

**City University of Hong Kong**

**Department of Electrical Engineering**

**EE 2000 – Lab Manual 1**

**ZedBoard Basic On-board Experiments**

**Course Leader: WONG Steve Hang**

## **Log Book**

Students are required to prepare their own log book (A4-size). Students could use it to record all findings, data, analysis, preparation materials, discussions, and experimental results.

## **Check point**

There are some check points in the lab exercises. Students are required to demonstrate their own results to the lab tutors and the lab demonstrators.

## **The Lab Report**

Students are required to submit an individual lab report in Week 14 (to their lab tutors). All reports should be typed, and A4-size bound. Contents of the lab report should include:

### **1. Objectives:**

State the purposes of each exercise and the achievements in this lab. Please do not copy the objectives verbatim from the lab handout.

### **2. Design**

Describe your design with the circuit diagram. Explain the functionalities of each circuit. Please make sure all figures and tables are consistent, legible and well labeled.

### **3. Results and Discussions:**

Report and describe what you have observed in the lab. Present your results in a clear and concise manner.

### **4. Conclusions:**

In this section, you should attempt to answer the questions:

What did you learn from this lab?

What did you do wrong (or what went wrong)?

How could you have improved upon your design procedures?

Were your results as expected or did you find something unusual.

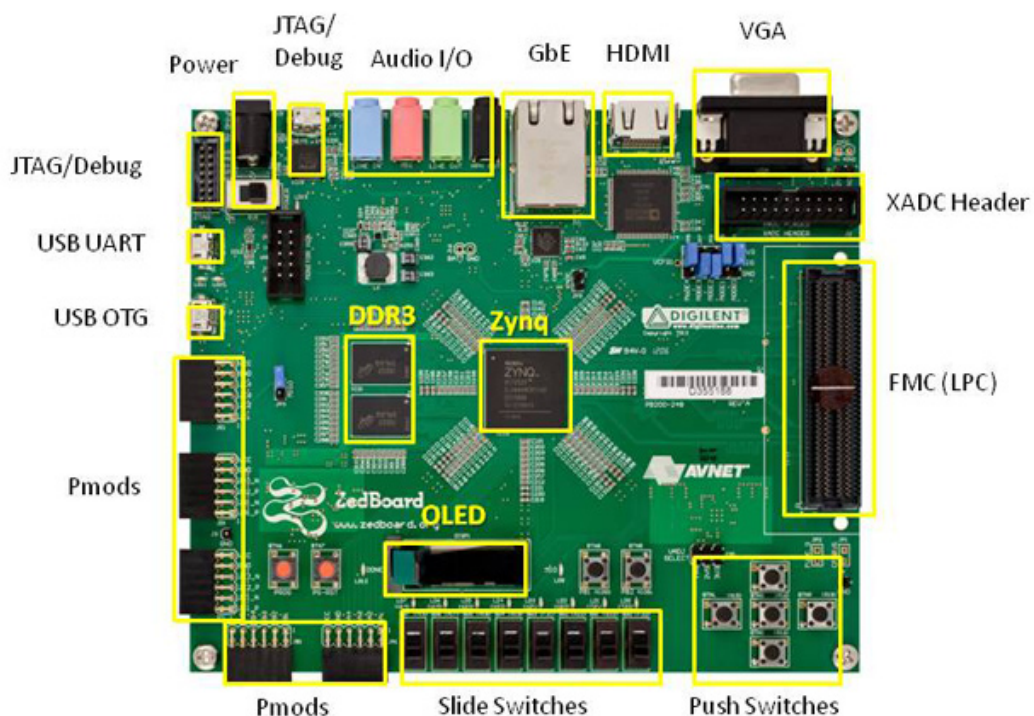
Try not to include information that you have included in previous sections. Present the significance of your results conceptually, if applicable.

## Objectives:

- Learn how to use the ZedBoard;
- Learn how to use the Xilinx Vivado Software;
- Learn how to create a simple digital circuit design using VHDL;
- Learn how to program the Zynq FPGA Chip on the ZedBoard;
- Learn how to access the basic I/O components on the ZedBoard.

**There are seven checkpoints in Page 4, 14, 19, 22 and 24. For each checkpoint, please take note/photo/screenshot.**

Before we start doing the experiments, we will first examine the ZedBoard as shown in the figure below. The reconfigurable FPGA chip on the ZedBoard is a Zynq™-7000 AP System-on-Chip (SoC) XC7Z020-CLG484-. The figure below shows the main components on ZedBoard.

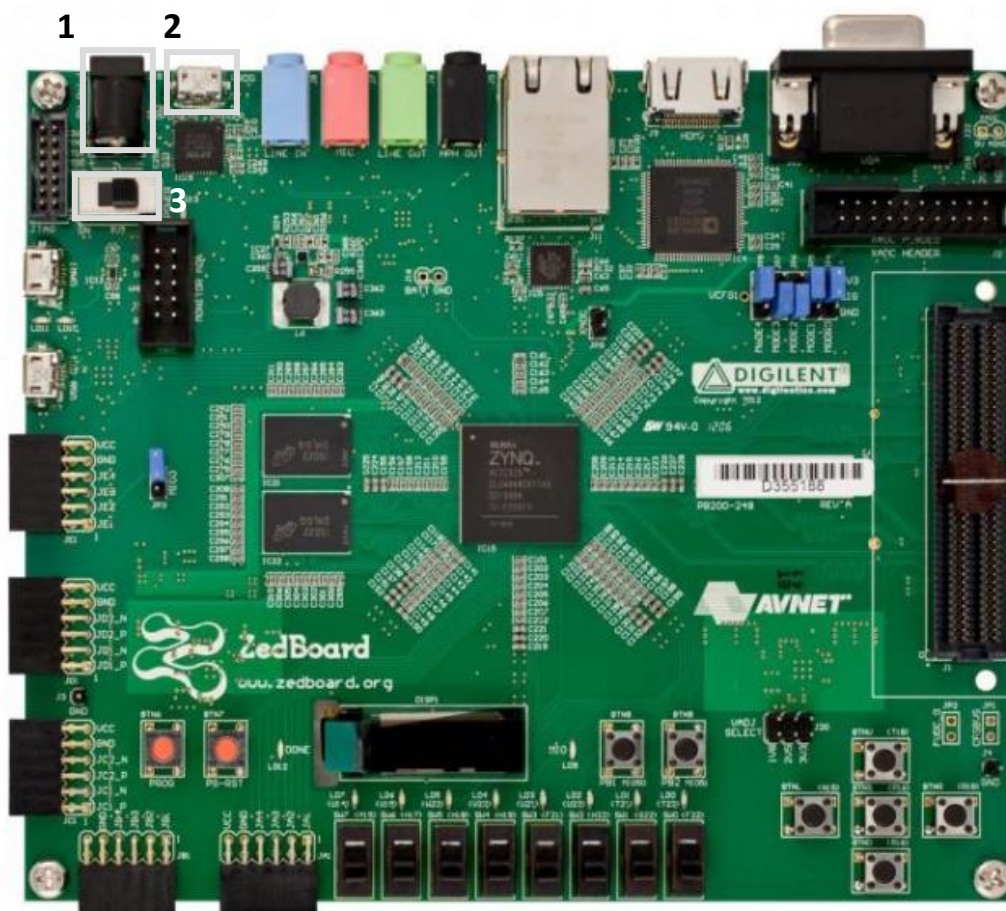


\* SD card cage and QSPI Flash reside on backside of board

**CHECKPOINT 1: Give the full name and functionality of the following terms: HDMI, XADC, UART, JTAG, DDR3, GbE, OLED? Please search the relevant materials on the Internet.**

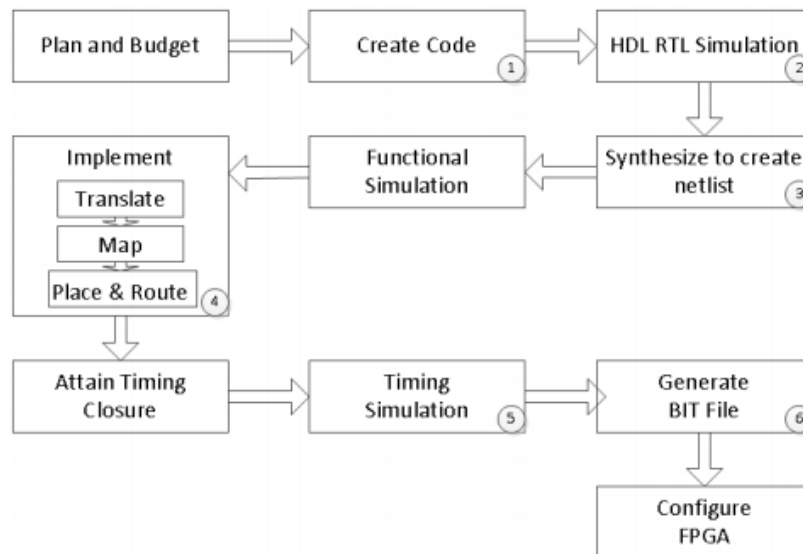
Before we start to program the FPGA chip on the ZedBoard, we need to setup the hardware connections as follows:

1. Connect a 12 V power supply to barrel jack (J20) using the power cable.
2. Connect the USB port, which is labeled as PROG (J17) to a PC using the MicroUSB cable. (Please ask for the help in the lab if you cannot find this cable.)
3. Turn the power switch (SW8) on the board to the ON position, which is located near the barrel jack for power supply.



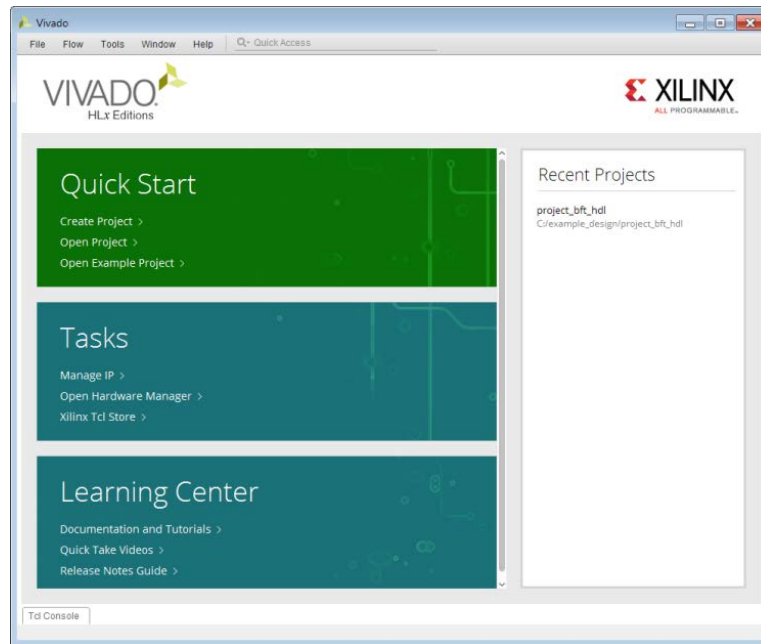
In the following part, you will learn the design flow using Xilinx Vivado software to create a simple digital circuit using Hardware Description Language (HDL). A typical design flow consists of 1) creating model(s), 2) creating user constraint file(s), 3) creating a new Vivado project, 4) importing the created models, 5) assigning created

constraint file(s), 6) optionally running behavioral simulation, 7) synthesizing the design, 8) implementing the design, 9) generating a bitstream (.bit) file for programming the FPGA chip, and 10) finally verifying the functionality in the hardware by downloading the generated .bit file. The whole design flow is shown as below. Since achieving the timing closure and performing timing simulation require more advanced knowledge of digital hardware design, we will cover these two parts in the lecture and later tutorials.

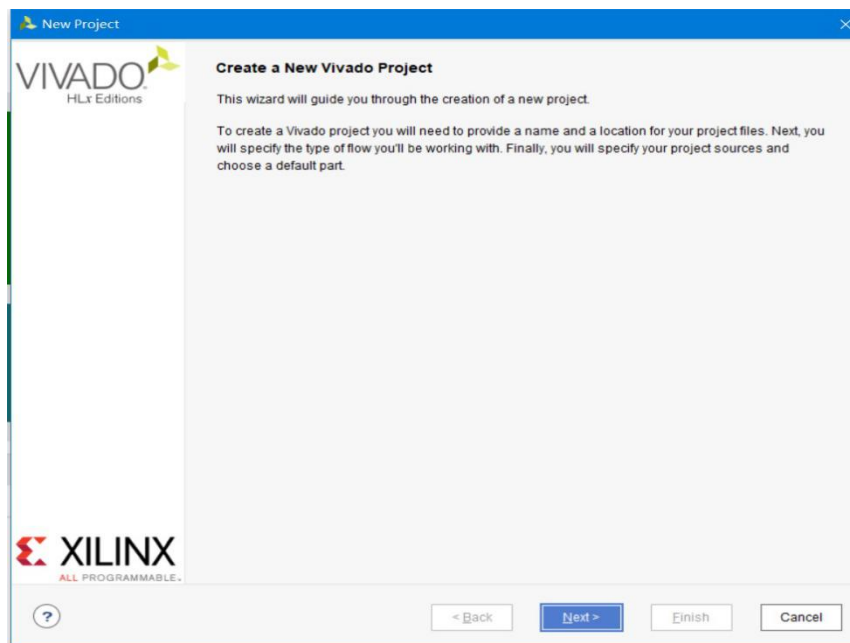


1. Login to one of the lab machines.
2. Launch the Vivado software and create a new project targeting the ZedBoard and using the VHDL.

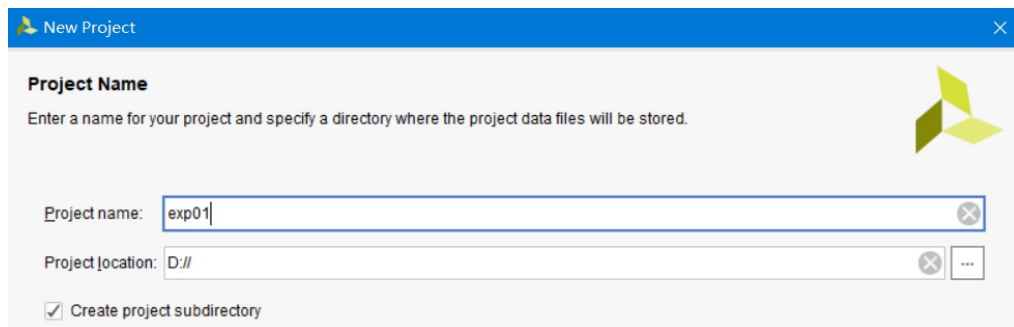
Once you launch the **Vivado** software, the getting started page will appear as follows.



- i. Choose **Create Project** to open New Project Wizard. Click **Next**.



- ii. Enter **Project name** and **Project location**. Click **Next**. Select **RTL Project**, Click **Do not specify sources at this time** and click **Next**.



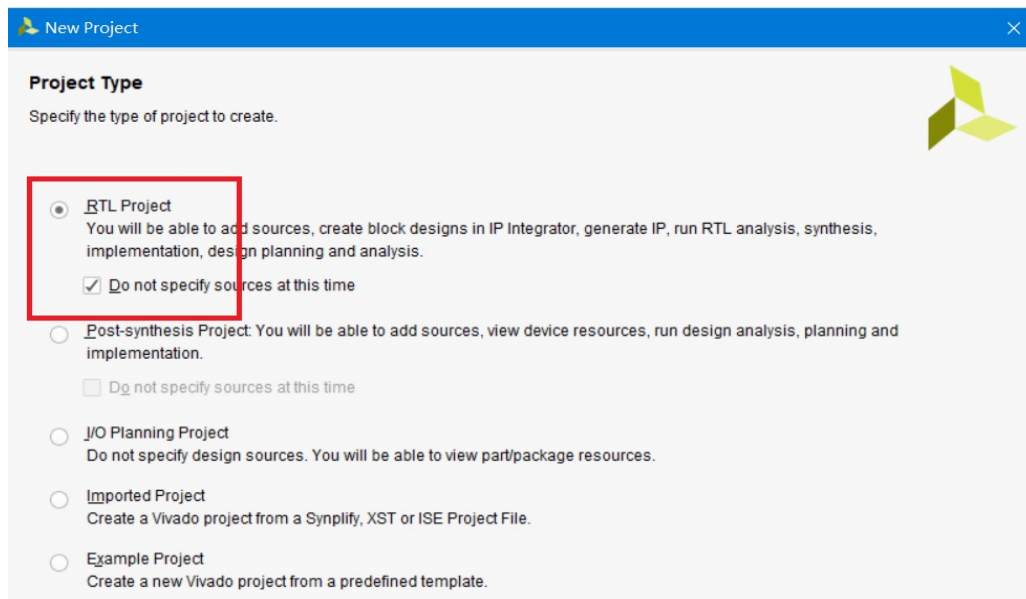
**New Project**

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

Project name:

Project location:

☒ Create project subdirectory



**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

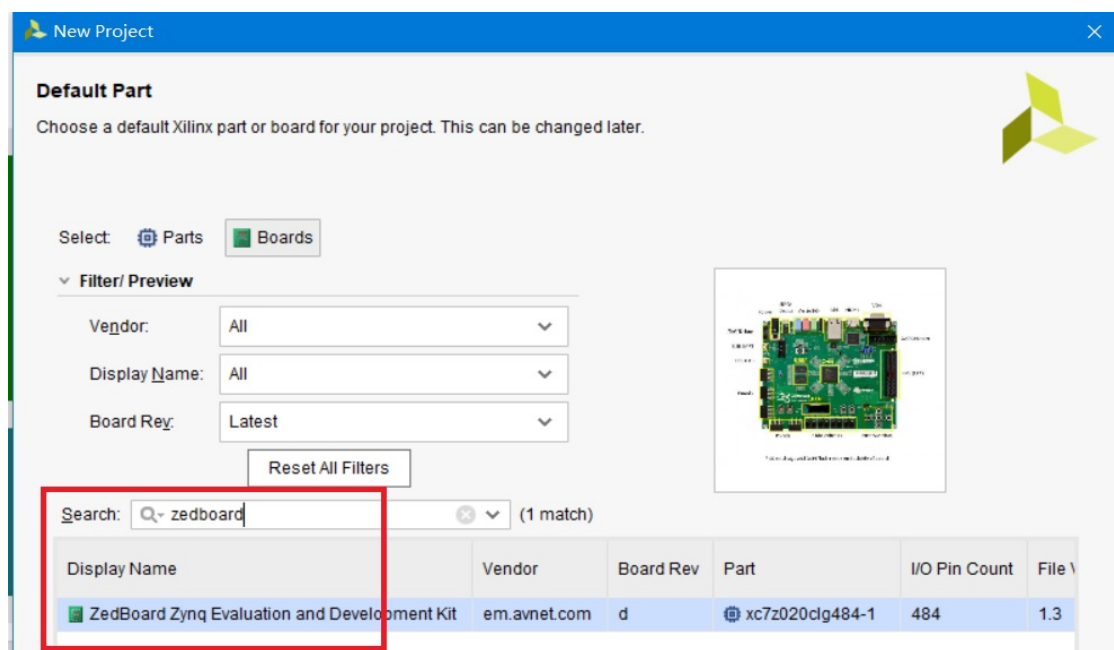
☐ **Post-synthesis Project**  
You will be able to add sources, view device resources, run design analysis, planning and implementation.  
☐ Do not specify sources at this time

☐ **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.

- iii. Select Board as **ZedBoard** and click **Next**. Alternatively, you can select the chip model. Then click **Finish** to create the project.



**New Project**

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

Select: ☒ Parts ☒ Boards



**Filter/ Preview**

Vendor:

Display Name:

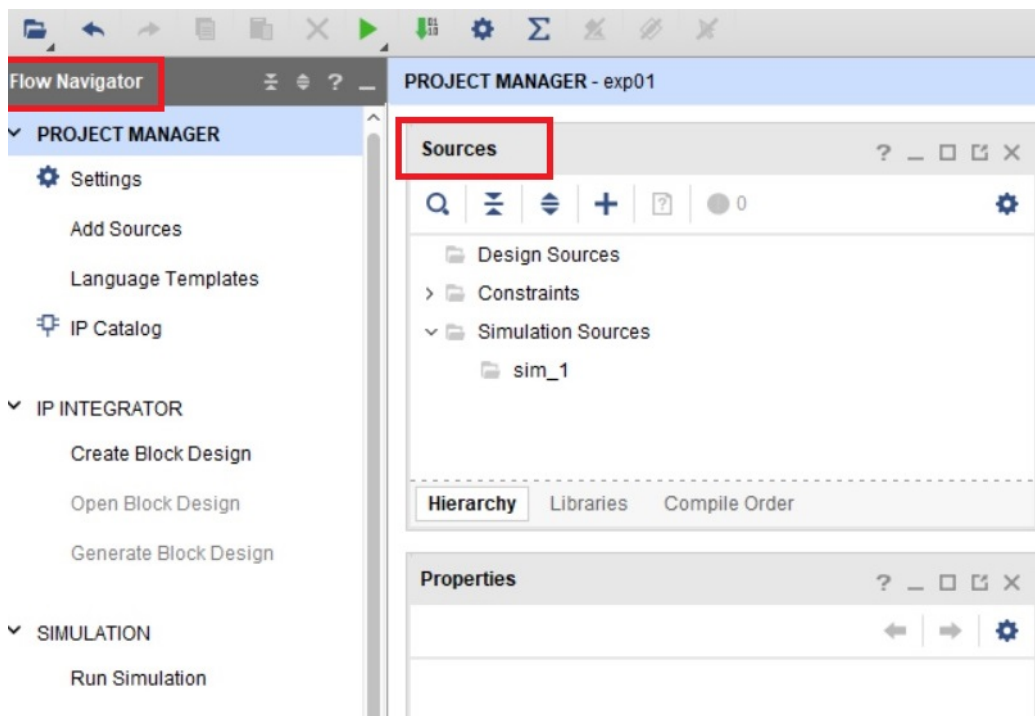
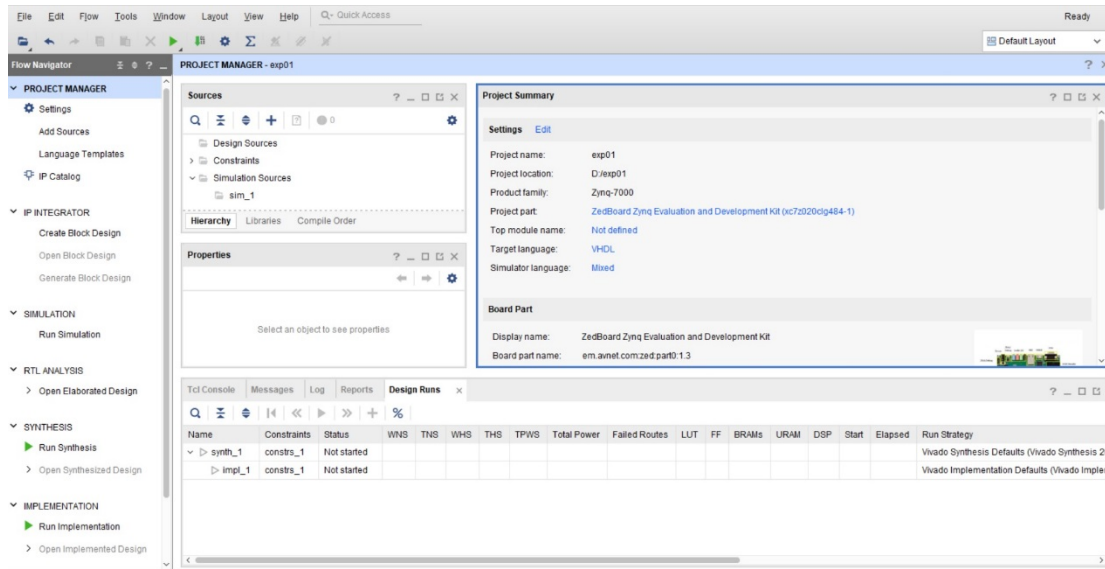
Board Rev:

Search:  (1 match)

| Display Name   | Vendor       | Board Rev | Part  | I/O Pin Count | File Version |
|--|--------------|-----------|---|---------------|--------------|
|  ZedBoard Zynq Evaluation and Development Kit | em.avnet.com | d         |  xc7z020clg484-1 | 484           | 1.3          |



- iv. You will see a screen look like this. In the left side, you can find **Flow Navigator** with a few click buttons. Besides is a **Sources** window showing your project hierarchy.



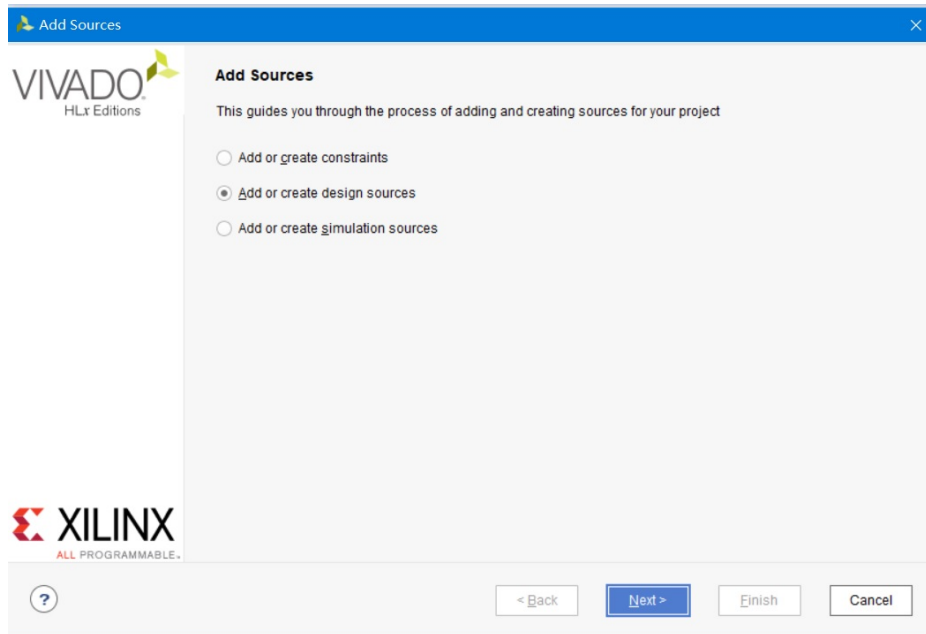
### 3. Create and edit two VHDL Files:

- exp01.vhd (Design model source HDL file)
- exp01\_tb.vhd (Design Testbench HDL file)

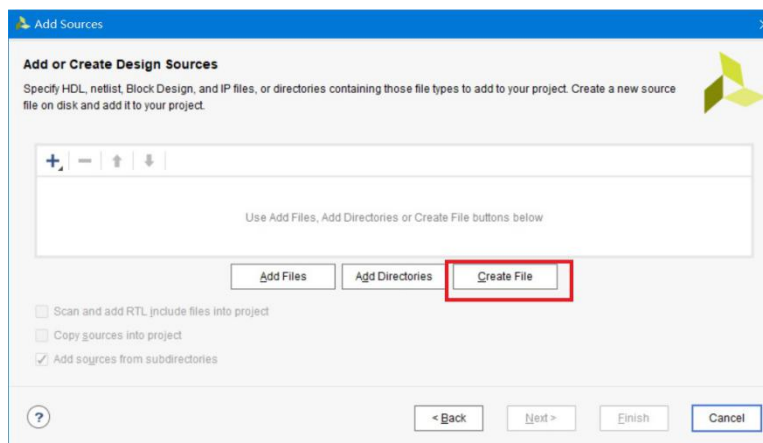
- i. In **Sources** window, right click **Design Sources** and click **Add Sources**. Alternatively, you can click **Add Sources** from **Flow Navigator**. You will open

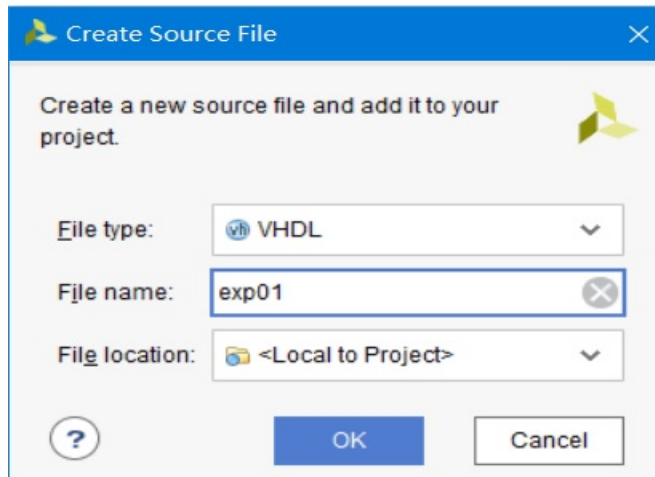


a new window.

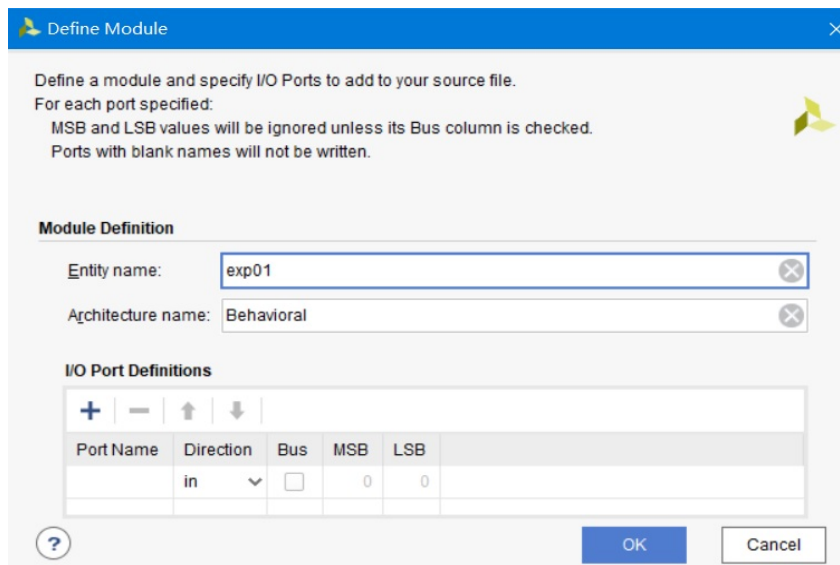


- ii. Select **Add or create design sources** and click **Next**. Select **Create File**. Make sure the **File type** is VHDL and enter the **File name**: exp01. Click **OK**, and then click **Finish**.

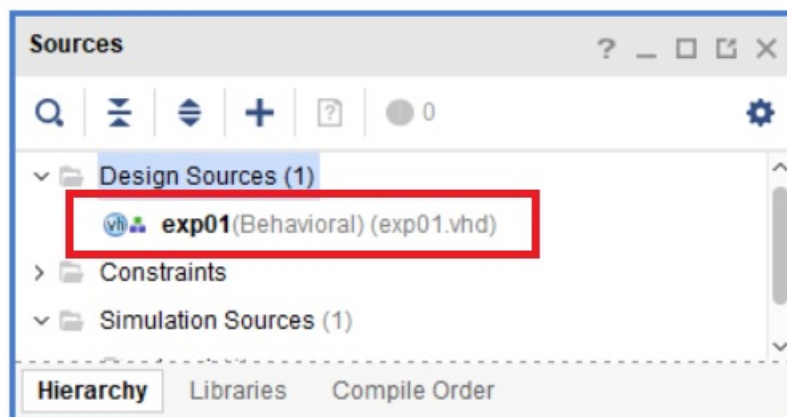




In the new window **Define Module**, click **OK** directly, then click **Yes**.



iii. Find your newly created file.



Edit exp01.vhd, and then save it. You are required to add your own code in the file.

```
entity exp01 is

    Port (
        sw: IN BIT;

        led: OUT BIT
    );

end exp01;

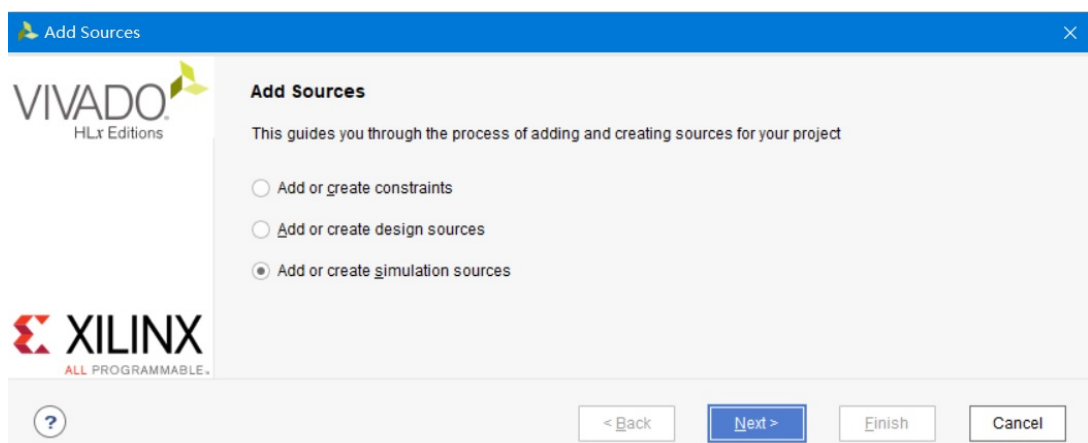

architecture Behavioral of exp01 is

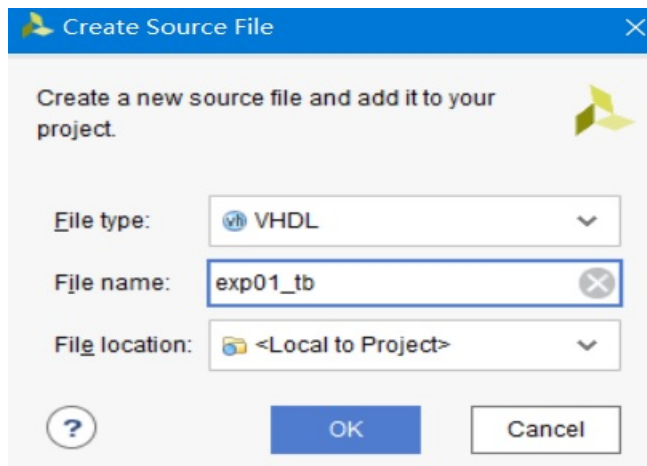
begin

    -- Put your own code here

end Behavioral;
```

- iv. Create a simulation file in the testbench. Right click on **Simulation Sources** in the source window and select **Add Sources**. Make sure you select **Add or create simulation sources** and click **Next**. Select **Create File**, choose **File Type** as VHDL and enter **File name**: exp01\_tb. The operation is similar to creating exp01.vhd.





- v. Edit exp01\_tb.v as follows and save it.

```
entity exp01_tb is
end exp01_tb;

architecture Behavioral of exp01_tb is
-- component declaration
component exp01 is
    Port (
        sw: IN BIT;
        led: OUT BIT
    );
end component;
-- signal declaration
signal sw: BIT;
signal led: BIT;
begin
exp01_inst: exp01
    port map (
```

```

        sw => sw,

        led => led

    );

simgen: process
begin

    sw <= '0';

    wait for 50ns;

    sw <= '1';

    wait for 100ns;

    sw <= '0';

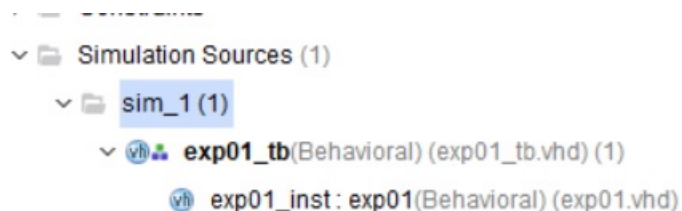
    wait for 50ns;

end process;

end Behavioral;

```

After saving the file, you will find that the hierarchy has been changed since exp01\_tb instantiates exp01.

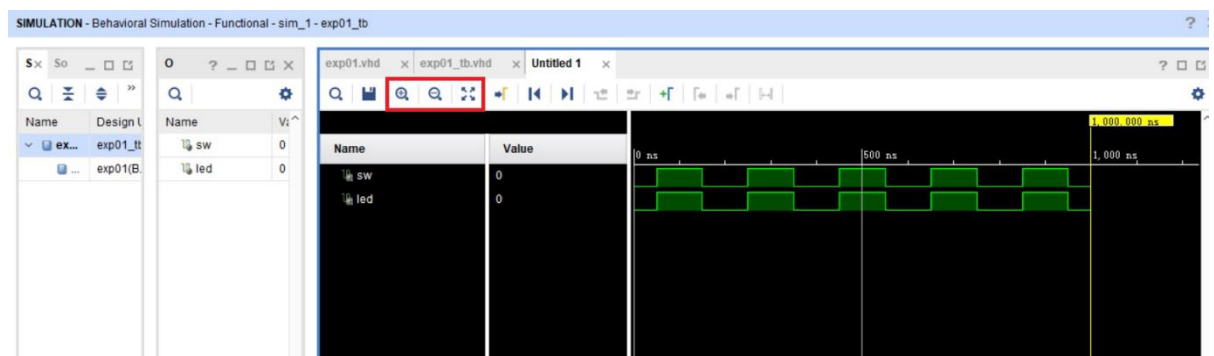


#### 4. Run simulation.

- i. Find and click **Run Simulation** in Flow Navigator, Select **Run Behavioral Simulation**. The default simulator is XSim and you can see the simulation waveform in Untitled 1. The example design is simple and you will learn more simulation skills later.



You can use the three buttons in the red box to scale your waveform.



**CHECKPOINT 2: How do you generate the waveform as shown in the following Figure 1, by modifying the exp01\_tb.vhd file? (Hint: please note the duration of 1 and duration of 0 as shown in the figure have changed.)**

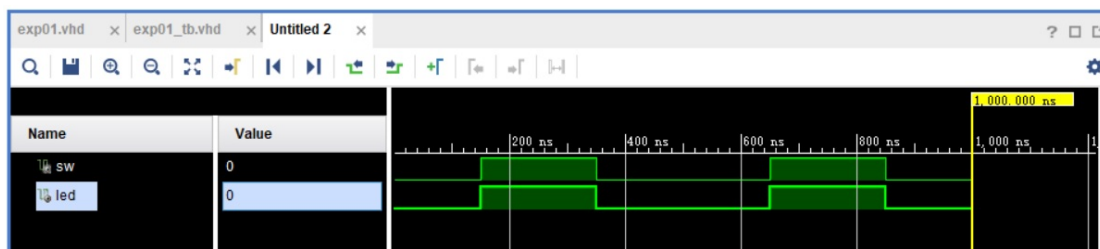


Figure 1. Checkpoint 2 waveform Example.

## 5. Synthesize, implement the design and generate bitstream file.

**At first**, the pins of FPGA chip connected with the above peripheral are shown as below. This information is included in Section 2.7 of “ZedBoard Hardware User’s Guide” on the website <http://zedboard.org/support/documentation/1521> .

**Table 12 - Push Button Connections**

| Signal Name | Subsection | Zynq pin     |
|-------------|------------|--------------|
| BTNU        | PL         | T18          |
| BTNR        | PL         | R18          |
| BTND        | PL         | R16          |
| BTNC        | PL         | P16          |
| BTNL        | PL         | N15          |
| PB1         | PS         | D13 (MIO 50) |
| PB2         | PS         | C10 (MIO 51) |

**Table 13 - DIP Switch Connections**

| Signal Name | Zynq pin |
|-------------|----------|
| SW0         | F22      |
| SW1         | G22      |
| SW2         | H22      |
| SW3         | F21      |
| SW4         | H19      |
| SW5         | H18      |
| SW6         | H17      |
| SW7         | M15      |

**Table 14 - LED Connections**

| Signal Name | Subsection | Zynq pin  |
|-------------|------------|-----------|
| LD0         | PL         | T22       |
| LD1         | PL         | T21       |
| LD2         | PL         | U22       |
| LD3         | PL         | U21       |
| LD4         | PL         | V22       |
| LD5         | PL         | W22       |
| LD6         | PL         | U19       |
| LD7         | PL         | U14       |
| LD9         | PS         | D5 (MIO7) |

- i. Close Simulation Window, right click **Constraints** and select **Add Sources**. Click **Next** and **Create File**. Enter **File name** as ZedBoard.xdc. Please make sure that the file type is .xdc. Click **OK**, and then click **Finish**.
- ii. Edit ZedBoard.xdc

```

set_property PACKAGE_PIN T22 [get_ports {led}]

set_property IOSTANDARD LVCMOS33 [get_ports {led}]

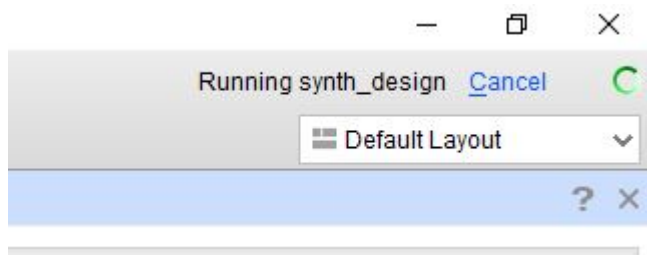
set_property PACKAGE_PIN F22 [get_ports {sw}]

set_property IOSTANDARD LVCMOS33 [get_ports {sw}]

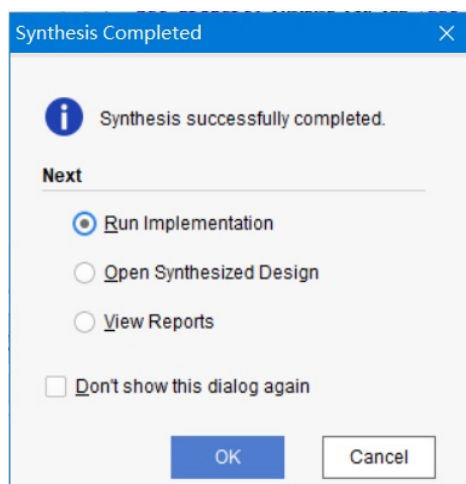
```



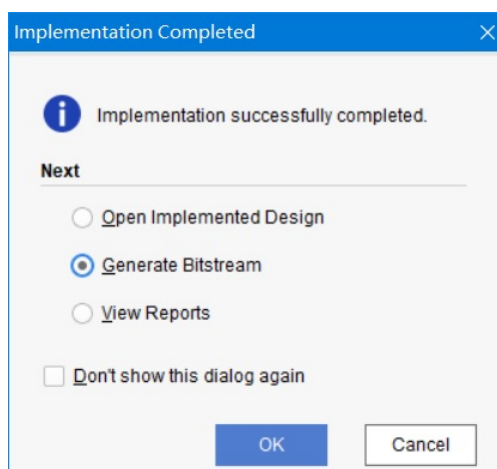
- iii. Click **Run Synthesis**, it may take some time and you can see the process on the top right corner.



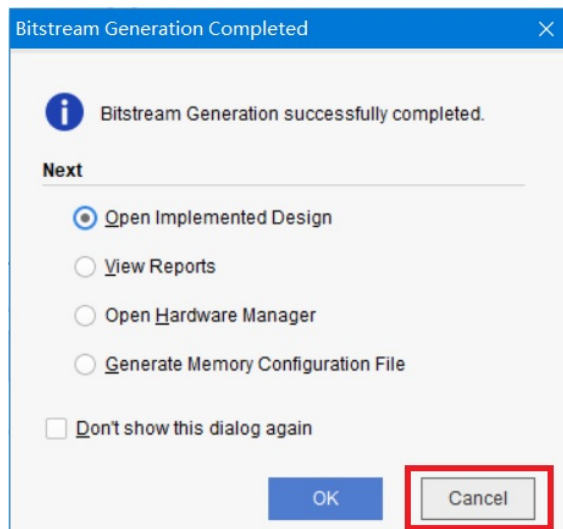
After synthesis is finished, you will see the following dialog, select **Run Implementation** and click **OK**. This process may take a few minutes.



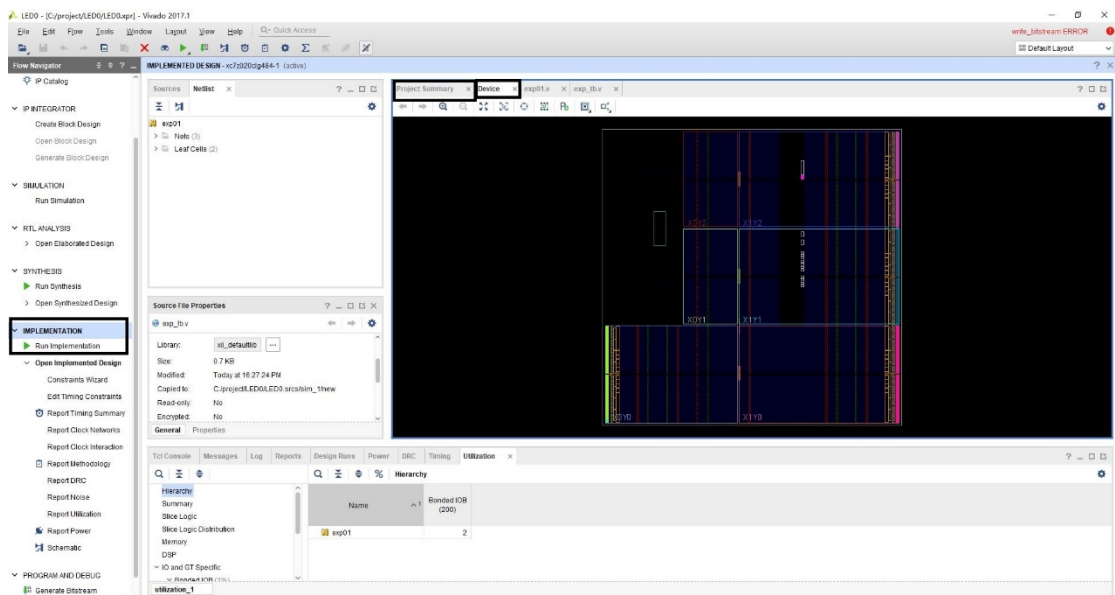
After implementation is finished, you will see the following dialog, select **Generate Bitstream** and click **OK**.



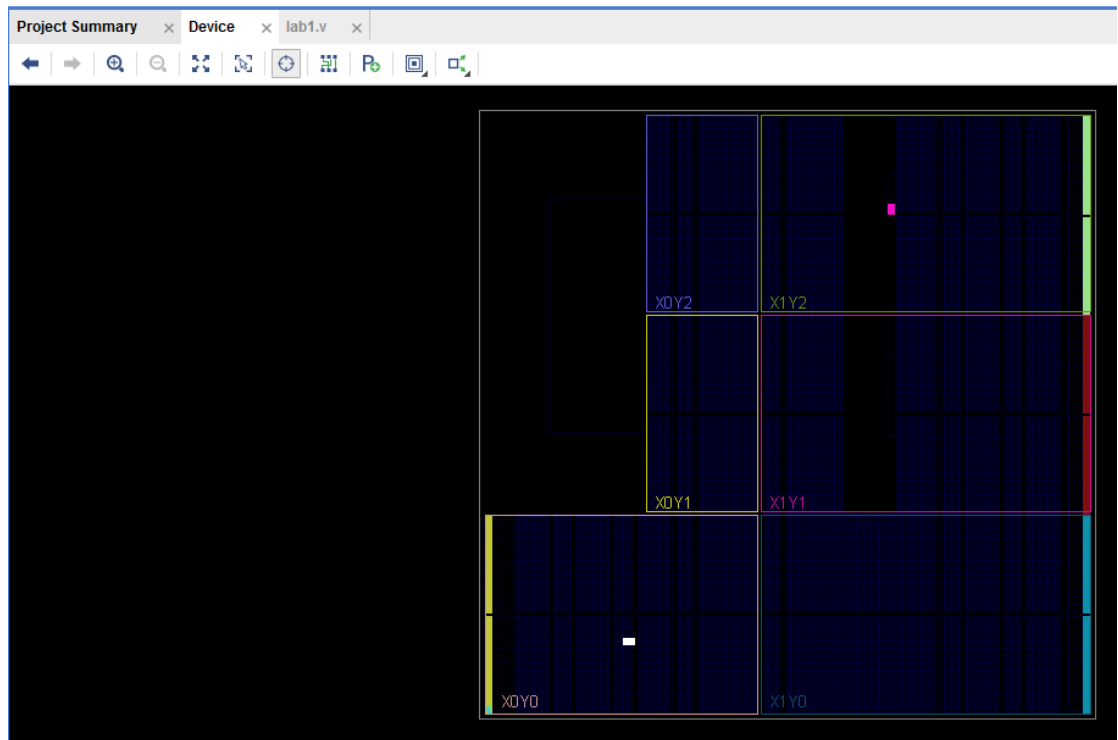
Finally the bitstream is generated, you will see the following dialog, click **Cancel**.



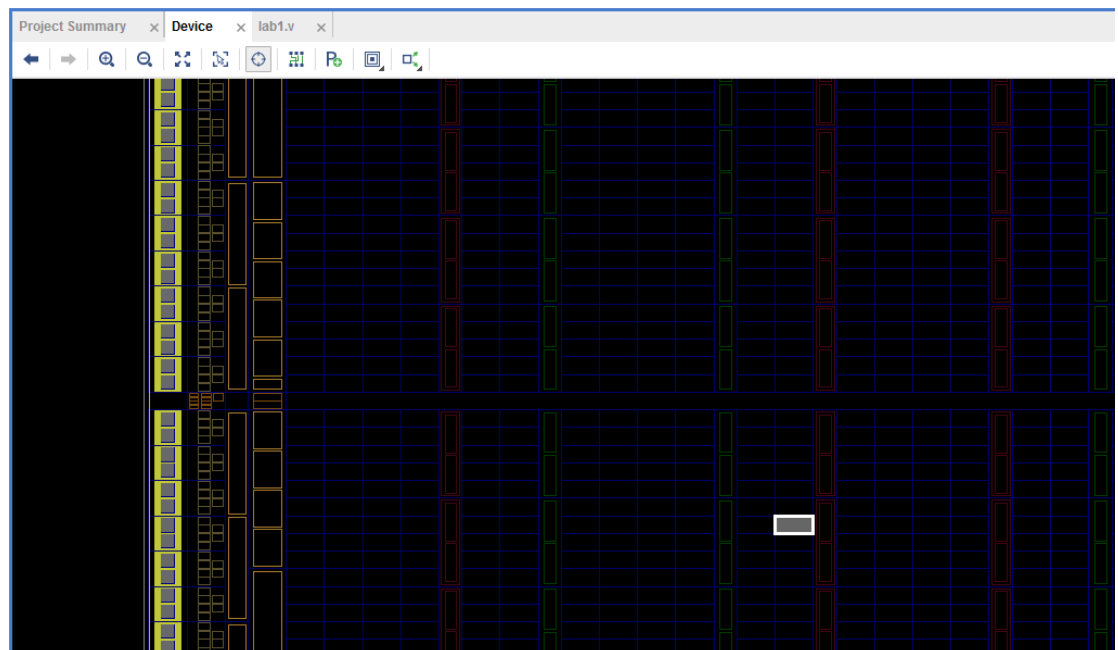
- iv. Click **Open Implementation**, after the synthesis is finished, you will see the occupation of device in Vivado.



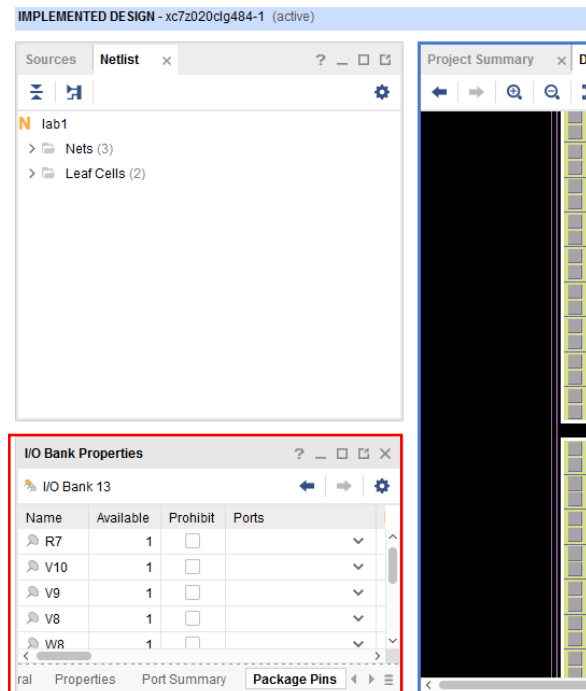
Press “ctrl” on the keyboard and turn the “scroll wheel” of the mouse, then we can zoom in/ zoom out the floorplan as shown below.



Zoom in the floorplan and scroll to the bottom left corner as shown below.



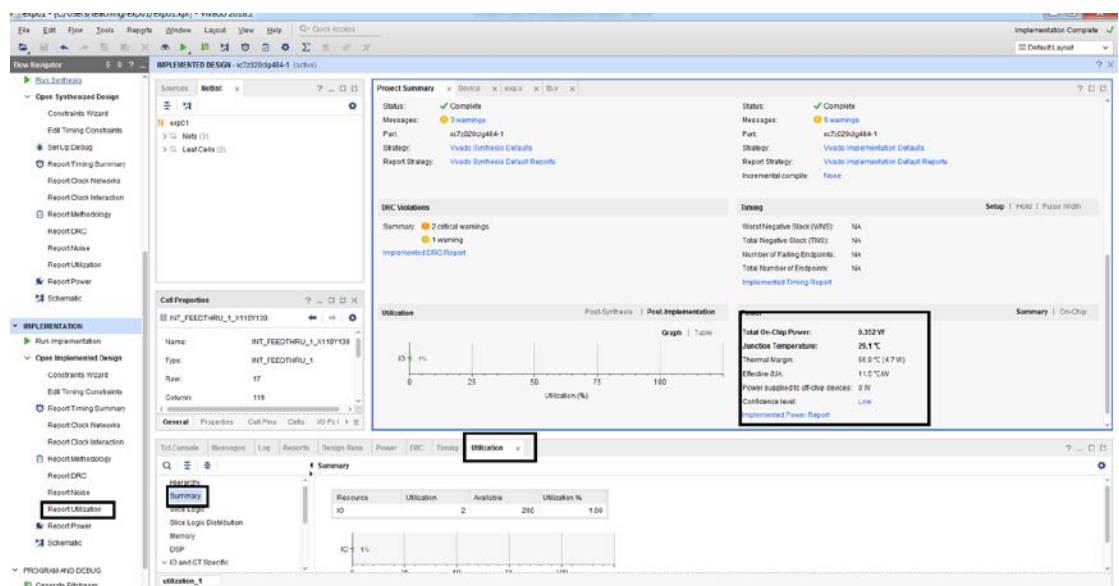
Click the yellow bar, we can find “I/O Bank Properties” in the left panel, we can see the detail information of the selected component.



**Checkpoint 3: Try to select the blue box, pink box and green box. And find out what are these components.**

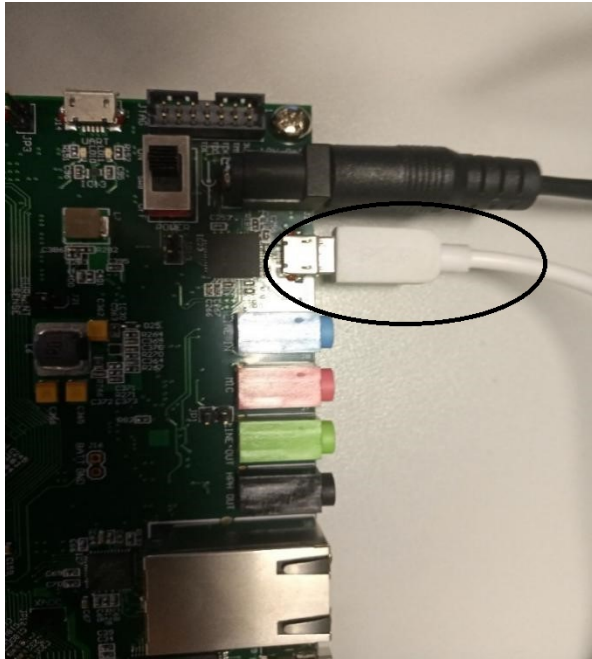
**Checkpoint 4: Please have a look at the synthesized results and the report. How many LUT are being used? Why? How about IOB?**

Then, click report utilization in the flow navigator under implementation to get the report.

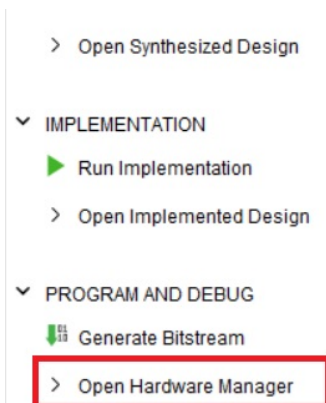


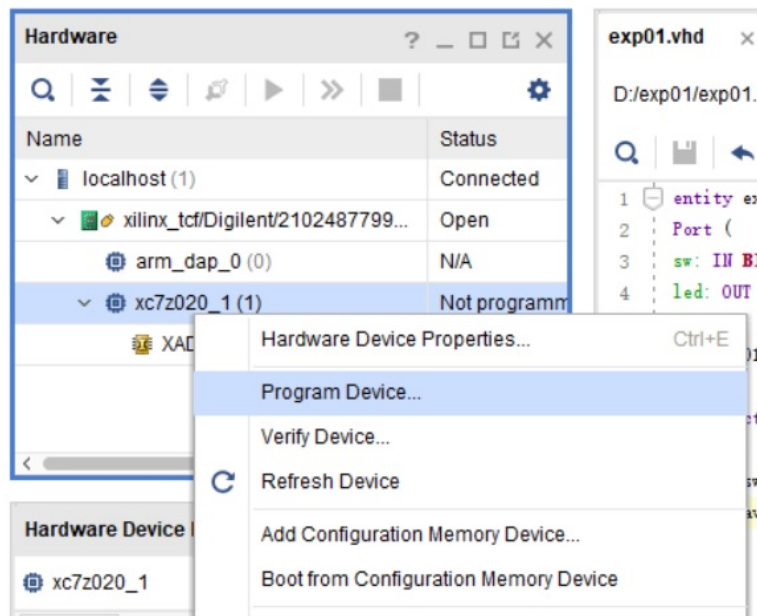
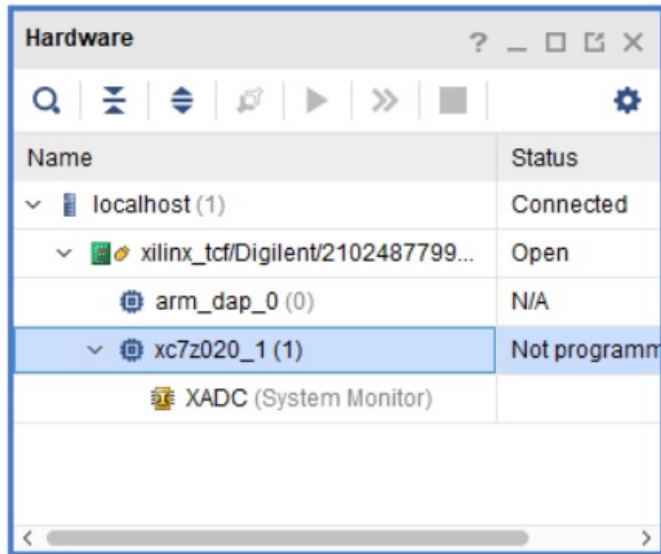
Besides the utilization, the power consumption and temperature are also shown in the picture.

- v. Make sure the ZedBoard has been correctly connected to PC. The USB cable is connected to the PC.



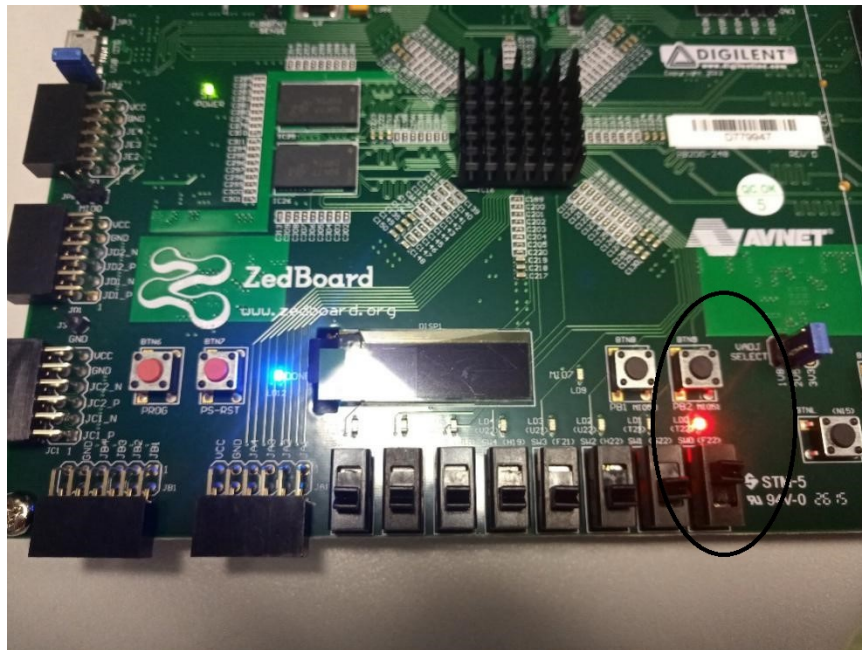
- vi. Open **Hardware Manager**, Click **Open target** and select **Auto Connect**. If ZedBoard is correctly connected with the PC, you will see the FPGA chip from the Vivado software. **Right click xc7z020\_1** and select **Program Device**. Then click **Program**.





You can find the file from impl\_1/exp01.bit

- vii. Turn on one of the LEDs, LED0.



**6. Create your own Xilinx account on the Xilinx official website, in order to get further supporting documents.**

First, enter [www.xilinx.com](http://www.xilinx.com) and click on the circle number 1 and create the account. In the shown figure, you can enter your CityU email and get some support, which is also shown in the figure.

**CHECKPOINT 5: Create your own Xilinx account.**

**CHECKPOINT 6: Can you implement a configurable logic design that can use**

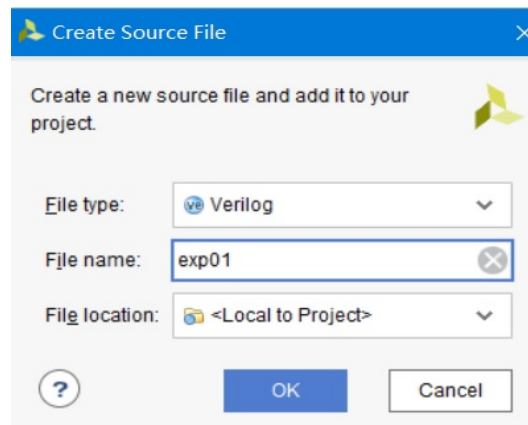


**20% or more of the FPGA chip?**

## 7. Create a Verilog design

The Verilog design flow is similar, but there are some differences need to be noticed.

- i. When creating the design file and testbench file, remember to choose the File type as **Verilog**. Then you can create two files **exp01.v** and **exp01\_tb.v**.



- ii. If you are not familiar with Verilog HDL syntax, you can google it.

Edit exp01.v

```
`timescale 1ns / 1ps

module exp01 (

    input sw,

    output led

);

    -- Put your own code here

endmodule
```

Edit exp01\_tb.v

```
`timescale 1ns / 1ps

module exp01_tb();
```

```

reg switch;

wire led;

exp01 exp01_inst (.sw(switch), .led(led));

initial

begin

    #0 switch = 1'b0; // # means a time delay, so at 5ns, switch will be set to 1

    #5 switch = 1'b1;

    #10 switch = 1'b0;

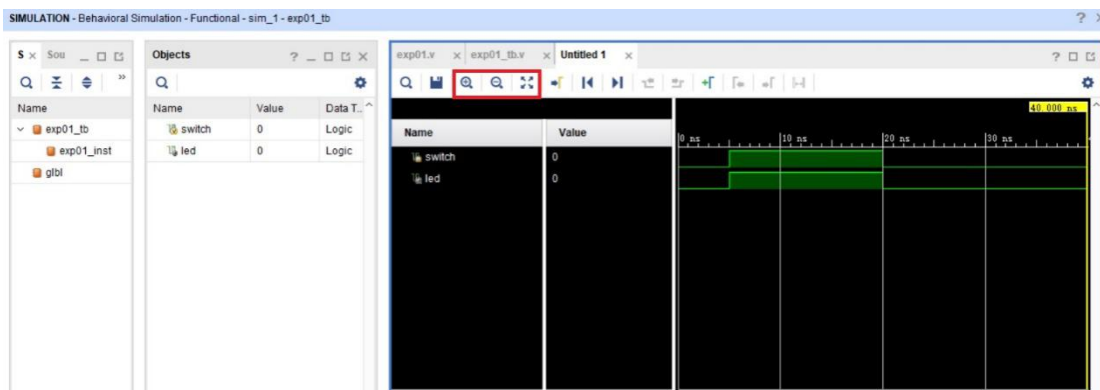
    #15 $stop; // $stop will stop the simulation

end

endmodule

```

iii. You can find the **simulation waveform** like this:



**CHECKPOINT 7: What are are basic differences between VHDL and Verilog syntax? Take a screenshot of your generated waveform using the Verilog.**

## Supplementary Materials:

VHDL Code Structure- Three Steps:

- i. Library Declaration
- ii. Entity Declaration
- iii. Architecture Declaration

*Library declaration:* locate the system library, IEEE Standard library is often included in the VHDL code. You can create your own library.

*Entity declaration:* defines the external interface to the current design [input (port) / output (port)].

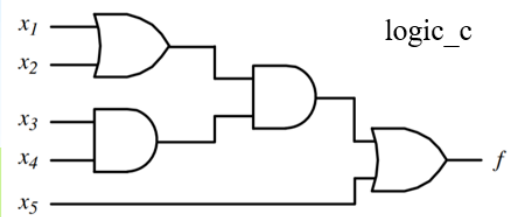
*Architecture declaration:* describes the internal circuit of the corresponding entity [logic circuit].

Example:

```
library ieee;  
use ieee.std_logic_1164.all;
```

```
entity logic_c is  
  port(  
    x1,x2,x3,x4,x5 : in  bit;  
    f               : out bit  
  );  
end logic_c;
```

```
architecture Behavior of logic_c is  
begin  
  f <= ((x1 or x2) and (x3 or x4) or x5);  
end Behavior;
```



## VHDL Simulation:

### *Testbench*

- Performing simulation to verify through simulation waveform;
- Provide stimulus for the UUT (unit under test) to check the output;

Template:

```
entity circuit_tb is
-- port (empty) (not required for simulation)
end entity circuit_tb;

architecture behavioral of circuit_tb is

    component declarations

    signal declarations

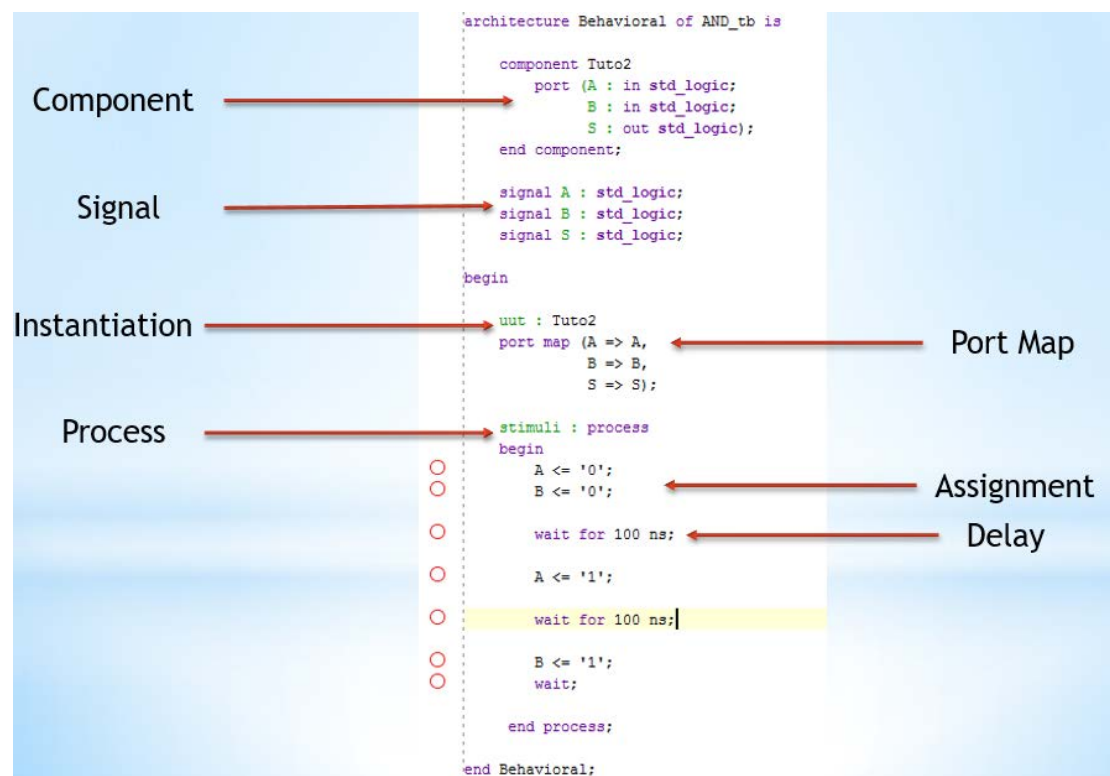
begin

    component instantiations

    process

end behavioral;
```

Example:



- Component declaration declares a virtual circuit template, which must be instantiated to take effect during the design.
- Port map is required for component instantiation. Port mapping is for connecting signals of the design in which the components are instantiated with the ports of the component itself.
- Multiple processes may exist in an architecture. All processes in a VHDL description file (.VHD) are executed in parallel – concurrent process blocks. Statements within a process are executed sequentially.