

EIE2810 Digital Systems Design Laboratory

## Laboratory Report #6

Name: Tengfei Ma

Student ID: 121090406

Date: 2024/5/11

The Chinese University of Hong Kong, Shenzhen

## Introduction

This laboratory consists of 2 experiments, they are:

- Experiment A: Using VHDL language to construct an 8-input AND Gate and 2-input-1-output XOR gate on the development board.
- Experiment B: Using VHDL language to construct a user-input down counter on the development board(EGO1).

## 1. Experiment A

### 1.1 Coding

#### 1.1.1 8-input AND gate

```
library IEEE;
```

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

```
entity MyAndGate is
```

```
    Port ( A : in STD_LOGIC_VECTOR (7 downto 0);
```

```
          Y : out STD_LOGIC);
```

```
end MyAndGate;
```

```
architecture Behavioral of MyAndGate is
```

```
begin
```

```
Y <= A(7) and A(6) and A(5) and A(4) and A(3) and A(2) and A(1) and  
A(0);
```

```
end Behavioral;
```

### 1.1.2 2-input-1-output XOR gate

```
library IEEE;
```

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

```
entity MyXORGate is
```

```
    Port ( A : in STD_LOGIC_VECTOR (1 downto 0);
```

```
          Y : out STD_LOGIC);
```

```
end MyXORGate;
```

```
architecture Behavioral of MyXORGate is
```

```
begin
```

```
Y <= A(1) xor A(0);
```

```
end Behavioral;
```

## 1.2 Results

### 1.2.1 8-input AND gate

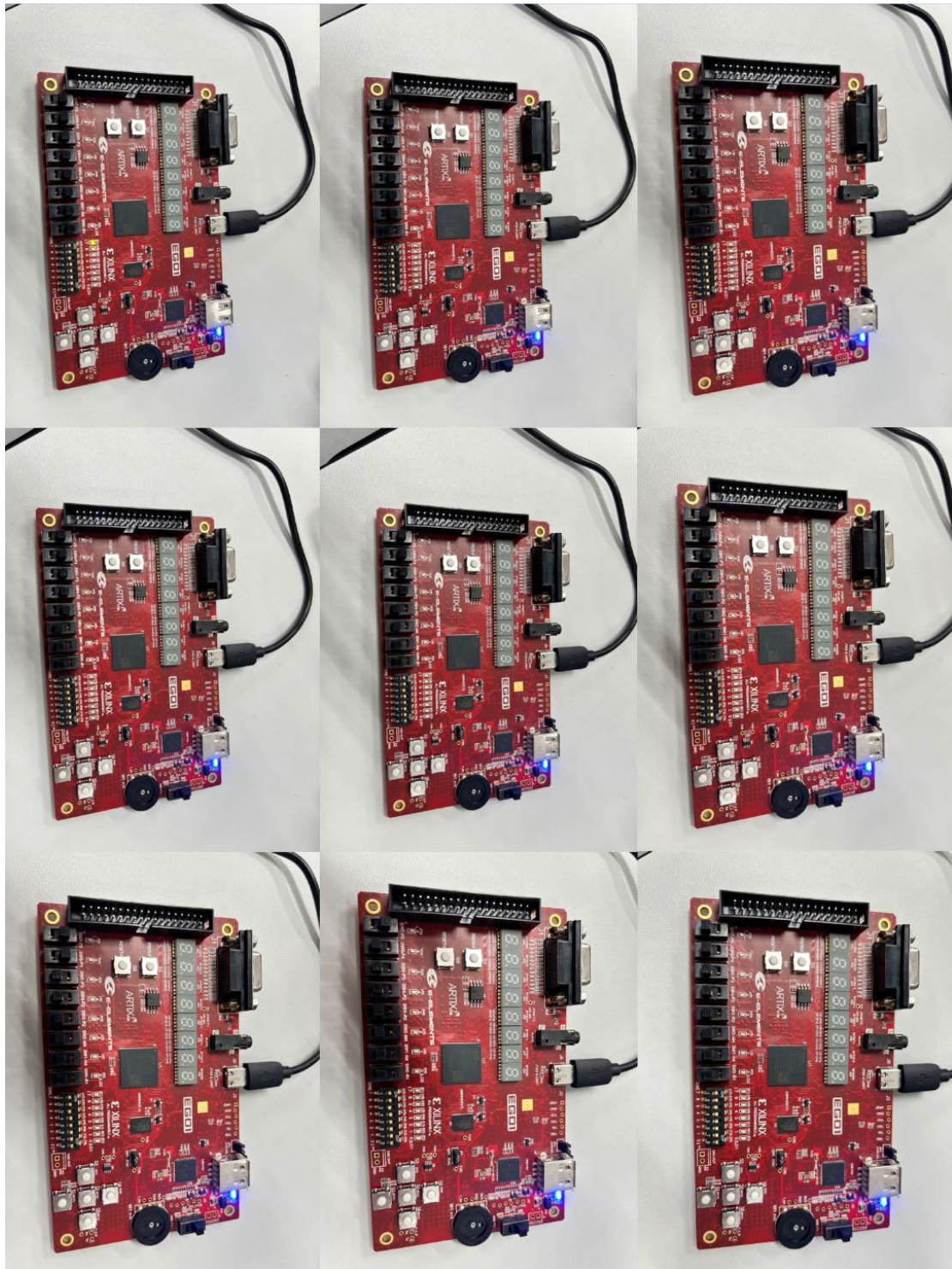


Figure 1 Results of 8-input AND gate

As shown in Figure 1, only when all the 8 switches (SW7~SW0) are on, the LD1(7) is on. Otherwise, the LD1(0) is off.



## 1.2.2 2-input-1-output XOR gate

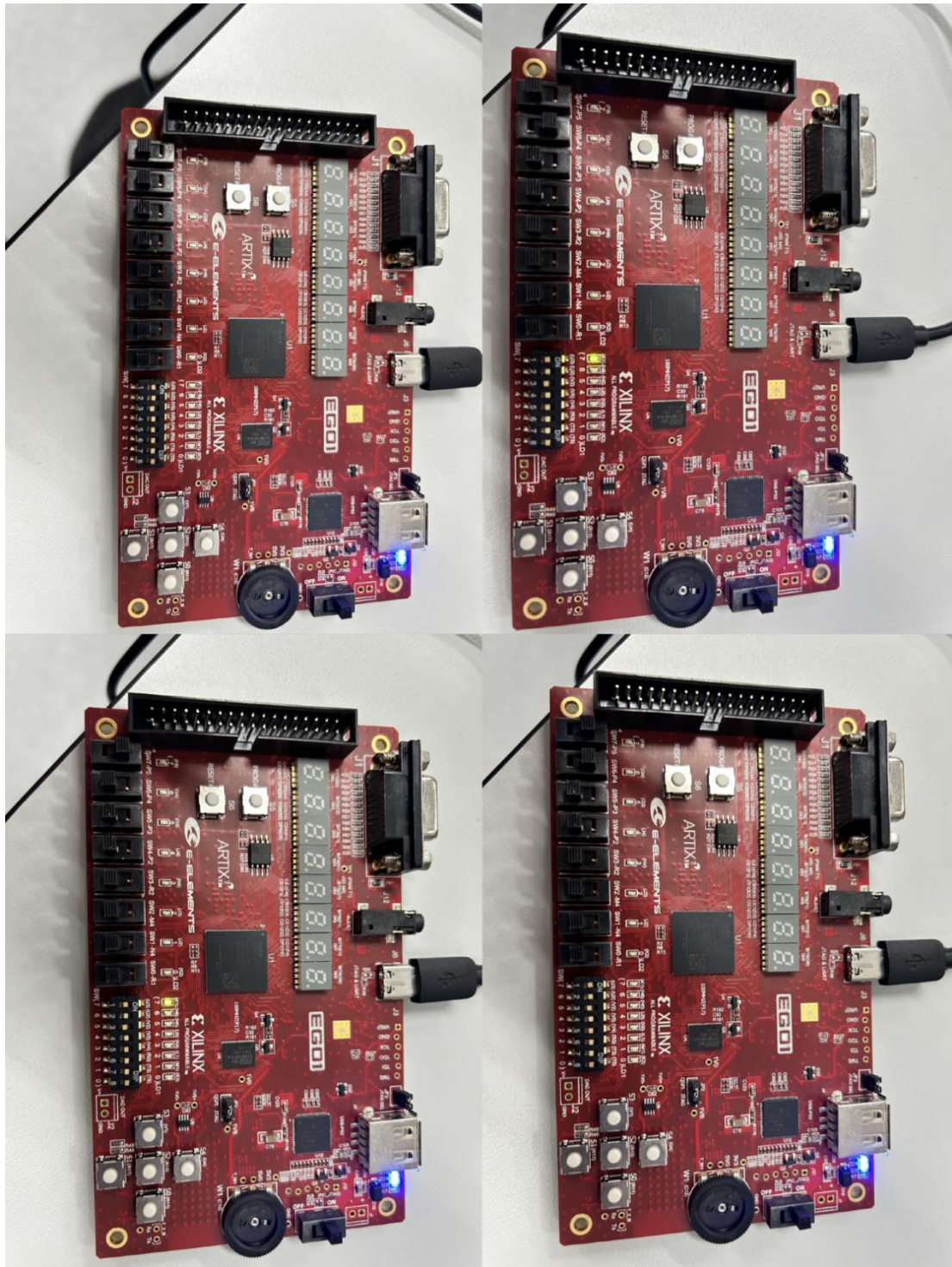


Figure 2 Results of 2-input-1-output XOR gate

As shown in Figure 2, when the states of SW7 and SW6 are different, LD(7) is on. When the states of them are the same, LD(7) is off.

### 1.3 Questions

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity FullAdder is

port( A: in std\_logic;

      B: in std\_logic;

      Cin: in std\_logic;

      S: out std\_logic;

      C: out std\_logic);

end entity;

architecture adding of FullAdder is

begin

S <= (A xor B) xor Cin;

C <= (A and (B or Cin)) or (Cin and B);

end adding;

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

entity 4BitAdder is

port( BCD1: in std\_logic\_vector(3 downto 0);

      BCD2: in std\_logic\_vector(3 downto 0);

      BCD3: out std\_logic\_vector(3 downto 0);

      carry: out std\_logic);

end entity;

architecture Add of 4BitAdder is

component FullAdder is

```
port( A: in std_logic;  
      B: in std_logic;  
      Cin: in std_logic;  
      S: out std_logic;  
      C: out std_logic);
```

end component;

```
signal carry1: std_logic:='0';
```

```
signal carry2: std_logic:='0';
```

```
signal carry3: std_logic:='0';
```

```
signal carry4: std_logic:='0';
```

```
signal carry5: std_logic:='0';
```

begin

```
Adding0: FullAdder port map (BCD1(0), BCD2(0), carry1, BCD3(0), carry2);
```

```
Adding1: FullAdder port map (BCD1(1), BCD2(1), carry2, BCD3(1), carry3);
```

```
Adding2: FullAdder port map (BCD1(2), BCD2(2), carry3, BCD3(2), carry4);
```

```
Adding3: FullAdder port map (BCD1(3), BCD2(3), carry4, BCD3(3), carry5);
```

```
carry <= carry5;
```

end Add;

## 2. Experiment B

### 2.1 Manual

#### 2.1.1 Creating a project

Firstly, click the icon of vivado shown in Figure 3.



Figure 3 icon of vivado

As shown in Figure 4, click “Create Project”.

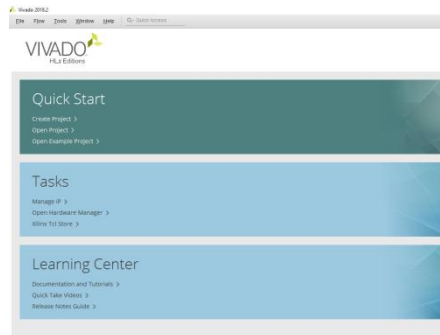


Figure 4 Start Creating a Project

As shown in Figure 5, click “Next”.

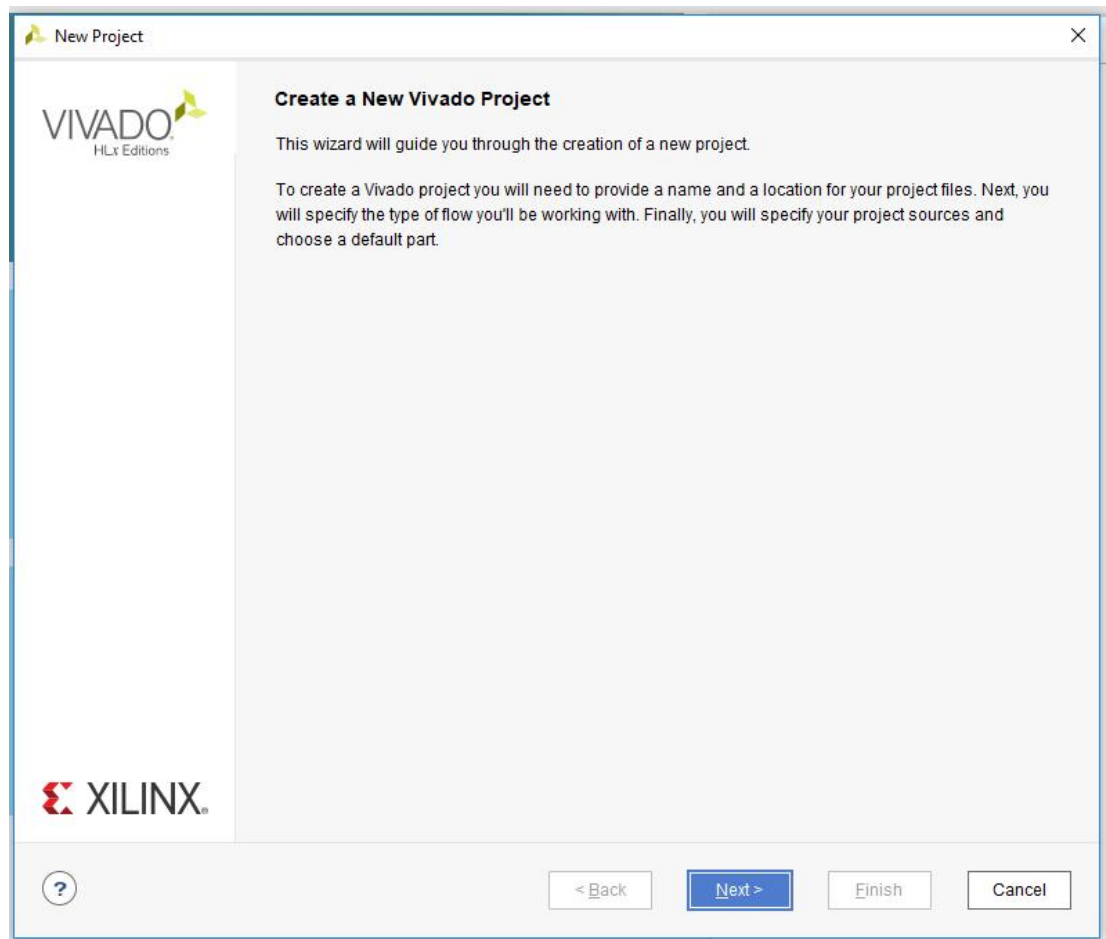
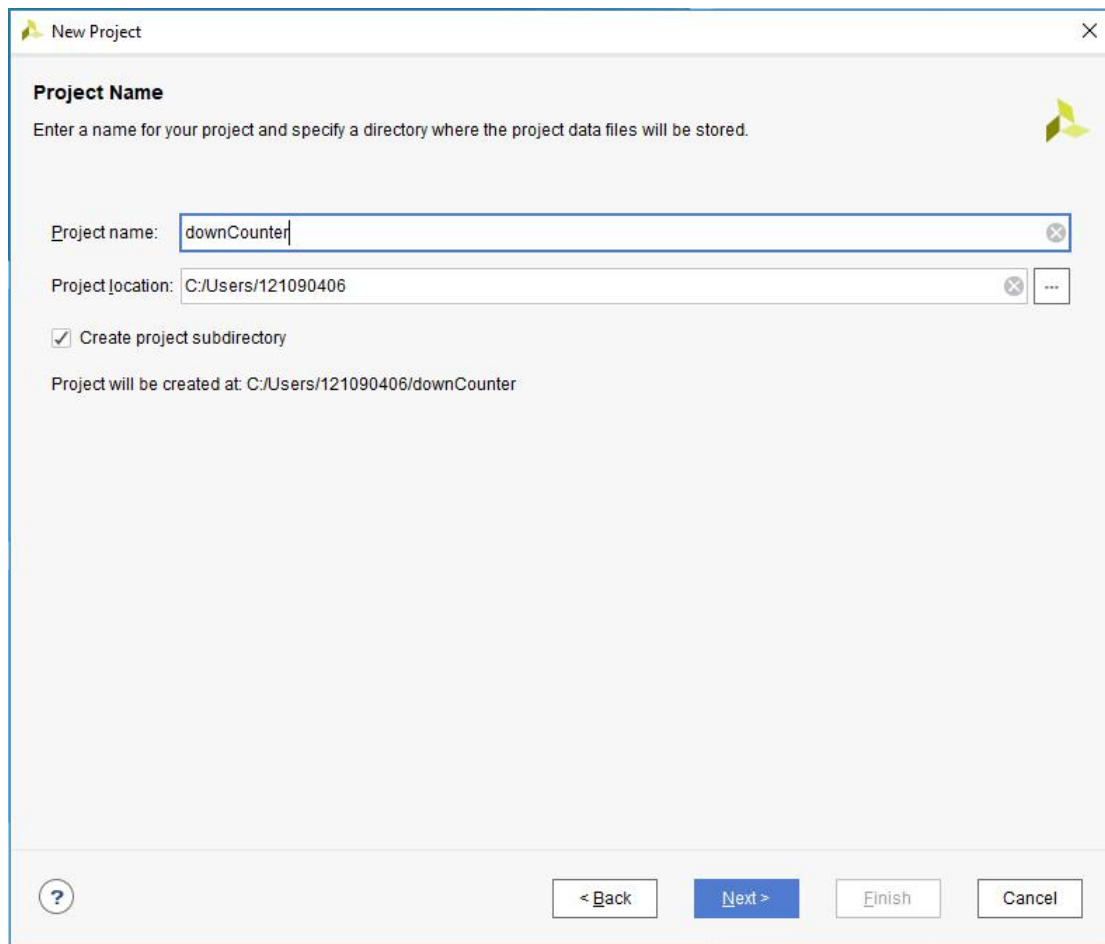


Figure 5 Creating a Project



As shown in Figure 6, enter the name of project and then 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: downCounter

Project location: C:/Users/121090406

☒ Create project subdirectory

Project will be created at: C:/Users/121090406/downCounter

? < Back Next > Finish Cancel

Figure 6 Name the Project

As shown in Figure 7, choose “RTL Project” and then click “Next”.

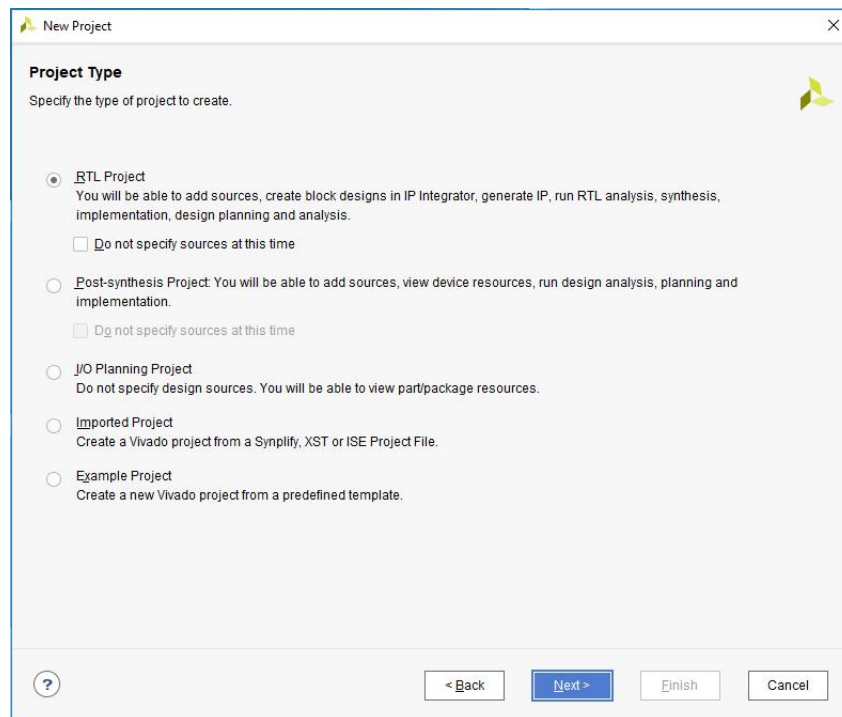


Figure 7 Choosing the Project Type

As shown in Figure 8, choose VHDL language and add a source file to the project.

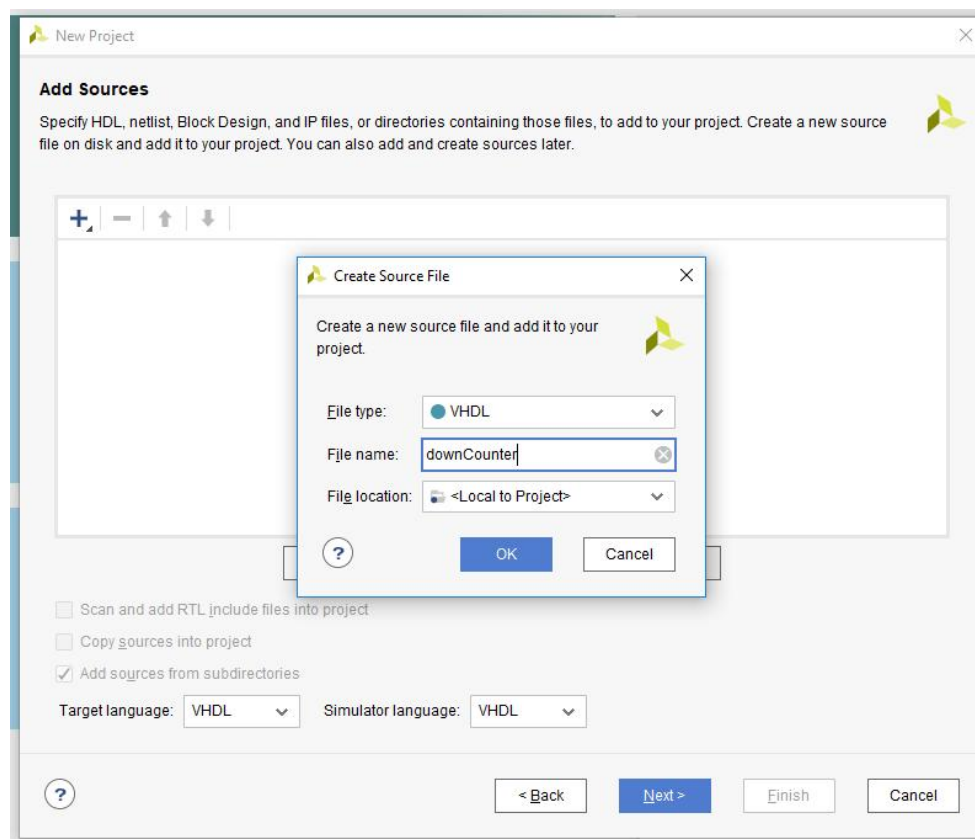


Figure 8 Choosing the Language and Creating Source File

As shown in Figure 9, click “Next”.

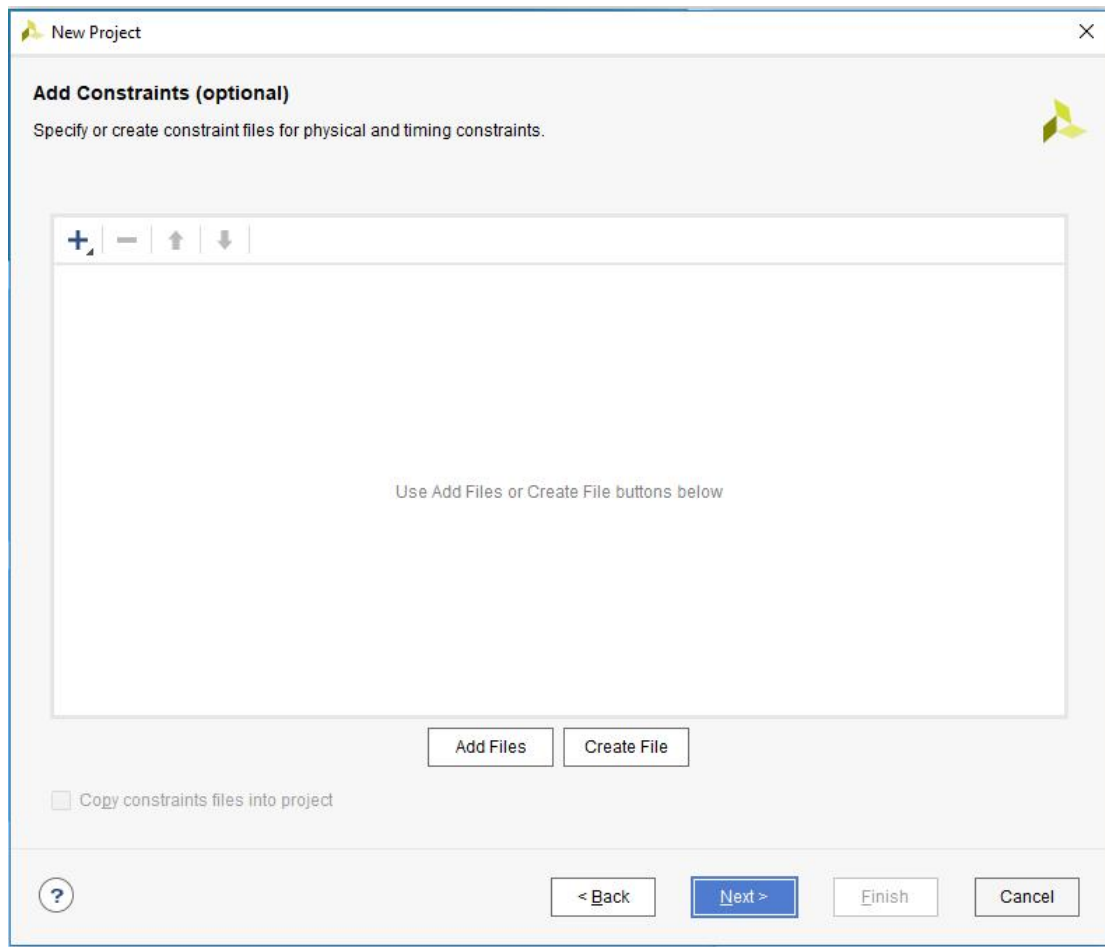


Figure 9 Neglecting Adding Constraints

As shown in Figure 10, in the “Search”, enter “xc7a35tcs324-1”, and select the corresponding chip. After that, click “Next”.

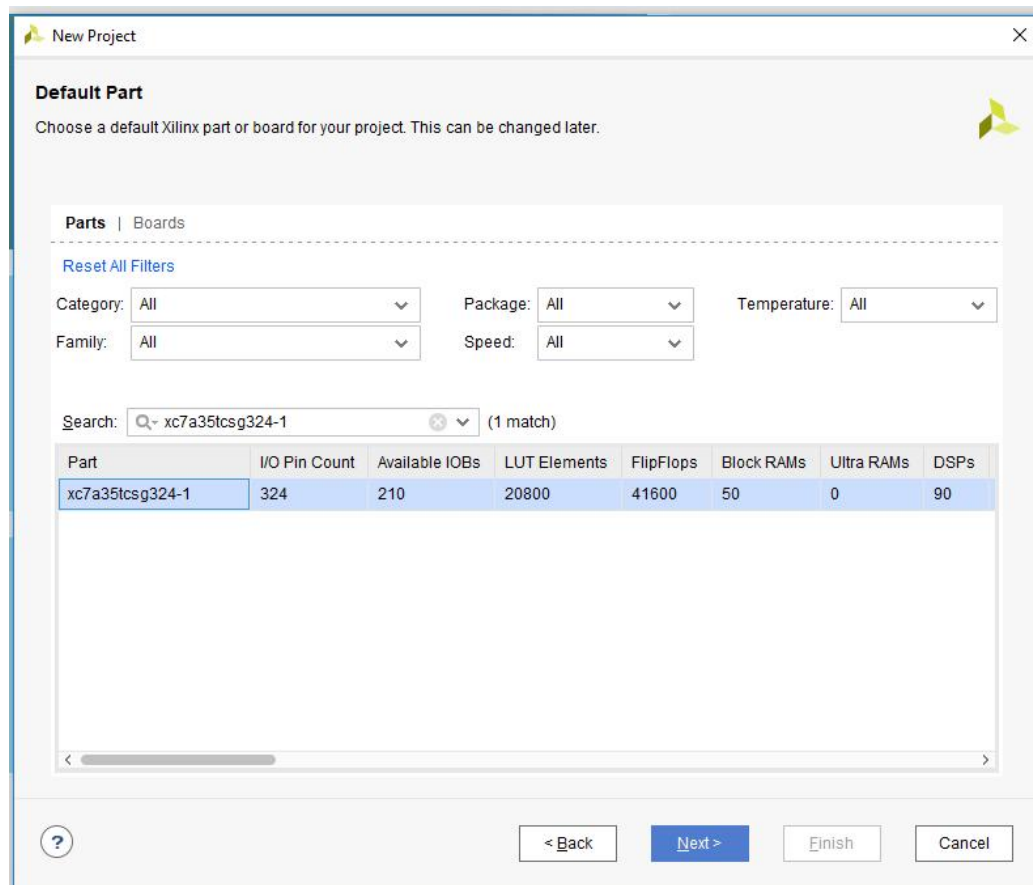


Figure 10 Selecting the Proper Chip for the Experiment

As shown in Figure 11, click “Finish”.

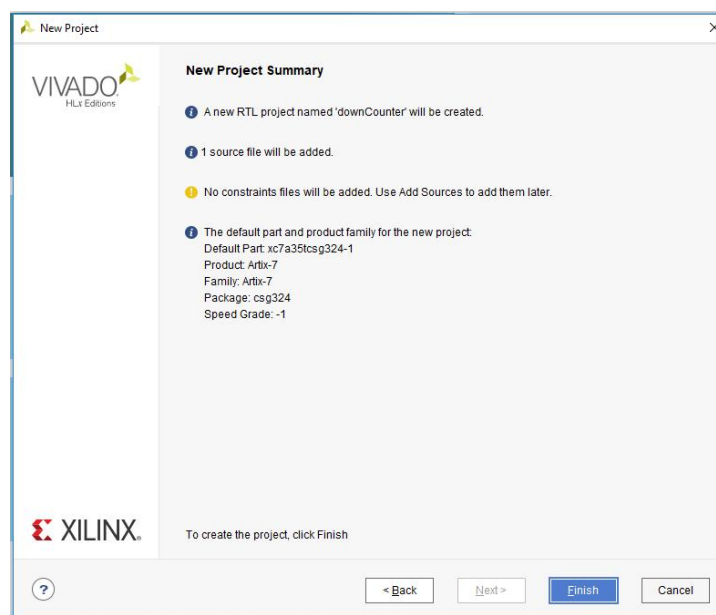


Figure 11 End of Creating the Project

### 2.1.2 Coding and Combining the Codes to the Development Board

As shown in Figure 12, input port names, directions, bus choices, MSB and LSB.

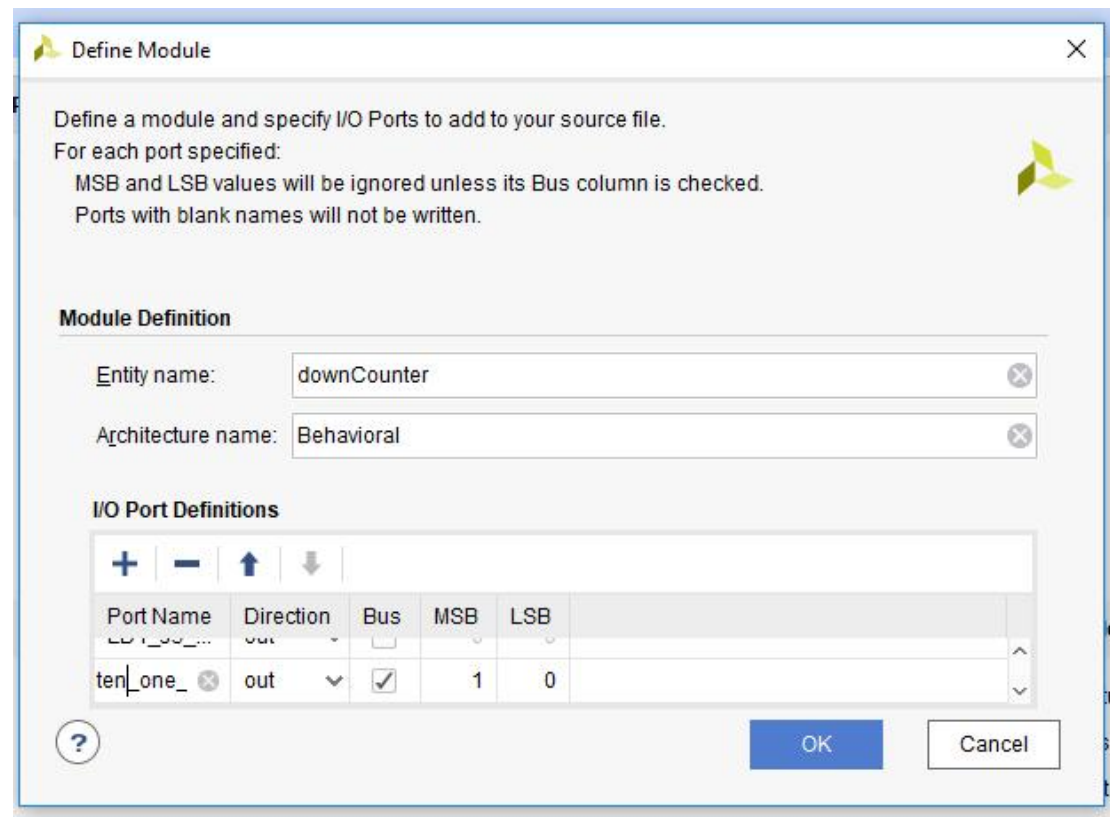


Figure 12 Creating Entity

As shown in Figure 13, double-click the source file in the Design Source to open the coding box.



Figure 13 The Source Box

As shown in Figure 14, write your code.

```
62  -- generate 8 Hz : divide8
63  process (clk_p17)
64  begin
65  if (rising_edge(clk_p17)) then
66      count <= count+1;
67      if (count = 6250000) then
68          divide8 <= NOT divide8;
69          count <= 1;
70      end if;
71  end if;
72  end process;
73
74  -- generate 4 Hz : divide4
75  process (divide8)
76  begin
77  if (rising_edge(divide8)) then
78      divide4 <= NOT divide4;
79  end if;
80  end process;
81
82  -- generate 2 Hz : divide2
83  process(divide4)
```

Figure 14 The Coding Box

As Shown in Figure 15, click “Run Synthesis”.

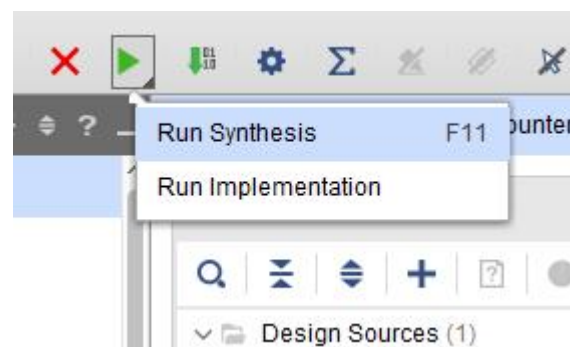


Figure 15 Run Synthesis



As shown in Figure 16, click “OK” with default settings.

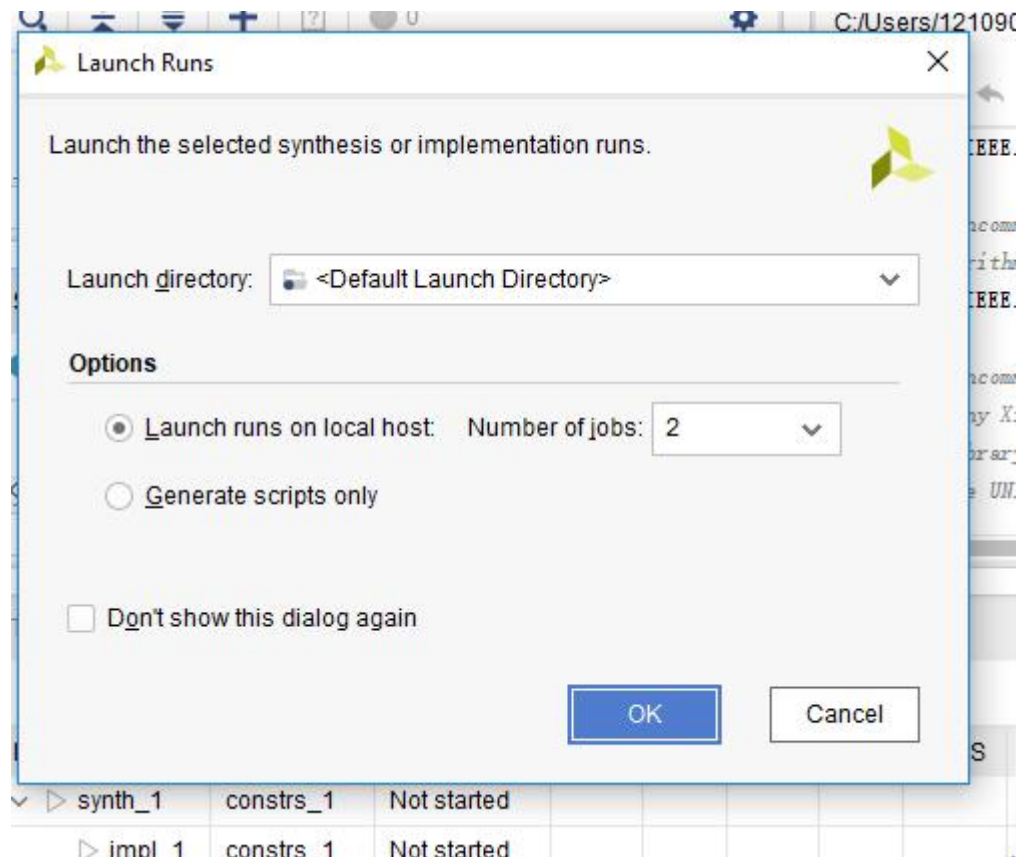


Figure 16 Settings of “Run Synthesis”

As shown in Figure 17, after finishing synthesis, click “Open Synthesis Design”.

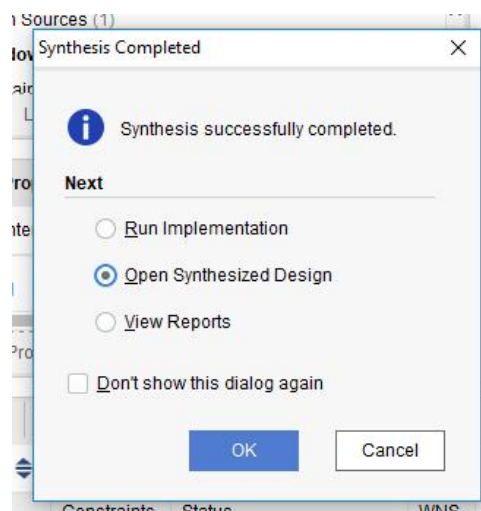


Figure 17 Open Synthesized Design

As shown in Figure 18, choose “I/O Planning”.

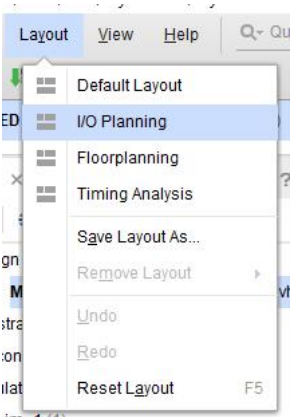


Figure 18 Choose I/O Planning

As shown in Figure 19 and Figure 20, Fix the ports and set “I/O Std” to “LVCMOS33\*”, and then “Ctrl + s” to save the file. Then click “OK”.

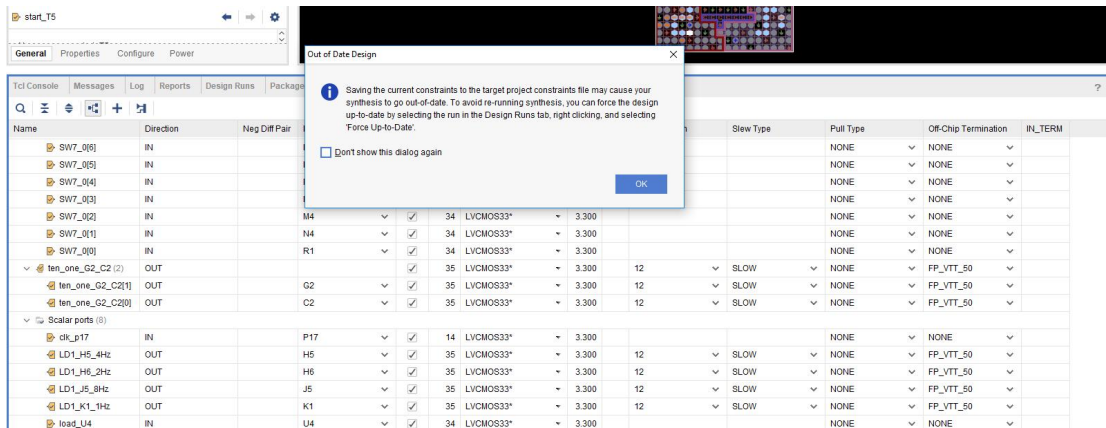


Figure 19 Choosing Open Implemented Design

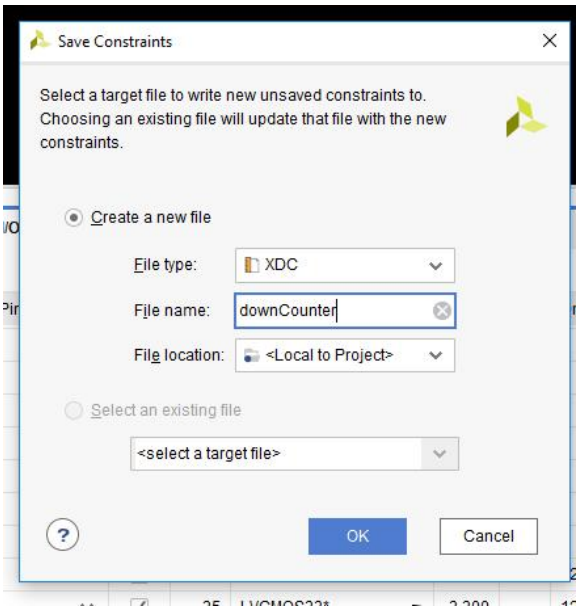


Figure 20 Save constraint file

As shown in Figure 21, click “Run Implementation”.

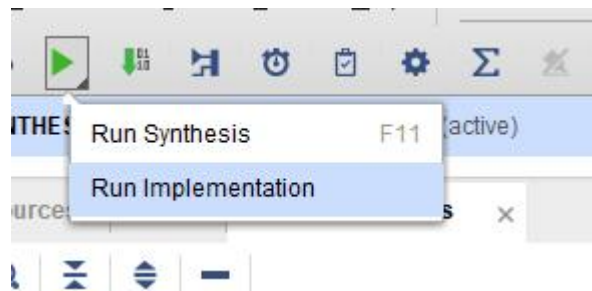


Figure 21 Run Implementation

As shown in Figure 22, keep default settings and click “OK”.

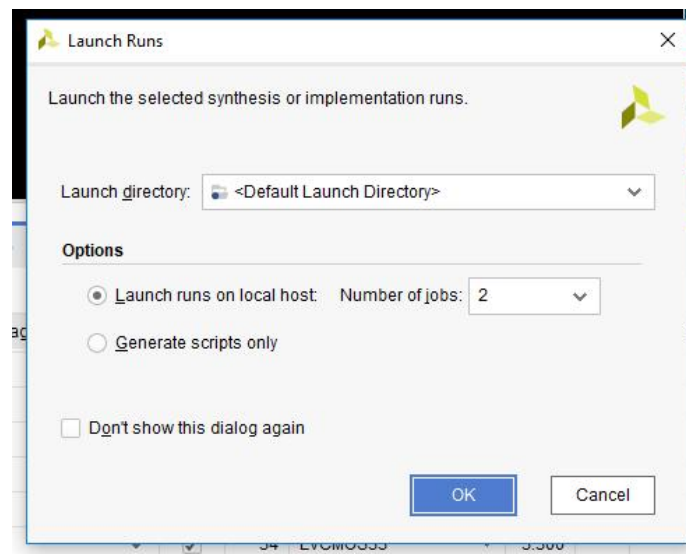


Figure 22 Settings of “Run implementation”

As shown in Figure 23, choose “Generate Bitstream” and then click “OK”.

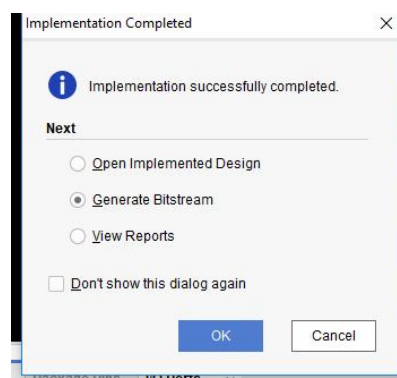


Figure 23 Generating Bitstream

As shown in Figure 24, 25, after bitstream generation is done, choose “Open Hardware Manager” and click “OK”. Then connect J6 port of the board with a computer through USB, turn on the switch. The driver will be installed. Click “Open Target” and click “Auto Connect”.

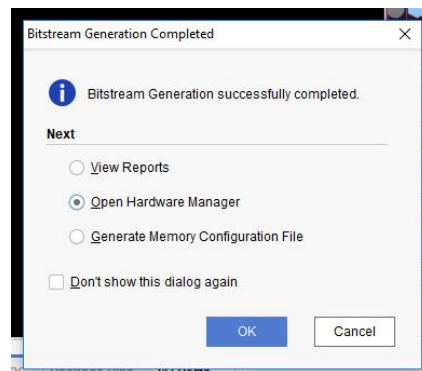


Figure 24 Opening Hardware Manager

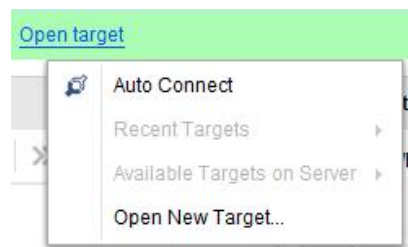


Figure 25 Connecting the Computer and the Development Board

Finally, click “Program device” and then “Program” shown in Figure 26, 27.



Figure 26 “Program device”

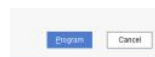


Figure 27 “Program”

## 2.2 Results

### 2.2.1 Experiment A, B

As shown in Figure 28, LD1(4-7) are flashing at the frequencies of 8Hz, 4Hz, 2Hz, 1Hz. And Load SW7-4 as 10s digit, as well as load SW3-0 as 1s digit into 7-segments displays by pressing button S4. Note that the BCD exceeding 0b1001 will be regarded as 0b1001.

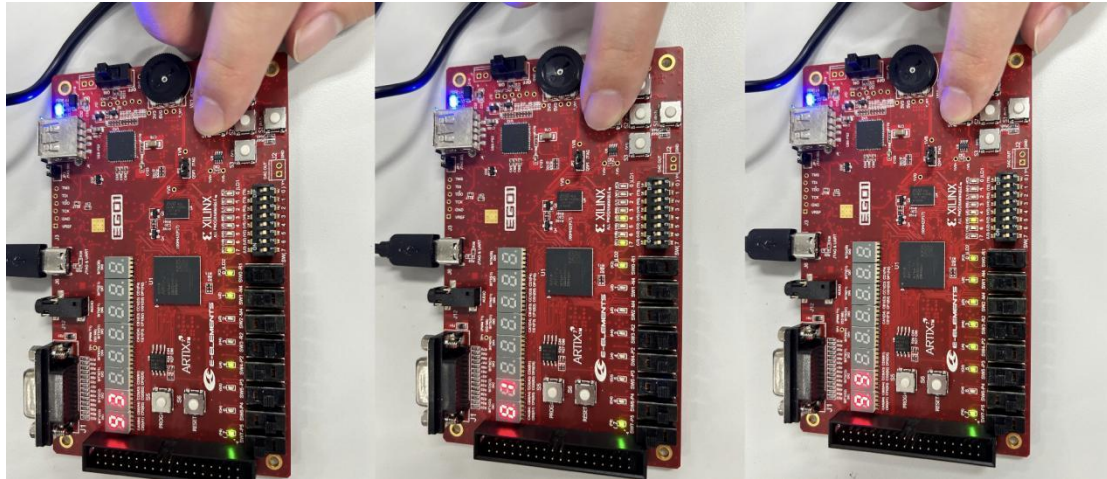


Figure 28 Results of Experiment A, B

### 2.2.2 Experiment C

As shown in Figure 29, we take 23 (0b0010, 0b0011) for example. First set the SW7-0 as 0010 0011 and then press S4 and turn SW(0) on. The counter will count down from 23 to 0. When reaching 0, it retains 0.



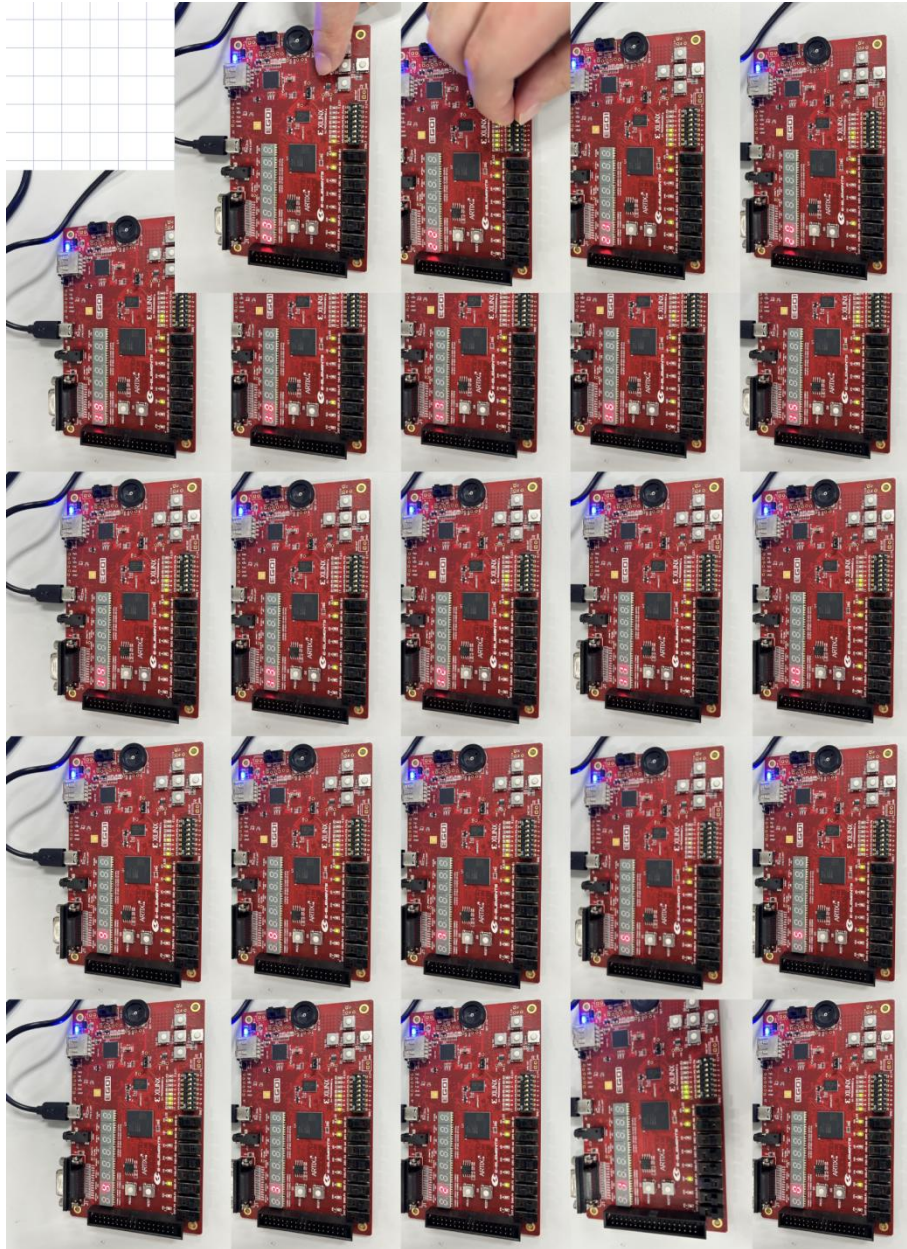


Figure 29 Result of Experiment C

## 2.3 Coding

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.NUMERIC\_STD.ALL;

entity labb is

Port ( clk\_p17 : in STD\_LOGIC;



```
    load_u4 : in STD_LOGIC;
    start_T5 : in STD_LOGIC;
    SW7_0 : in STD_LOGIC_VECTOR (7 downto 0);
    LD2 : out STD_LOGIC_VECTOR (7 downto 0);
    SevenSeg : out STD_LOGIC_VECTOR (6 downto 0);
    LD1_K1_1Hz : out STD_LOGIC;
    LD1_H6_2Hz : out STD_LOGIC;
    LD1_H5_4Hz : out STD_LOGIC;
    LD1_J5_8Hz : out STD_LOGIC;
    ten_one_G2_C2 : out STD_LOGIC_VECTOR (1 downto 0));
end labb;
```

architecture Behavioral of labb is

signal divide1: std\_logic:='0'; -- 1Hz

signal divide2: std\_logic:='0'; -- 2Hz

signal divide4: std\_logic:='0'; -- 4Hz

signal divide8: std\_logic:='0'; -- 8Hz

signal count: integer:=1; --used in 8Hz

signal count1: integer:=1; --used in special clock

signal c\_num: integer:=0;

signal spc\_clk: std\_logic:='0';

begin

-- generate 8 Hz : divide8

```
process (clk_p17)
begin
if (rising_edge(clk_p17)) then
    count <= count+1;
    if (count = 6250000) then
        divide8 <= NOT divide8;
        count <= 1;
    end if;
end if;
end process;

-- generate 4 Hz : divide4
process (divide8)
begin
if (rising_edge(divide8)) then
    divide4 <= NOT divide4;
end if;
end process;

-- generate 2 Hz : divide2
process(divide4)
begin
if (rising_edge(divide4)) then
    divide2 <= NOT divide2;
end if;
end process;
```

```
-- generate 1 Hz : divide1
process(divide2)
begin
if (rising_edge(divide2)) then
    divide1 <= NOT divide1;
end if;
end process;

-- LD1(4-7) (J5, H5, H6, K1) : 8Hz, 4Hz, 2Hz, 1Hz
LD1_J5_8Hz <= divide8;
LD1_H5_4Hz <= divide4;
LD1_H6_2Hz <= divide2;
LD1_K1_1Hz <= divide1;

-- clock division finish

-- read SW7-0 to the system
process(load_U4, divide1)
variable temp_c_num : integer;
variable tmp_c_num : integer;
variable isLoad : integer;
variable isCount : integer;
begin
if (load_U4 = '1') then
    case SW7_0(7 downto 4) is
        when "0000" => temp_c_num := 0;
        when "0001" => temp_c_num := 10;
```

```
when "0010" => temp_c_num := 20;
when "0011" => temp_c_num := 30;
when "0100" => temp_c_num := 40;
when "0101" => temp_c_num := 50;
when "0110" => temp_c_num := 60;
when "0111" => temp_c_num := 70;
when "1000" => temp_c_num := 80;
when "1001" => temp_c_num := 90;
when others => temp_c_num := 90;
end case;
```

case SW7\_0(3 downto 0) is

```
when "0000" => temp_c_num := temp_c_num + 0;
when "0001" => temp_c_num := temp_c_num + 1;
when "0010" => temp_c_num := temp_c_num + 2;
when "0011" => temp_c_num := temp_c_num + 3;
when "0100" => temp_c_num := temp_c_num + 4;
when "0101" => temp_c_num := temp_c_num + 5;
when "0110" => temp_c_num := temp_c_num + 6;
when "0111" => temp_c_num := temp_c_num + 7;
when "1000" => temp_c_num := temp_c_num + 8;
when "1001" => temp_c_num := temp_c_num + 9;
when others => temp_c_num := temp_c_num + 9;
end case;
```

```
LD2 <= SW7_0;
```

```
isLoad := 1;
```

end if;

if (start\_T5 = '1' and rising\_edge(divide1)) then

    if c\_num = 0 then

        tmp\_c\_num := 0;

    else

        tmp\_c\_num := c\_num - 1;

    end if;

    isCount := 1;

end if;

if (isLoad = 1) then

    c\_num <= temp\_c\_num;

elsif (isCount = 1) then

    c\_num <= tmp\_c\_num;

else

    c\_num <= temp\_c\_num;

end if;

isLoad := 0;

isCount := 0;

end process;

-- generate a special clock (200Hz) for the ten and the unit

process (clk\_p17)

begin

if (rising\_edge(clk\_p17)) then

```
        count1 <= count1+1;
    if (count1 = 250000) then
        spc_clk <= NOT spc_clk;
        count1 <= 1;
    end if;
end if;
end process;

-- display
process (spc_clk)
begin
    if spc_clk = '1' then
        ten_one_G2_C2 <= "10";
        case (c_num/10) is
            when 0 => SevenSeg <="0000000"; --do not let it appear 0 when the
ten equals 0
            when 1 => SevenSeg <="0110000";
            when 2 => SevenSeg <="1101101";
            when 3 => SevenSeg <="1111001";
            when 4 => SevenSeg <="0110011";
            when 5 => SevenSeg <="1011011";
            when 6 => SevenSeg <="1011111";
            when 7 => SevenSeg <="1110000";
            when 8 => SevenSeg <="1111111";
            when 9 => SevenSeg <="1111011";
            when others => SevenSeg <= "0000000";
        end case;
    else
```



```
ten_one_G2_C2 <= "01";
case (c_num mod 10) is
  when 0 => SevenSeg <="1111110"; --do not let it appear 0 when the
ten equals 0
  when 1 => SevenSeg <="0110000";
  when 2 => SevenSeg <="1101101";
  when 3 => SevenSeg <="1111001";
  when 4 => SevenSeg <="0110011";
  when 5 => SevenSeg <="1011011";
  when 6 => SevenSeg <="1011111";
  when 7 => SevenSeg <="1110000";
  when 8 => SevenSeg <="1111111";
  when 9 => SevenSeg <="1111011";
  when others => SevenSeg <= "0000000";
end case;

end if;
end process;

end Behavioral;
```

## 2.4 Question

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

entity CountdownTimer is
```

---

```

Port ( CLK_100MHz : in STD_LOGIC;
      START      : in STD_LOGIC;
      RST        : in STD_LOGIC;
      LED        : out STD_LOGIC);
end CountdownTimer;

architecture Behavioral of CountdownTimer is
    constant Divider : natural := 50000000; -- Half the 100 MHz clock for
    1 Hz
    signal Clock_Counter : natural := 0;
    signal CLK_1Hz      : STD_LOGIC := '0';
    signal Timer_Counter : unsigned(23 downto 0) := (others => '0'); -- 24-
    bit enough to count 7200
    signal Timer_Active : STD_LOGIC := '0';
begin
    -- Clock divider process
    Clock_Divider : process(CLK_100MHz, RST)
    begin
        if RST = '1' then
            Clock_Counter <= 0;
            CLK_1Hz <= '0';
        elsif rising_edge(CLK_100MHz) then
            if Clock_Counter = Divider - 1 then
                Clock_Counter <= 0;
                CLK_1Hz <= not CLK_1Hz; -- Toggle the clock to generate 1
    Hz signal
            else
                Clock_Counter <= Clock_Counter + 1;
            end if;
        end if;
    end process;
end architecture;

```

```
        end if;
    end if;
end process Clock_Divider;

-- Countdown timer process
Timer_Logic : process(CLK_1Hz, RST)
begin
    if RST = '1' then
        Timer_Counter <= (others => '0');
        Timer_Active <= '0';
        LED <= '0';
    elsif rising_edge(CLK_1Hz) then
        if START = '1' and Timer_Active = '0' then
            Timer_Active <= '1';
            Timer_Counter <= to_unsigned(7200, 24); -- Set counter for 2
hours
        elsif Timer_Active = '1' then
            if Timer_Counter = 0 then
                LED <= '1'; -- Turn on LED when time expires
                Timer_Active <= '0'; -- Stop timer
            else
                Timer_Counter <= Timer_Counter - 1;
            end if;
        end if;
    end if;
end process Timer_Logic;
end Behavioral;
```

### **3. Conclusion**

In this lab, we used VHDL language to construct an 8-input AND Gate, 2-input-1-output XOR gate and a down counter, then we loaded these codes to a development board to verify the function. From the lab, we know:

- 1) The basic syntax and structure of VHDL language.
- 2) The way to connect a development board to the computer using Vivado.
- 3) How to realize a down counter step by step.