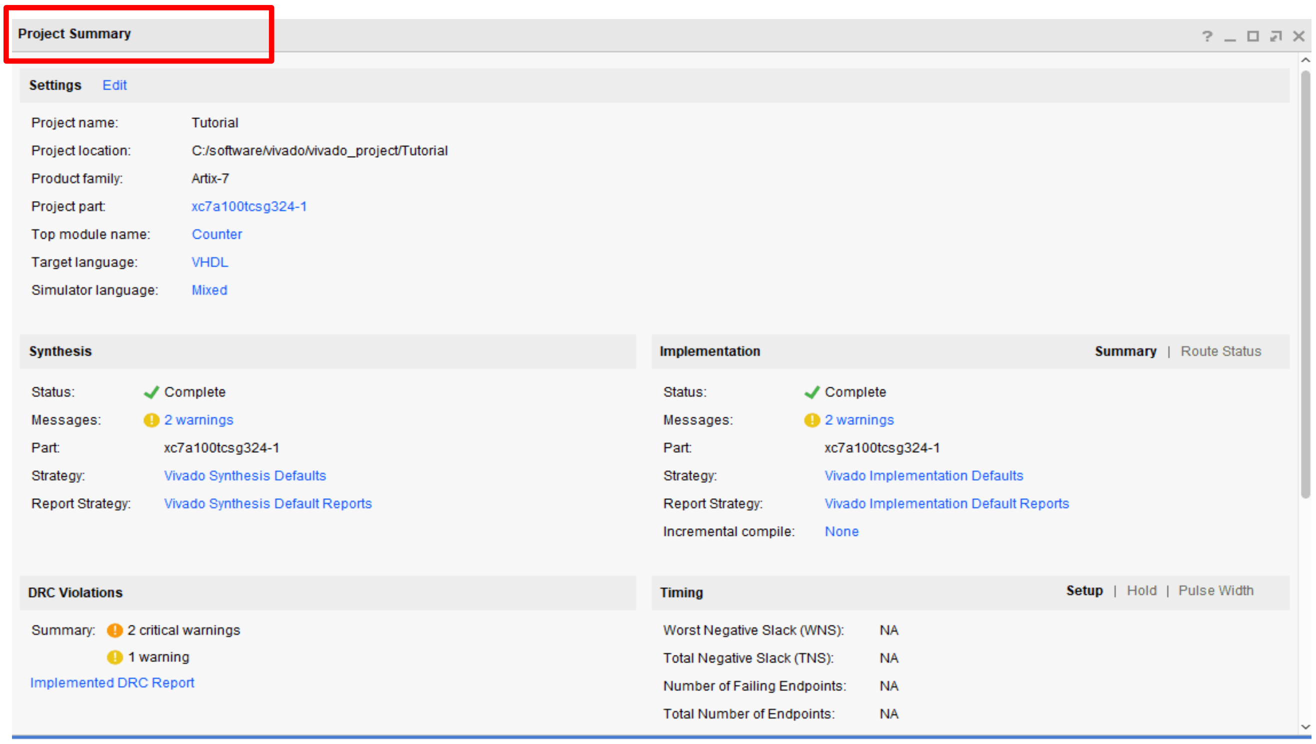
You can zoom to view the schematic in detail.
You may also view the RTL design under **Flow Navigator -> RTL analysis -> Open Elaborated Design -> Schematic**.



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.


Project Summary

Settings Edit

Project name:	Tutorial
Project location:	C:/software/vivado/vivado_project/Tutorial
Product family:	Artix-7
Project part:	xc7a100tcsg324-1
Top module name:	Counter
Target language:	VHDL
Simulator language:	Mixed

Synthesis

Status:	✓ Complete
Messages:	2 warnings
Part:	xc7a100tcsg324-1
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports

Implementation **Summary** | Route Status

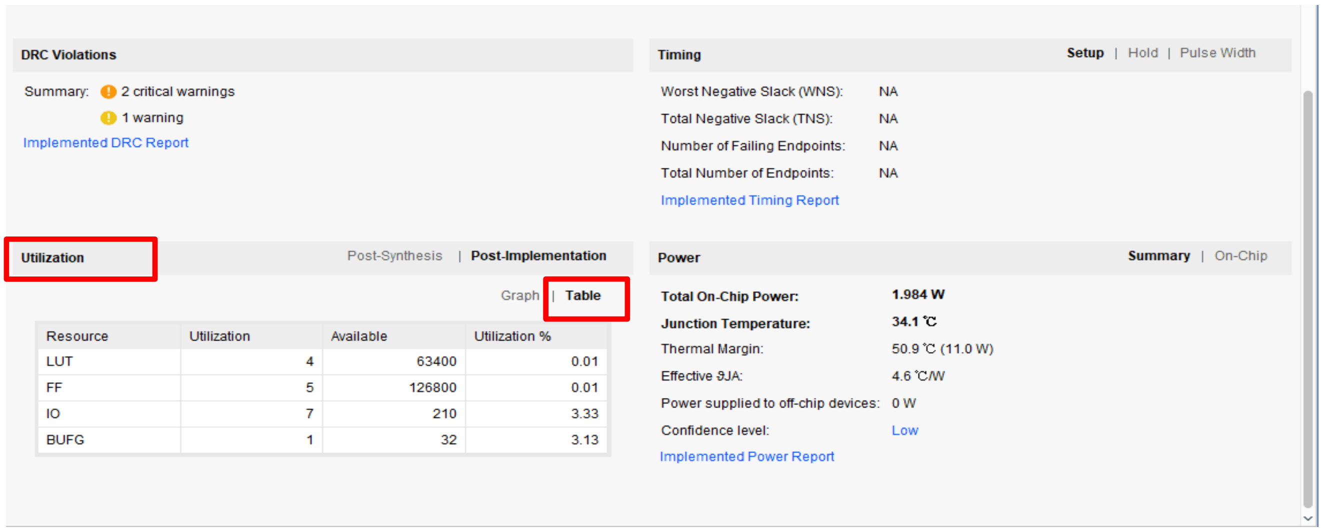
Status:	✓ Complete
Messages:	2 warnings
Part:	xc7a100tcsg324-1
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental compile:	None

DRC Violations

Summary: 2 critical warnings
1 warning
Implemented DRC Report

Timing **Setup** | Hold | Pulse Width

Worst Negative Slack (WNS):	NA
Total Negative Slack (TNS):	NA
Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA



DRC Violations

Summary: 2 critical warnings
1 warning
Implemented DRC Report

Timing — Setup | Hold | Pulse Width

Worst Negative Slack (WNS):	NA
Total Negative Slack (TNS):	NA
Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA

Implemented Timing Report

Utilization — Post-Synthesis | Post-Implementation — Graph | Table

Resource	Utilization	Available	Utilization %
LUT	4	63400	0.01
FF	5	126800	0.01
IO	7	210	3.33
BUFG	1	32	3.13

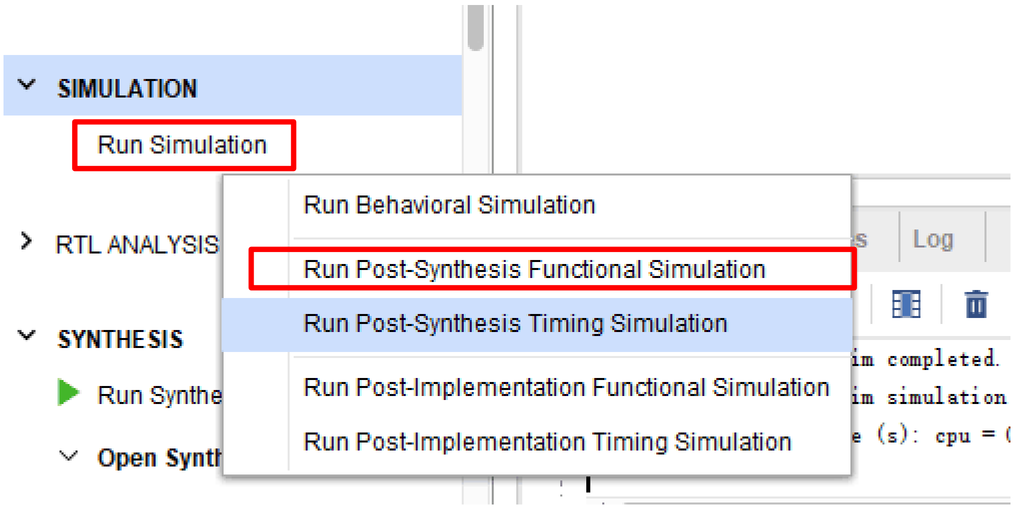
Power — Summary | On-Chip

Total On-Chip Power:	1.984 W
Junction Temperature:	34.1 ℃
Thermal Margin:	50.9 ℃ (11.0 W)
Effective ϑJA:	4.6 ℃/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

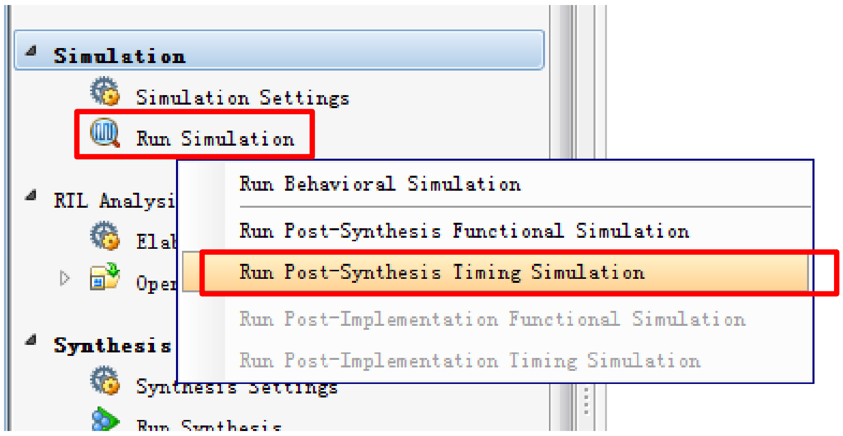
Implemented Power Report

Post-Synthesis Functional and Timing Simulation

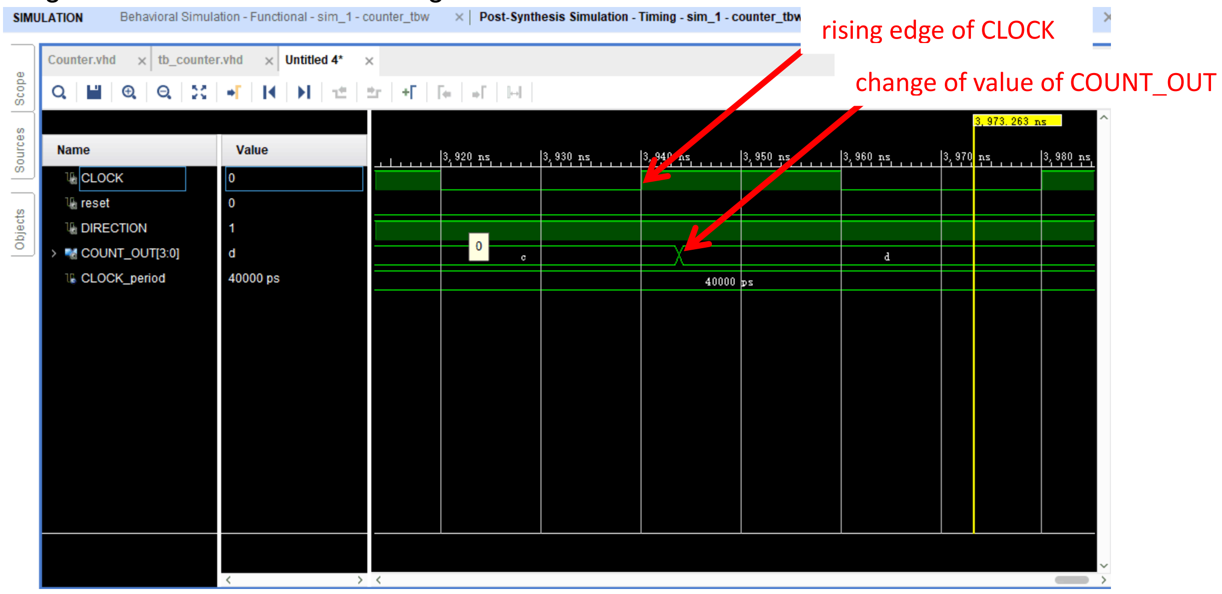
Run the **post-synthesis functional simulation** in the same way as **behavior simulation** by selecting **Post-synthesis function simulation** as follows.


SIMULATION
Run Simulation
RTL ANALYSIS
SYNTHESIS
Run Synthe
Open Synth
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
Run Post-Implementation Timing Simulation
Log

Once the function is verified, run the **Post-Synthesis Timing Simulation**.


Simulation
Simulation Settings
Run Simulation
Run Behavioral Simulation
Run Post-Synthesis Functional Simulation
Run Post-Synthesis Timing Simulation
Run Post-Implementation Functional Simulation
Run Post-Implementation Timing Simulation
RTL Analysi
Synthesis
Synthesis Settings
Run Synthesis

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

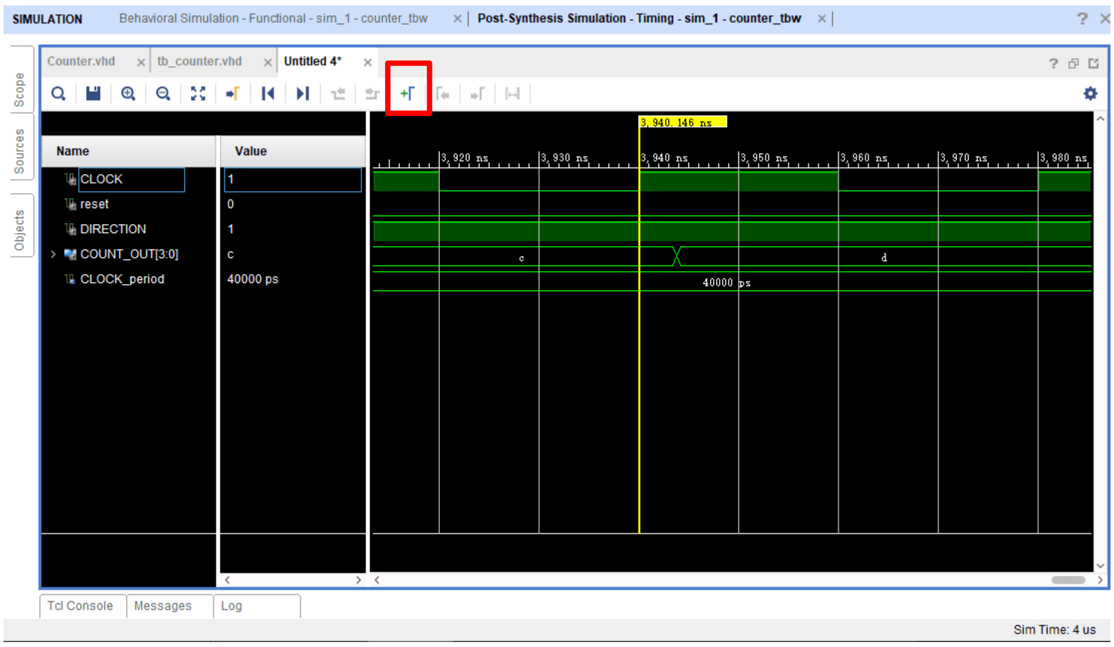


Name	Value
CLOCK	0
reset	0
DIRECTION	1
COUNT_OUT[3:0]	d
CLOCK_period	40000 ps

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

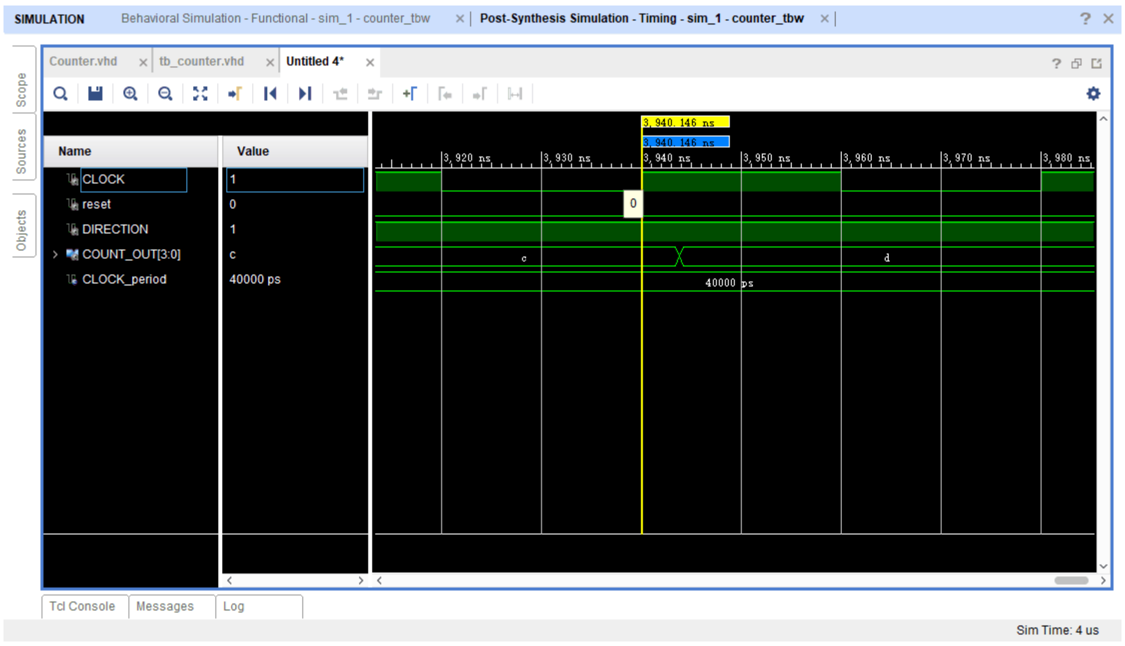
Using Markers in Simulation Waveforms

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



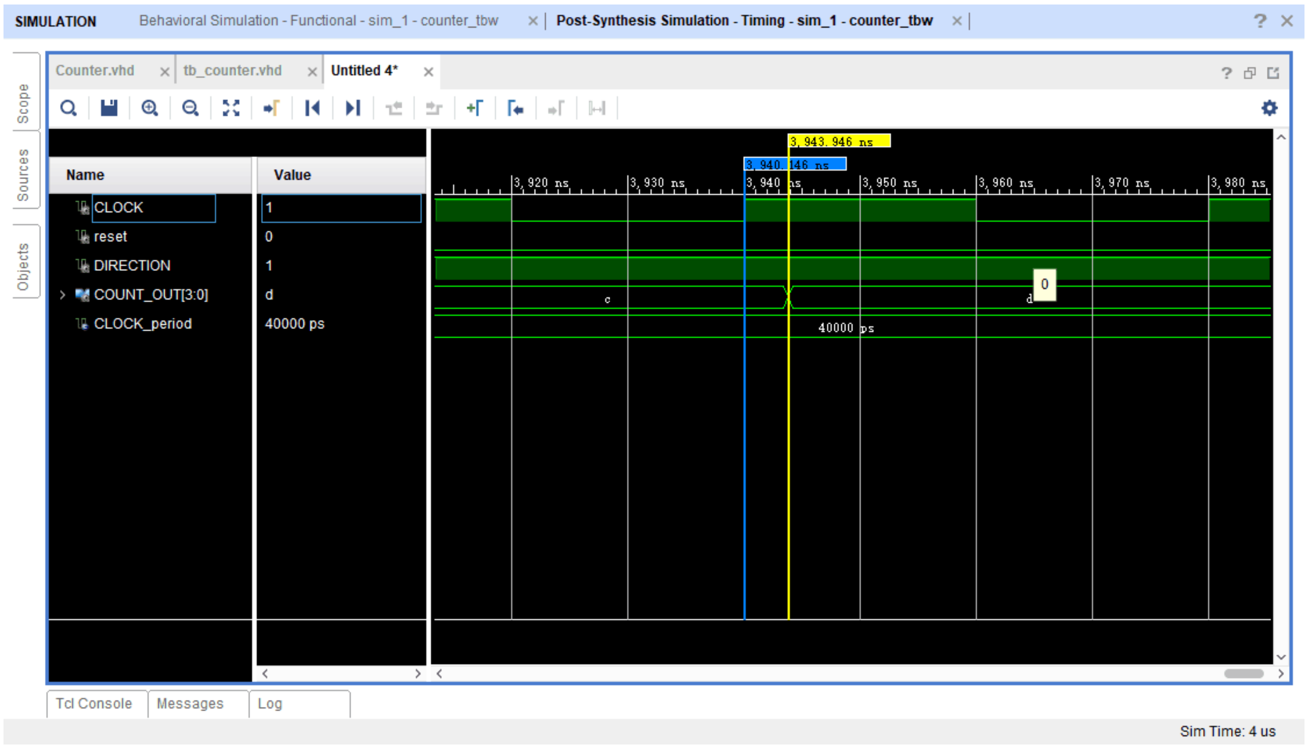
Name	Value
CLOCK	1
reset	0
DIRECTION	1
COUNT_OUT[3:0]	c
CLOCK_period	40000 ps

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

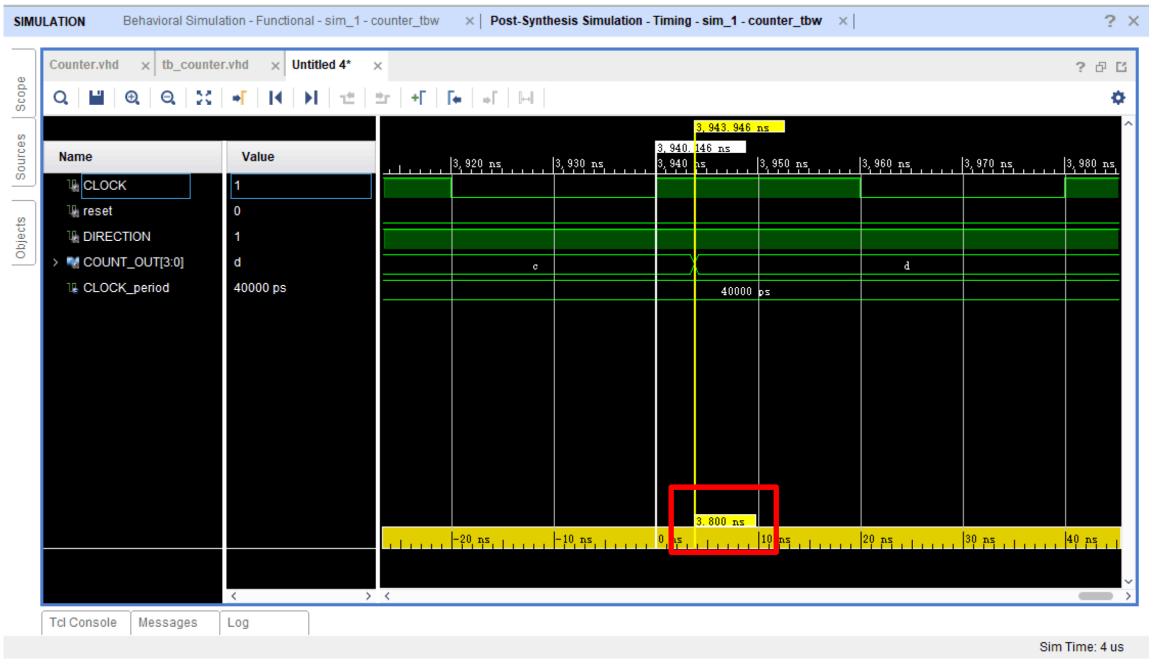


Name	Value
CLOCK	1
reset	0
DIRECTION	1
COUNT_OUT[3:0]	c
CLOCK_period	40000 ps

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.

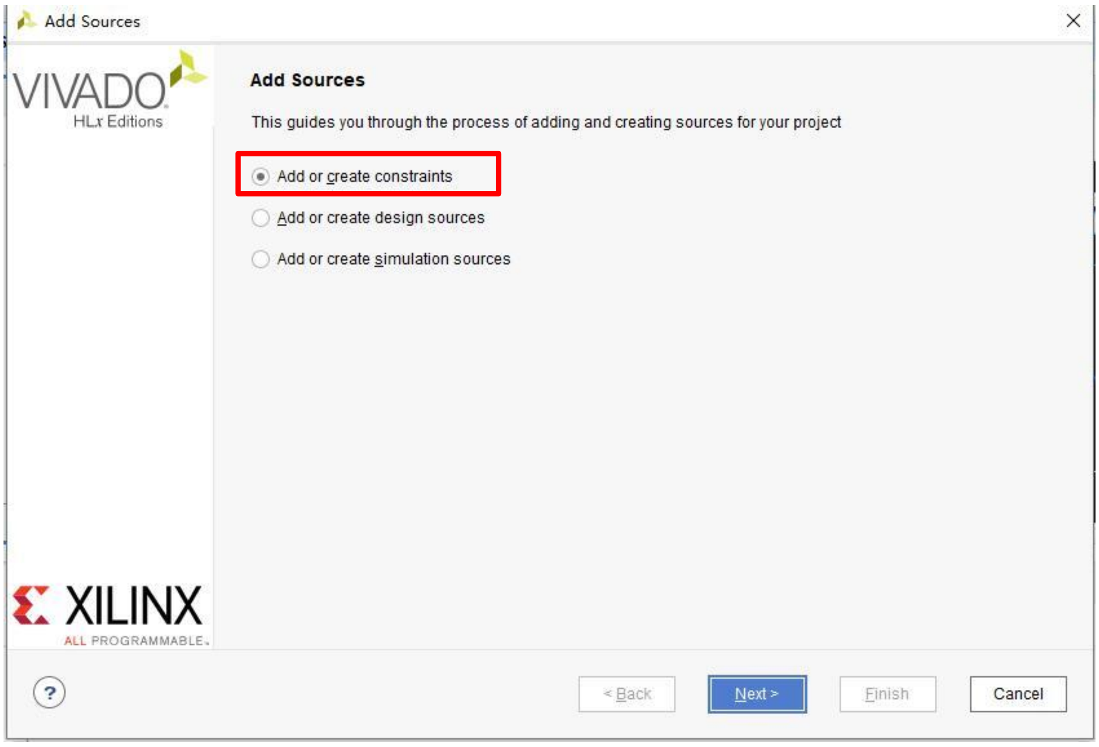
Creating constraint files – Step 5

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**.



Add Sources

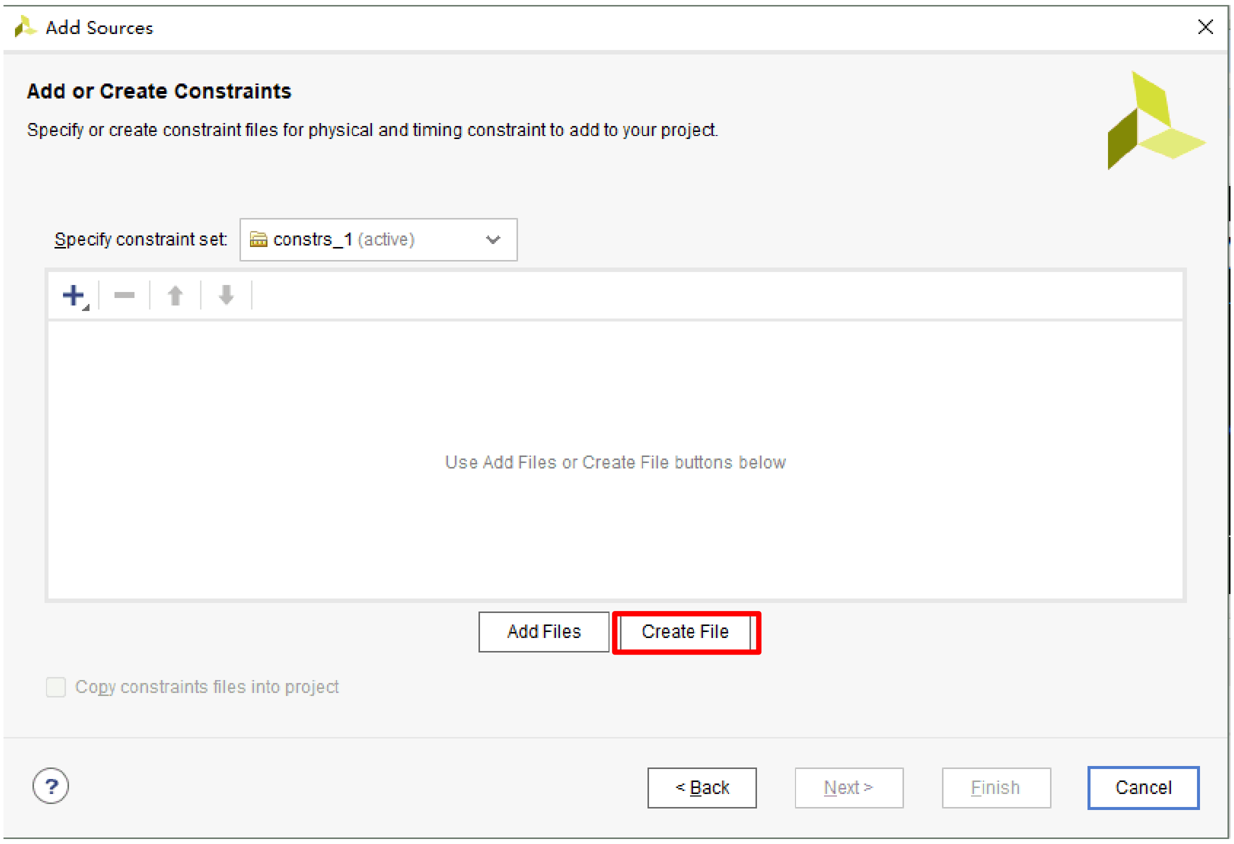
This guides you through the process of adding and creating sources for your project

[✓ Add or create constraints]

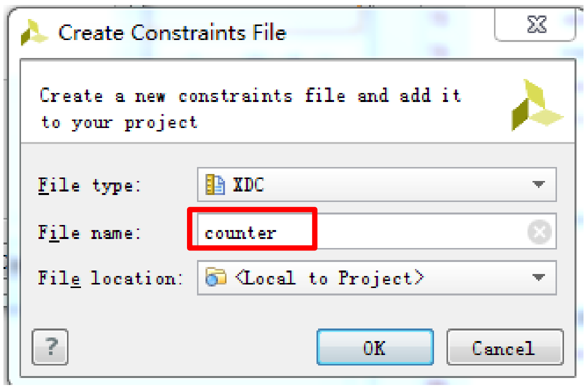
Add or create design sources

Add or create simulation sources

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

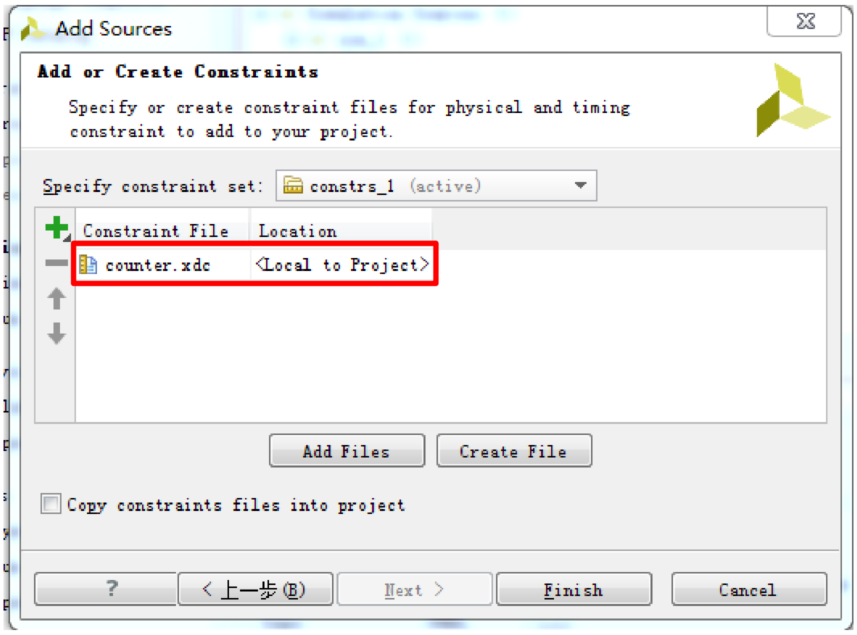

Add Sources
Add or Create Constraints
Specify or create constraint files for physical and timing constraint to add to your project.
Specify constraint set: constrs_1 (active)
Use Add Files or Create File buttons below
Add Files
Create File
Copy constraints files into project
< Back
Next >
Finish
Cancel

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

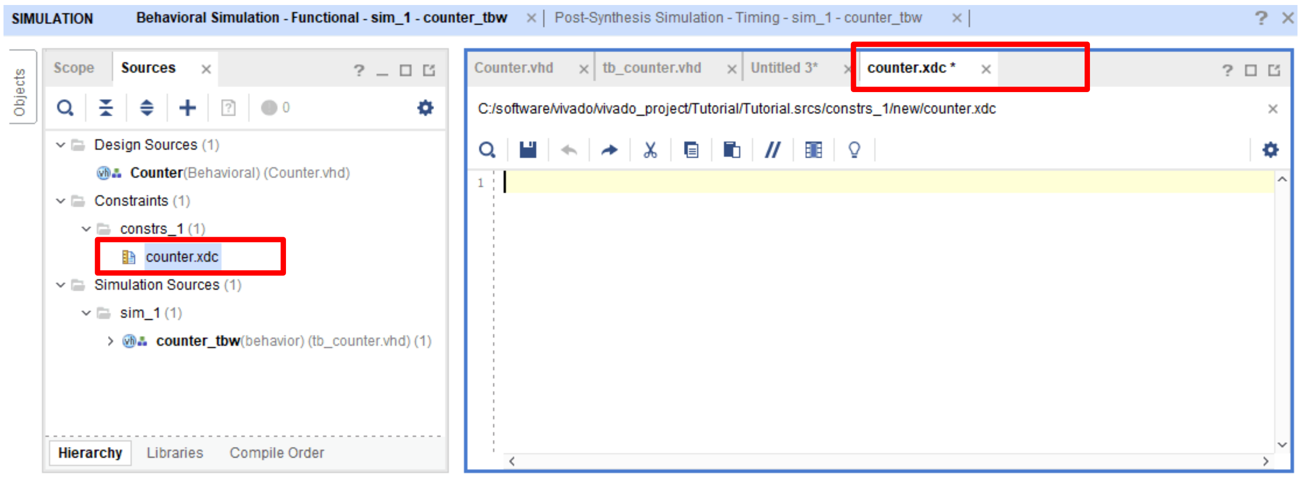


File type: XDC
File name: counter
File location: <Local to Project>

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


Add Sources
Add or Create Constraints
Specify or create constraint files for physical and timing constraint to add to your project.
Specify constraint set: constrs_1 (active)
Constraint File
Location
counter.xdc
<Local to Project>
Add Files
Create File
Copy constraints files into project
?
< 上一步(B)
Next >
Finish
Cancel

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.


SIMULATION
Behavioral Simulation - Functional - sim_1 - counter_tbw
Post-Synthesis Simulation - Timing - sim_1 - counter_tbw
Scope
Sources
Design Sources (1)
Counter(Behavioral) (Counter.vhd)
Constraints (1)
constrs_1 (1)
counter.xdc
Simulation Sources (1)
sim_1 (1)
counter_tbw(behavior) (tb_counter.vhd) (1)
Hierarchy
Libraries
Compile Order
Counter.vhd
tb_counter.vhd
Untitled 3*
counter.xdc *
C:/software/vivado/vivado_project/Tutorial/Tutorial.srcs/constrs_1/new/counter.xdc

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.

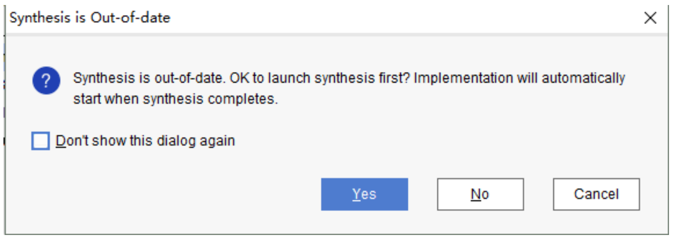
```
set_property PACKAGE_PIN E3 [get_ports CLOCK]
set_property PACKAGE_PIN M17 [get_ports RESET]
set_property PACKAGE_PIN J15 [get_ports DIRECTION]
set_property PACKAGE_PIN H17 [get_ports COUNT_OUT[0]]
set_property PACKAGE_PIN K15 [get_ports COUNT_OUT[1]]
set_property PACKAGE_PIN J13 [get_ports COUNT_OUT[2]]
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.

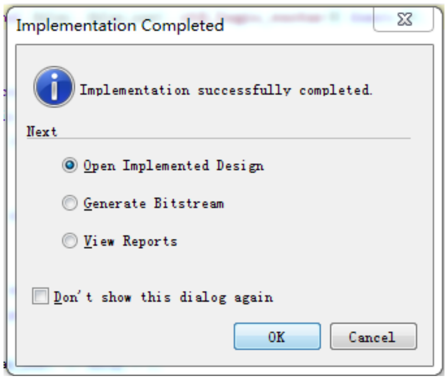


Synthesis is Out-of-date

Synthesis is out-of-date. OK to launch synthesis first? Implementation will automatically start when synthesis completes.

☐ Don't show this dialog again

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



Implementation Completed

Implementation successfully completed.

Next

● Open Implemented Design

○ Generate Bitstream

○ View Reports

☐ Don't show this dialog again

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.

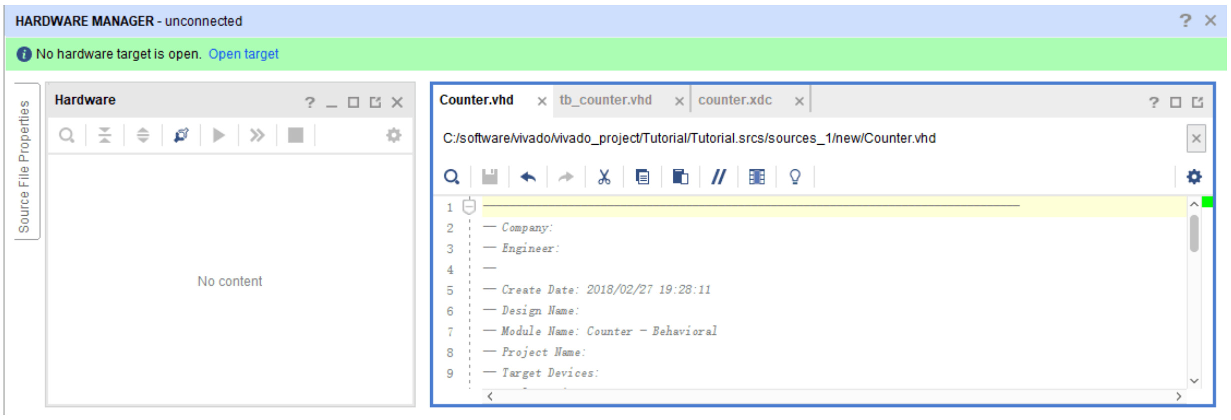
Generating FPGA Configuration File

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

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

<u>Download FPGA configuration file into Nexys4 DDR board via JTAG.</u>

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



HARDWARE MANAGER - unconnected

No hardware target is open. Open target

Source File Properties

Hardware

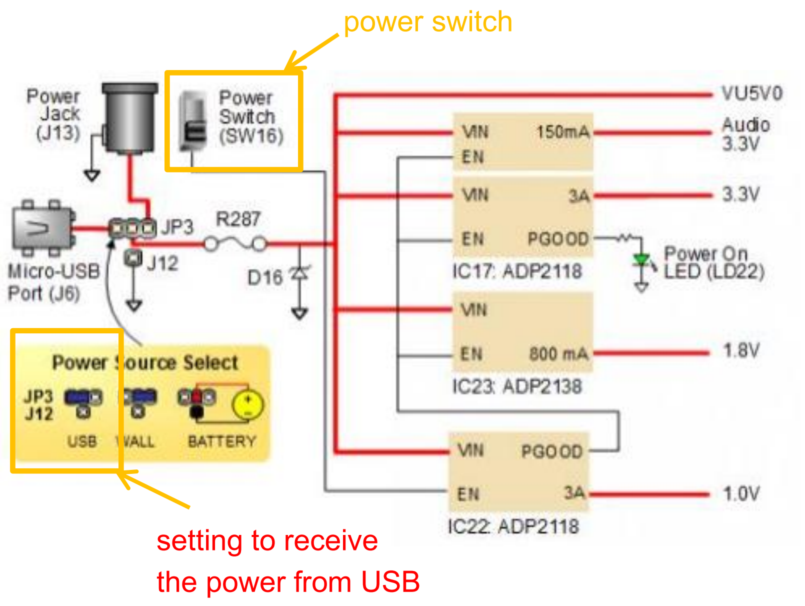
No content

Counter.vhd | tb_counter.vhd | counter.xdc

C:/software/vivado/vivado_project/Tutorial/Tutorial.srcs/sources_1/new/Counter.vhd

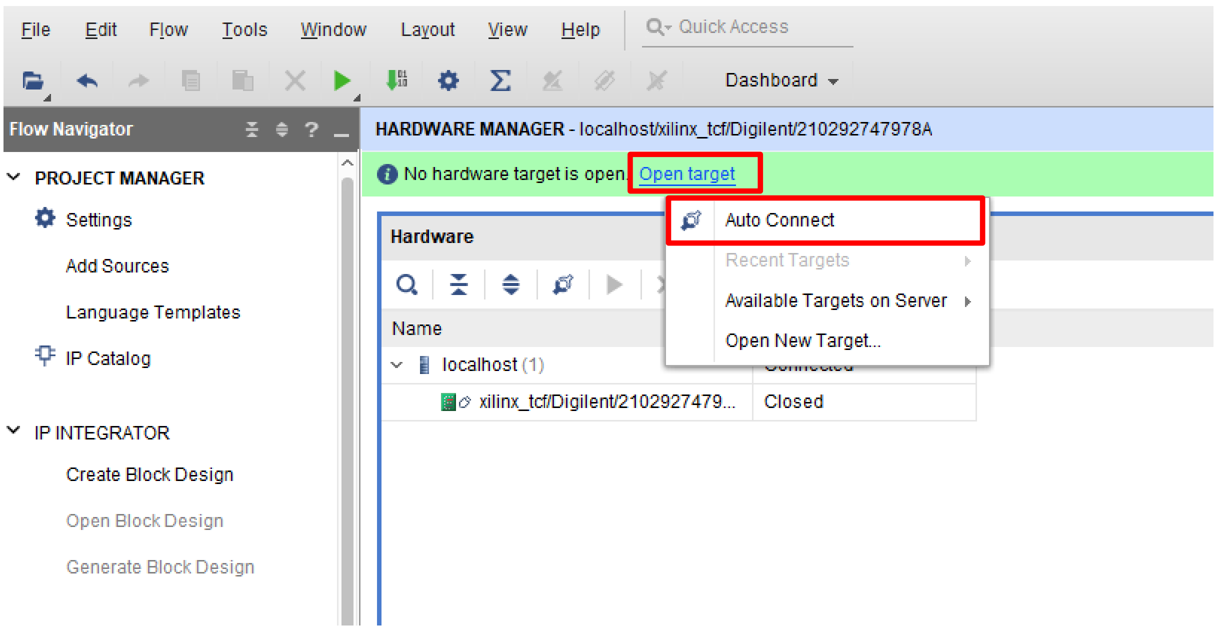
```
----------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 2018/02/27 19:28:11
-- Design Name:
-- Module Name: Counter - Behavioral
-- Project Name:
-- Target Devices:
```

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**.

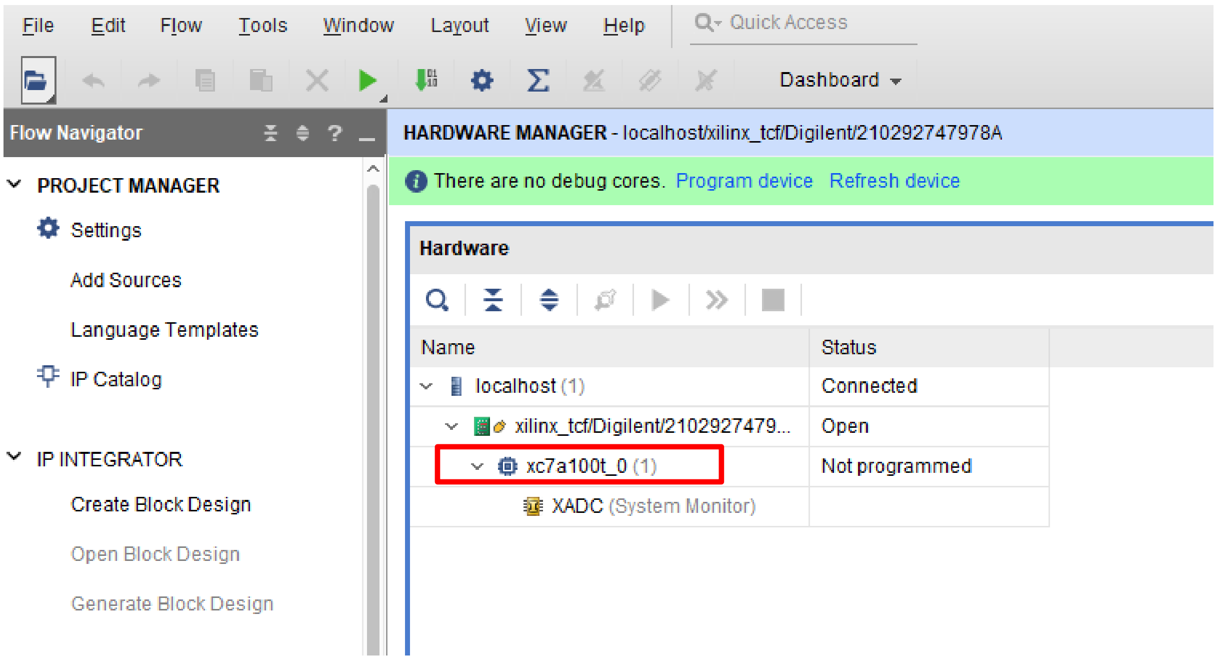


HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210292747978A

No hardware target is open. Open target

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/2102927479...	Closed

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.

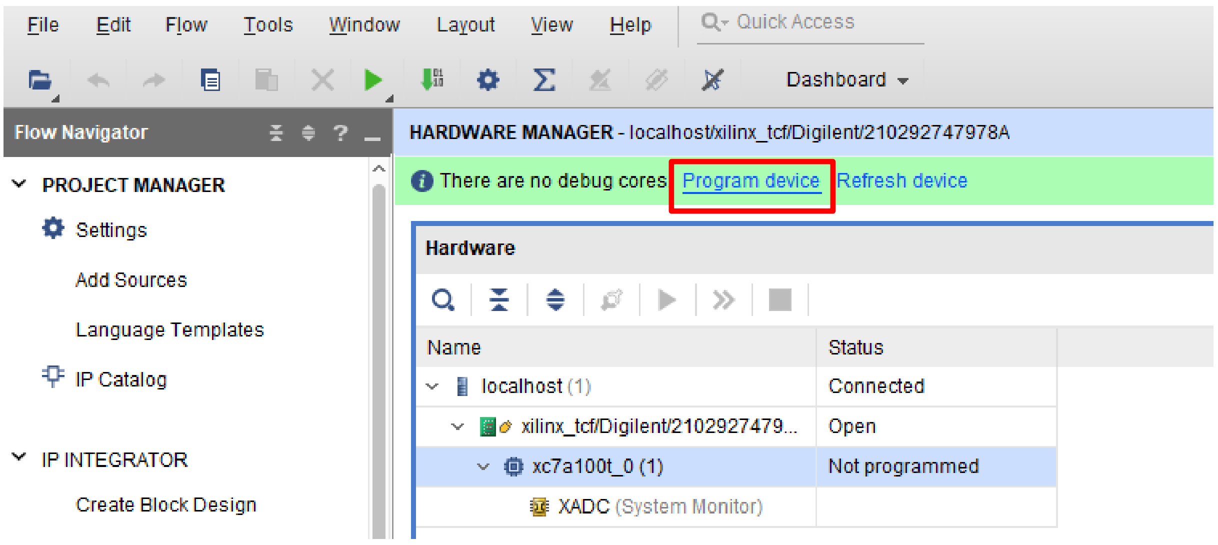


HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210292747978A

There are no debug cores. Program device Refresh device

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/2102927479...	Open
xc7a100t_0 (1)	Not programmed
XADC (System Monitor)	

Click **Program device**.



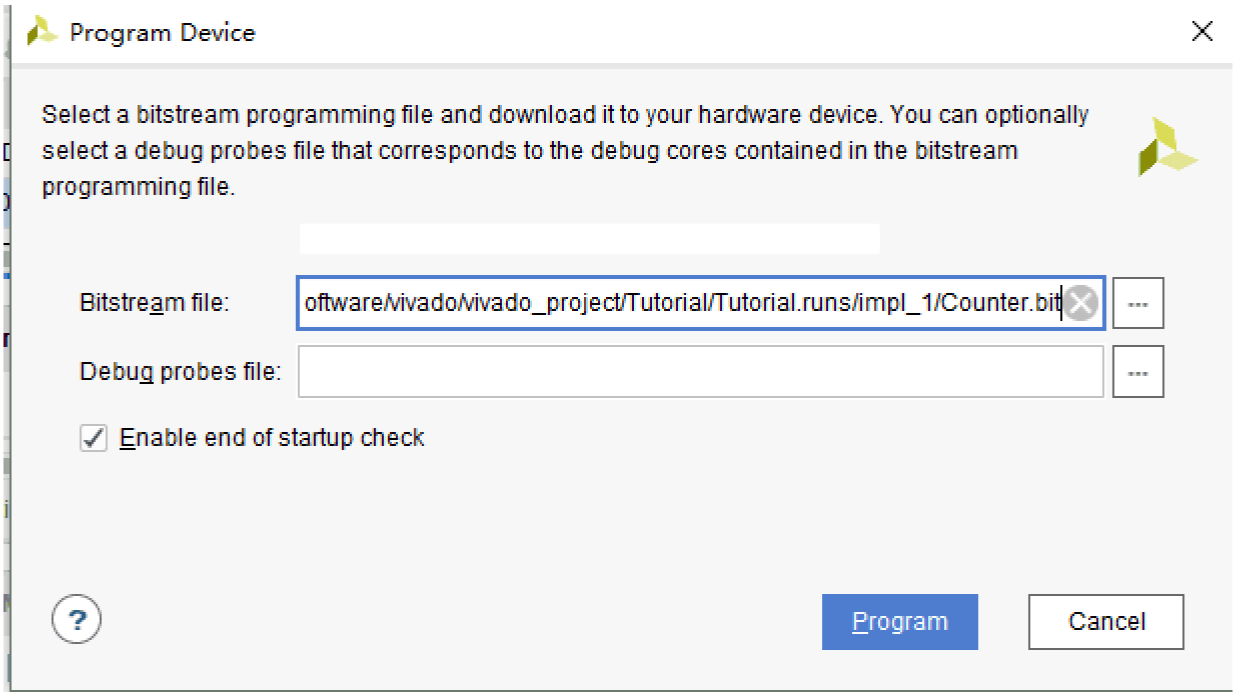
HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210292747978A

There are no debug cores. Program device Refresh device

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/2102927479...	Open
xc7a100t_0 (1)	Not programmed
XADC (System Monitor)	

The following dialog is popped out.   Select the **bitstream file** from the

/project_dir/Tutorial/Tutorial.runs/impl_1/Counter.bit, and click **Program**.



Program Device

Select a bitstream programming file and download it to your hardware device. You can optionally select a debug probes file that corresponds to the debug cores contained in the bitstream programming file.

Bitstream file: oftware/vivado/vivado_project/Tutorial/Tutorial.runs/impl_1/Counter.bit

Debug probes file:

[✓] Enable end of startup check

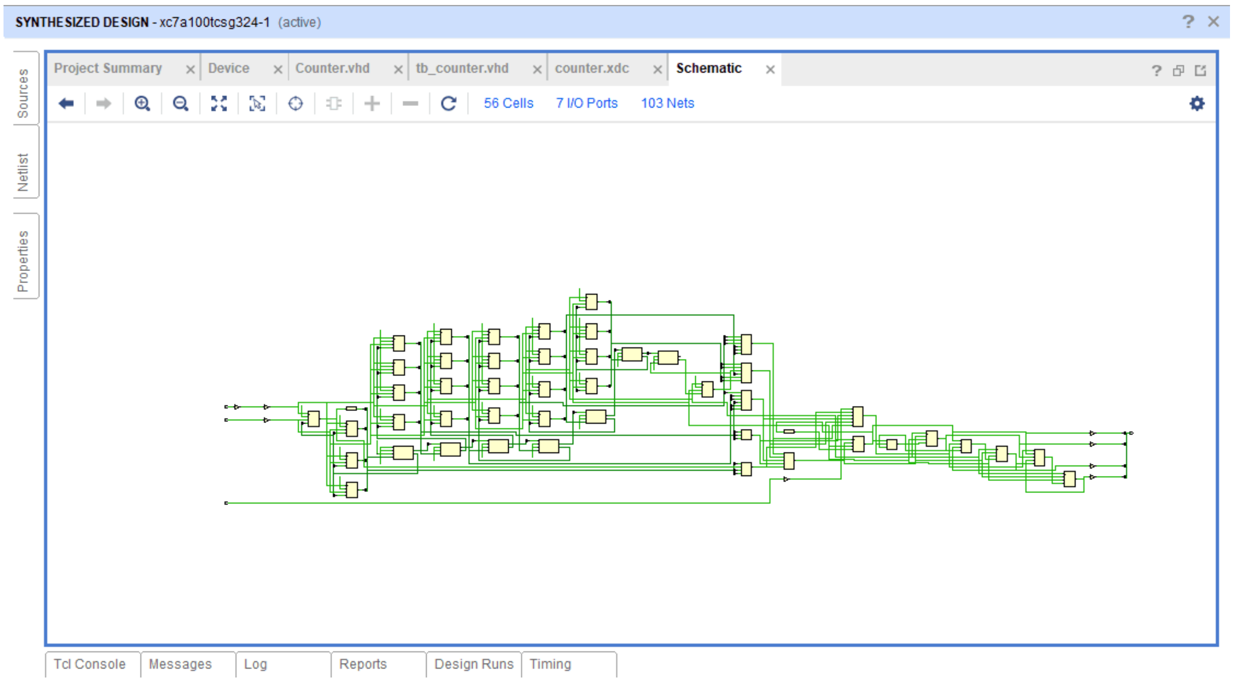
- When programming is completed, On the board, LEDs 0, 1, 2 and 3 are lit, indicating that the counter is running.
- Note that all the LEDs seem to be on all the time, instead of counting on the desired sequence. ***Can you explain why it is so?***

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.

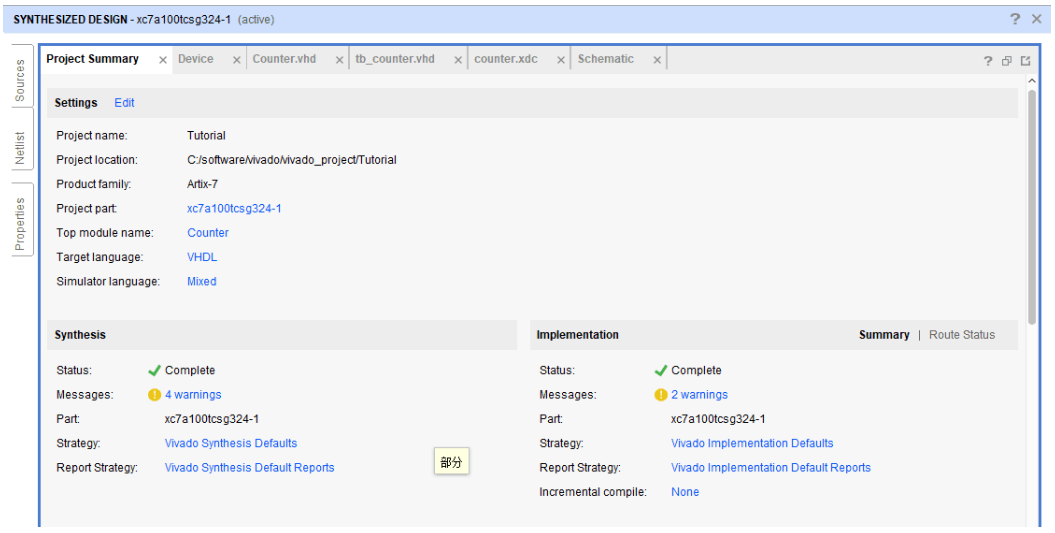
- Modify the VHDL source code to slow down the counting as follow:
  signal delay, delay_next: std_logic_vector(24 downto 0);
- Save the file

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



Settings Edit

Project name:	Tutorial
Project location:	C:/software/vivado/vivado_project/Tutorial
Product family:	Artix-7
Project part:	xc7a100tcsg324-1
Top module name:	Counter
Target language:	VHDL
Simulator language:	Mixed

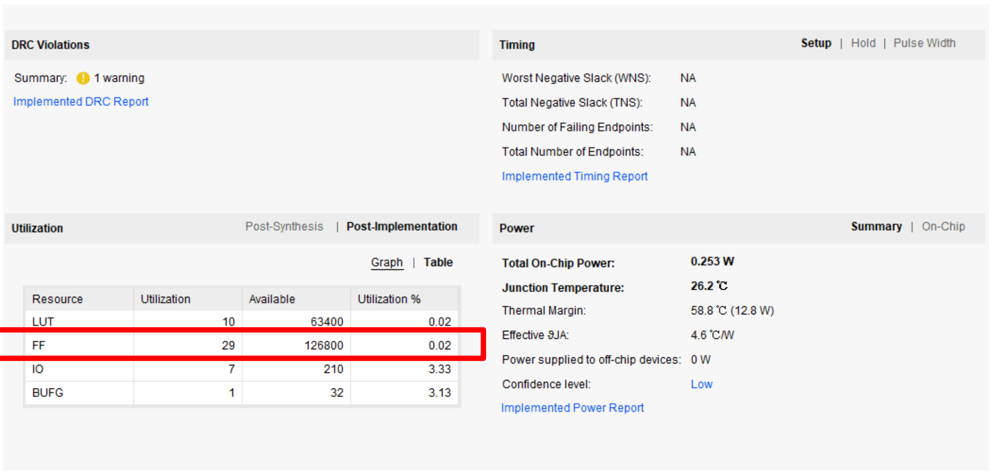
Synthesis

Status:	✓ Complete
Messages:	4 warnings
Part:	xc7a100tcsg324-1
Strategy:	Vivado Synthesis Defaults
Report Strategy:	Vivado Synthesis Default Reports

部分

Implementation Summary | Route Status

Status:	✓ Complete
Messages:	2 warnings
Part:	xc7a100tcsg324-1
Strategy:	Vivado Implementation Defaults
Report Strategy:	Vivado Implementation Default Reports
Incremental compile:	None


DRC Violations

Summary: 1 warning

Implemented DRC Report

Timing — Setup | Hold | Pulse Width

Worst Negative Slack (WNS):	NA
Total Negative Slack (TNS):	NA
Number of Failing Endpoints:	NA
Total Number of Endpoints:	NA

Implemented Timing Report

Utilization — Post-Synthesis | Post-Implementation

Graph | Table

Resource	Utilization	Available	Utilization %
LUT	10	63400	0.02
FF	29	126800	0.02
IO	7	210	3.33
BUFG	1	32	3.13

Power — Summary | On-Chip

Total On-Chip Power:	**0.253 W**
Junction Temperature:	**26.2 ℃**
Thermal Margin:	58.8 ℃ (12.8 W)
Effective ϑJA:	4.6 ℃/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Implemented Power Report

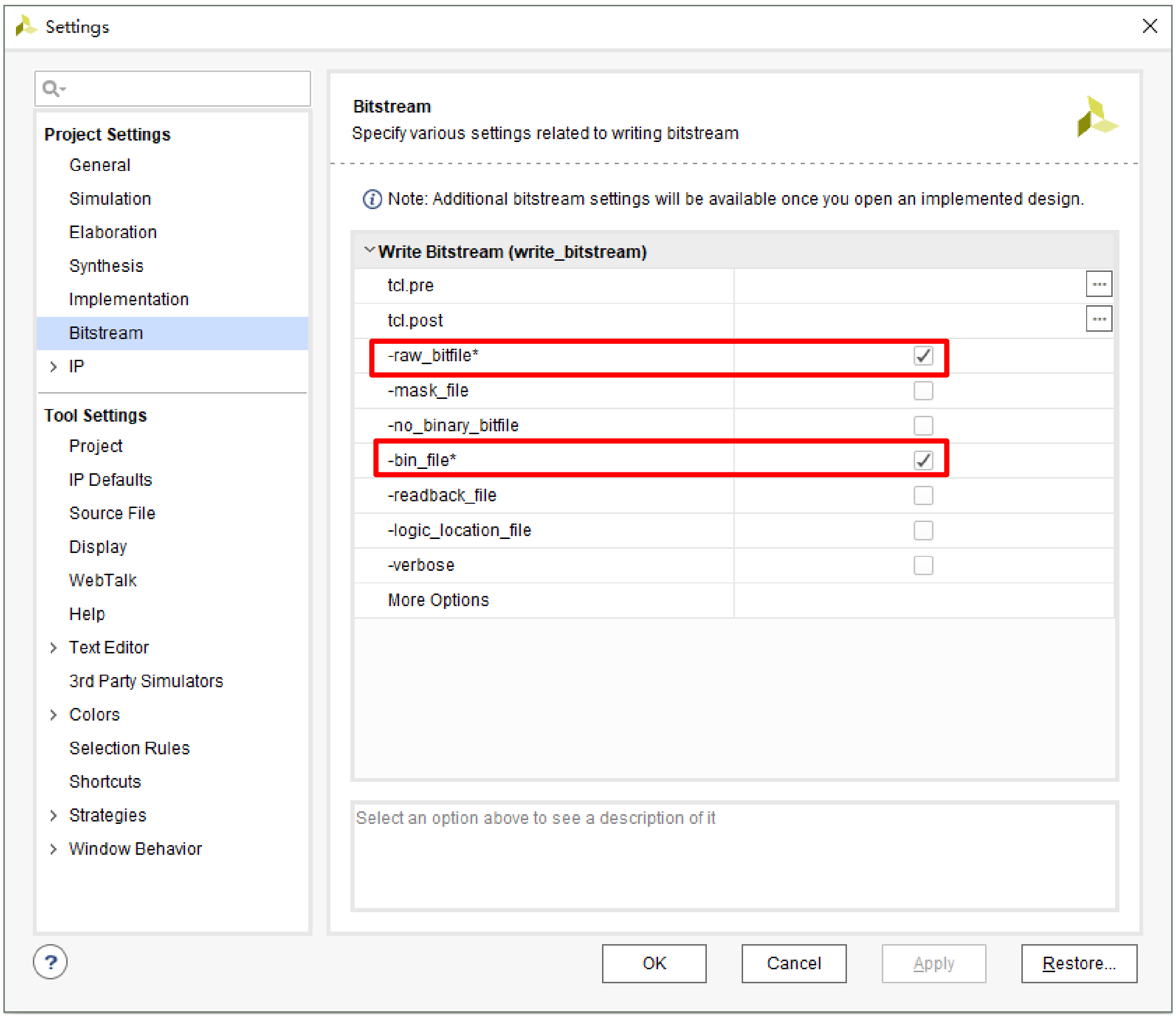
Look at the change in number of the Flip Flops (why is it 29 instead of 5?)

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**.


Settings

Bitstream
Specify various settings related to writing bitstream

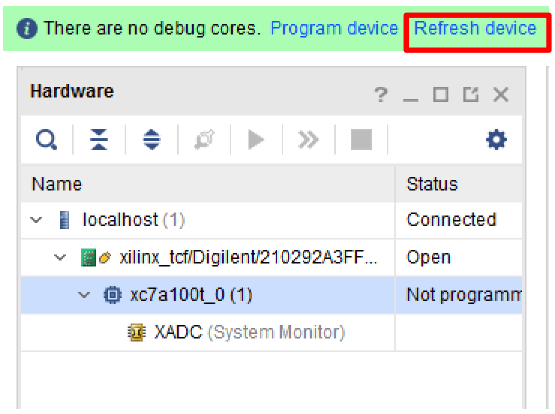
Note: Additional bitstream settings will be available once you open an implemented design.

Write Bitstream (write_bitstream)

tcl.pre	
tcl.post	
-raw_bitfile*	✓
-mask_file	☐
-no_binary_bitfile	☐
-bin_file*	✓
-readback_file	☐
-logic_location_file	☐
-verbose	☐
More Options	

Select an option above to see a description of it

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

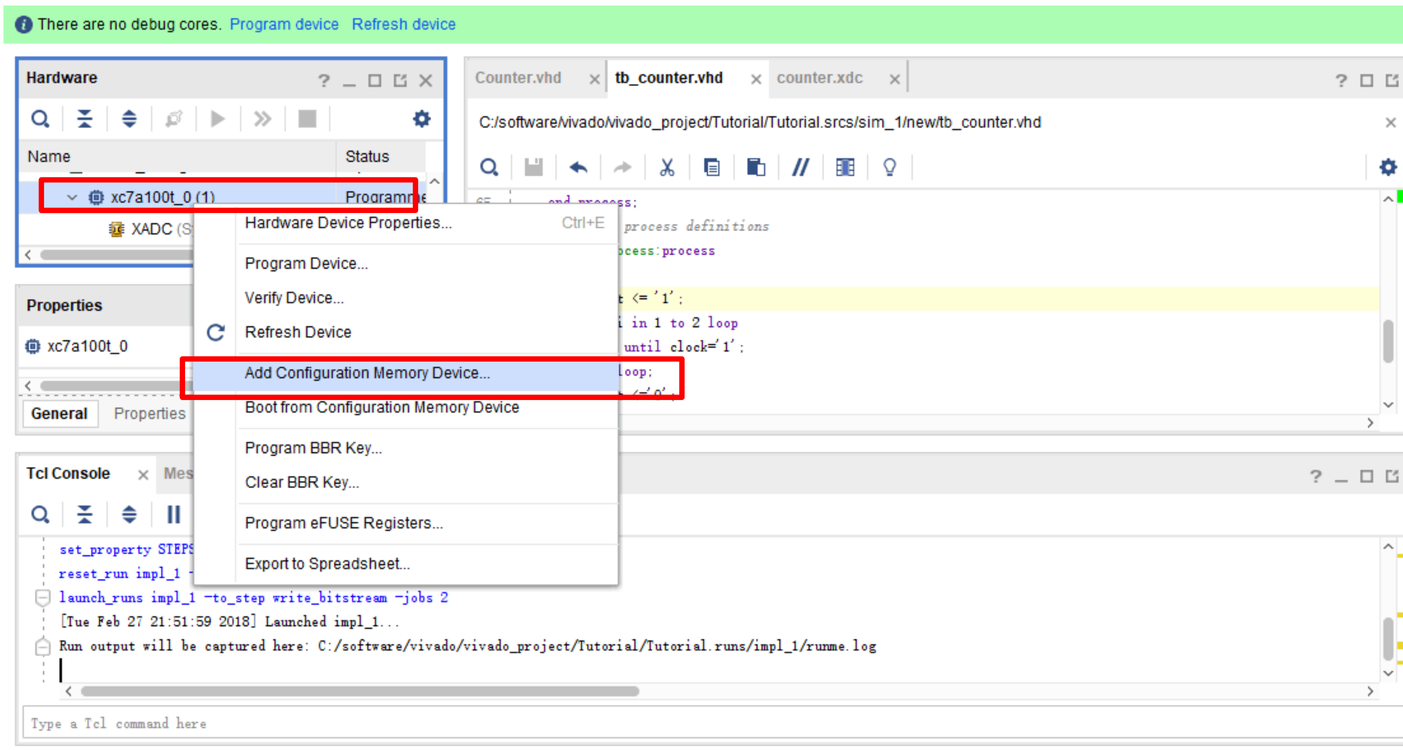

There are no debug cores. Program device Refresh device

Hardware

Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210292A3FF...	Open
xc7a100t_0 (1)	Not programm
XADC (System Monitor)	

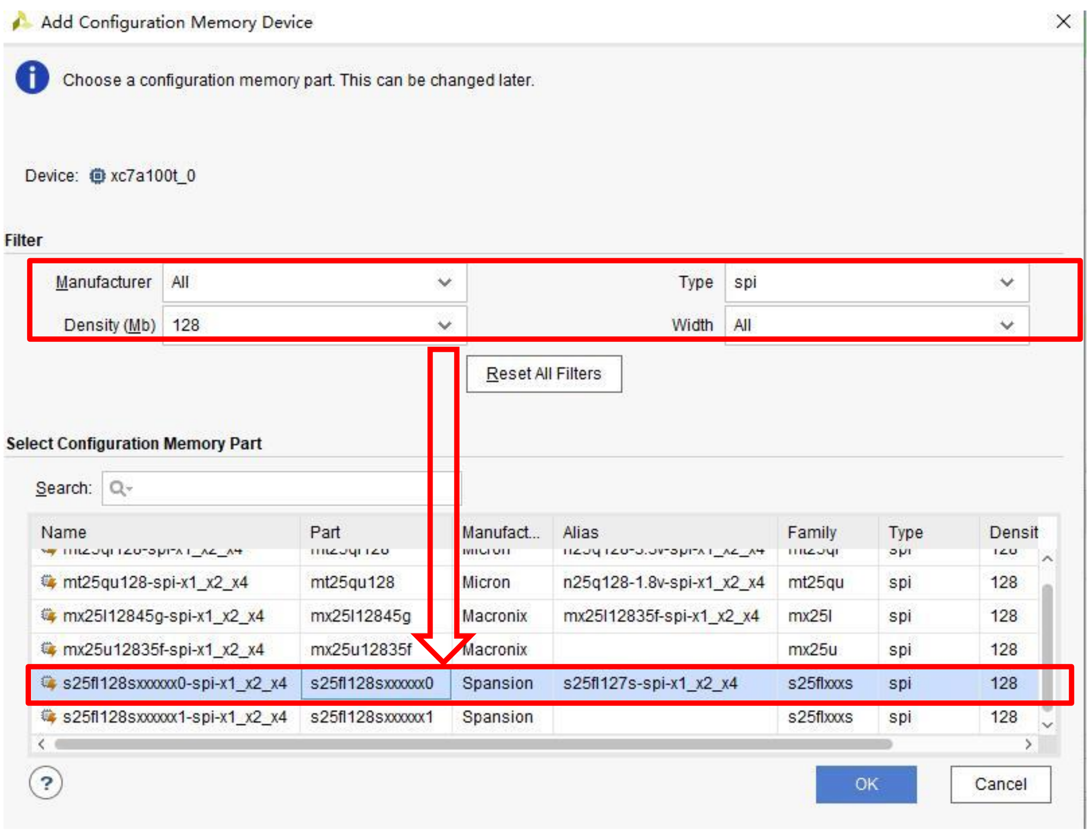
- Generate bitstream again by clicking **Flow Navigator -> Program and Debug -> Generate Bitstream**.
- Open the **Hardware Manage** once the bitstream is generated by clicking **Flow Navigator -> Program and Debug -> Hardware Manager**.

- **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.


There are no debug cores. Program device Refresh device
Hardware
Name
Status
xc7a100t_0 (1)
Programme
XADC (S
Properties
xc7a100t_0
General
Properties
Hardware Device Properties... Ctrl+E
Program Device...
Verify Device...
Refresh Device
Add Configuration Memory Device...
Boot from Configuration Memory Device
Program BBR Key...
Clear BBR Key...
Program eFUSE Registers...
Export to Spreadsheet...
Counter.vhd
tb_counter.vhd
counter.xdc
C:/software/vivado/vivado_project/Tutorial/Tutorial.srcs/sim_1/new/tb_counter.vhd
Tcl Console
Mes
Type a Tcl command here

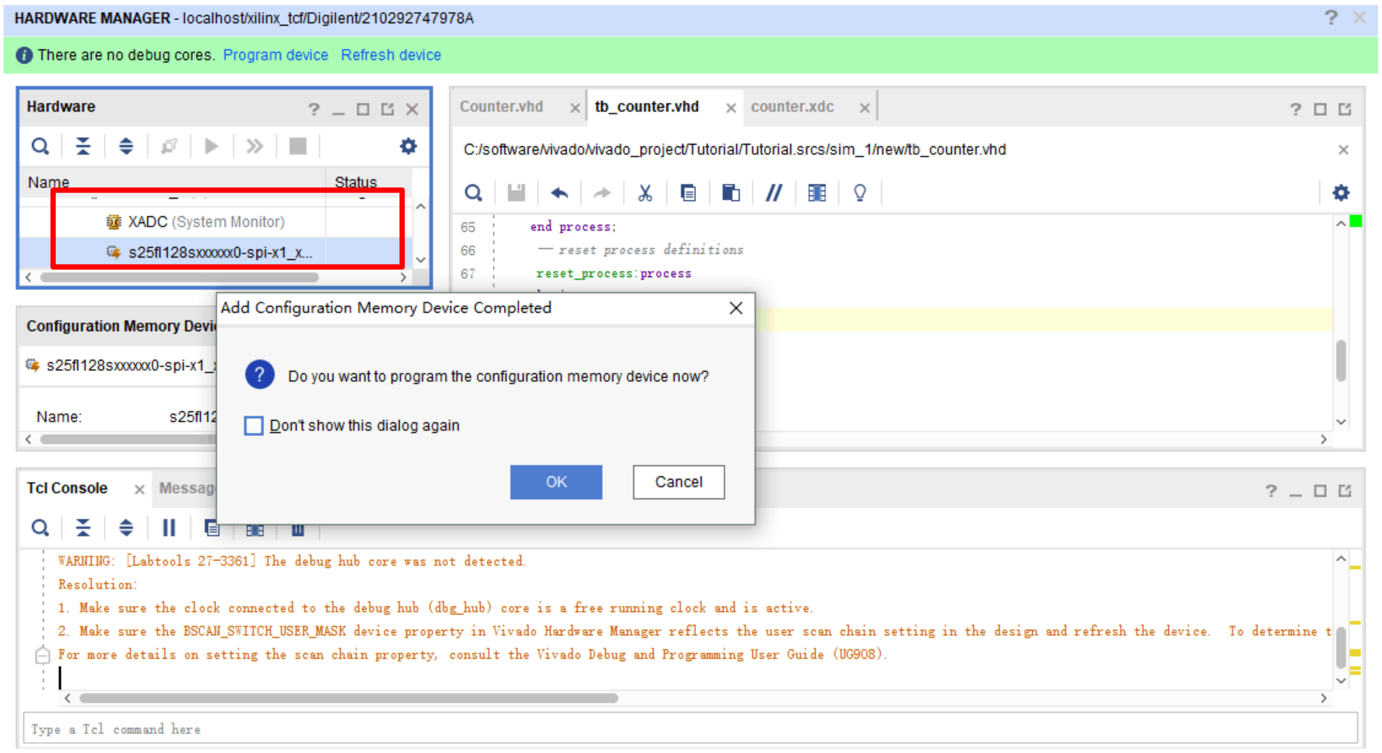
```
set_property STEPS
reset_run impl_1 -
launch_runs impl_1 -to_step write_bitstream -jobs 2
[Tue Feb 27 21:51:59 2018] Launched impl_1...
Run output will be captured here: C:/software/vivado/vivado_project/Tutorial/Tutorial.runs/impl_1/runme.log
```

- Select **Density 128**, **Type spi** as follows, and select **s25fl128xxxxxx0** in the appeared list. Then **OK**.


Add Configuration Memory Device
Choose a configuration memory part. This can be changed later.
Device: xc7a100t_0
Filter
Manufacturer All
Type spi
Density (Mb) 128
Width All
Reset All Filters
Select Configuration Memory Part
Search:
OK
Cancel

Name	Part	Manufact...	Alias	Family	Type	Densit
mt25qu128-spi-x1_x2_x4	mt25qu128	Micron	n25q128-1.8v-spi-x1_x2_x4	mt25qu	spi	128
mx25l12845g-spi-x1_x2_x4	mx25l12845g	Macronix	mx25l12835f-spi-x1_x2_x4	mx25l	spi	128
mx25u12835f-spi-x1_x2_x4	mx25u12835f	Macronix		mx25u	spi	128
s25fl128sxxxxxx0-spi-x1_x2_x4	s25fl128sxxxxxx0	Spansion	s25fl127s-spi-x1_x2_x4	s25flxxxs	spi	128
s25fl128sxxxxxx1-spi-x1_x2_x4	s25fl128sxxxxxx1	Spansion		s25flxxxs	spi	128

- You will see that **s25fl128xxxxxx0** has been attached to **xc7a100t_0**. In the popped up dialog, click **OK** to program the configuration memory device.



HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210292747978A

There are no debug cores. Program device Refresh device

Hardware

Name	Status
XADC (System Monitor)	
s25fl128sxxxxxx0-spi-x1_x...	

Configuration Memory Device

s25fl128sxxxxxx0-spi-x1_

Name: s25fl12

Counter.vhd | tb_counter.vhd | counter.xdc

C:/software/vivado/vivado_project/Tutorial/Tutorial.srcs/sim_1/new/tb_counter.vhd

```
    end process;
    -- reset process definitions
    reset_process:process
```

Add Configuration Memory Device Completed

Do you want to program the configuration memory device now?

☐ Don't show this dialog again

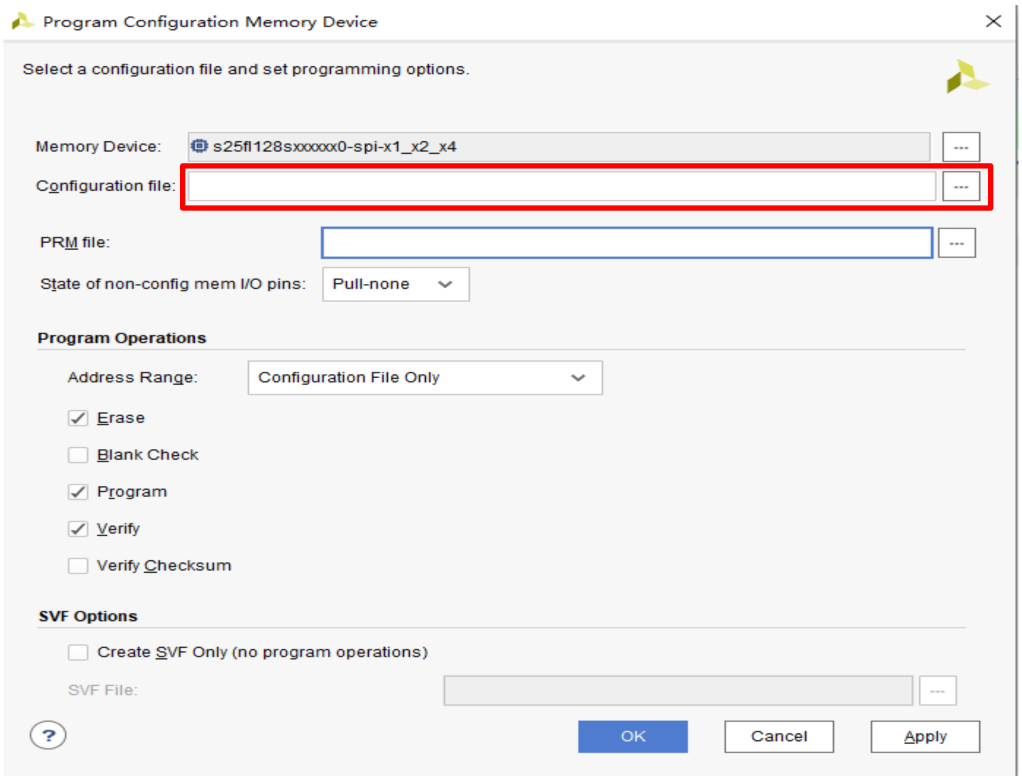
OK Cancel

Tcl Console | Messag

```
WARNING: [Labtools 27-3361] The debug hub core was not detected.
Resolution:
1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active.
2. Make sure the BSCAN_SWITCH_USER_MASK device property in Vivado Hardware Manager reflects the user scan chain setting in the design and refresh the device.  To determine t
For more details on setting the scan chain property, consult the Vivado Debug and Programming User Guide (UG908).
```

Type a Tcl command here

- 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**.



Program Configuration Memory Device

Select a configuration file and set programming options.

Memory Device: s25fl128sxxxxxx0-spi-x1_x2_x4

Configuration file:

PRM file:

State of non-config mem I/O pins: Pull-none

Program Operations

Address Range: Configuration File Only

☑ Erase

☐ Blank Check

☑ Program

☑ Verify

☐ Verify Checksum

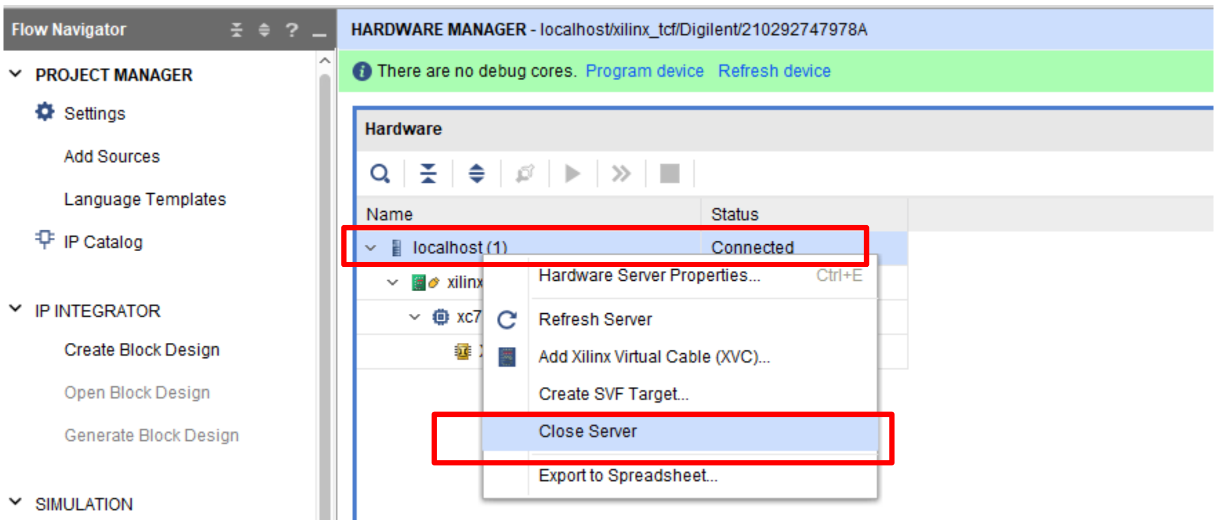
SVF Options

☐ Create SVF Only (no program operations)

SVF File:

OK Cancel Apply

- 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.

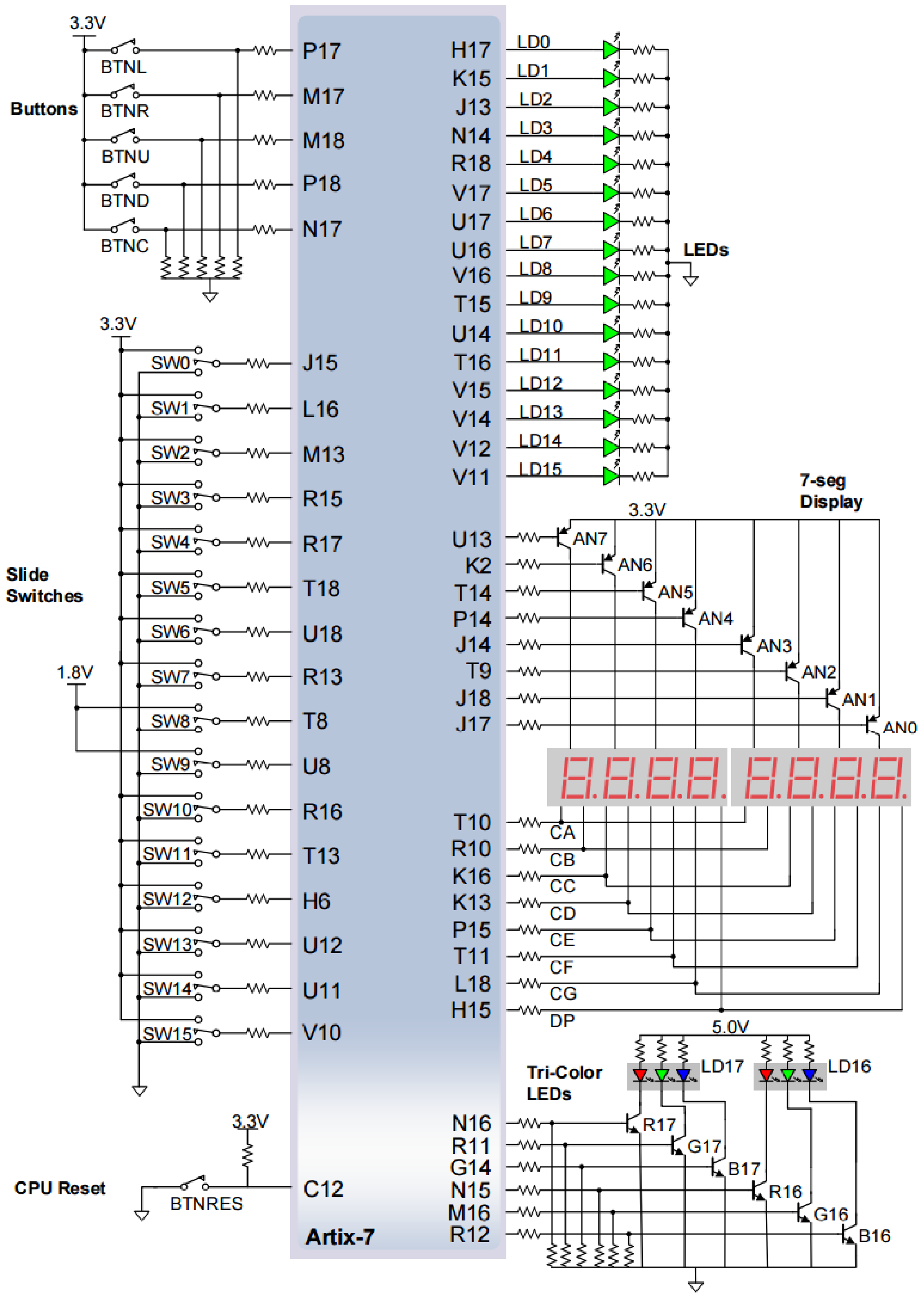
Appendix A:

```
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.