EIE2810 Digital Systems Design Laboratory

Laboratory Report #6

Name: 关嘉俊

Student ID: 116010060

Date: April 26th

The Chinese University of Hong Kong, Shenzhen

- Experiment A: e.g. Follow the instruction and learn to programme Xlinx FPGA for digital logic functions. The goal is to achieve the 7 inputs AND Gate and then create the XOR Gate.
- Experiment B: e.g. First tried to use the create the specific frequencies, which are 8Hz, 4Hz, 2Hz and 1Hz respectively, and show it on the led light.
- Experiment C: e.g. Display the two digit decimal number with 8 digit BCD code, and convert it on the 7-segment display simultaneously by flashing the digit.
- Experiment D: e.g. Complete the counter by setting the frequency of each four digit. It is requited to count down and make it stop at 0

I. Experiment A

(a) Design:

To construct the 7 inputs AND Gate, that is, it is required for 7 ports for the input and one port for the output. Set it in the entity, next we just need to figure out the logic. Similar way could be applied to build the XOR Gate. The codes of two programs are shown as followed: (The right one is the XOR gate)

```
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;
```

```
entity MyAndGate is
    Port(A: in STD_LOGIC_VECTOR;
        B: in STD_LOGIC_VETOR;
        Y: out STD_LOGIC);
end MyAndGate;

Architecture Behavioral of MyAndGate is

begin
    Y<= A XOR B;

end Behavioral;
```

(b) Results:

After programming, next we run the simulation with the software and set the I/O planning. After setting, we conduct synthesis and implementation and pass the data to the hardware through USB. The experimental results are as followed:

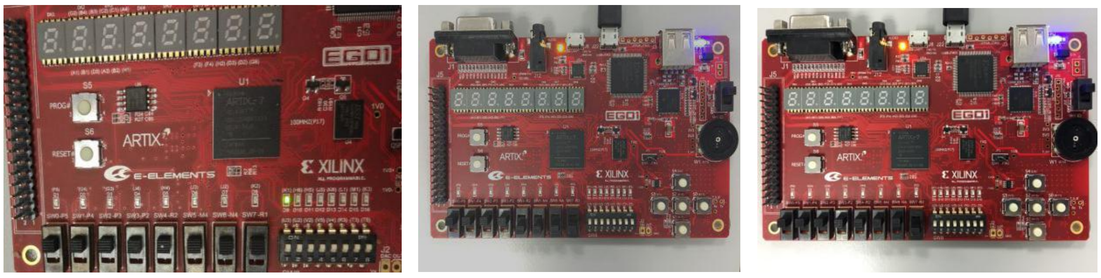


Figure 1 The led is on with all 7 switches on, off with not all 7 switches on

And the XOR Gate test results are:

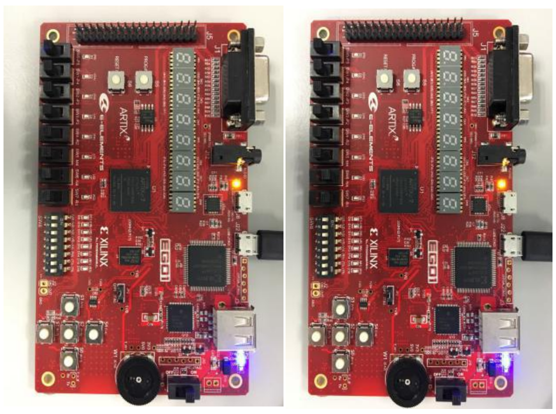


Figure 2 ‘0’’0’ and ‘1’’1’ have low voltage input

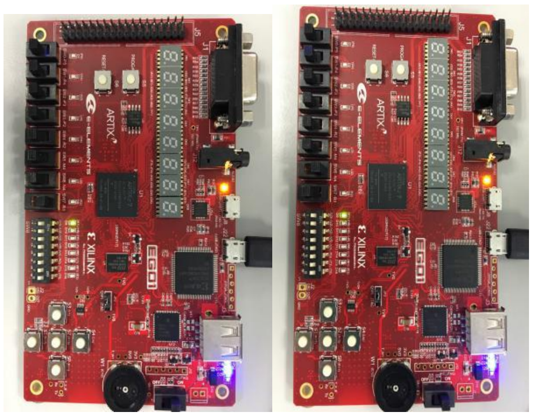


Figure 3 ‘0’and‘1’have high voltage input

II. Experiment B

(a) Design:

In this experiment, we should create the frequency 8Hz, 4Hz, 2Hz and 1Hz in 4 led lights. To achieve it, we need to apply the 100Mhz clock in the development board. We can calculate that, when $6.25*10^6$ rising edge cross, (It is calculated by 100M/8/2) the 8Hz-led would change its state, when $1.25*10^7$ rising edges cross, the 4Hz-led would change its state. As $2.5*10^7$ rising edges cross, the 2Hz-led would change its state. As $5*10^7$ rising edge cross, the 1Hz-led would change its state. Define the ports of inputs and outputs, set one input in the entity as the clock for P17. Define the output pin to the led D9 to D12 for different frequencies individually, the VHDL codes are shown as followed:

```
Library ieee;
        use ieee.std_logic_1164.all;
        use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity ShowNumber is
```

```
port (
        clk : in std_logic;
        eight_Hz : out std_logic;
        four_Hz : out std_logic;
        two_Hz : out std_logic;
        one_Hz : out std_logic
        );
end ShowNumber;


architecture Behavioral of ShowNumber is

Begin
variable eight_count: integer:= 0;
variable eight_count: integer:= 0;
variable eight_count: integer:= 0;
variable eight_count: integer:= 0;
variable eight_count: integer:= 0;

process(clk)
begin
        if rising_edge(clk) then
                eight_count := eight_count + 1;
                four_count := four_count + 1;
                two_count := two_count + 1;
                one_count := one_count + 1;
                change_count := change_count + 1;


        if (eight_count >= 6250000) then
                eight_Hz <= '1';
                if (eight_count = 12500000) then
                        eight_Hz <= '0';
                        eight_count :=0;
                end if;
        end if;
        if (four_count >= 12500000) then
                four_Hz <= '1';
                if (four_count = 25000000) then
                        four_Hz <= '0';
                        four_count :=0;
                end if;
        end if;


        if (two_count >= 25000000) then
                two_Hz <= '1';
                if (two_count = 50000000) then
                        two_Hz <= '0';
                        two_count :=0;
                end if;
        end if;


        if (one_count >= 50000000) then
                one_Hz <= '1';
                if (one_count = 100000000) then
                        one_Hz <= '0';
                        one_count :=0;
                end if;
        end if;
        end process;
end Behavioral;
```

I/O Connection:

Clk = P17
eight_Hz = K1
four_Hz = H6
two_HZ = H5
one_Hz = J6

(c) Result:

The testing results are shown as followed:

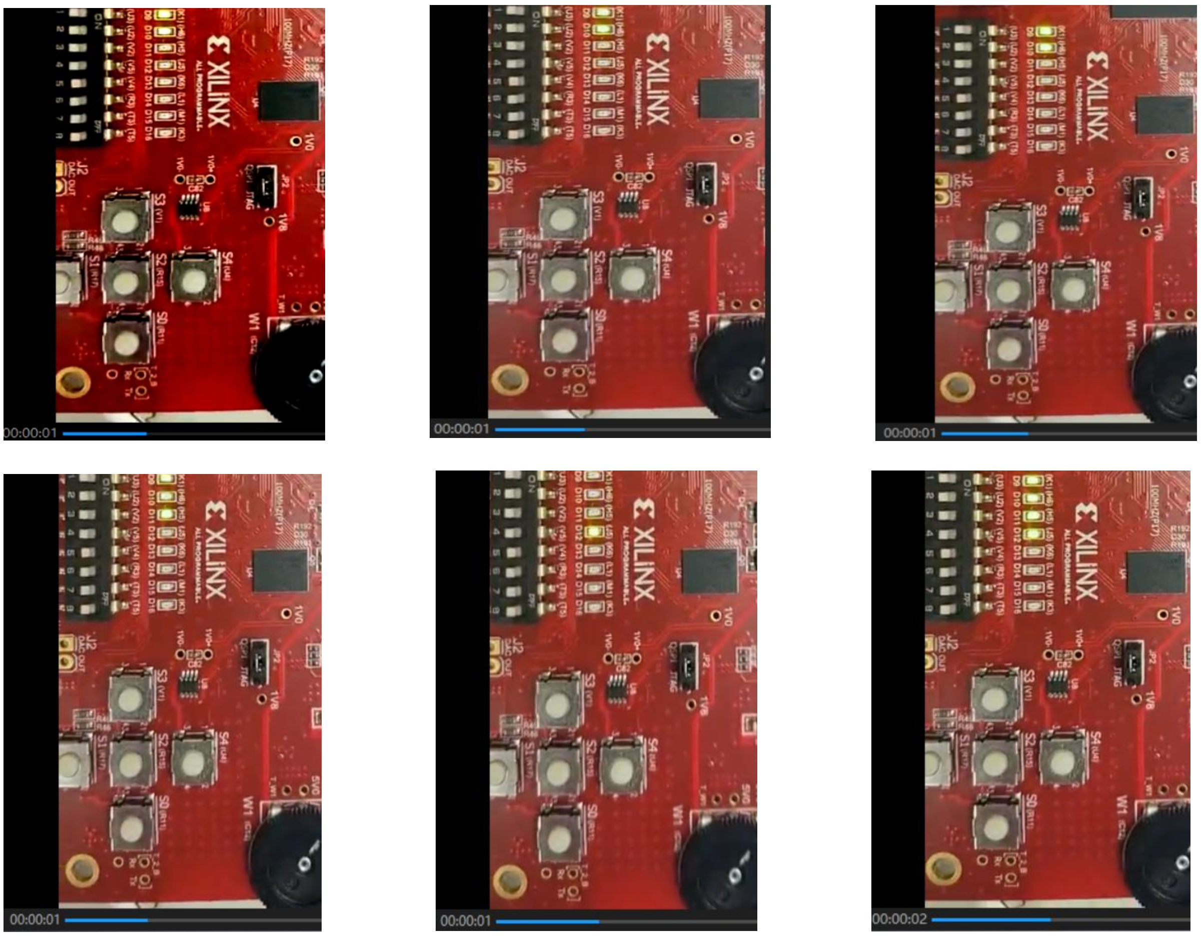


Figure process: 0001-->0010-->-->0011-->0100-->.....-->1000-->1001

From the screenshot of the video we can see that, from 0001 to 1001, the time changing is 1s accurately. That is, the D12 led shine with 1Hz in this experiment. And we can count that, D9,D10,D11 flash for 8 times, 4 times and two times in one second, which indicated their frequency is 8Hz, 4Hz and 2Hz respectively. Thus we have solved the frequency problem.

II. Experiment C

(a) Design:

In this experiment we are required to set the BCD codes of 2 digits (from decimals 99 to 0) by the switches SW0-SW7. SW0 and SW3 are the MSB and LSB of the 10s digit, respectively. SW4 and SW7 are the MSB and LSB of the 1s digit, respectively. Any BCD code exceeding 0b1001 should be regarded as illegal, and then will be regarded as 0b1001.

The requirement above are easy to achieve by setting the judgement of input. In this experiment we define the user-input are A_0 for the single digit and A_1 for the decade digit. And we creating the signal BCD_A0 and signal BCD_A1 (std_logic_vector) to make it display 9 when the input has been exceeding. Furthermore, the ShowLED0 is to display the single digit in the LED D5 to D8 , ShowLED1 display the LED D1 to D4.

To show both the two digits onto two 7-segment, we first define the output vector for SEGMENT We can iteratively show different values of A~G numbers onto the 1st and 2nd 7-segments, and use DN0_K1="1" or DN0_K0="1" to switch from them at a frequency that your eyes can not identify. Thus we set the variable change_count for creating the frequency for 0.1HZ in another signal con_dition for flashing the K1 and K2. The code are shown as followed:

```
Library ieee;
    use ieee.std_logic_1164.all;
    use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity ShowNumber is
    port (
        clk : in std_logic;
        A_0 : in std_logic_vector(3 downto 0);
        A_1 : in std_logic_vector(3 downto 0);
        S4 : in std_logic;
        SEL_0 : out std_logic;
        SEL_1 : out std_logic;
        SEGMENT : out std_logic_vector(6 downto 0);
        ShowLEDA0: out std_logic_vector(3 downto 0);
        ShowLEDA1: out std_logic_vector(3 downto 0);
        );
end ShowNumber;


architecture Behavioral of ShowNumber is
    signal BCD_A0 : std_logic_vector(3 downto 0) := "0000";
    signal BCD_A1 : std_logic_vector(3 downto 0) := "0000";
    signal con_dition : std_logic;
    signal LED_A0 :    std_logic_vector(3 downto 0) :='0000';
    signal LED_A1 :    std_logic_vector(3 downto 0) :='0000';
```

```
Begin
        variable change_count: integer:= 0;

        process(clk)
        Begin
        if rising_edge(clk) then
                change_count := change_count + 1;

                if (change_count >= 1000) then
                        con_dition <= '1';
                        if (change_count = 2000) then
                                con_dition <= '0';
                                change_count :=0 ;
                        end if;
                end if;

                        if (A_0 > "1001") then
                                LED_A0 <= "1001";
                        else
                                LED_A0 <= A_0;
                        end if;

                        if (A_1 > "1001") then
                                LED_A1 <= "1001";
                        else
                                LED_A1 <= A_1;
                        end if;

                        ShowLEDA1 <= LED_A1;
                        ShowLEDA0 <= LED_A0;

if(con_dition = '0') then
SEL_0 <= '1';
SEL_1 <= '0';
if (S4 = '1') then
        BCD_A0 <= LED_A0;
        case BCD_A0 is
                when "0000" => SEGMENT <= "1111110";
                when "0001" => SEGMENT <= "0110000";
                when "0010" => SEGMENT <= "1101101";
                when "0011" => SEGMENT <= "1111001";
                when "0100" => SEGMENT <= "0110011";
                when "0101" => SEGMENT <= "1011011";
                when "0110" => SEGMENT <= "1011111";
                when "0111" => SEGMENT <= "1110000";
                when "1000" => SEGMENT <= "1111111";
                when "1001" => SEGMENT <= "1111011";
                when others => SEGMENT <= "1111011";
        end case;
end if;
end if;

if (con_dition='1') then
SEL_0 <= '0';
SEL_1 <= '1';

if (S4 = '1') then
        BCD_A1 <= LED_A1;
        case BCD_A1 is
                when "0000" => SEGMENT <= "0000000";
                when "0001" => SEGMENT <= "0110000";
                when "0010" => SEGMENT <= "1101101";
                when "0011" => SEGMENT <= "1111001";
                when "0100" => SEGMENT <= "0110011";
                when "0101" => SEGMENT <= "1011011";
                when "0110" => SEGMENT <= "1011111";
                when "0111" => SEGMENT <= "1110000";
                when "1000" => SEGMENT <= "1111111";
                when "1001" => SEGMENT <= "1111011";
                when others => SEGMENT <= "1111011";
        end case;
end if;
end if;
end process;
end Behavioral;
```

The I/O port planning are as followed:
A_0(3)= P5; A_0(2)= P4; A_0(1) = P3; A_0(0) = P2;
A_1(3)= R2; A_1(2) = M4; A_1(1) = N4; A_1(0) = R1;
Clk = P17; SEL_0 = C2, SEL_1 = G2; S4= U4
SEGMENT(6) (5) (4) (3) (2) (1) (0) are B4 A4 A3 B1 A1 B3 B2 respectively
ShowLEDA0(3) (2) (1) (0) are H4 J3 H2 K2 , ShowLEDA1(3) (2) (1) (0) are F6 G4 G3 J4.

(b) Result:

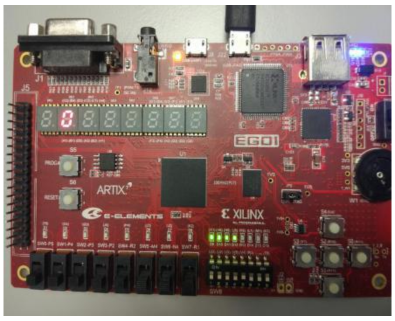


Figure 4 00000000 display 0

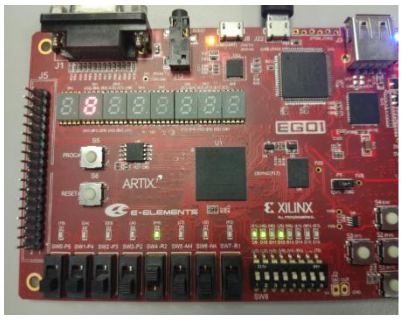


Figure 5 00001000 display 8

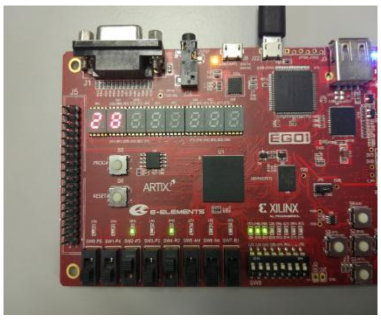


Figure 6 00101000 display 28

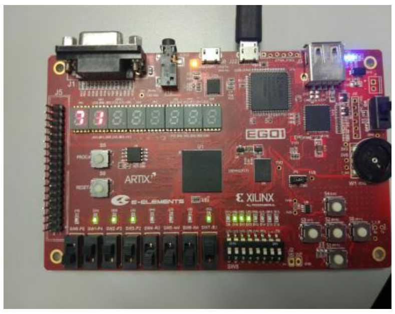


Figure 7 01110001 display 71

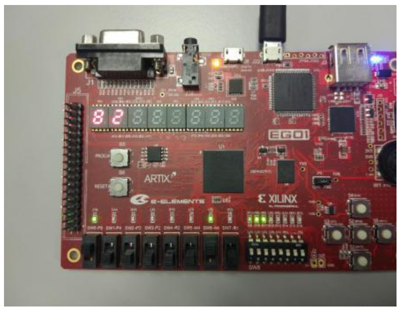


Figure 8 10000010 display 82

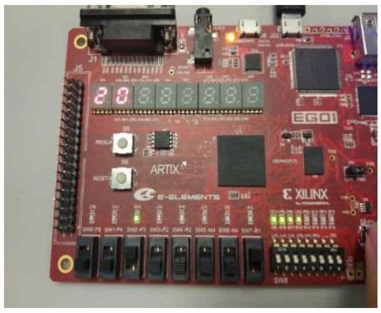


Figure 9 00100000 display 20

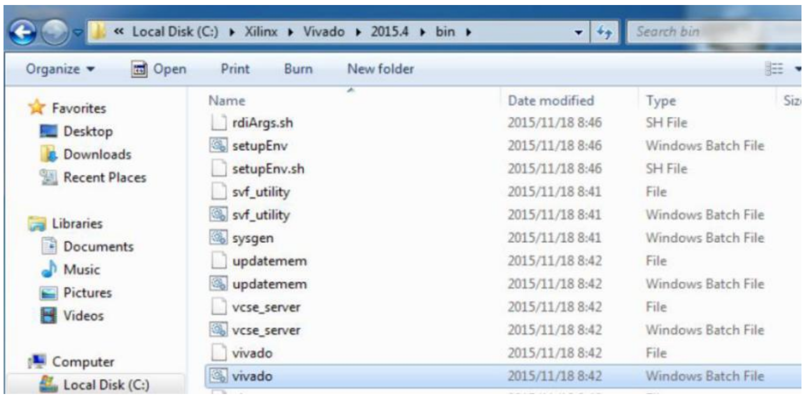
After the hardware demonstration, the display function of the board has been accomplished. Also, we can discover that there would be double image if the switchover of the DN0K1 and DN0K2 is under 10000Hz or higher than 10MHz. Next step we would finish the down-counter.

Experiment C

(a) Design:

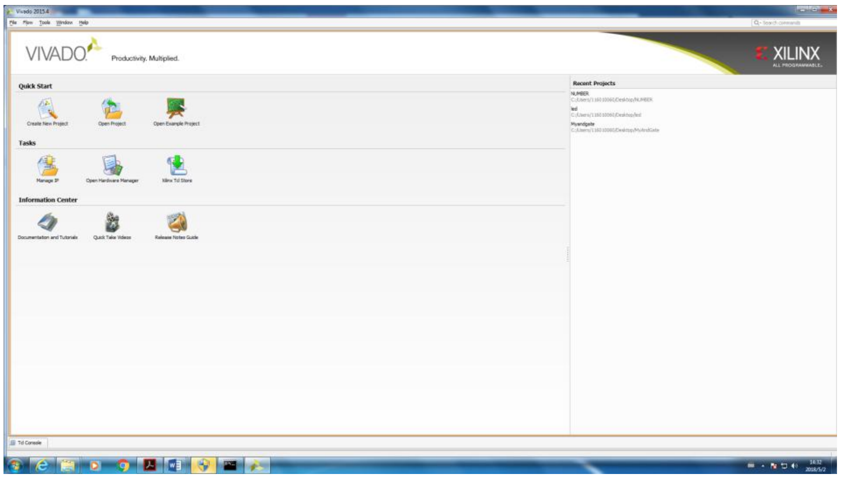
In this experiment, we show the detailed step as followed for the instruction:

1. Double click the Windows batch file "vivado" at the directory C:\Xilinx\Vivado\2015.4\bin

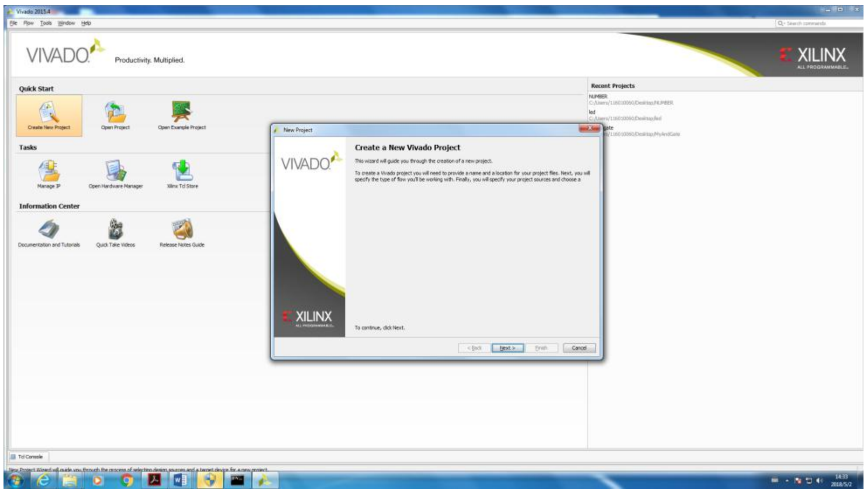


Name	Date modified	Type	Siz
rdiArgs.sh	2015/11/18 8:46	SH File	
setupEnv	2015/11/18 8:46	Windows Batch File	
setupEnv.sh	2015/11/18 8:46	SH File	
svf_utility	2015/11/18 8:41	File	
svf_utility	2015/11/18 8:41	Windows Batch File	
sysgen	2015/11/18 8:41	Windows Batch File	
updatemem	2015/11/18 8:42	File	
updatemem	2015/11/18 8:42	Windows Batch File	
vcse_server	2015/11/18 8:42	File	
vcse_server	2015/11/18 8:42	Windows Batch File	
vivado	2015/11/18 8:42	File	
vivado	2015/11/18 8:42	Windows Batch File	

2. Few seconds later the Gui would be launched. Double Click the “Create New Project” on the top left corner to create the new project.



3. Click “Next” as the window pop up ==> input the name of the project “Counting”==>keep the default items ==> “Next”

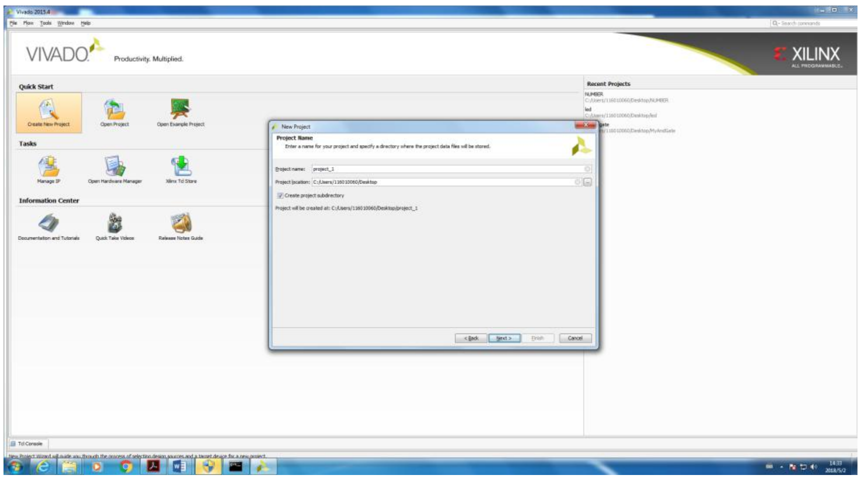


Create a New Vivado Project

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

To continue, click Next.



Project Name

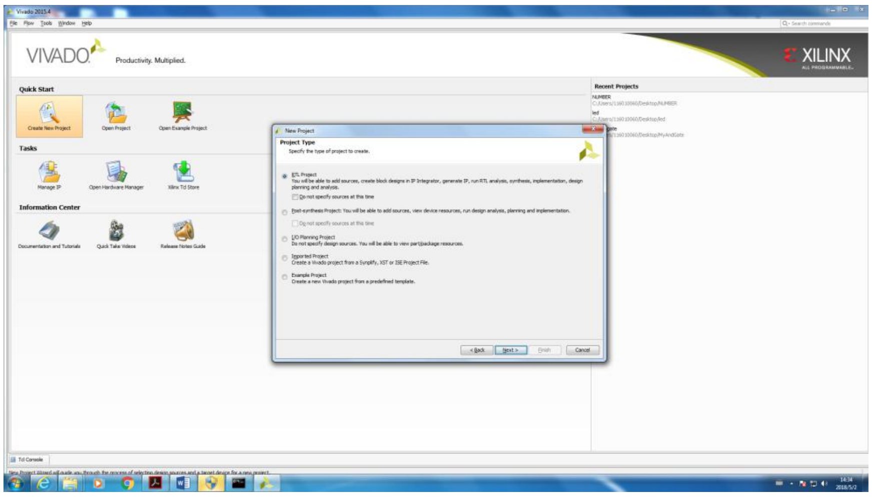
Enter a name for your project and specify a directory where the project data files will be stored.

Project name: project_1

Project location: C:/Users/116010060/Desktop

[✓] Create project subdirectory

Project will be created at: C:/Users/116010060/Desktop/project_1

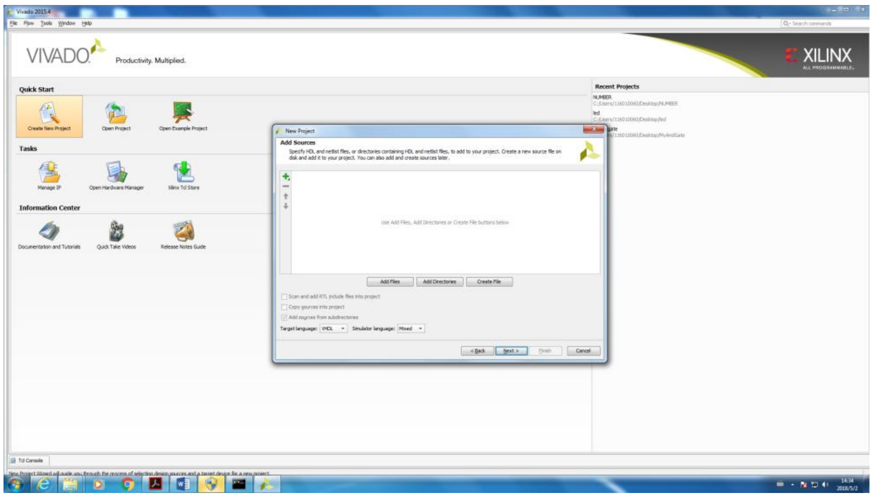

New Project

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.

4. If the previous process are correct, In the "Add Existing IP(optional)",click "Next"


New Project

Add Sources

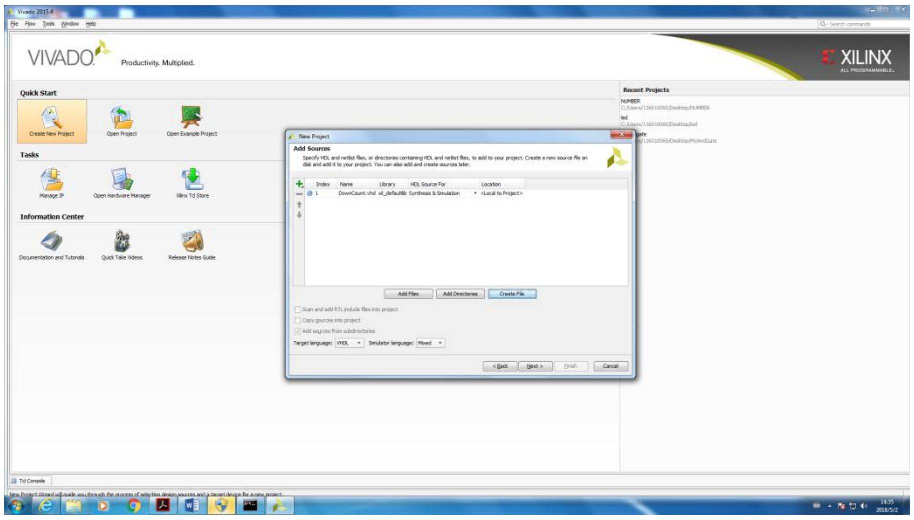
Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

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

5. On the bottom left corner, first change the "Target Language" to "VHDL"==> click the "Create File"==>Input the file name "Counting"==>Click Next. The window is shown here


New Project

Add Sources

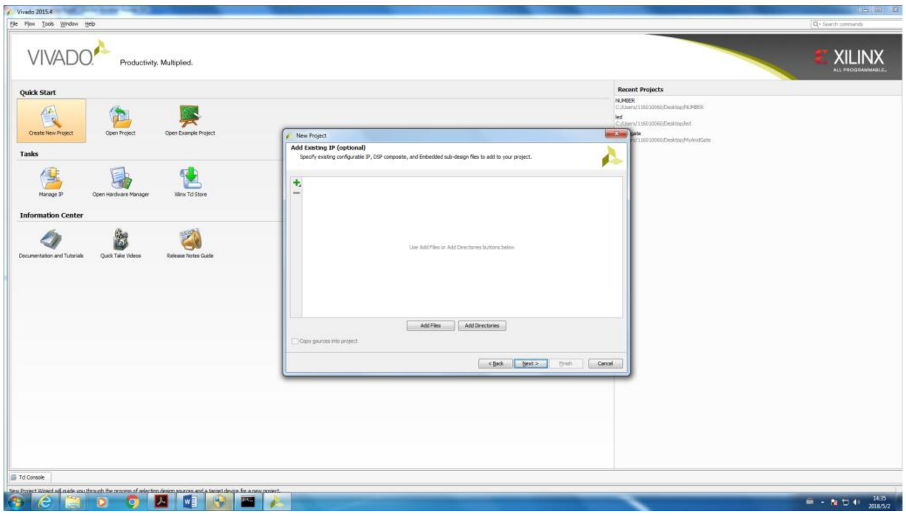
Specify HDL and netlist files, or directories containing HDL and netlist files, to add to your project. Create a new source file on disk and add it to your project. You can also add and create sources later.

Index	Name	Library	HDL Source For	Location
1	DownCount.vhd	xil_defaultlib	Synthesis & Simulation	<Local to Project>

- [ ] Scan and add RTL include files into project
- [ ] Copy sources into project
- [✓] Add sources from subdirectories

Target language: VHDL Simulator language: Mixed

6. Click “Next” in the “Adding Existing IP”

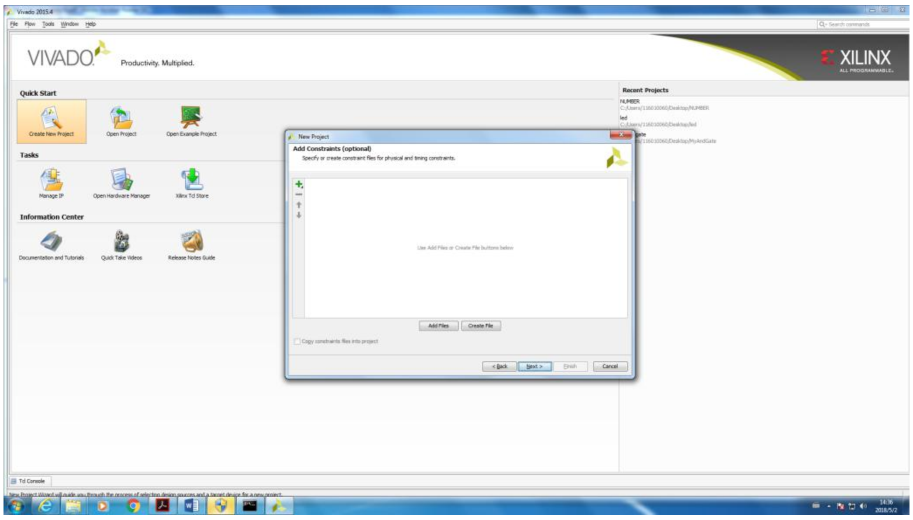

New Project

Add Existing IP (optional)

Specify existing configurable IP, DSP composite, and Embedded sub-design files to add to your project.

Use Add Files or Add Directories buttons below

7. Click “Next” in the “Add Constrain”

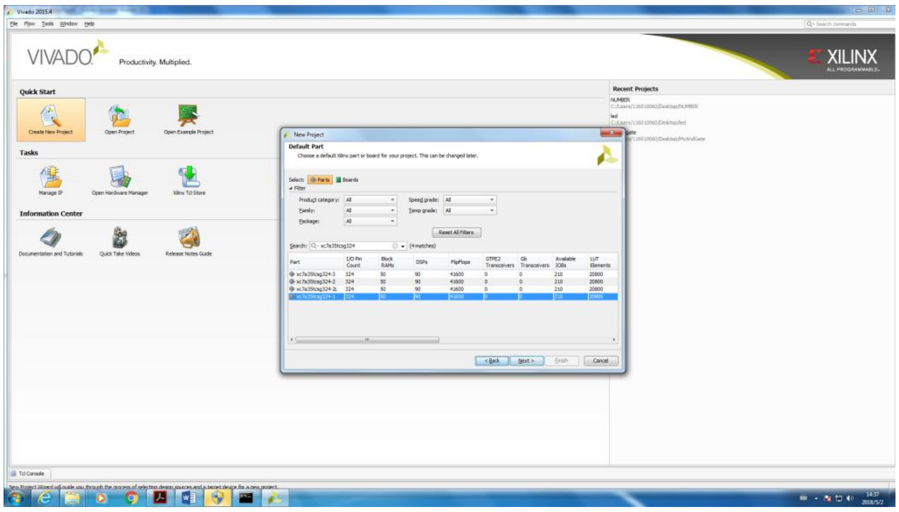

New Project

Add Constraints (optional)

Specify or create constraint files for physical and timing constraints.

Use Add Files or Create File buttons below

8. Input the chip type by searching “xc7a35txsg324-1==>Click “Next” ==> “Finish”


New Project

Default Part

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

Select: Parts Boards

Product category: All Speed grade: All

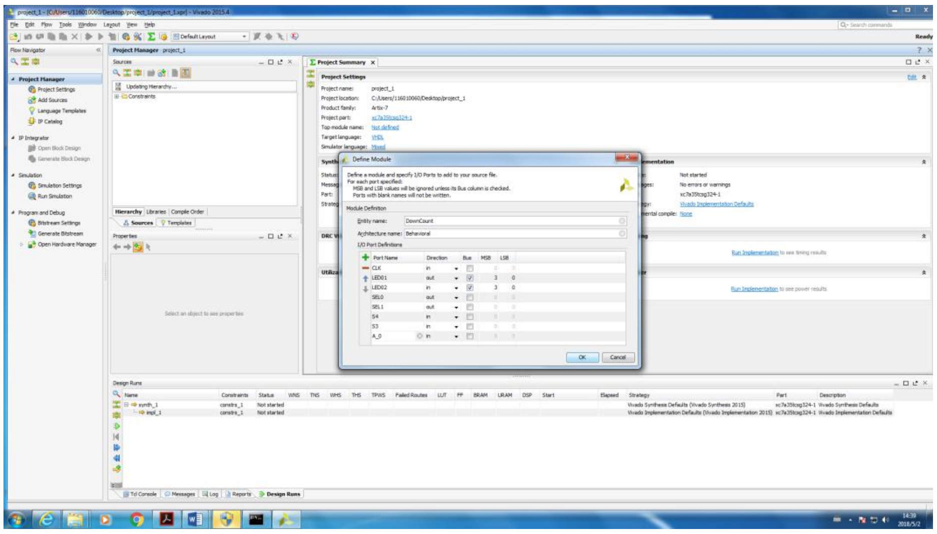
Family: All Temp grade: All

Package: All

Search: xc7a35tcsg324 (4 matches)

Part	I/O Pin Count	Block RAMs	DSPs	FlipFlops	GTPE2 Transceivers	Gb Transceivers	Available IOBs	LUT Elements
xc7a35tcsg324-3	324	50	90	41600	0	0	210	20800
xc7a35tcsg324-2	324	50	90	41600	0	0	210	20800
xc7a35tcsg324-2L	324	50	90	41600	0	0	210	20800
xc7a35tcsg324-1	324	50	90	41600	0	0	210	20800

9. In the “Define Module” window, input the port names, direction, bus choice, MSB and LSB as below. After checking and click “OK”.



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

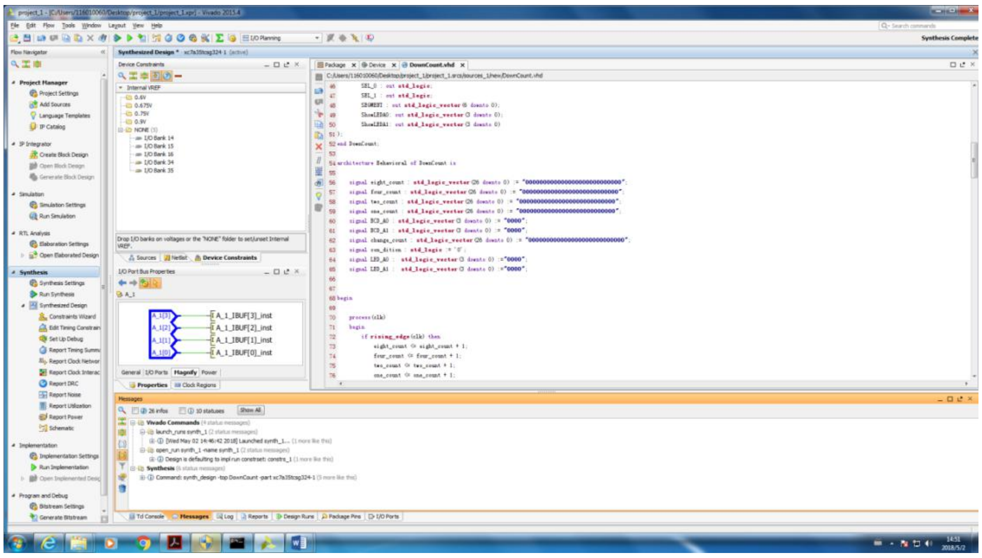
Entity name: DownCount

Architecture name: Behavioral

I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
CLK	in	☐	0	0
LED01	out	✓	3	0
LED02	in	✓	3	0
SEL0	out	☐	0	0
SEL1	out	☐	0	0
S4	in	☐	0	0
S3	in	☐	0	0
A_0	in	☐	0	0

10. In the “Design source” panel, double click the “Counting”, the port information has been generated and shown in the “Counting.vhd” window.



C:/Users/116010060/Desktop/project_1/project_1.srcs/sources_1/new/DownCount.vhd

```
        SEL_0 : out std_logic;
        SEL_1 : out std_logic;
        SEGMENT : out std_logic_vector(6 downto 0);
        ShowLEDA0: out std_logic_vector(3 downto 0);
        ShowLEDA1: out std_logic_vector(3 downto 0)
);
end DownCount;

architecture Behavioral of DownCount is

    signal eight_count : std_logic_vector(26 downto 0) := "000000000000000000000000000";
    signal four_count : std_logic_vector(26 downto 0) := "000000000000000000000000000";
    signal two_count : std_logic_vector(26 downto 0) := "000000000000000000000000000";
    signal one_count : std_logic_vector(26 downto 0) := "000000000000000000000000000";
    signal BCD_A0 : std_logic_vector(3 downto 0) := "0000";
    signal BCD_A1 : std_logic_vector(3 downto 0) := "0000";
    signal change_count : std_logic_vector(26 downto 0) := "000000000000000000000000000";
    signal con_dition : std_logic := '0';
    signal LED_A0 :  std_logic_vector(3 downto 0) :="0000";
    signal LED_A1 :  std_logic_vector(3 downto 0) :="0000";


begin

    process(clk)
    begin
        if rising_edge(clk) then
            eight_count <= eight_count + 1;
            four_count <= four_count + 1;
            two_count <= two_count + 1;
            one_count <= one_count + 1;
```

11. The next step is to input the programme code and debug until there are no errors warning. To construct the counting, we first copy the digit showing in the last experiment for showing two numbers in the seven segment. The Only difference is we create a new port S3 for changing the value that was counted backward. Furthermore, we apply similar method used in the first experiment to set the counting frequency in 1Hz. Furthermore, the counting can be stopped by pressing S4. The whole structure of the codes are shown as followed:

1

```
Library ieee;
use ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ShowNumber is
port (
clk : in std_logic;
A_0 : in std_logic_vector(3 downto 0);
A_1 : in std_logic_vector(3 downto 0);
S4 : in std_logic;
S3 : in std_logic;
SEL_0 : out std_logic;
SEL_1 : out std_logic;
one_Hz : out std_logic;
SEGMENT : out std_logic_vector(6 downto 0);
ShowLEDA0: out std_logic_vector(3 downto 0);
ShowLEDA1: out std_logic_vector(3 downto 0);
);
end ShowNumber;

architecture Behavioral of ShowNumber is

signal BCD_A0 : std_logic_vector(3 downto 0) := "0000";
signal BCD_A1 : std_logic_vector(3 downto 0) := "0000";
signal con_dition : std_logic := '0'
signal LED_A0 : std_logic_vector(3 downto 0) :="0000";
signal LED_A1 : std_logic_vector(3 downto 0) :="0000";
```

2

```
Begin
variable change_count: integer:= 0;
variable one_count:interger:=0;

process(clk)
begin
if rising_edge(clk) then
    one_count := one_count + 1;
    change_count := change_count + 1;

    if (change_count >= 10000) then
        con_dition <= '1';
        if (change_count = 20000) then
            con_dition <= '0';
            change_count :=0 ;
        end if;
    end if;
     if (one_count >= 50000000) then
        one_Hz <= '1';
        if (one_count = 100000000) then
            one_Hz <= '0';
            one_count :=0 ;
        end if;
    end if;
```

/** The program above aimed at create two specific frequency 10000Hz and 1Hz, 10000Hz is used for switching the DN0_K1 and DN0_K2, and 1Hz is used for the decreasing of the number**/

```
        if (S3 = '0') then
                if (A_0 > "1001") then
                        LED_A0 <= "1001";
                else
                        LED_A0 <= A_0;
                end if;

                if (A_1 > "1001") then
                        LED_A1 <= "1001";
                else
                        LED_A1 <= A_1;
                end if;
        end if;
        ShowLEDA1 <= LED_A1;
        ShowLEDA0 <= LED_A0;

        if (S3 = '1') then
                if (one_Hz='1') then
                        if    (LED_A0 = 0) then
                                if (LED_A1 <='0000') then
                                        LED_A0 <='0000';
                                else
                                        LED_A0 <=”1001”;
                                        LED_A1 <= LED_A1-1;
                                end if;
                        else
                                LED_A0 <= LED_A0-1;
                        end if;
                end if;
        end if;
if (con_dition='1') then
SEL_0 <= '0';
SEL_1 <= '1';
```

3

4

/** The code in the left is using the switch of S3 to control the working situation of the counter. When the S3 button is pressed, the input data which is controlled by eight switches SW0 to SW8 would be assigned to LED_A0 and LED_A1**/

```
if(con_dition = '0') then
SEL_0 <= '1';
SEL_1 <= '0';

if (S4 = '0') then
BCD_A0 <= LED_A0;
        case BCD_A0 is
        when    "0000"    =>    SEGMENT    <=
"1111110";
        when    "0001"    =>    SEGMENT    <=
"0110000";
        when    "0010"    =>    SEGMENT    <=
"1101101";
        when    "0011"    =>    SEGMENT    <=
"1111001";
        when    "0100"    =>    SEGMENT    <=
"0110011";
        when    "0101"    =>    SEGMENT    <=
"1011011";
        when    "0110"    =>    SEGMENT    <=
"1011111";
        when    "0111"    =>    SEGMENT    <=
"1110000";
        when    "1000"    =>    SEGMENT    <=
"1111111";
        when    "1001"    =>    SEGMENT    <=
"1111011";
        when    others    =>    SEGMENT    <=
"1111011";
        end case;
end if;
```

```
if (S4 = '0') then
BCD_A1 <= LED_A1;
        case BCD_A1 is
        when "0000" => SEGMENT <= "0000000";
        when "0001" => SEGMENT <= "0110000";
        when "0010" => SEGMENT <= "1101101";
        when "0011" => SEGMENT <= "1111001";
        when "0100" => SEGMENT <= "0110011";
        when "0101" => SEGMENT <= "1011011";
        when "0110" => SEGMENT <= "1011111";
        when "0111" => SEGMENT <= "1110000";
        when "1000" => SEGMENT <= "1111111";
        when "1001" => SEGMENT <= "1111011";
        when others => SEGMENT <= "1111011";
        end case;
end if;
end if;

end if;
end process;
end Behavioral;
```

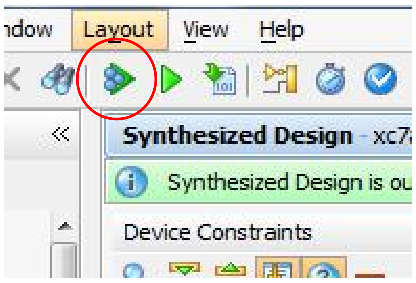
5

The final step is the copy of the last showing experiment, we display the two digit decimal number by using a high frequency switching. The I/O port connection are shown as followed:

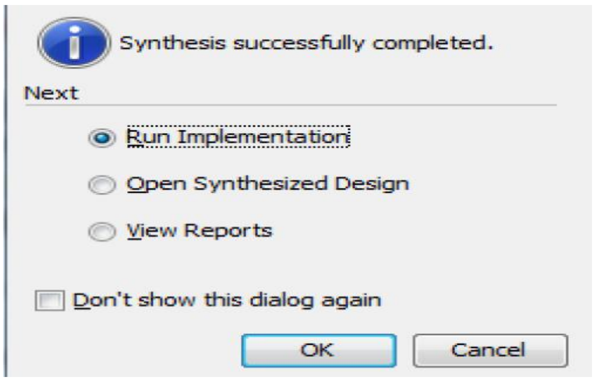
A_0(3) = P5; A_0(2)= P4; A_0(1) = P3;
A_0(0) = P2; A_1(3) = R2; A_1(2) = M4;
A_1(1) = N4; A_1(0) = R1;
Clk = P17; SEL_0 = C2, SEL_1 = G2;
S4= U4; S3= V1;
SEGMENT(6) (5) (4) (3) (2) (1) (0) are B4 A4 A3 B1 A1 B3 B2 respectively
ShowLEDA0(3) (2) (1) (0) are H4 J3 H2 K2 , ShowLEDA1(3) (2)(1)(0) are F6 G4 G3 J4.

Process Continue

12. After typing all the codes, you need to debug unless there is no critical warning in the message box. Next you are supposed to click the icon for “Run Synthesis”. This process would take several minutes.



13. When the Synthesis is finished, the dialogue box below shows that the synthesis is completed. Choose “Open synthesized design” and click “OK”.


Synthesis successfully completed.

Next

[✓ Run Implementation]

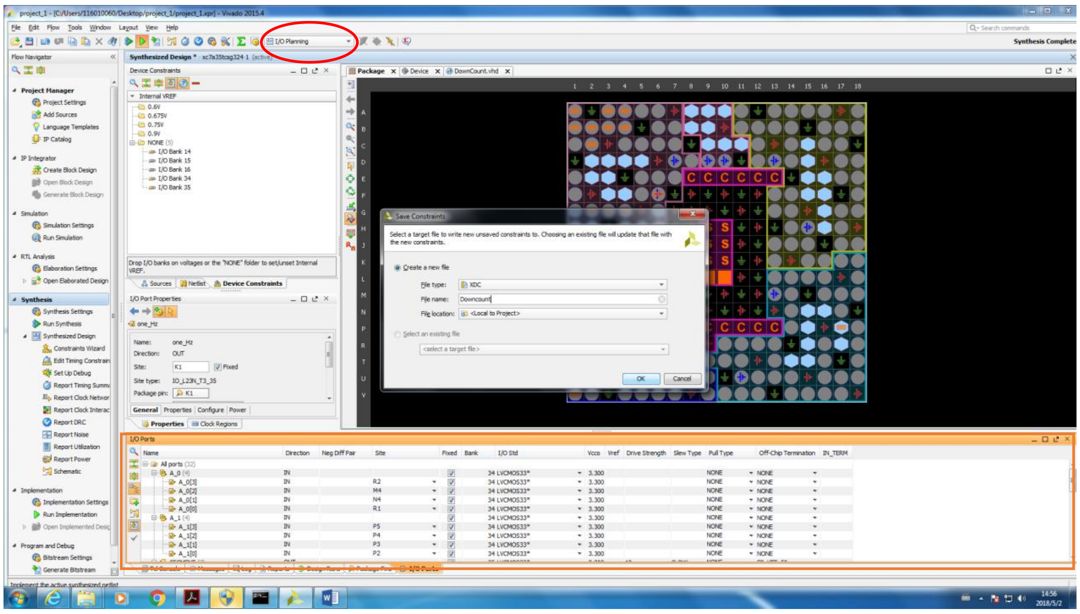
Open Synthesized Design

View Reports

☐ Don't show this dialog again

OK Cancel

14. Select I/O planning in the red cycle options. In the orange box, connect the input and output pin to the switches and LED according the instruction which has been shown at the end of the coding. Remember to select the all the voltage as LVCMOS33*. When you finish all the assignments, save the document by pressing “Ctrl+S”, in the dialog box input your file name.



Select a target file to write new unsaved constraints to. Choosing an existing file will update that file with the new constraints.

(● Create a new file)

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

Select an existing file

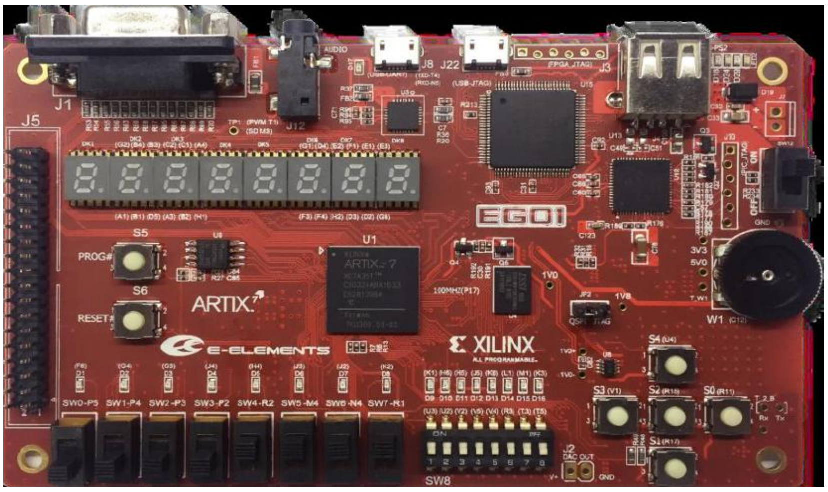
<select a target file>

Name: one_Hz
Direction: OUT
Site: K1 ✓ Fixed
Site type: IO_L23N_T3_35
Package pin: K1

I/O Ports

Name	Direction	Neg Diff Pair	Site	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
All ports (32)													
A_0 (4)	IN			✓	34	LVCMOS33*	3.300				NONE	NONE	
A_0[3]	IN		R2	✓	34	LVCMOS33*	3.300				NONE	NONE	
A_0[2]	IN		M4	✓	34	LVCMOS33*	3.300				NONE	NONE	
A_0[1]	IN		N4	✓	34	LVCMOS33*	3.300				NONE	NONE	
A_0[0]	IN		R1	✓	34	LVCMOS33*	3.300				NONE	NONE	
A_1 (4)	IN			✓	34	LVCMOS33*	3.300				NONE	NONE	
A_1[3]	IN		P5	✓	34	LVCMOS33*	3.300				NONE	NONE	
A_1[2]	IN		P4	✓	34	LVCMOS33*	3.300				NONE	NONE	
A_1[1]	IN		P3	✓	34	LVCMOS33*	3.300				NONE	NONE	
A_1[0]	IN		P2	✓	34	LVCMOS33*	3.300				NONE	NONE	

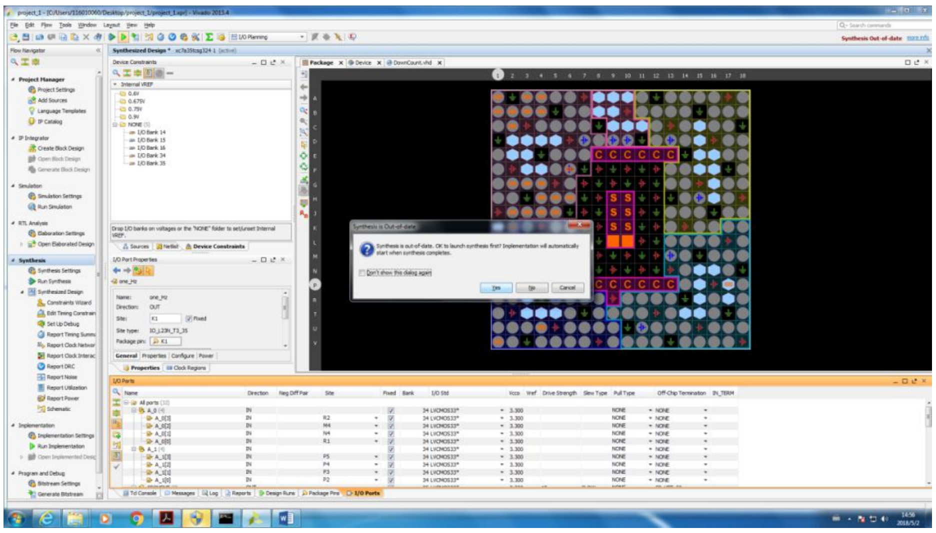
P5, P4, P3, P2, R2, M4, N4, R1, K1, P17, C2, G2, U4, V1, B4, A4, A3, B1, A1, B3, B2, H4, J3, H2, K2, F6, G4, G3, J4 are connected to the I/O of the FPGA chip. You are recommended check the pins’ names by the text on the development board.



Also, you can check all the connection in the “EGO1 Users Manual” shown as follow:

EGO1_User_Manual_v04_20161221.pdf	2018/4/17 15:45	Adobe Acrobat ...	1,982 KB

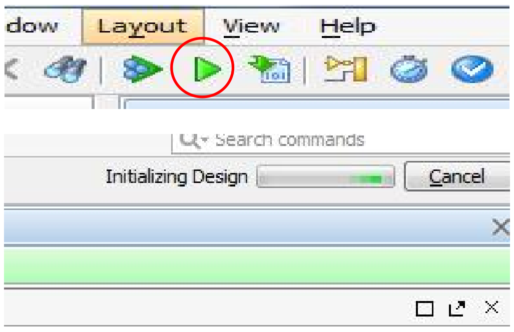
15. If the dialog indicates that the Synthesis has been out of date, you can just click the Yes to skip it.

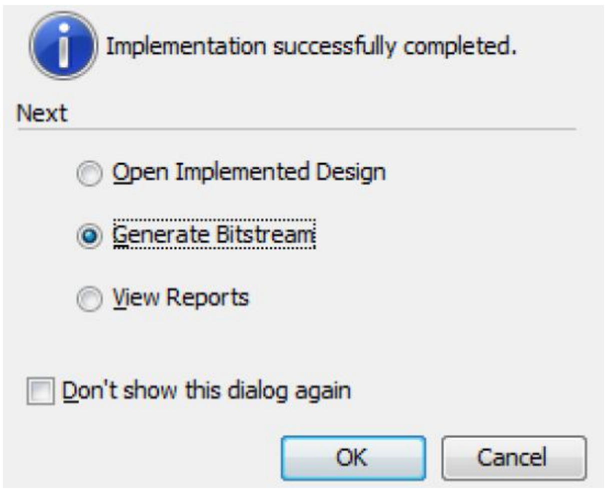

Synthesis is Out-of-date

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

Don't show this dialog again

16. Next run the program by clicking the icon for "Run implementation". The running might take about 2 minutes, You can see the running state at the top Right corner.



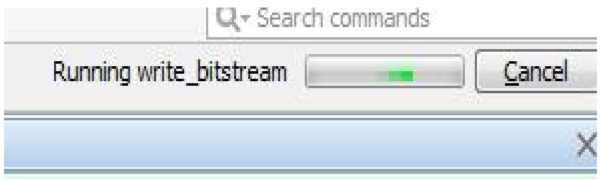

Implementation successfully completed.

Next

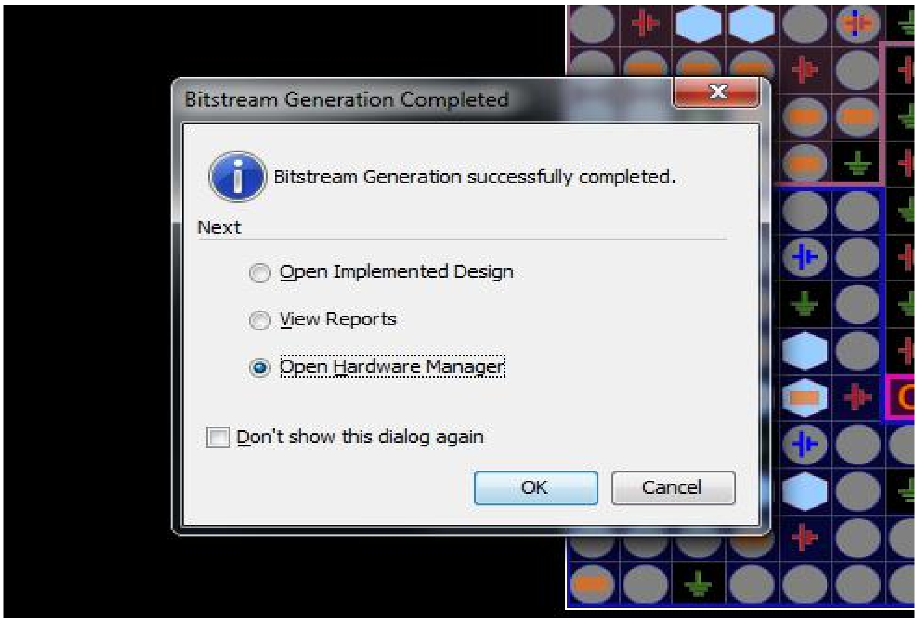
- Open Implemented Design
- [✓ Generate Bitstream]
- View Reports

Don't show this dialog again

17. When the following dialogue box prompts out, choose "Generate Bitstream" and click "OK" . You can also see the Generating state at the top right corner.



18. When the progress is completed, select the open hardware manager in the box and click ok. Next you are required to connect the J22 port of the development board with computer through USB, and then turn on the switch. If the board have proper function, you can see the 7-Segments are all 0, with other 4 LEDs on the upper part of the board on.



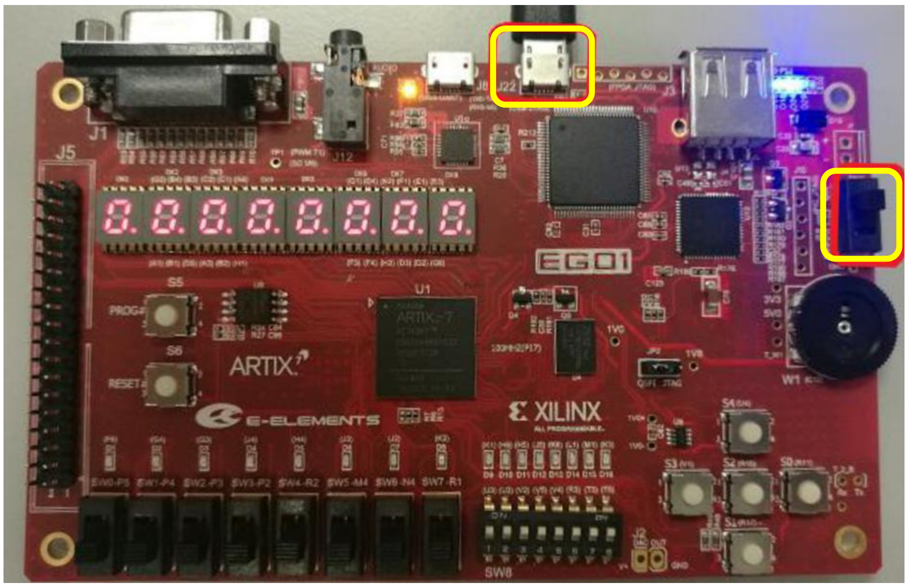
Bitstream Generation Completed

Bitstream Generation successfully completed.

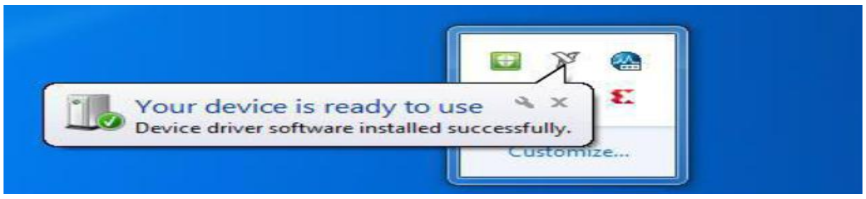
Next

- Open Implemented Design
- View Reports
- [✓ Open Hardware Manager]

Don't show this dialog again

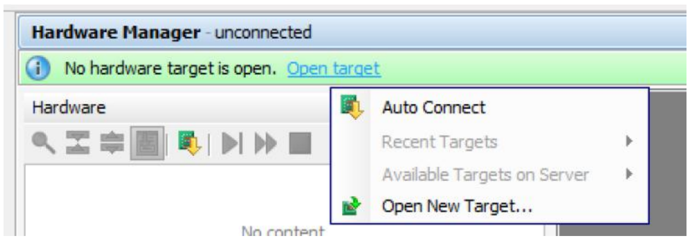


It will take from a few seconds to a few minutes to install the driver. When you see the indicating text below, the drivers are installed successfully.

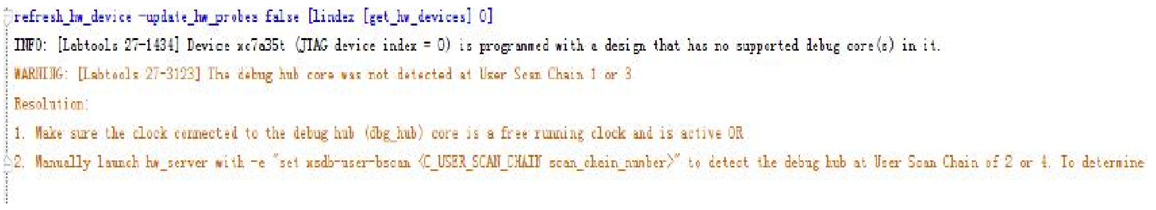


Your device is ready to use
Device driver software installed successfully.

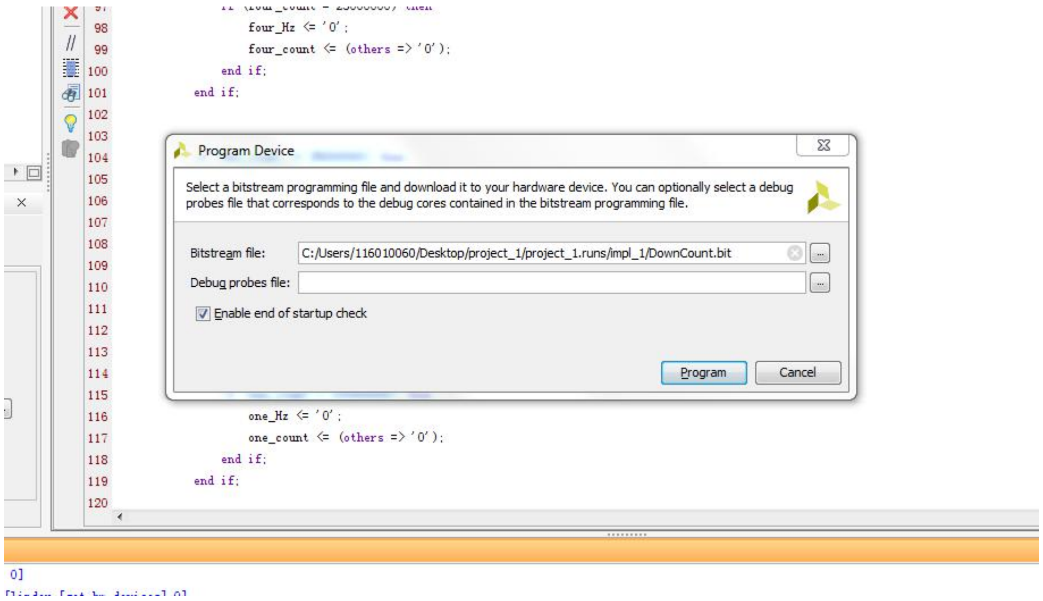
19. After USB connection, click the Auto connect. If the connection work well, you would see these orange information follow:


Hardware Manager - unconnected

No hardware target is open. Open target


```
refresh_hw_device -update_hw_probes false [lindex [get_hw_devices] 0]
INFO: [Labtools 27-1434] Device xc7a35t (JTAG device index = 0) is programmed with a design that has no supported debug core(s) in it.
WARNING: [Labtools 27-3123] The debug hub core was not detected at User Scan Chain 1 or 3.
Resolution:
1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active OR
2. Manually launch hw_server with -e "set xsdb-user-bscan <C_USER_SCAN_CHAIN scan_chain_number>" to detect the debug hub at User Scan Chain of 2 or 4. To determine
```

20.Next you are able to download coding the FPGA by selecting program device. Remain the default items and click the program icon:


```
            four_Hz <= '0';
            four_count <= (others => '0');
        end if;
    end if;
```

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.

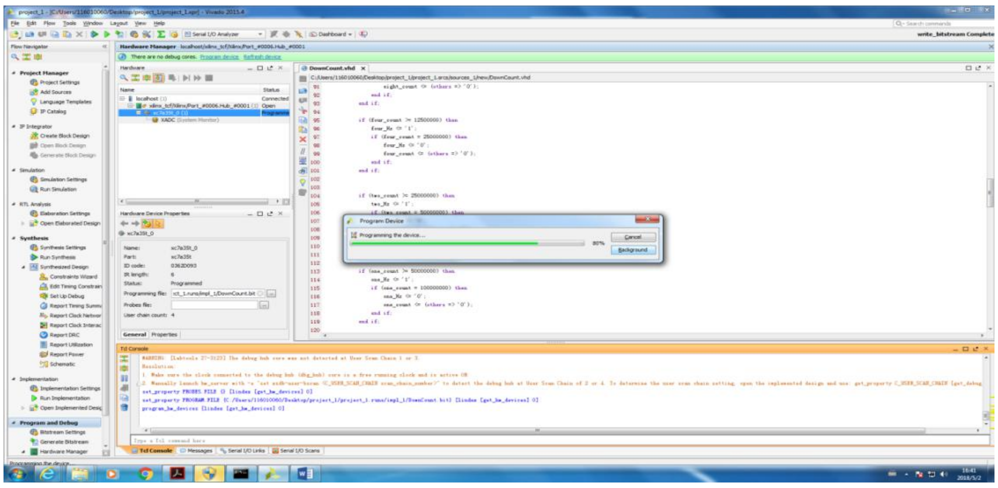
Bitstream file: C:/Users/116010060/Desktop/project_1/project_1.runs/impl_1/DownCount.bit

Debug probes file:

[✓] Enable end of startup check

```
            one_Hz <= '0';
            one_count <= (others => '0');
        end if;
    end if;
```

21.After few seconds, we can see the results of this counting program.


Name: xc7a35t_0
Part: xc7a35t
ID code: 0362D093
IR length: 6
Status: Programmed
Programming file: ct_1.runs/impl_1/DownCount.bit
Probes file:
User chain count: 4

Programming the device... 80%

```
WARNING: [Labtools 27-3123] The debug hub core was not detected at User Scan Chain 1 or 3.
Resolution:
1. Make sure the clock connected to the debug hub (dbg_hub) core is a free running clock and is active OR
2. Manually launch hw_server with -e "set xsdb-user-bscan <C_USER_SCAN_CHAIN scan_chain_number>" to detect the debug hub at User Scan Chain of 2 or 4. To determine the user scan chain setting, open the implemented design and use: get_property C_USER_SCAN_CHAIN [get_debug
set_property PROBES.FILE {} [lindex [get_hw_devices] 0]
set_property PROGRAM.FILE {C:/Users/116010060/Desktop/project_1/project_1.runs/impl_1/DownCount.bit} [lindex [get_hw_devices] 0]
program_hw_devices [lindex [get_hw_devices] 0]
```

22. The expected experiment results are shown as followed, If there exist distinct differences, you can return to check the codes and I/O connection.

Figure 10 The initial setting “01101001” to “69”

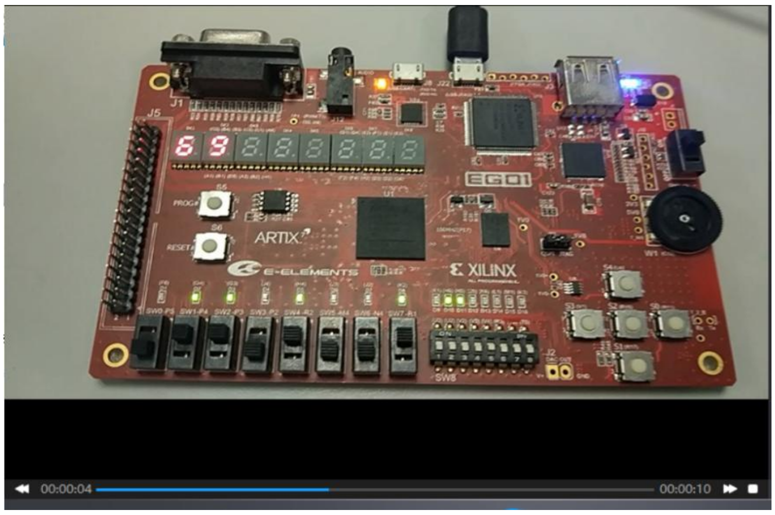
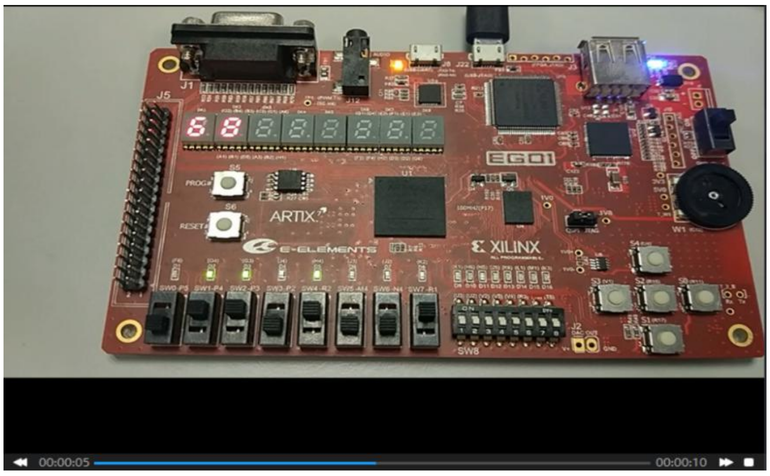


Figure 11 One second later figure



In the counter, we first preset the expected number by controlling the output SW0 to SW7. In this experiment, we preset 0110-1001 for decimal number 69, and then press the S3, the counter would start its function, which count down one second later. Next the key function is the number change from 0 to 9 and finally stay at 0. These are shown as followed in another video.

Figure 12 Counter count down to 11 at 8’s

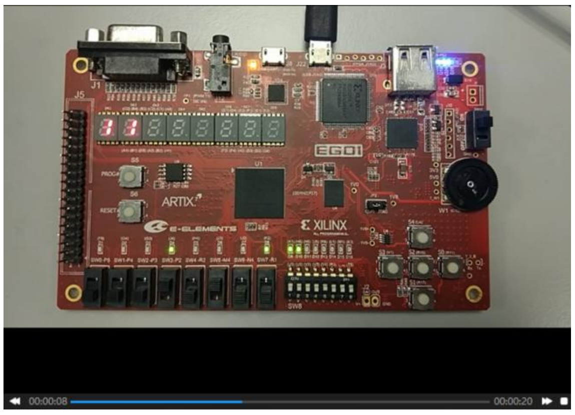


Figure 13 Counter count down to 9 at 9’s

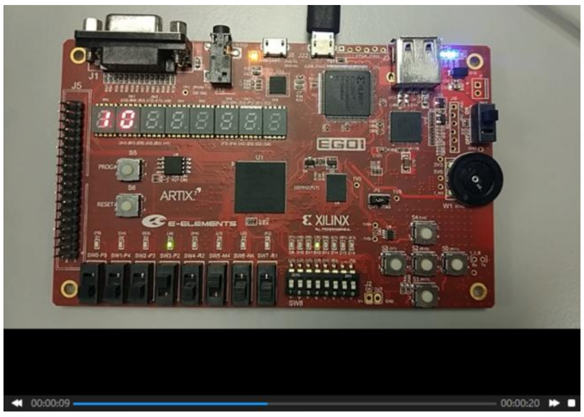


Figure 14 Counter count down to ‘9’ at 10s

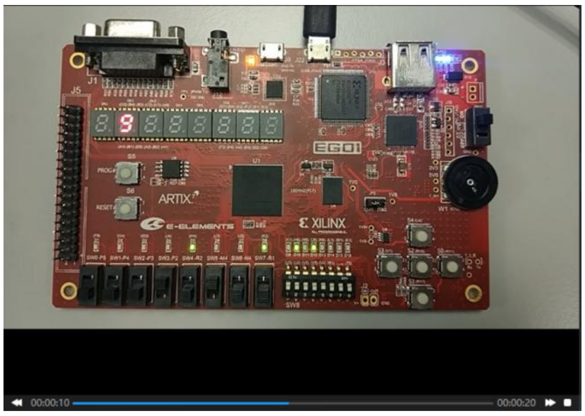


Figure 15 Counter count down to ‘0’ at 19s

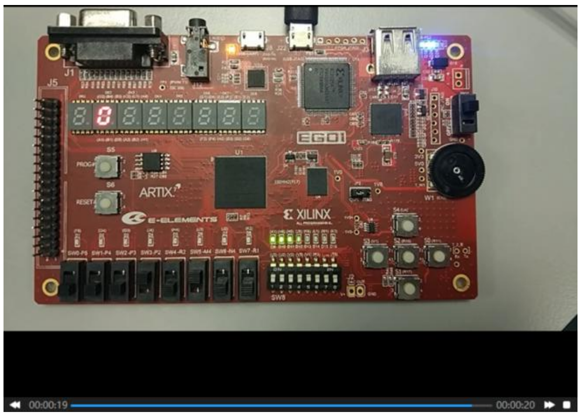


Figure 16 Counter count down to ‘0’ at 20s

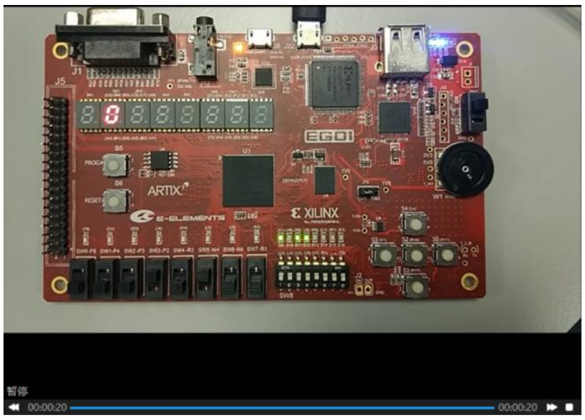
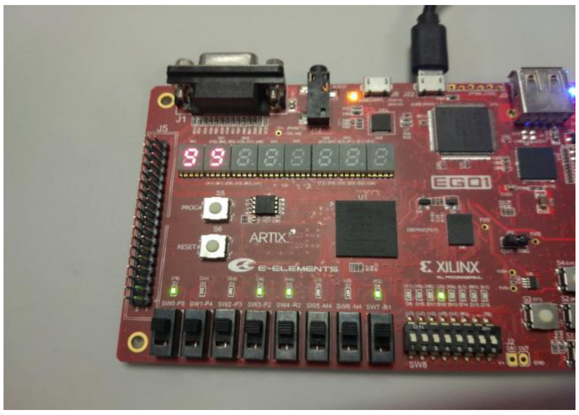
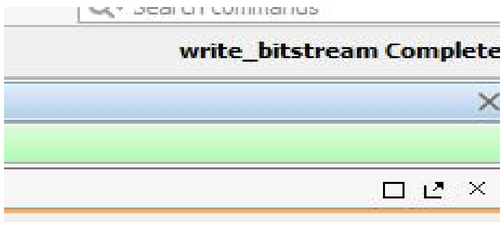
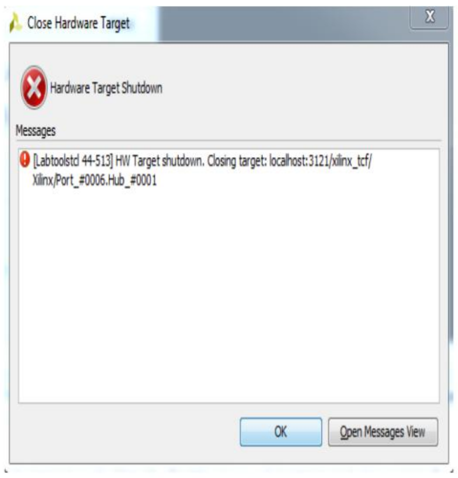


Figure 17 Addition case: 11111111 indicate ‘99’



23. If you finish the test and try to disconnect the board and computer, first close the hardware manager at the top right corner, or you may meet the error below.





Close Hardware Target

Hardware Target Shutdown

Messages

[Labtoolstcl 44-513] HW Target shutdown. Closing target: localhost:3121/xilinx_tcf/Xilinx/Port_#0006.Hub_#0001

Figure 18 Error warning

In this case, follow these steps to recover from the error: close the Hardware Manager => turn off the power switch => disconnect the usb cable => reconnect the usb cable => turn on the power switch => wait for driver installation (about 10 seconds) => run Vivado and get back to the Hardware Manager => click “Open target” and “Auto Connect”. Finally close the Hardware Manager and quit from the Vivado, the experiment has been accomplished.

Question's solution:

The code on the left can theoretically create the count down timert for 2 hours. The port connection are provided as followed:

Clk = P17; SEL_0 = G2, SEL_1 = C2;

SEL_2 = C1; SEL_3 = H1,SEL_4=G1;

S3= V1; one_Hz = F6;

SEGMENT(6) (5) (4) (3) (2) (1) (0) are

B4, A4, A3, B1, A1, B3, B2 respectively

```
Library ieee;
use ieee.std_logic_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity ShowNumber is
port (
clk : in std_logic;
S3 : in std_logic;
SEL_0 : out std_logic;
SEL_1 : out std_logic;
SEL_2 : out std_logic;
SEL_3 : out std_logic;
SEL_4 : out std_logic;
one_Hz : out std_logic;
SEGMENT : out std_logic_vector(6 downto 0);
);
end ShowNumber;

architecture Behavioral of ShowNumber is

signal BCD_A0 : std_logic_vector(3 downto 0) := "0000";
signal BCD_A1 : std_logic_vector(3 downto 0) := "0000";
signal BCD_A2 : std_logic_vector(3 downto 0) := "0000";
signal BCD_A3 : std_logic_vector(3 downto 0) := "0000";
signal BCD_A4 : std_logic_vector(3 downto 0) := "0000";


Begin
    variable change_count: integer:= 0;
    variable one_count:interger:=0;

    process(clk)
    begin
        if rising_edge(clk) then
            one_count := one_count + 1;
            change_count := change_count + 1;


            if (change_count = 50000) then          /   Using for switching the showing of the/
                change_count :=0 ;                  /different digits of the time/
            end if;
```

```
if (one_count >= 50000000) then
        one_Hz <= '1';
        if (one_count = 100000000) then       / Code for generating the 1Hz signal/
                one_Hz <= '0';
                one_count :=0 ;
        end if;
end if;

if (S3 = '1') then
        BCD_A4 <= "0010";                      / Reset the timer to two o'clock/
        BCD_A3 <= "0000";
        BCD_A2 <= "0000";
        BCD_A1 <= "0000";
        BCD_A0 <= "0000";
end if;


if (S3 = '0') then                             /Start to count down/
        if (one_Hz='1') then
                if    (BCD_A0 <= 0) then
                        if (BCD_A1 ='0000') then
                                if (BCD_A2 = "0000") then
                                        if (BCD_A3 = "0000") then
                                                if (BCD_A4 = "0000") then
                                                        BCD_A0 <= "0000";
                                                else
                                                        BCD_A4 <= BCD_A4-1;
                                                        BCD_A3 <= "0110";
                                                        BCD_A2 <= "1001";
                                                        BCD_A1 <= "0110";
                                                        BCD_A0 <= "1001";
                                        else
                                                BCD_A3 <= BCD_A3-1;
                                                BCD_A2 <= "1001";
                                                BCD_A1 <= "0110";
                                                BCD_A0 <= "1001";
                                else
                                        BCD_A2 <= BCD_A3-1;
                                        BCD_A1 <= "0110";
                                        BCD_A0 <= "1001";
                        else
                                BCD_A1<=BCD_A1-1;
                                BCD_A0<="1001"
                else
```

```
                    BCD_A0 <= BCD_A0 -1;
        end if;
end if;

if(change_count>=10000 ) then               /Show the number of the least bit digit of/
        SEL_0 <= '1';                       / second/
        SEL_1 <= '0';
        SEL_2 <= '0';
        SEL_3 <= '0';
        SEL_4 <= '0';


        if (S3 = '0') then
                BCD_A0 <= LED_A0;
                case BCD_A0 is
                        when "0000" => SEGMENT <= "1111110";
                        when "0001" => SEGMENT <= "0110000";
                        when "0010" => SEGMENT <= "1101101";
                        when "0011" => SEGMENT <= "1111001";
                        when "0100" => SEGMENT <= "0110011";
                        when "0101" => SEGMENT <= "1011011";
                        when "0110" => SEGMENT <= "1011111";
                        when "0111" => SEGMENT <= "1110000";
                        when "1000" => SEGMENT <= "1111111";
                        when "1001" => SEGMENT <= "1111011";
                        when others => SEGMENT <= "1111011";
                end case;
        end if;

if(change_count>=10000 & -1*change_count>=-20000) then
        SEL_0 <= '0';
        SEL_1 <= '1';
        SEL_2 <= '0';
        SEL_3 <= '0';
        SEL_4 <= '0';


        if (S3 = '0') then
                BCD_A0 <= LED_A0;
                case BCD_A0 is
                        when "0000" => SEGMENT <= "1111110";
                        when "0001" => SEGMENT <= "0110000";
                        when "0010" => SEGMENT <= "1101101";
                        when "0011" => SEGMENT <= "1111001";
                when "0100" => SEGMENT <= "0110011";
                when "0101" => SEGMENT <= "1011011";
                when "0110" => SEGMENT <= "1011111";
                when "0111" => SEGMENT <= "1110000";
                when "1000" => SEGMENT <= "1111111";
                when "1001" => SEGMENT <= "1111011";
                when others => SEGMENT <= "1111011";
            end case;
        end if;

    if(change_count>=20000 & -1*change_count>=-30000) then
        SEL_0 <= '0';
        SEL_1 <= '0';
        SEL_2 <= '1';
        SEL_3 <= '0';
        SEL_4 <= '0';


        if (S3 = '0') then
            BCD_A0 <= LED_A0;
            case BCD_A0 is
                when "0000" => SEGMENT <= "1111110";
                when "0001" => SEGMENT <= "0110000";
                when "0010" => SEGMENT <= "1101101";
                when "0011" => SEGMENT <= "1111001";
                when "0100" => SEGMENT <= "0110011";
                when "0101" => SEGMENT <= "1011011";
                when "0110" => SEGMENT <= "1011111";
                when "0111" => SEGMENT <= "1110000";
                when "1000" => SEGMENT <= "1111111";
                when "1001" => SEGMENT <= "1111011";
                when others => SEGMENT <= "1111011";
            end case;
        end if;

    if(change_count>=30000 & -1*change_count>=-40000) then
        SEL_0 <= '0';
        SEL_1 <= '0';
        SEL_2 <= '0';
        SEL_3 <= '1';
        SEL_4 <= '0';
```

```
            if (S3 = '0') then
                BCD_A0 <= LED_A0;
                case BCD_A0 is
                    when "0000" => SEGMENT <= "1111110";
                    when "0001" => SEGMENT <= "0110000";
                    when "0010" => SEGMENT <= "1101101";
                    when "0011" => SEGMENT <= "1111001";
                    when "0100" => SEGMENT <= "0110011";
                    when "0101" => SEGMENT <= "1011011";
                    when "0110" => SEGMENT <= "1011111";
                    when "0111" => SEGMENT <= "1110000";
                    when "1000" => SEGMENT <= "1111111";
                    when "1001" => SEGMENT <= "1111011";
                    when others => SEGMENT <= "1111011";
                end case;
            end if;

        if(change_count>=40000 & -1*change_count>=-50000) then
            SEL_0 <= '0';
            SEL_1 <= '0';
            SEL_2 <= '0';
            SEL_3 <= '0';                               /Show the number of the hour/
            SEL_4 <= '1';

            if (S3 = '0') then
                BCD_A0 <= LED_A0;
                case BCD_A0 is
                    when "0000" => SEGMENT <= "1111110";
                    when "0001" => SEGMENT <= "0110000";
                    when "0010" => SEGMENT <= "1101101";
                    when "0011" => SEGMENT <= "1111001";
                    when "0100" => SEGMENT <= "0110011";
                    when "0101" => SEGMENT <= "1011011";
                    when "0110" => SEGMENT <= "1011111";
                    when "0111" => SEGMENT <= "1110000";
                    when "1000" => SEGMENT <= "1111111";
                    when "1001" => SEGMENT <= "1111011";
                    when others => SEGMENT <= "1111011";
                end case;
            end if;
        end if;
    end if;
  end process;
end Behavioral;
```