EIE2810 Digital Systems Design Laboratory

Laboratory Report #6

Name: Student ID:
Date: 2024.5.16

The Chinese University of Hong Kong, Shenzhen

Student ID: Name:

- Experiment A: Learn to programme Xilinx FPGA for digit logic function
- Experiment B: Learn to programme Xilinx FPGA for multiple 7-segment outputs simultaneously displayed; Learn to programme Xilinx FPGA to divide high-frequency clocks into low frequency clocks; Learn to programme Xilinx FPGA for a count-down timer

1. Experiment A

1.1 Design

- 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;
```

Figure 1. Code of AND gate

- 2-input-1-output XOR

```
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;
```

Figure 2. Code of XOR gate

1.2 Result

- 8-input AND gate

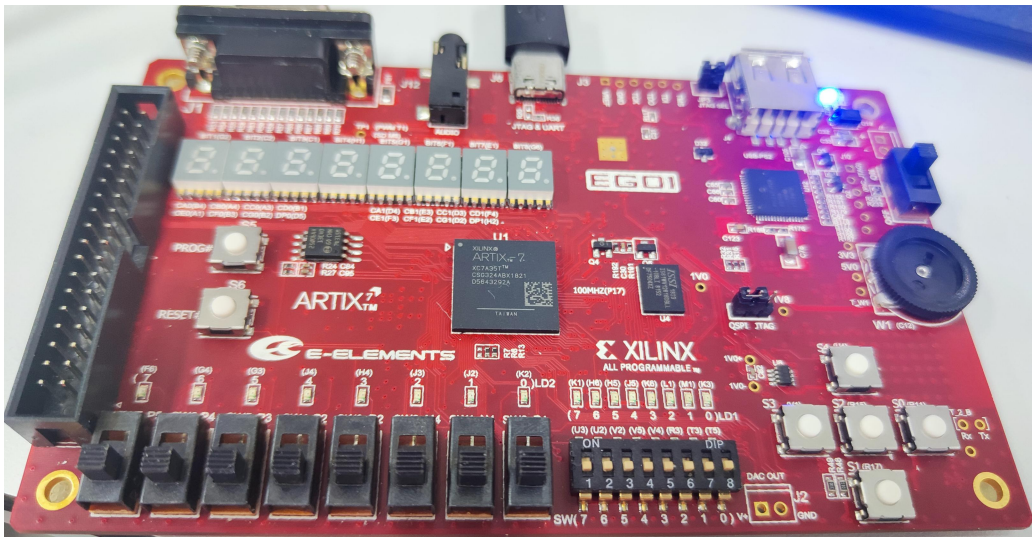

Figure 3. Input is (0, 0, 0, 0, 0, 0, 0, 0), output is 0

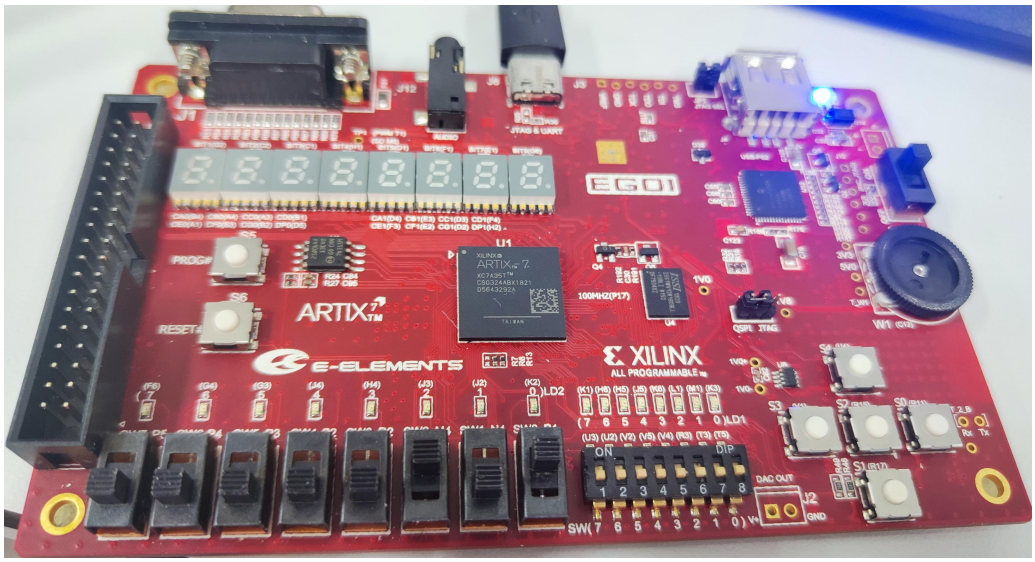

Figure 4. Input is (0, 0, 0, 0, 0, 1, 0, 1), output is 0

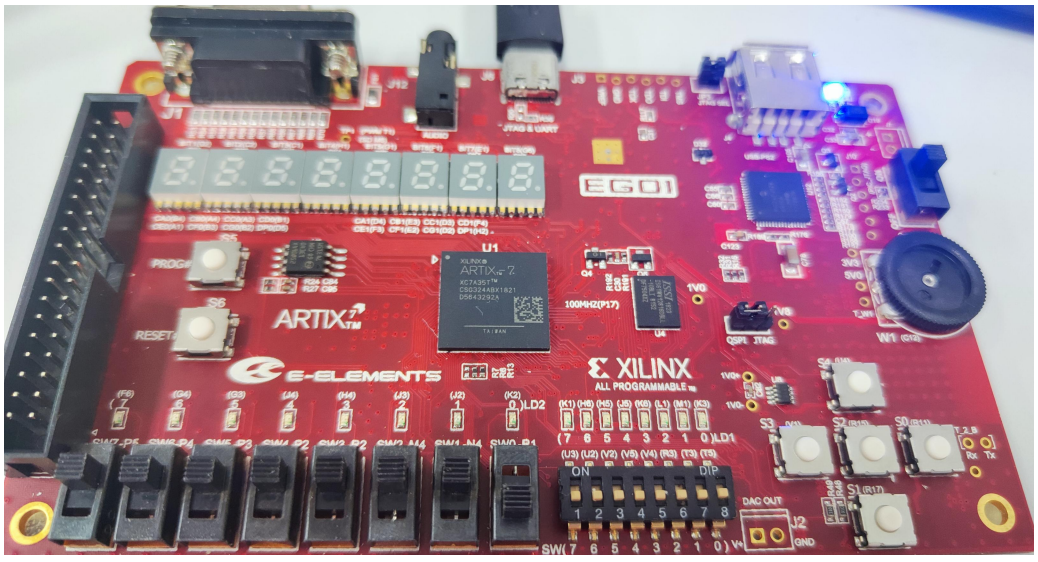

Figure 5. Input is (1, 1, 1, 1, 1, 1, 1, 0), output is 0

Figure 6. Input is (1, 1, 1, 1, 1, 1, 0, 1), output is 0

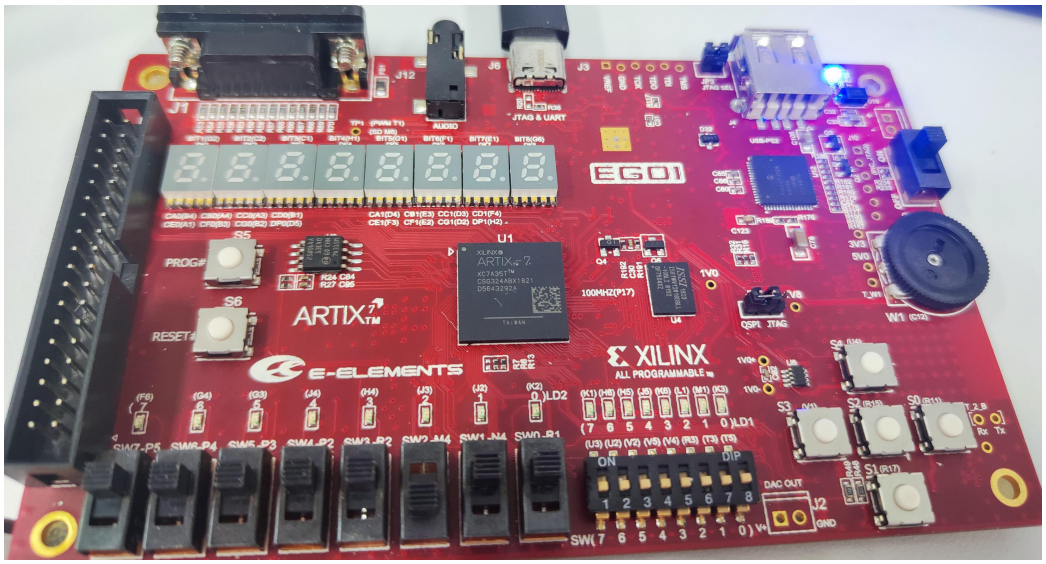

Figure 7. Input is (1, 1, 1, 1, 1, 0, 1, 1), output is 0

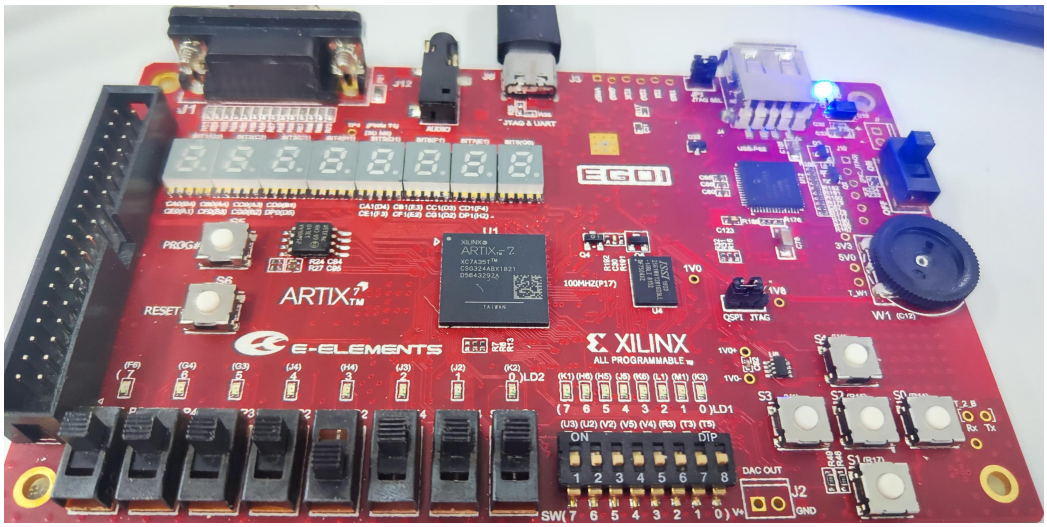

Figure 8. Input is (1, 1, 1, 1, 0, 1, 1, 1), output is 0

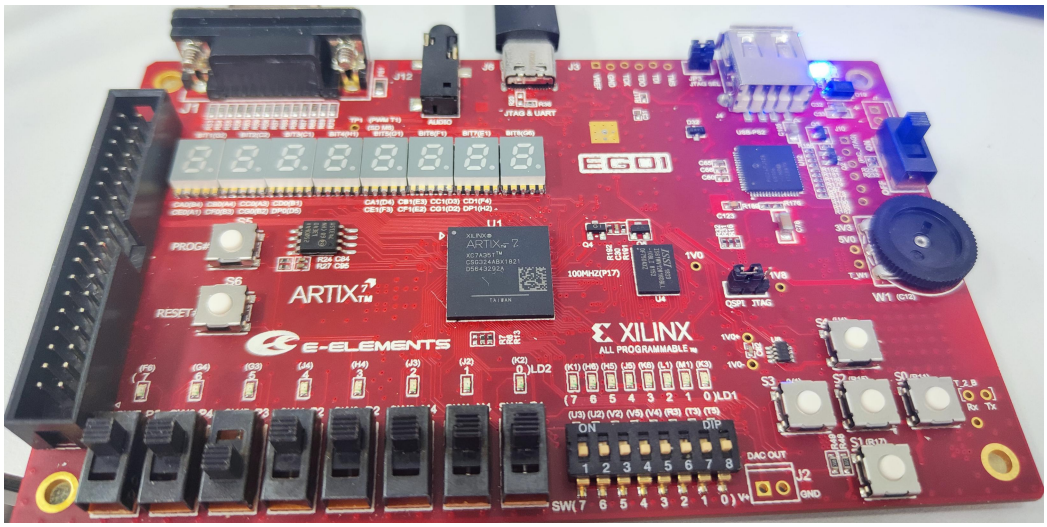

Figure 9. Input is (1, 1, 0, 1, 1, 1, 1, 1), output is 0

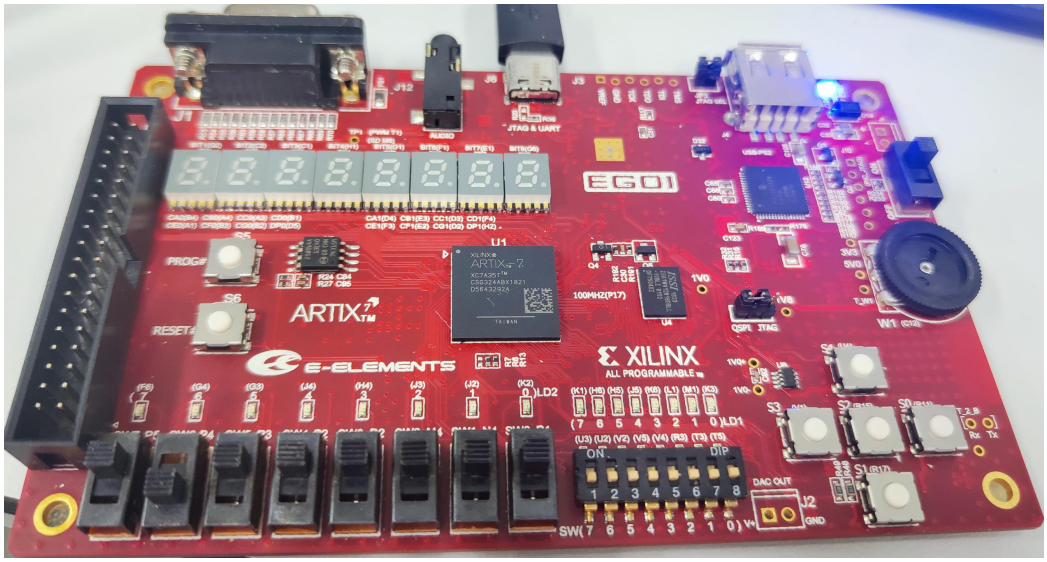

Figure 10. Input is (1, 0, 1, 1, 1, 1, 1, 1), output is 0

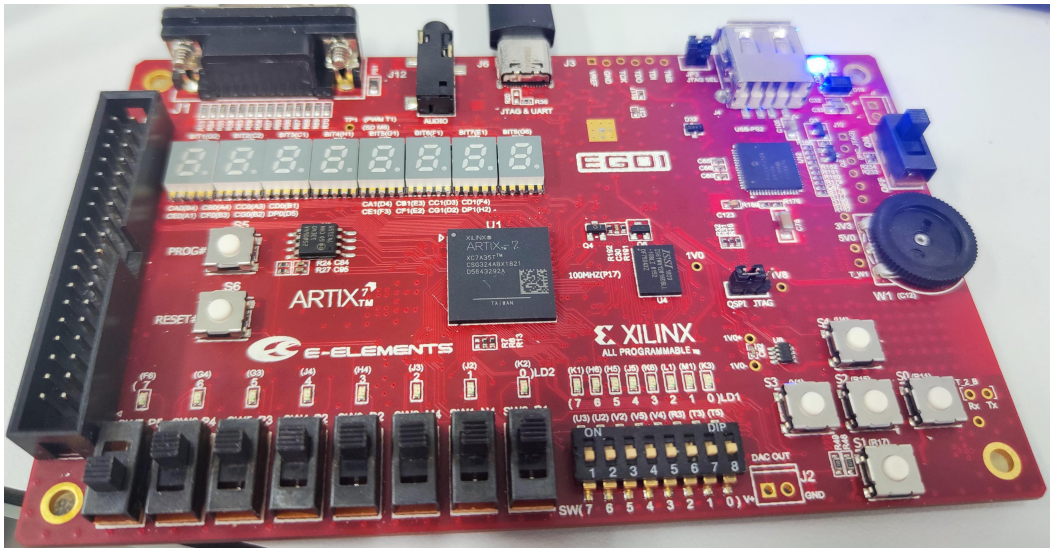

Figure 11. Input is (0, 1, 1, 1, 1, 1, 1, 1), output is 0

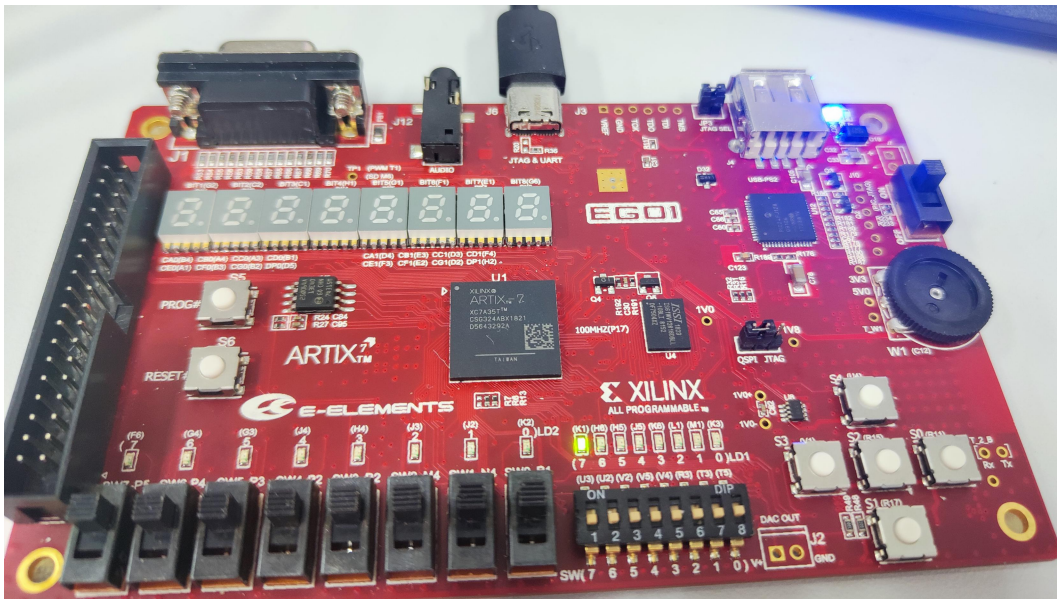

Figure 12. Input is (1, 1, 1, 1, 1, 1, 1, 1), output is 1

Figure1 ~ Figure 12 shows that only when all inputs are HIGH, the output is HIGH. AND gate is successfully implemented.

- 2-input-1-output XOR

SW0 and SW1 are connected to the input ports.

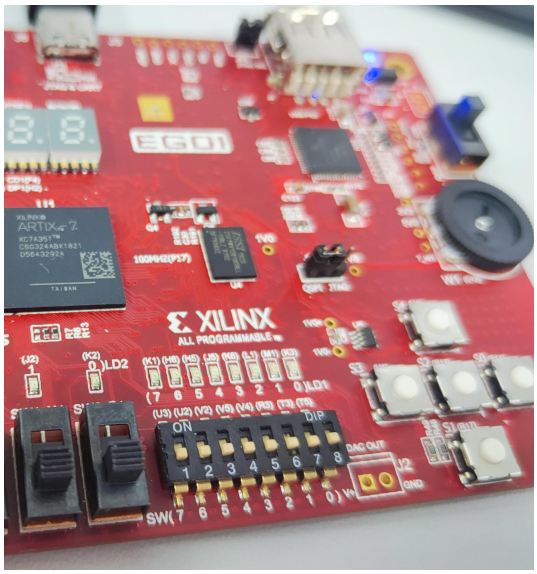

Figure 13. Input is (0, 0), output is 0

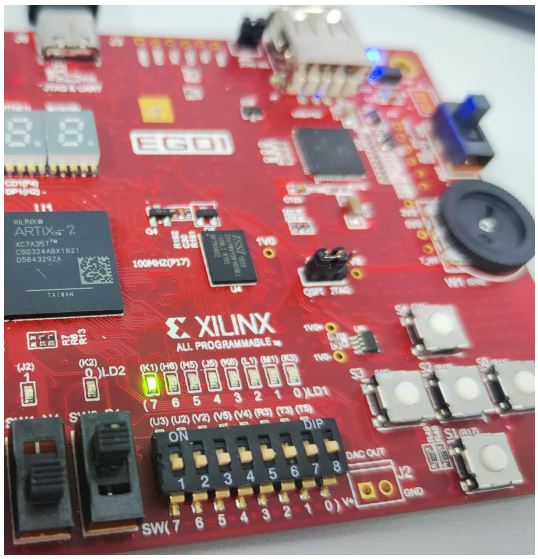

Figure 14. Input is (0, 1), output is 1

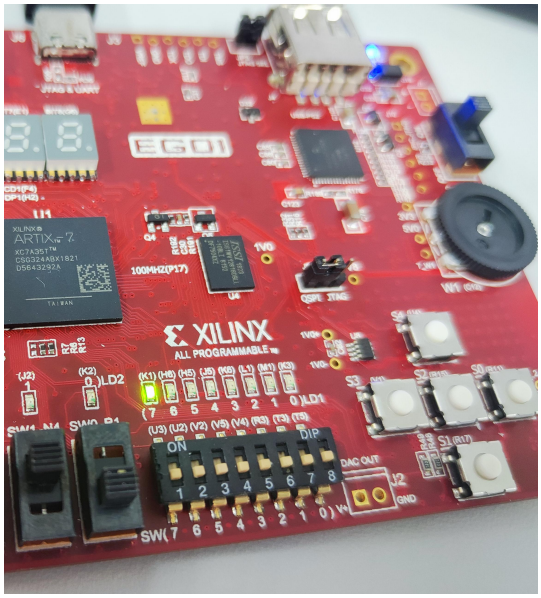

Figure 15. Input is (1, 0), output is 1

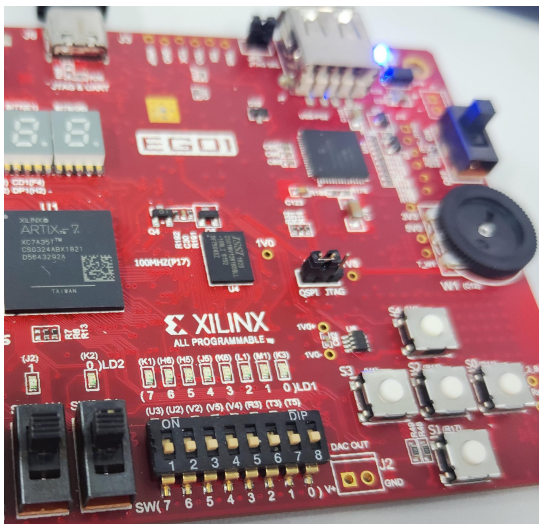

Figure 16. Input is (1, 1), output is 0

When the two inputs are the same, the output is LOW. When the two inputs are different, the output is HIGH.

XOR gate is successfully implemented.

1.3 Questions

Write the entity and architecture for the combinational logic of a 4-bits plus 4-bits adder, with no input carry and 1-bit output carry.

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

entity myAdder is
    Port (
        A : in STD_LOGIC_VECTOR (4 downto 1);
        B : in  STD_LOGIC_VECTOR (4 downto 1);
        Sum : out STD_LOGIC_VECTOR (4 downto 1);
        Cout: out STD_LOGIC);
end myAdder;

architecture Behavioral of myAdder is
begin
    process(A,B)
    variable carry1 : STD_LOGIC := '0';
    variable carry2 : STD_LOGIC := '0';
    variable carry3 : STD_LOGIC := '0';
    variable carry4 : STD_LOGIC := '0';
    begin
        Sum(1)<= A(1) xor B(1);
        carry1 := A(1) and B(1);

        Sum(2)<= A(2) xor B(2) xor carry1;
        carry2 := (A(2) and B(2)) or (B(2) and carry1) or (carry1 and A(2));

        Sum(3)<= A(3) xor B(3) xor carry2;
        carry3 := (A(3) and B(3)) or (B(3) and carry2) or (carry2 and A(3));

        Sum(4)<= A(4) xor B(4) xor carry3;
        carry4 := (A(4) and B(4)) or (B(4) and carry3) or (carry3 and A(4));
        Cout <= carry4;
    end process;
end Behavioral;
```

Figure 17. Code of 4-bit adder

2. Experiment B

2.1 Design

- Part A: Make 4 LEDs flashing at the frequency of 8Hz, 4Hz, 2Hz, and 1Hz

In the following code, the CockDivider entity divides the clock signal (100MHz) to generate four different output signals with frequencies of 1 Hz, 2 Hz, 4 Hz, and 8 Hz, which then drive the LEDs.

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

entity ClockDivider is
  Port (
      ClkFPGA: in STD_LOGIC;
      LED: out STD_LOGIC_VECTOR(4 downto 1)
  );
end ClockDivider;

architecture Behavioral of ClockDivider is
    signal led_state_8HZ: STD_LOGIC:='0';
    signal led_state_4HZ: STD_LOGIC:='0';
    signal led_state_2HZ: STD_LOGIC:='0';
    signal led_state_1HZ: STD_LOGIC:='0';
    constant DIVISOR_8HZ: integer := 6250000-1;
    constant DIVISOR_4HZ: integer := 12500000-1;
    constant DIVISOR_2HZ: integer := 25000000-1;
    constant DIVISOR_1HZ: integer := 50000000-1;

begin
    process(ClkFPGA)
    variable counter1 : integer :=0;
    variable counter2 : integer :=0;
    variable counter3 : integer :=0;
    variable counter4 : integer :=0;
    begin
        if rising_edge(ClkFPGA) then
            if counter1 = DIVISOR_1HZ then
                led_state_1HZ <= not led_state_1HZ;
                counter1 := 0;
            else
                counter1 := counter1+1;
            end if ;
            if counter2 = DIVISOR_2HZ then
                led_state_2HZ <= not led_state_2HZ;
                counter2 := 0;
            else
                counter2 := counter2+1;
            end if ;
            if counter3 = DIVISOR_4HZ then
                led_state_4HZ <= not led_state_4HZ;
                counter3 := 0;
            else
                counter3 := counter3+1;
            end if ;
            if counter4 = DIVISOR_8HZ then
                led_state_8HZ <= not led_state_8HZ;
                counter4 := 0;
            else
                counter4 := counter4+1;
            end if ;
        end if;
    end process;

    led(1) <= led_state_1HZ;
    led(2) <= led_state_2HZ;
    led(3) <= led_state_4HZ;
    led(4) <= led_state_8HZ;
end Behavioral;
```

Figure 18. CockDivider entity

- Part B: Set the BCD codes of 2 digits by the switches SW0~SW7

In the following code, the part_B entity is a structural design that uses "port map" to combine the BCD_display and ClockDivider components, thus fulfilling the functionality of flashing LEDs (requirement A) and a BCD display (requirement B). The BCD_display entity is designed to display BCD values on the 7-segment display. The ClockDivider entity is the same as in part A and is therefore not shown below.

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

entity part_B is
    Port (
        ClkFPGA : in STD_LOGIC;
        switches : in STD_LOGIC_VECTOR(7 downto 0);
        btn_s4 : in STD_LOGIC;
        anodes : out STD_LOGIC_VECTOR(1 downto 0);
        segments : out STD_LOGIC_VECTOR(6 downto 0);
        LED : out STD_LOGIC_VECTOR(4 downto 1)
    );
end part_B;

architecture Structural of part_B is
    component BCD_display is
        Port (
            ClkFPGA : in STD_LOGIC;
            switches : in STD_LOGIC_VECTOR(7 downto 0);
            c_s4 : in STD_LOGIC;
            anodes : out STD_LOGIC_VECTOR(1 downto 0);
            segments : out STD_LOGIC_VECTOR(6 downto 0)
        );
    end component;

    component ClockDivider is
        Port (
            ClkFPGA : in STD_LOGIC;
            LED : out STD_LOGIC_VECTOR(4 downto 1)
        );
    end component;


begin
    DisplayInst : BCD_display
        port map (
            ClkFPGA => ClkFPGA,
            switches => switches,
            c_s4 => btn_s4,
            anodes => anodes,
            segments => segments
        );

    ClockDivInst : ClockDivider
        port map (
            ClkFPGA => ClkFPGA,
            LED => LED
        );
end Structural;
```

Figure 19. Part_B entity

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

entity BCD_display is
   Port (
        ClkFPGA : in STD_LOGIC;
        switches : in STD_LOGIC_VECTOR(7 downto 0);
        c_s4 : in STD_LOGIC;  --S4
        anodes : out STD_LOGIC_VECTOR(1 downto 0);  -- DN0_K1, DN0_K2
        segments : out STD_LOGIC_VECTOR(6 downto 0)  -- 7-segment display
    );
end BCD_display;

architecture Behavioral of BCD_display is
signal bcd_digit_1, bcd_digit_2 : STD_LOGIC_VECTOR(3 downto 0);
signal anode_control : STD_LOGIC_VECTOR(1 downto 0) := "01";
signal display_active : STD_LOGIC := '0';
signal segments1, segments2 : STD_LOGIC_VECTOR(6 downto 0);

begin
    -- BCD Input Sanitization
    process(switches)
    begin
        if switches(3 downto 0) > "1001" then
            bcd_digit_1 <= "1001";
        else
            bcd_digit_1 <= switches(3 downto 0);
        end if;

        if switches(7 downto 4) > "1001" then
            bcd_digit_2 <= "1001";
        else
            bcd_digit_2 <= switches(7 downto 4);
        end if;
    end process;

    process(ClkFPGA)
    variable count :integer := 0;
    begin
        if rising_edge(ClkFPGA) then
            if count = 1000000 then
                anode_control <= not anode_control
                count := 0;
            else
                count := count + 1;
            end if;
        end if;
    end process;

    -- Display Control
    process(anode_control, c_s4)
    begin
    if c_s4 = '1' then
        if anode_control = "01" then
            if (bcd_digit_1 = "0000") then
                 segments1 <= "1111110";
                 segments <= segments1;
            elsif (bcd_digit_1 = "0001") then
                 segments1 <= "0110000";
                 segments <= segments1;
            elsif (bcd_digit_1 = "0010") then
                 segments1 <= "1101101";
                 segments <= segments1;
            elsif (bcd_digit_1 = "0011") then
```

```
                segments1 <= "1111001";
                segments <= segments1;
            elsif (bcd_digit_1 = "0100") then
                segments1 <= "0110011";
                segments <= segments1;
            elsif (bcd_digit_1 = "0101") then
                segments1 <= "1011011";
                segments <= segments1;
            elsif (bcd_digit_1 = "0110") then
                segments1 <= "0011111";
                segments <= segments1;
            elsif (bcd_digit_1 = "0111") then
                segments1 <= "1110000";
                segments <= segments1;
            elsif (bcd_digit_1 = "1000") then
                segments1 <= "1111111";
                segments <= segments1;
            elsif (bcd_digit_1 = "1001") then
                segments1 <= "1110011";
                segments <= segments1;
            else
                segments1 <= "1110011";
                segments <= segments1;
            end if;
        else
            if (bcd_digit_2 = "0000") then
                segments2 <= "0000000";
                segments <= segments2;
            elsif (bcd_digit_2 = "0001") then
                segments2 <= "0110000";
                segments <= segments2;
            elsif (bcd_digit_2 = "0010") then
                segments2 <= "1101101";
                segments <= segments2;
            elsif (bcd_digit_2 = "0011") then
                segments2 <= "1111001";
                segments <= segments2;
            elsif (bcd_digit_2 = "0100") then
                segments2 <= "0110011";
                segments <= segments2;
            elsif (bcd_digit_2 = "0101") then
                segments2 <= "1011011";
                segments <= segments2;
            elsif (bcd_digit_2 = "0110") then
                segments2 <= "0011111";
                segments <= segments2;
            elsif (bcd_digit_2 = "0111") then
                segments2 <= "1110000";
                segments <= segments2;
            elsif (bcd_digit_2 = "1000") then
                segments2 <= "1111111";
                segments <= segments2;
            elsif (bcd_digit_2 = "1001") then
                segments2 <= "1110011";
                segments <= segments2;
            else
                segments2 <= "1110011";
                segments <= segments2;
            end if;
        end if;
    else
        if anode_control = "01" then
            segments <= segments1;
        end if;
        if anode_control = "10" then
            segments <= segments2;
        end if;
    end if;
    end process;

    anodes <= anode_control;
end Behavioral;
```

Figure 20. BCD_display entity

- Part C: 2-digit display counts down at the frequency of 1Hz

In the following code, the part_C entity is a structural design that uses “port map” to combine the Countdown and ClockDivider components, thus fulfilling the functionality of flashing LEDs (requirement A) and a countdown clock (requirement C). The Countdown entity is designed to display BCD values on the 7-segment display and count down at the frequency of 1Hz. The ClockDivider entity is the same as in part A and is therefore not shown below.

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

entity part_C is
    Port (
        ClkFPGA : in STD_LOGIC;
        switches : in STD_LOGIC_VECTOR(7 downto 0);
        sw_0 : in STD_LOGIC;
        anodes : out STD_LOGIC_VECTOR(1 downto 0);
        segments : out STD_LOGIC_VECTOR(6 downto 0);
        LED : out STD_LOGIC_VECTOR(4 downto 1);
        c_s4 : in STD_LOGIC;
        L : out STD_LOGIC_VECTOR (7 downto 0)
    );
end part_C;

architecture Structural of part_C is
    component ClockDivider is
        Port (
            ClkFPGA : in STD_LOGIC;
            LED : out STD_LOGIC_VECTOR(4 downto 1)
        );
    end component;

    component Countdown is
        Port (
           switches : in STD_LOGIC_VECTOR (7 downto 0);
           c_s4 : in STD_LOGIC;
           sw_0: in std_logic;
           anodes : out STD_LOGIC_VECTOR(1 downto 0);
           ClkFPGA : in STD_LOGIC;
           L : out STD_LOGIC_VECTOR (7 downto 0);
           segments : out STD_LOGIC_VECTOR (6 downto 0)
        );
    end component;

begin
    ClockDivInst : ClockDivider
        port map (
            ClkFPGA => ClkFPGA,
            LED => LED
        );
```

```
    CountdownInst : Countdown
        port map (
            ClkFPGA => ClkFPGA,
            sw_0 => sw_0,
            switches => switches,
            c_s4 => c_s4,
            anodes => anodes,
            segments => segments,
            L=>L
        );
end Structural;
```

Figure 21. part_C entity

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

entity Countdown is
    Port ( switches : in STD_LOGIC_VECTOR (7 downto 0);
           c_s4 : in STD_LOGIC;
           sw_0 : in STD_LOGIC;
           anodes : out STD_LOGIC_VECTOR(1 downto 0);
           ClkFPGA : in STD_LOGIC;
           L : out STD_LOGIC_VECTOR (7 downto 0);
           segments : out STD_LOGIC_VECTOR (6 downto 0));
end Countdown;

architecture Behavioral of Countdown is
     signal n_change: integer := 0;
     signal n_1: integer := 0;
     signal state_1HZ: std_logic;
     signal anode_control: std_logic := '0';
     signal sw0, sw1, sw2, sw3, sw4, sw5, sw6, sw7: std_logic;
     signal start: std_logic;
     signal segments1, segments2: STD_LOGIC_VECTOR (6 downto 0);
     signal bcd_digit_1, bcd_digit_2: std_logic_vector (3 downto 0);
     constant DIVISOR_1HZ: integer := 50000000-1;
     constant change_HZ: integer := 1000000-1;
begin
    process (ClkFPGA)
    begin
        if rising_edge(ClkFPGA) then
            n_change<= n_change+1;
            n_1<=n_1+1;
            if n_1 = DIVISOR_1HZ then
                n_1 <= 0;
                state_1HZ <= not(state_1HZ);
            end if;
            if n_change = change_HZ then
                n_change <= 0;
                anode_control <= not(anode_control);
            end if;
        end if;
    end process;

    process(anode_control)
    begin
```

```
    anodes(0) <= anode_control;
    anodes(1)<= not anode_control;
    if anode_control = '1' then
        segments <= segments2;
    else
        segments <= segments1;
    end if;
end process;

process(bcd_digit_1, bcd_digit_2)
begin
    if start = '1' then
        if (bcd_digit_1 = "0000") then
            segments1<="1111110";
        elsif (bcd_digit_1 = "0001") then
            segments1<="0110000";
        elsif (bcd_digit_1 = "0010") then
            segments1<="1101101";
        elsif (bcd_digit_1 = "0011") then
            segments1<="1111001";
        elsif (bcd_digit_1 = "0100") then
            segments1<="0110011";
        elsif (bcd_digit_1 = "0101") then
            segments1<="1011011";
        elsif (bcd_digit_1 = "0110") then
            segments1<="0011111";
        elsif (bcd_digit_1 = "0111") then
            segments1<="1110000";
        elsif (bcd_digit_1 = "1000") then
            segments1<="1111111";
        elsif (bcd_digit_1 = "1001") then
            segments1<="1110011";
        else
            segments1<="1110011";
        end if;

        if (bcd_digit_2 = "0000") then
            segments2<="0000000";
        elsif (bcd_digit_2 = "0001") then
            segments2<="0110000";
        elsif (bcd_digit_2 = "0010") then
            segments2<="1101101";
        elsif (bcd_digit_2 = "0011") then
            segments2<="1111001";
        elsif (bcd_digit_2 = "0100") then
            segments2<="0110011";
        elsif (bcd_digit_2 = "0101") then
            segments2<="1011011";
        elsif (bcd_digit_2 = "0110") then
            segments2<="0011111";
        elsif (bcd_digit_2 = "0111") then
            segments2<="1110000";
        elsif (bcd_digit_2 = "1000") then
            segments2<="1111111";
        elsif (bcd_digit_2 = "1001") then
            segments2<="1110011";
        else
            segments2<="1110011";
        end if;
    end if;
end process;

process(ClkFPGA)
begin
    if rising_edge(ClkFPGA) then
        if c_s4 = '1' then
```

```
                start <= '1';
                bcd_digit_1 <= sw3 & sw2 & sw1 & sw0;
                bcd_digit_2 <= sw7 & sw6 & sw5 & sw4;
                if bcd_digit_1 > "1001" then
                    bcd_digit_1 <= "1001";
                end if;
                if bcd_digit_2 > "1001" then
                    bcd_digit_2 <= "1001";
                end if;
            end if;
            if (rising_edge(state_1HZ) and start = '1') then
                if bcd_digit_2 /= "0000" and bcd_digit_1 = "0000" then
                    bcd_digit_2 <= bcd_digit_2 - "0001";
                    bcd_digit_1 <= "1001";
                elsif bcd_digit_2 = "0000" and bcd_digit_1 = "0000" then
                else
                    bcd_digit_1 <= bcd_digit_1 - "0001";
                end if;
            end if;
        end if;
    end process;

    process(switches)
    begin
        sw0 <= switches(0);
        sw1 <= switches(1);
        sw2 <= switches(2);
        sw3 <= switches(3);
        sw4 <= switches(4);
        sw5 <= switches(5);
        sw6 <= switches(6);
        sw7 <= switches(7);
        L<= switches;
    end process;
end Behavioral;
```

Figure 22. Countdown entity

2.2 Result

- Part A: Make 4 LEDs flashing at the frequency of 8Hz, 4Hz, 2Hz, and 1Hz

Figure 23. Flashing LEDs (1)

Figure 24. Flashing LEDs (2)

Figure 25. Flashing LEDs (3)

Figure 26. Flashing LEDs (4)

- Part B: Set the BCD codes of 2 digits by the switches SW0~SW7

In the following figures, the switches are (SW7, SW6, SW5, SW4, SW3, SW2, SW1, SW0) from left to right. SW7~SW4 are for the 10s digit, SW3~SW0 are for the 1s digit. SW0 and SW3 are the LSB and MSB of the 1s digit, SW4 and SW7 are the LSB and MSB of the 10s digit. Any BCD code exceed 0b1001 is regarded as 0b1001. If the 10s digit is 0, the left 7-segment display will show nothing.

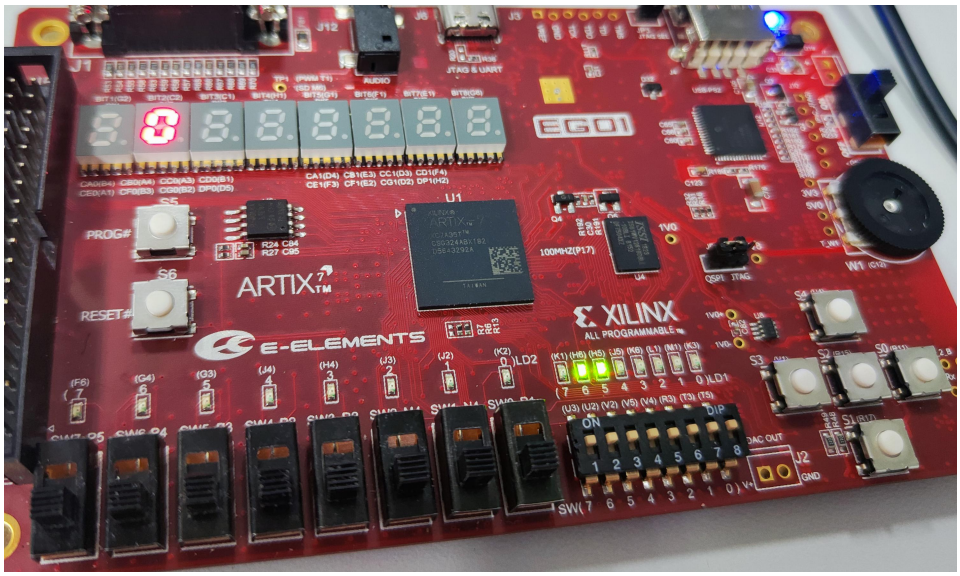

Figure 27. Switches are (0, 0, 0, 0, 0, 0, 0, 0), display “0”

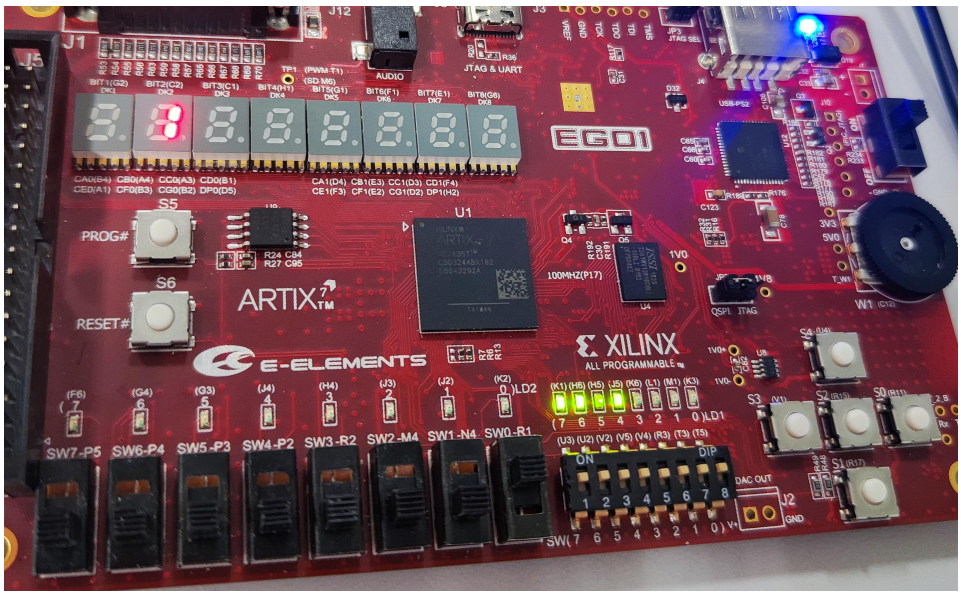

Figure 28. Switches are (0, 0, 0, 0, 0, 0, 0, 1), display “1”

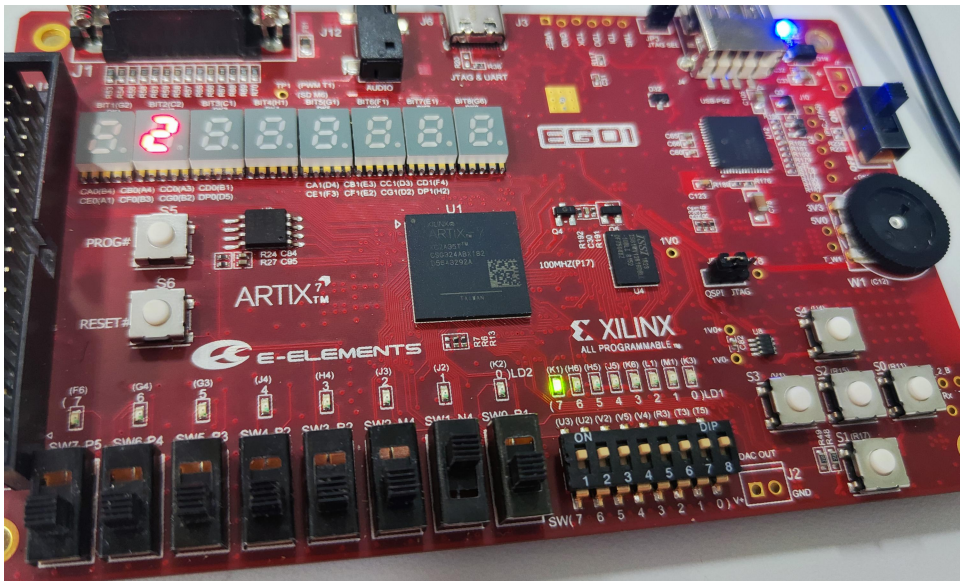

Figure 29. Switches are (0, 0, 0, 0, 0, 0, 1, 0), display “2”

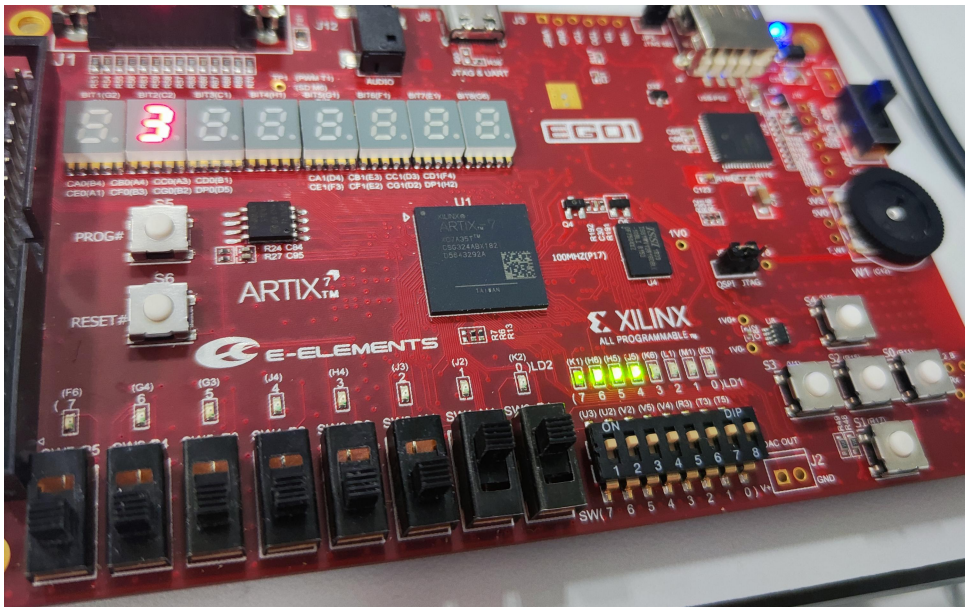

Figure 30. Switches are (0, 0, 0, 0, 0, 0, 1, 1), display “3”

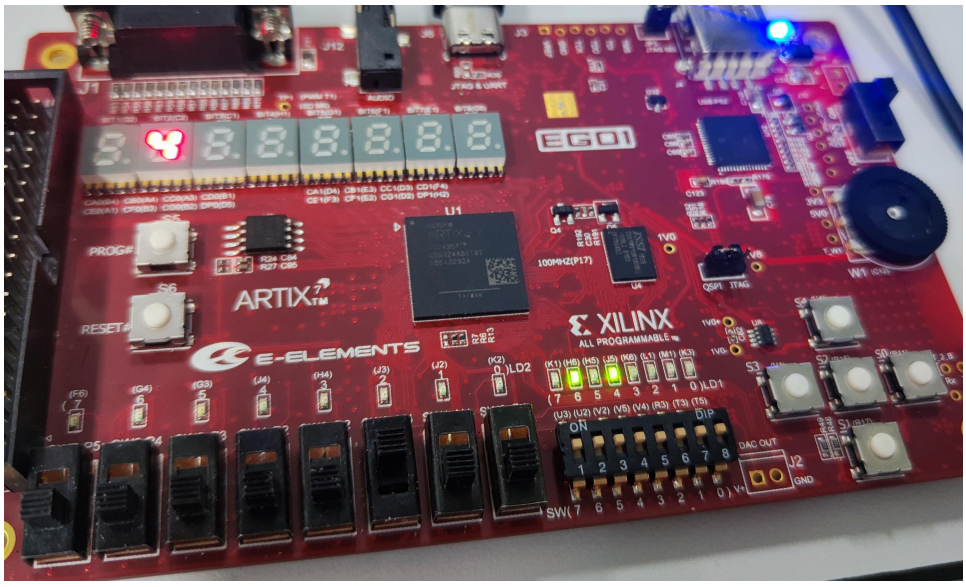

Figure 31. Switches are (0, 0, 0, 0, 0, 1, 0, 0), display “4”

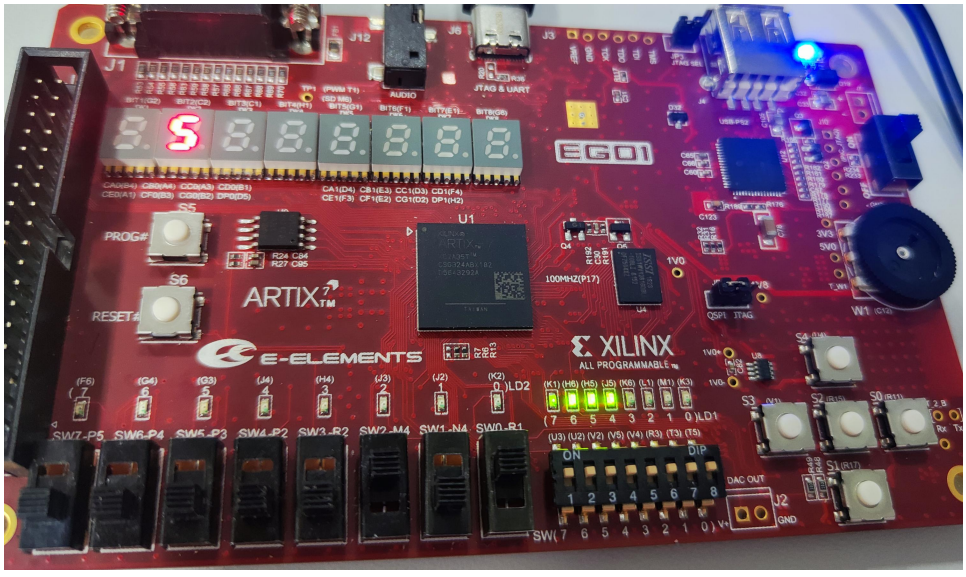

Figure 32. Switches are (0, 0, 0, 0, 0, 1, 0, 1), display “5”

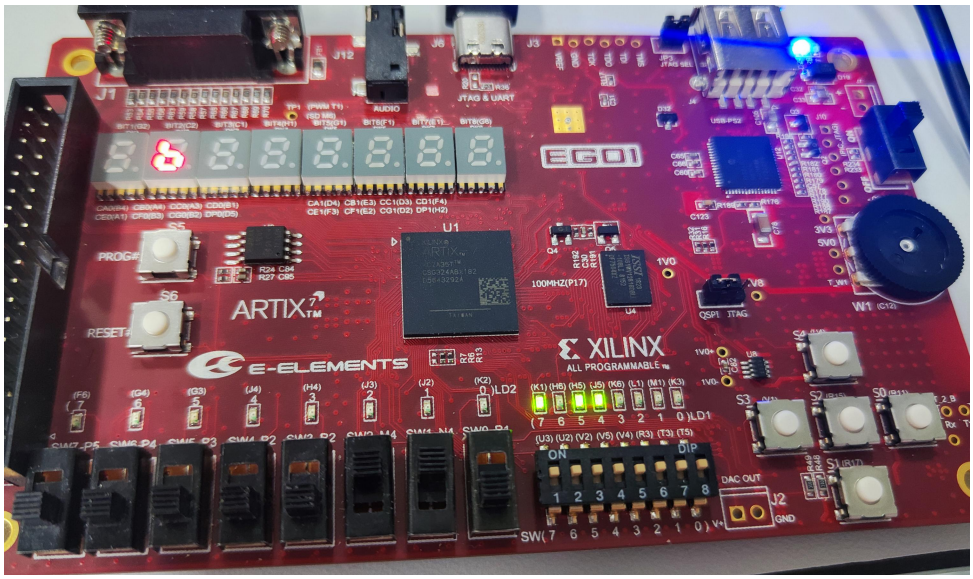

Figure 33. Switches are (0, 0, 0, 0, 0, 1, 1, 0), display “6”

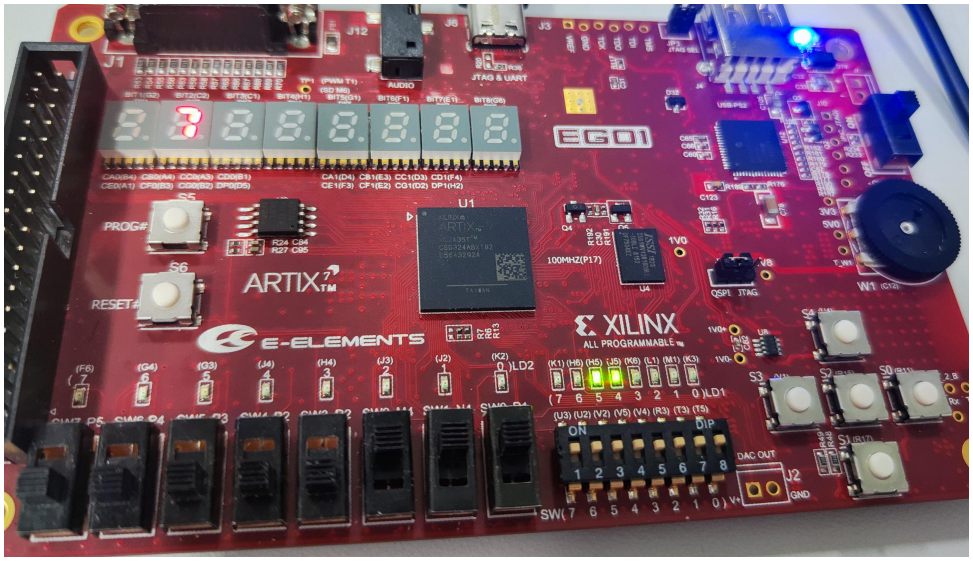

Figure 34. Switches are (0, 0, 0, 0, 0, 1, 1, 1), display “7”

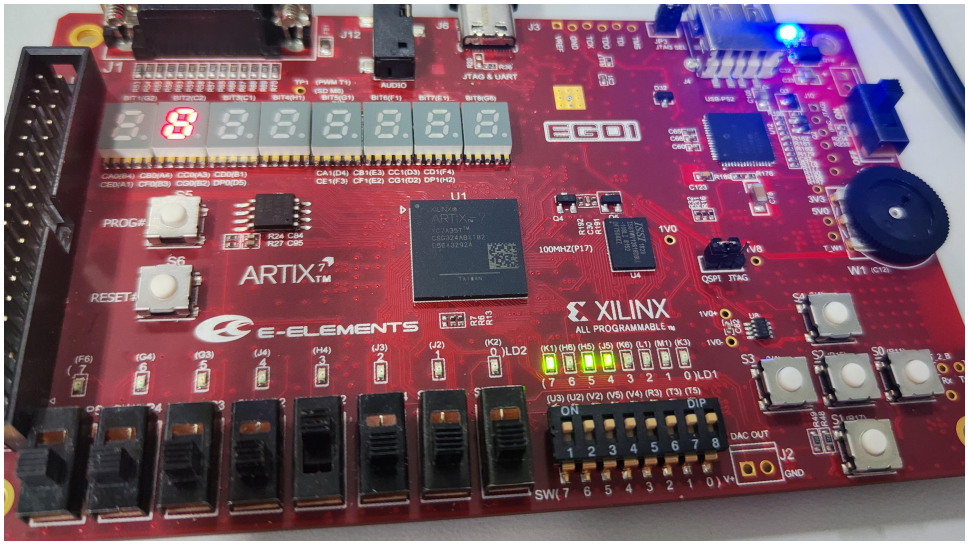

Figure 35. Switches are (0, 0, 0, 0, 1, 0, 0, 0), display “8”

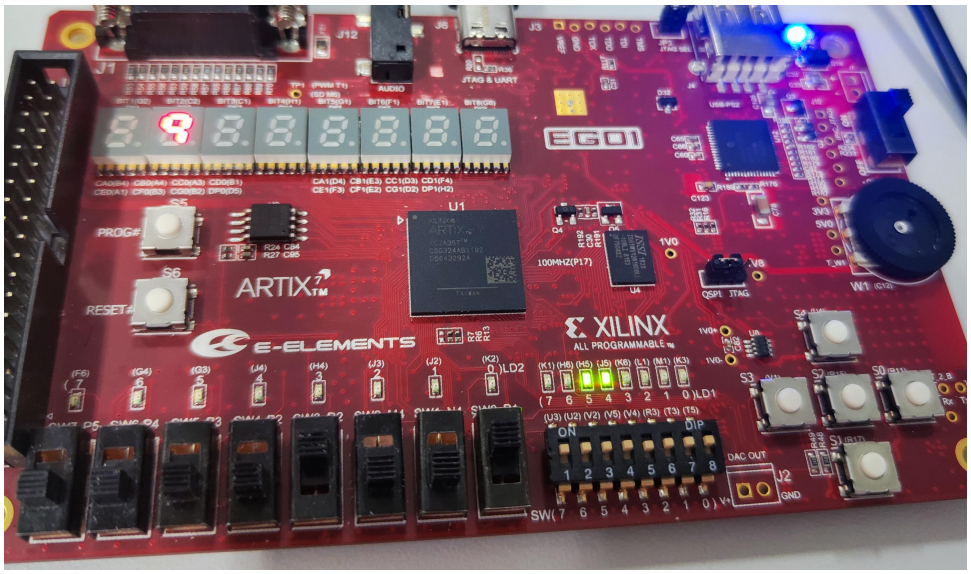

Figure 36. Switches are (0, 0, 0, 0, 1, 0, 0, 1), display “9”

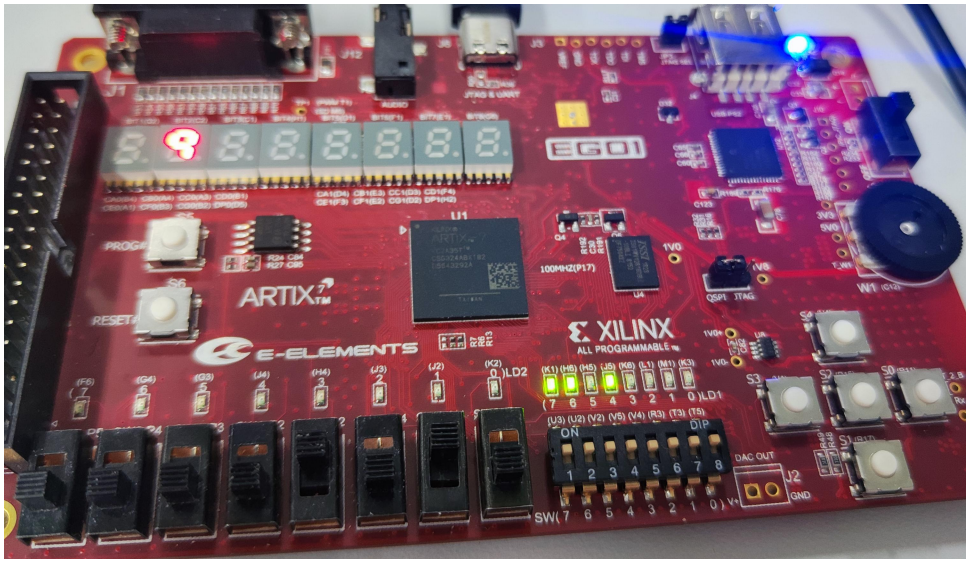

Figure 37. Switches are (0, 0, 0, 0, 1, 0, 1, 0), display “9”

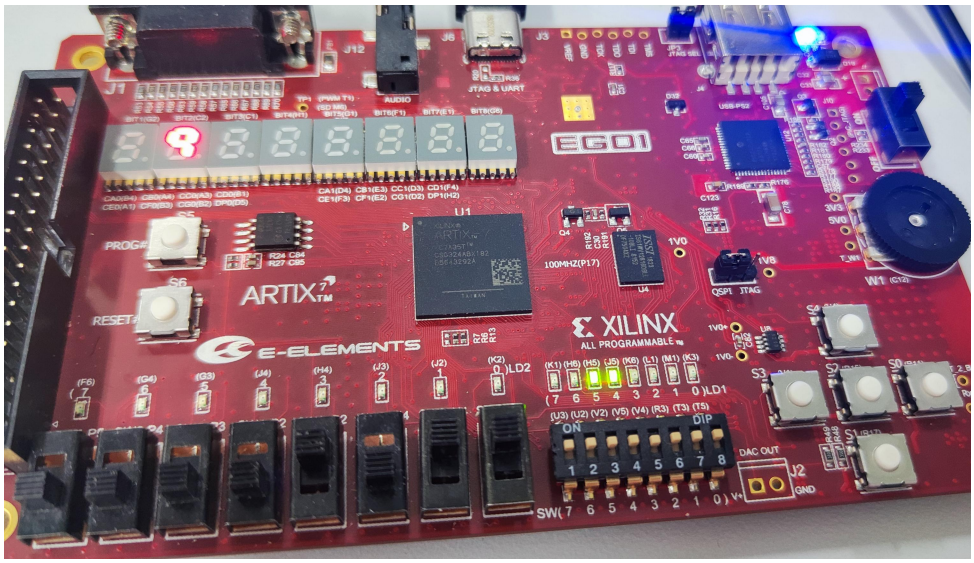

Figure 38. Switches are (0, 0, 0, 0, 1, 0, 1, 1), display “9”

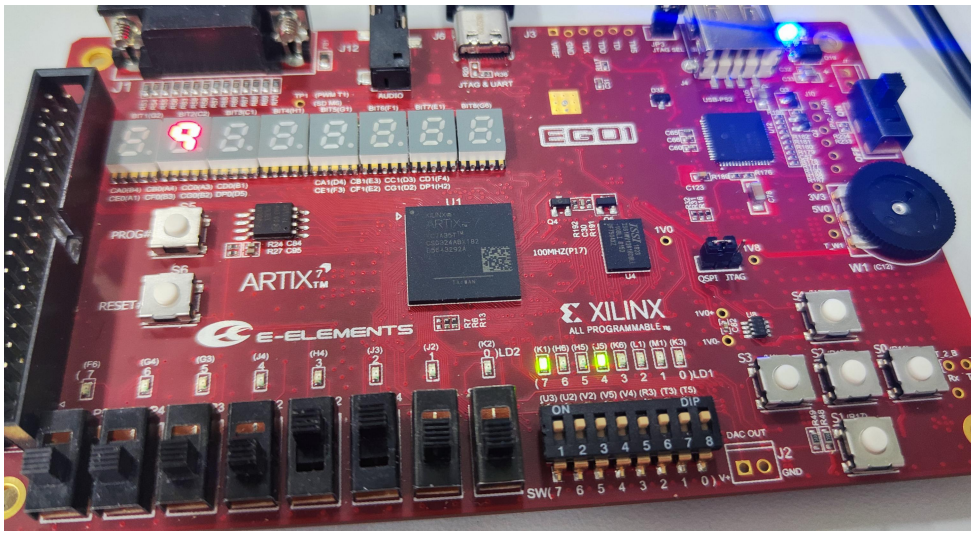

Figure 39. Switches are (0, 0, 0, 0, 1, 1, 0, 0), display “9”

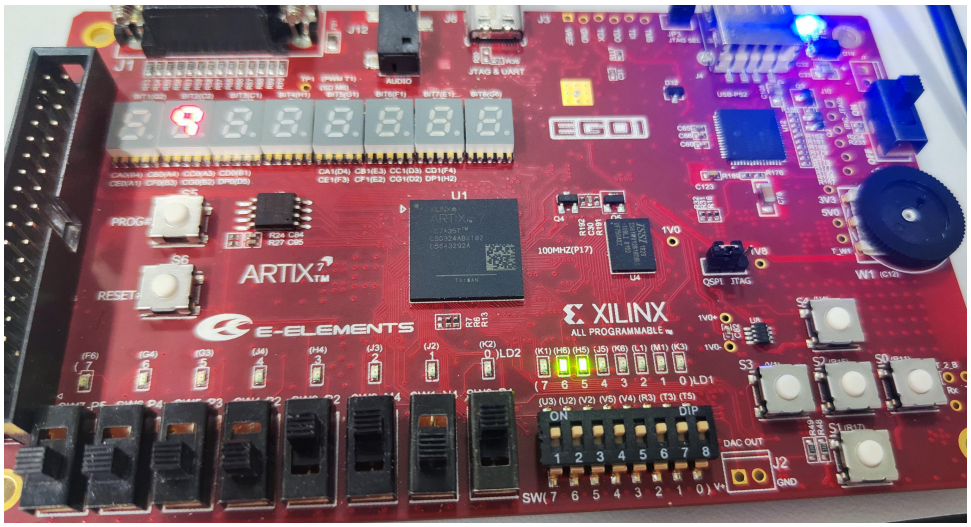

Figure 40. Switches are (0, 0, 0, 0, 1, 1, 0, 1), display “9”

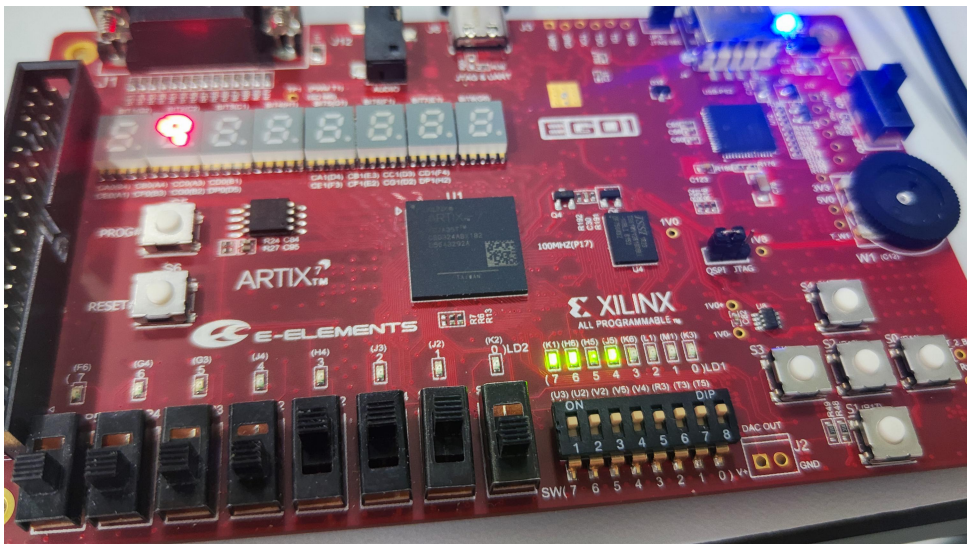

Figure 41. Switches are (0, 0, 0, 0, 1, 1, 1, 0), display “9”

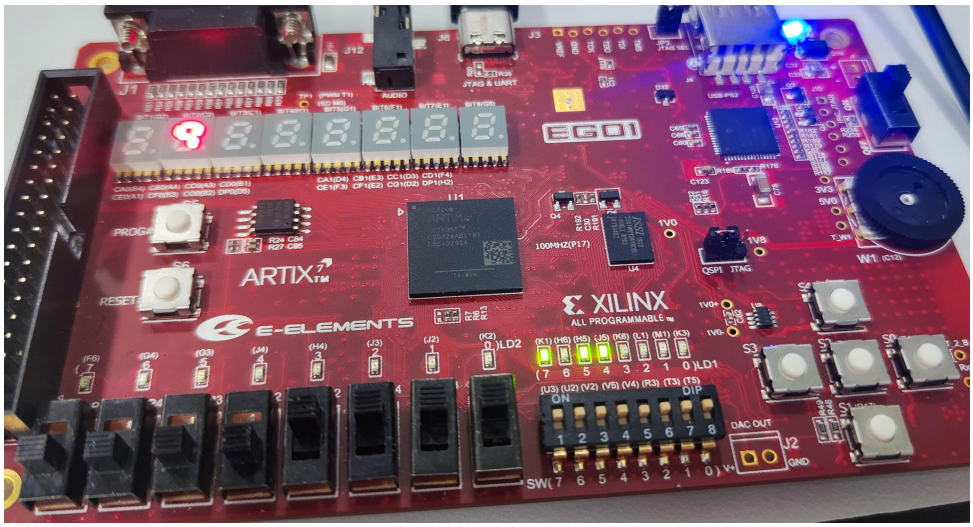

Figure 42. Switches are (0, 0, 0, 0, 1, 1, 1, 1), display “9”

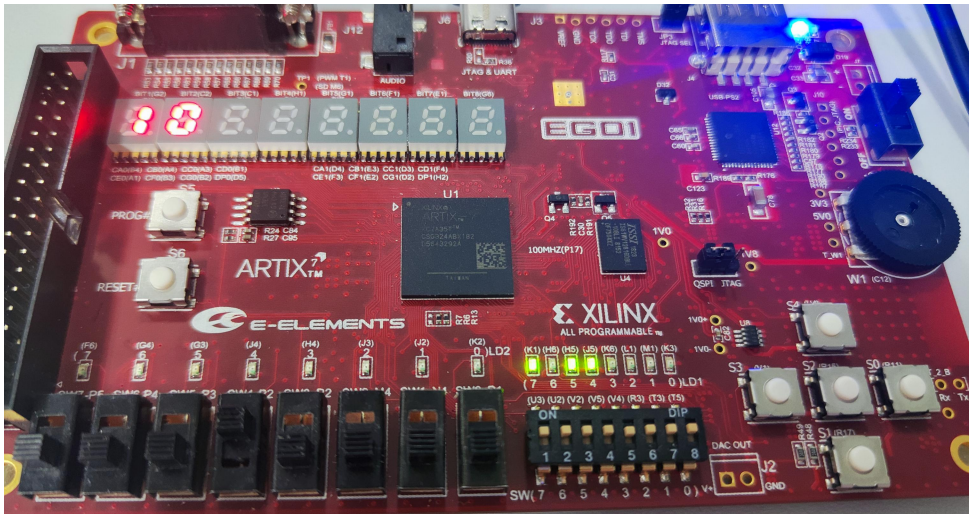

Figure 43. Switches are (0, 0, 0, 1, 0, 0, 0, 0), display “10”

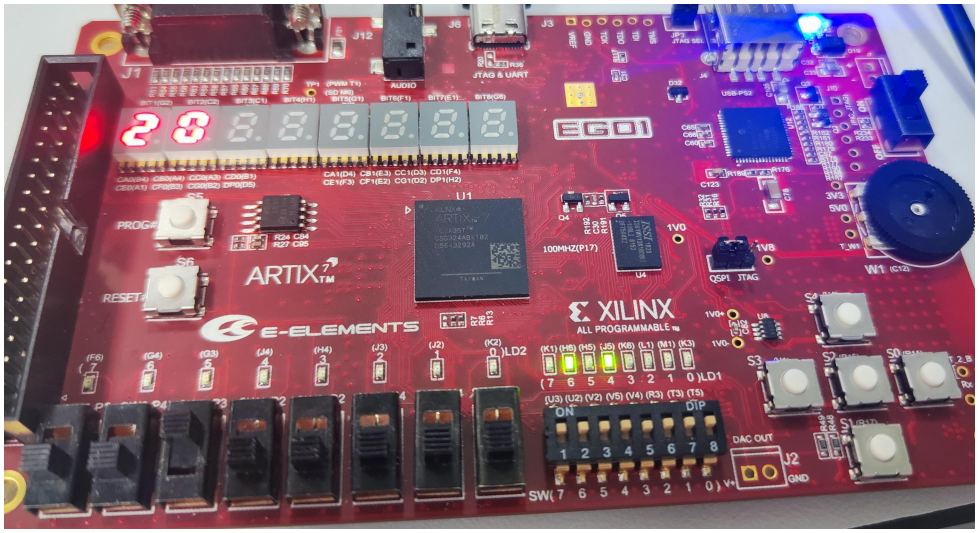

Figure 44. Switches are (0, 0, 1, 0, 0, 0, 0, 0), display “20”

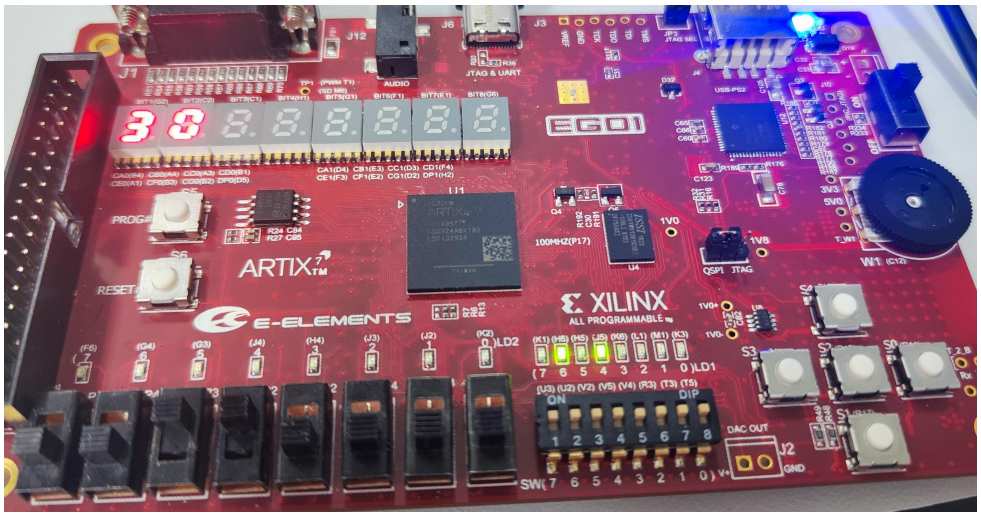

Figure 45. Switches are (0, 0, 1, 1, 0, 0, 0, 0), display “30”

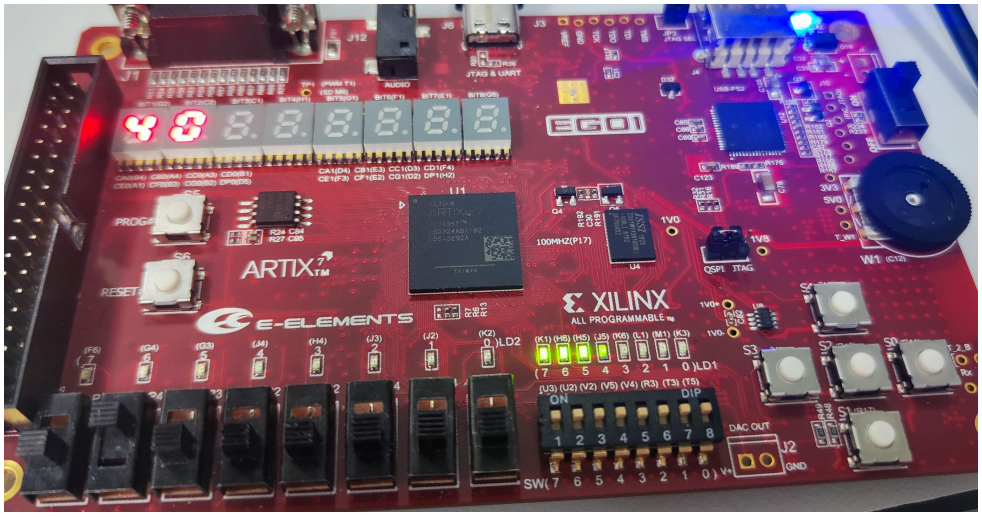

Figure 46. Switches are (0, 1, 0, 0, 0, 0, 0, 0), display “40”

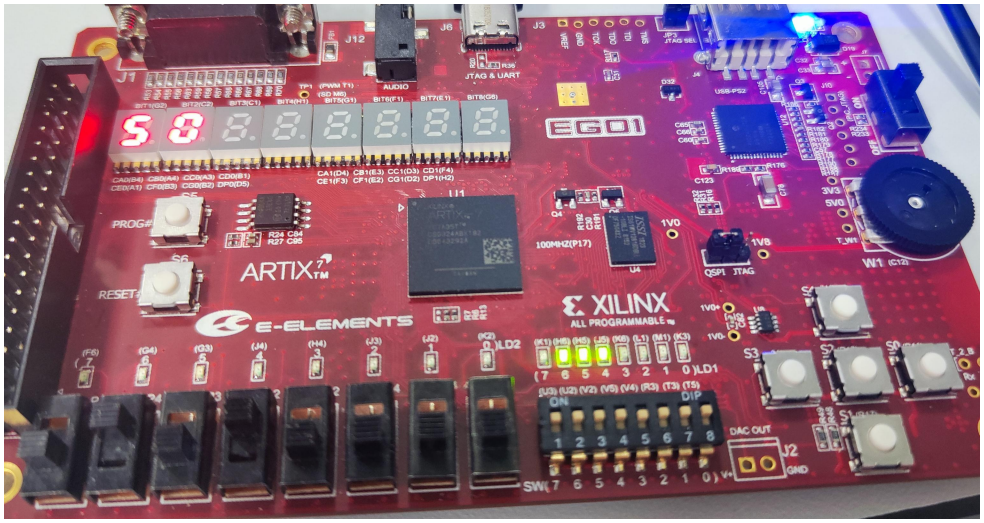

Figure 47. Switches are (0, 1, 0, 1, 0, 0, 0, 0), display “50”

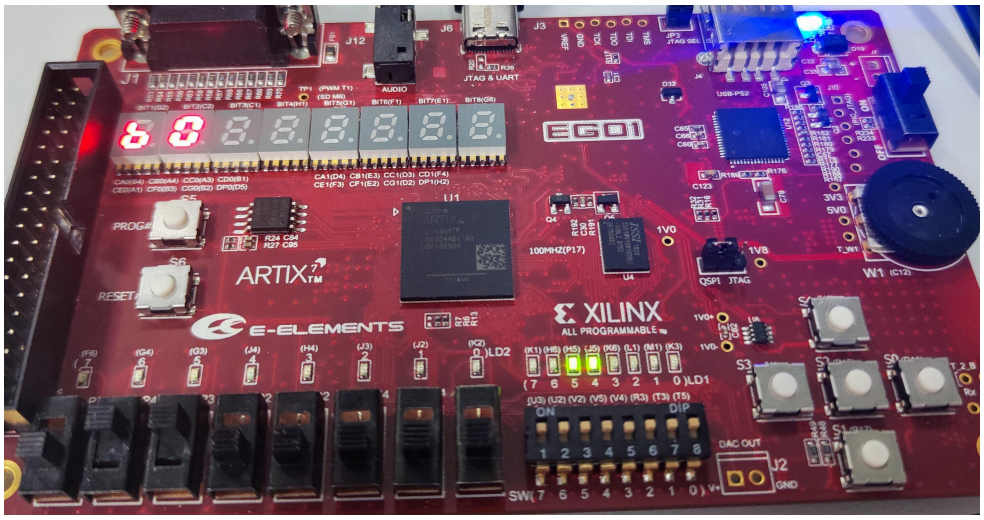

Figure 48. Switches are (0, 1, 1, 0, 0, 0, 0, 0), display “60”

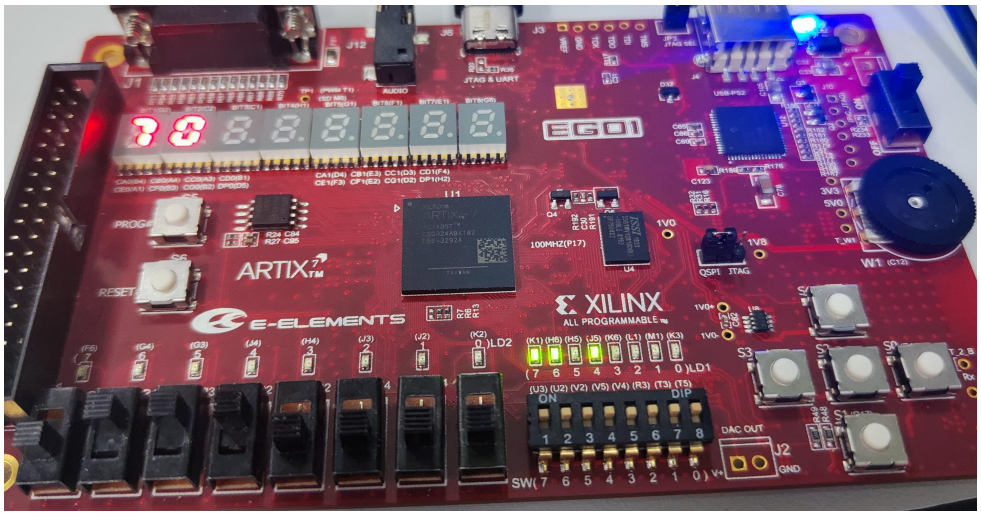

Figure 49. Switches are (0, 1, 1, 1, 0, 0, 0, 0), display “70”

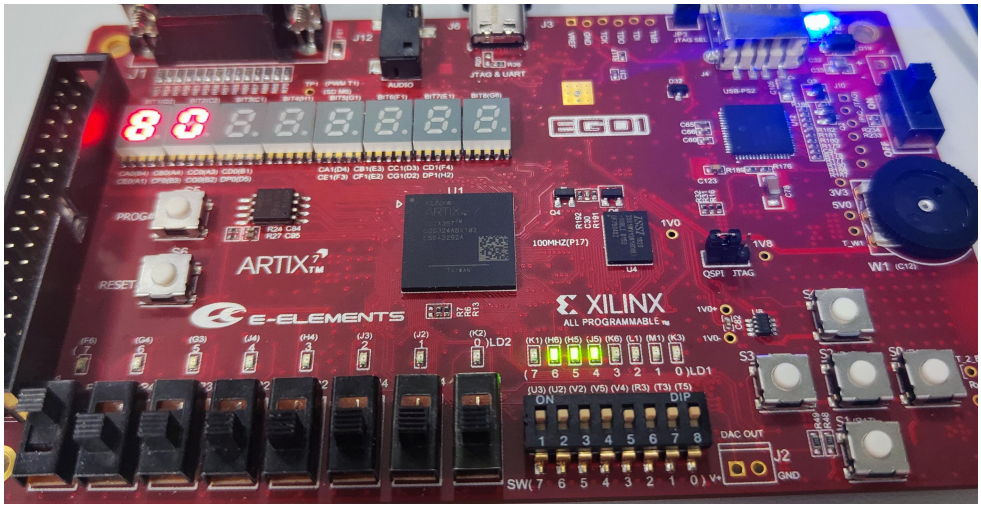

Figure 50. Switches are (1, 0, 0, 0, 0, 0, 0, 0), display “80”

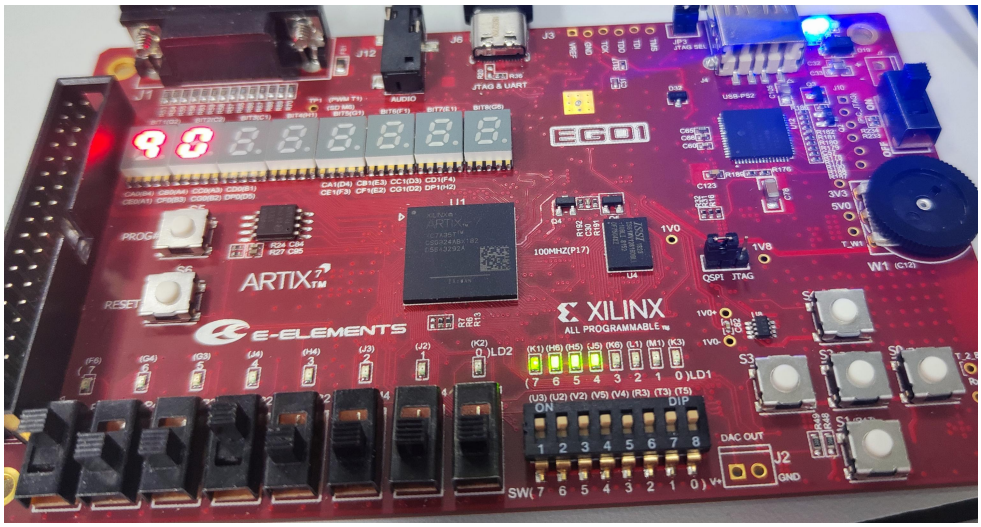

Figure 51. Switches are (1, 0, 0, 1, 0, 0, 0, 0), display “90”

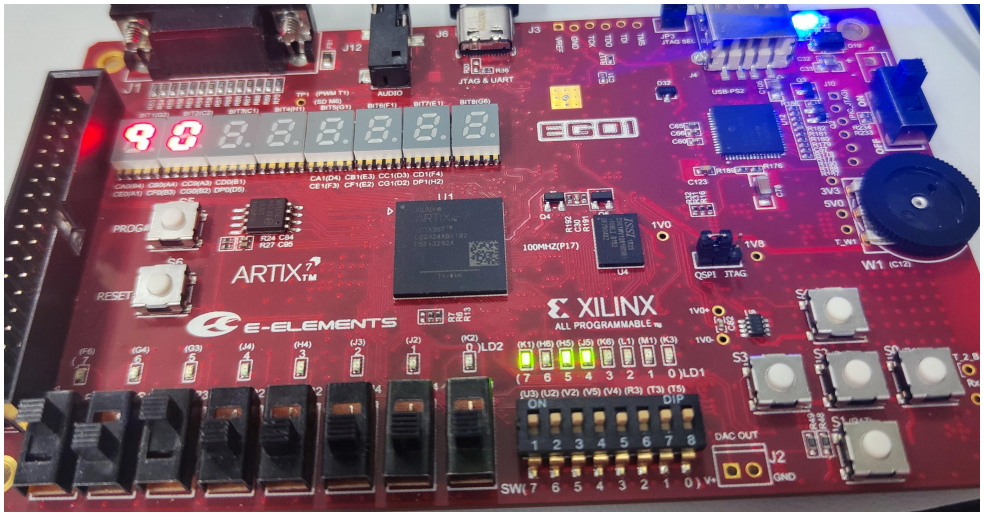

Figure 52. Switches are (1, 0, 1, 0, 0, 0, 0, 0), display “90”

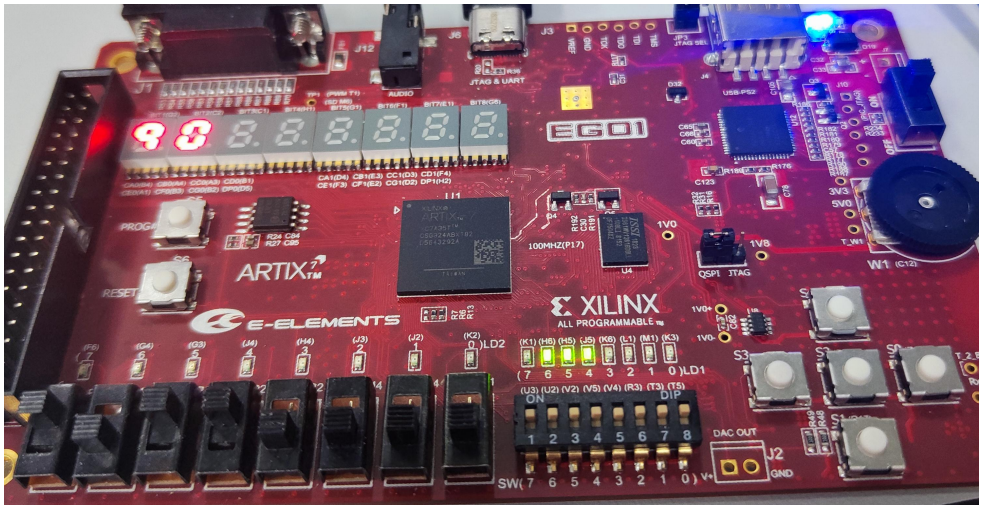

Figure 53. Switches are (1, 0, 1, 1, 0, 0, 0, 0), display “90”

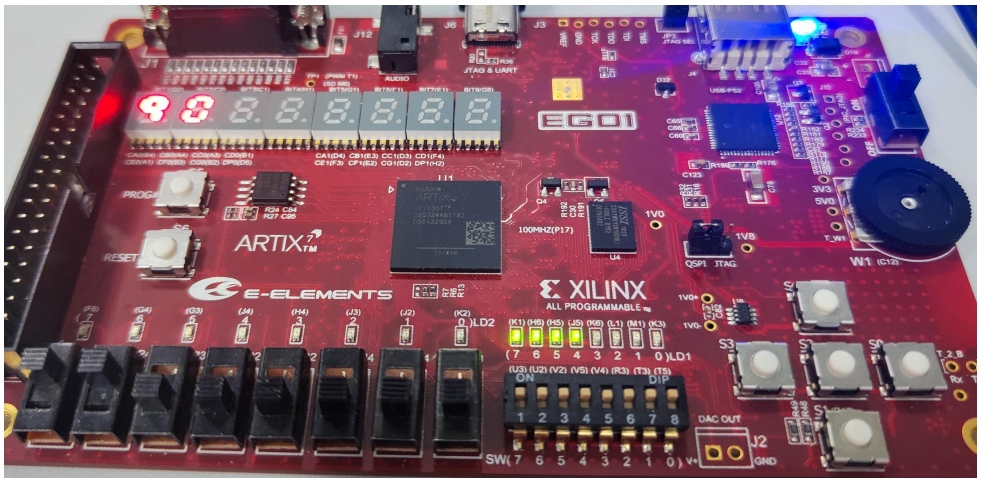

Figure 54. Switches are (1, 1, 0, 0, 0, 0, 0, 0), display “90”

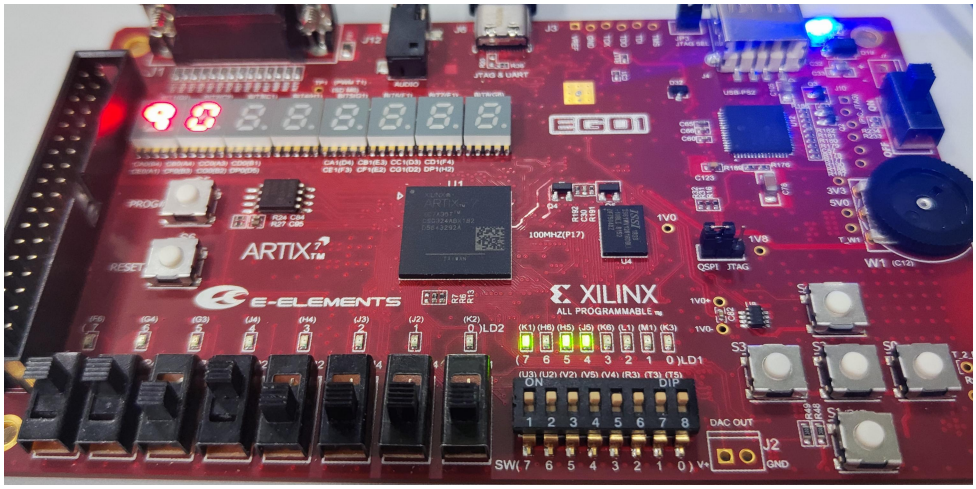

Figure 55. Switches are (1, 1, 0, 1, 0, 0, 0, 0), display “90”

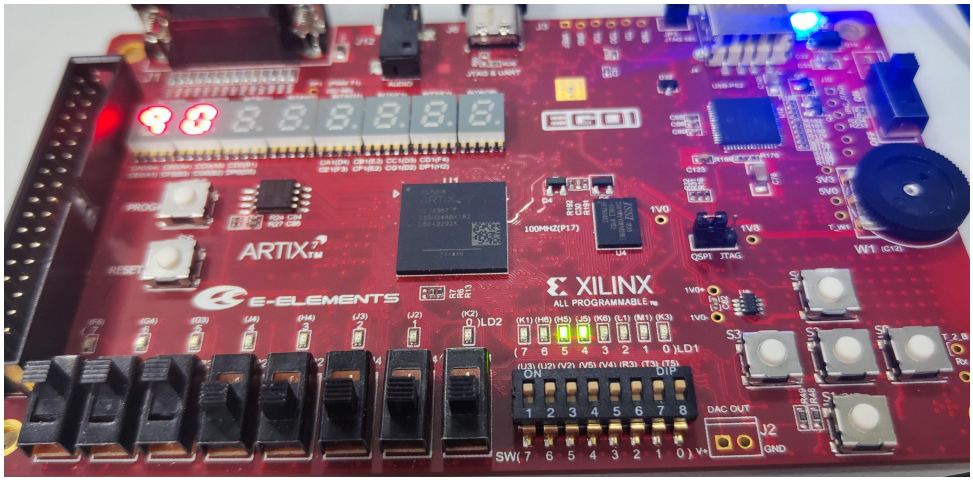

Figure 56. Switches are (1, 1, 1, 0, 0, 0, 0, 0), display “90”

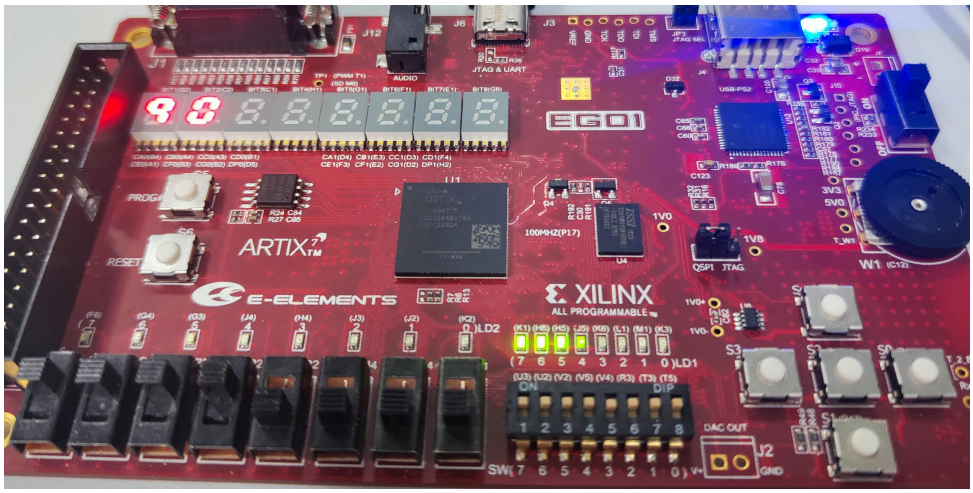

Figure 57. Switches are (1, 1, 1, 1, 0, 0, 0, 0), display “90”

- Part C: 2-digit display counts down at the frequency of 1Hz

The switches are (0, 0, 0, 1, 0, 0, 1, 1), so the initial number shown on the display is “13” (Figure 58). In Figure 59 ~ Figure 71, the number on the display decrement by one every second. At the end of the process, the display stops at “0” (Figure 72).

Figure 58. display “13”

Figure 59. display “12”

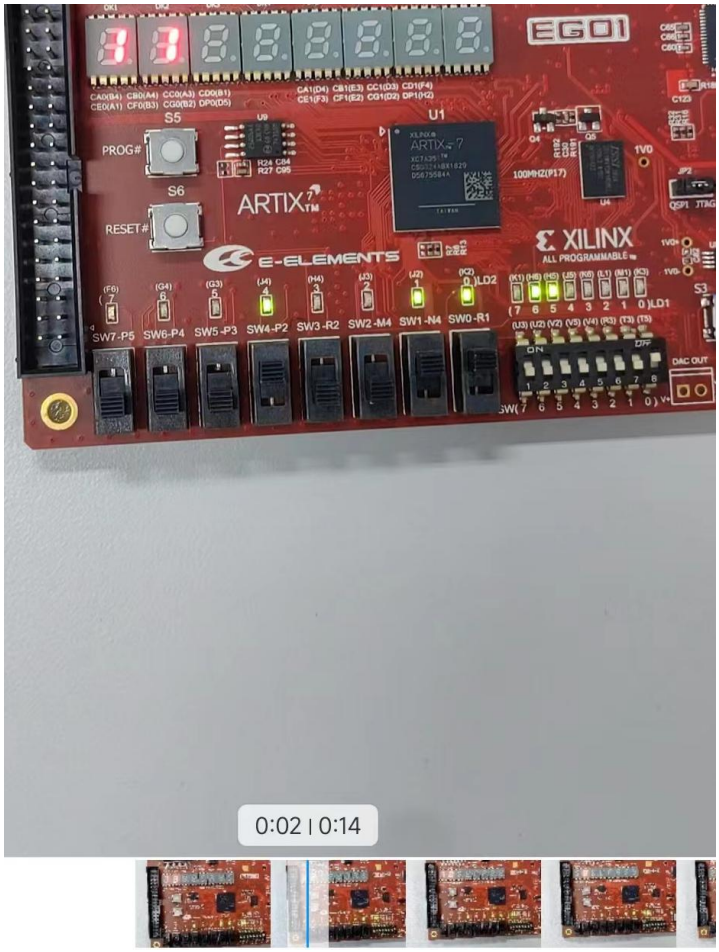

Figure 60. display “11”

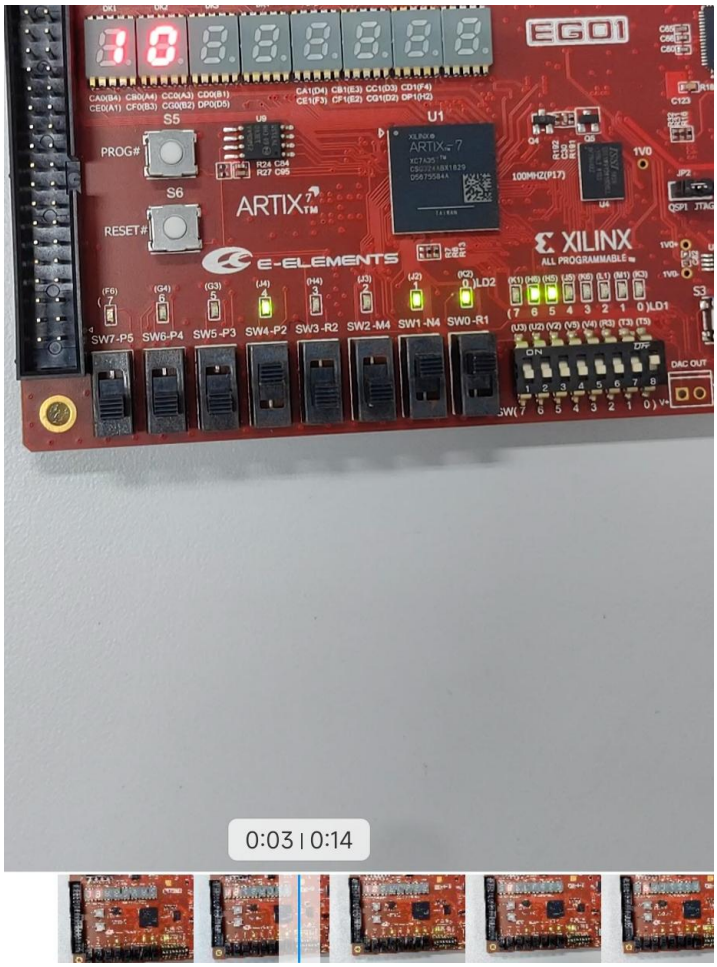

Figure 61. display “10”

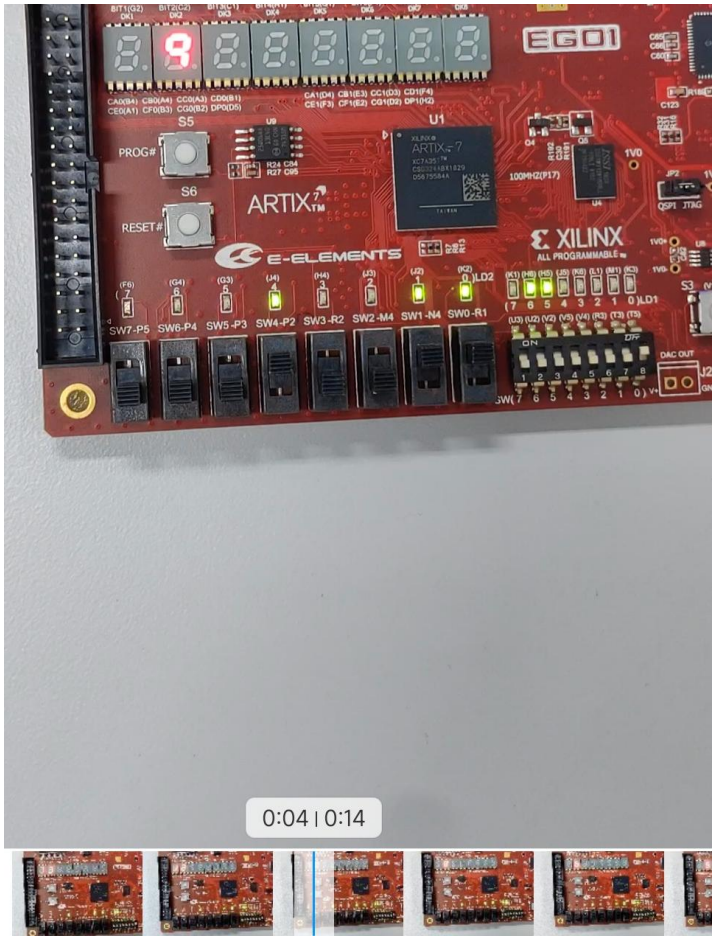

Figure 62. display “9”

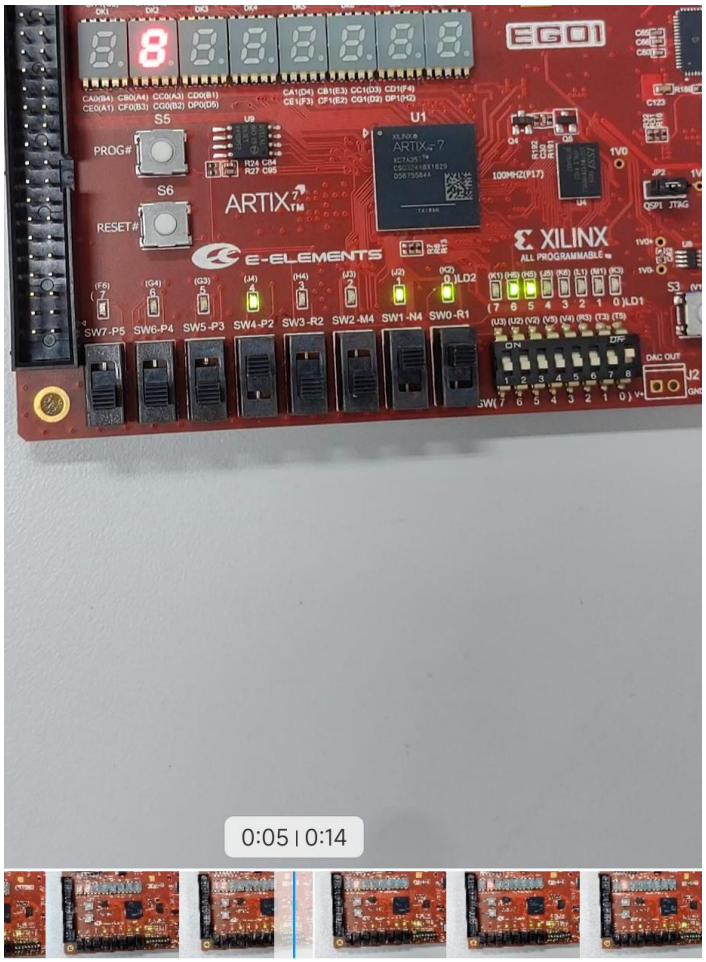

Figure 63. display “8”

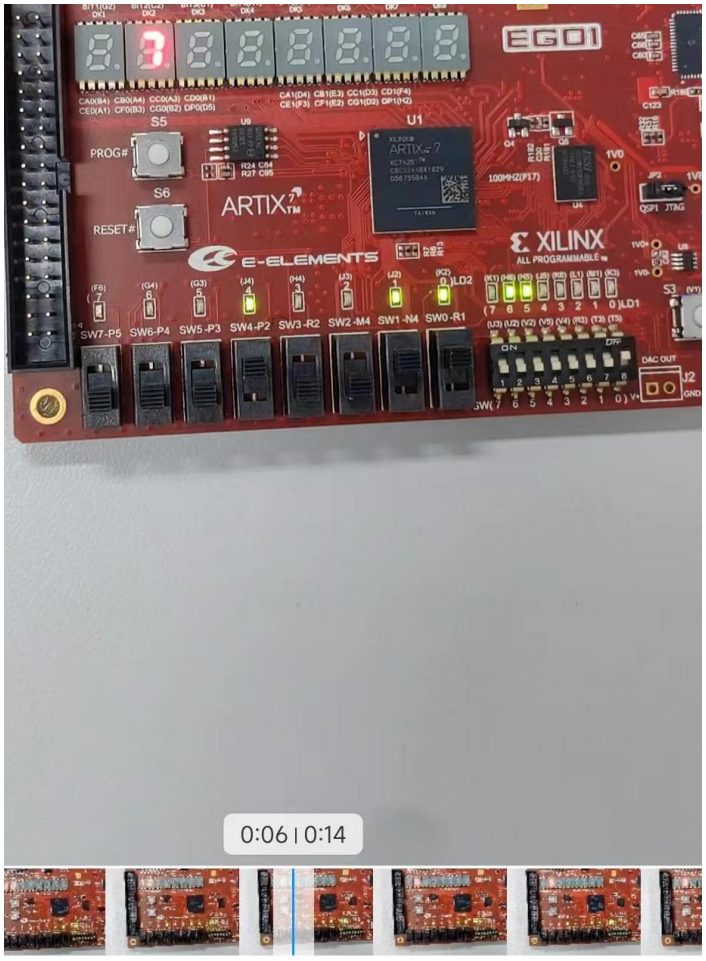

Figure 64. display “7”

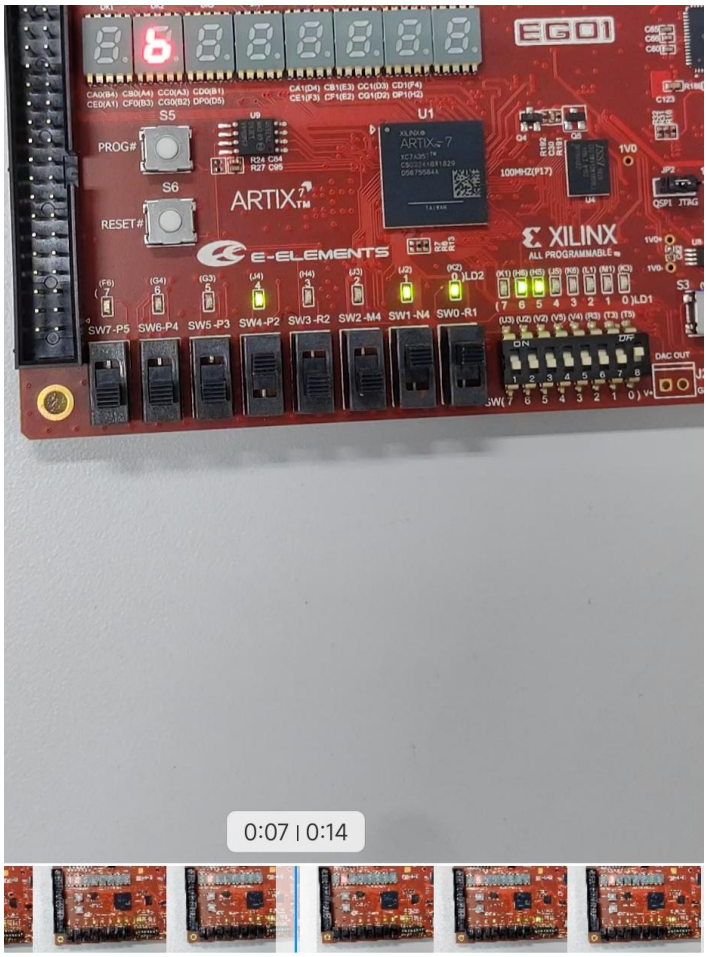

Figure 65. display “6”

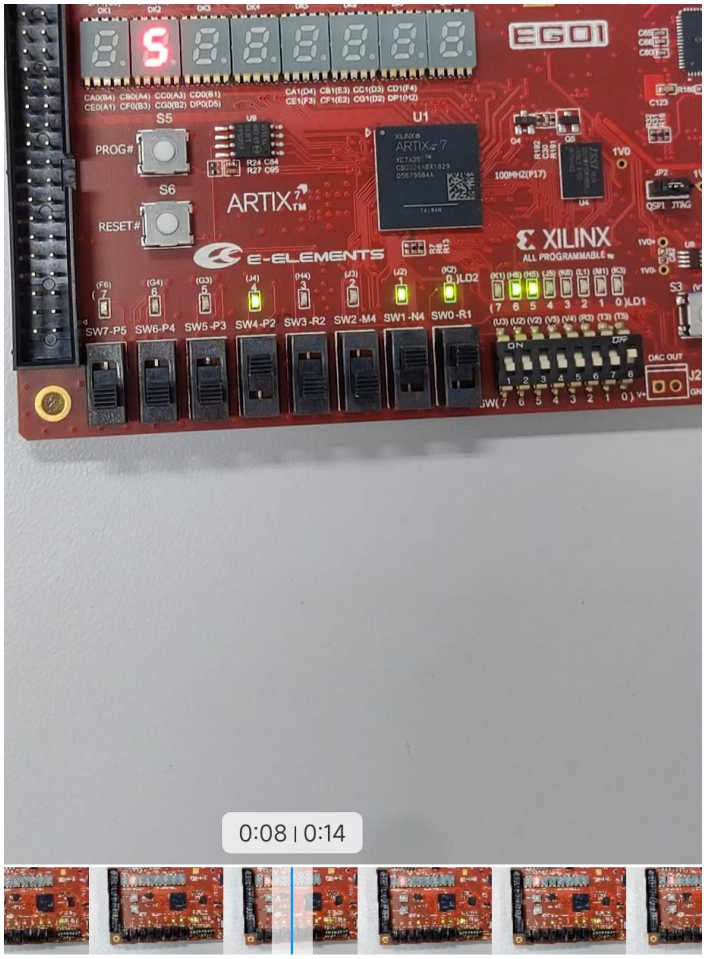

Figure 66. display “5”

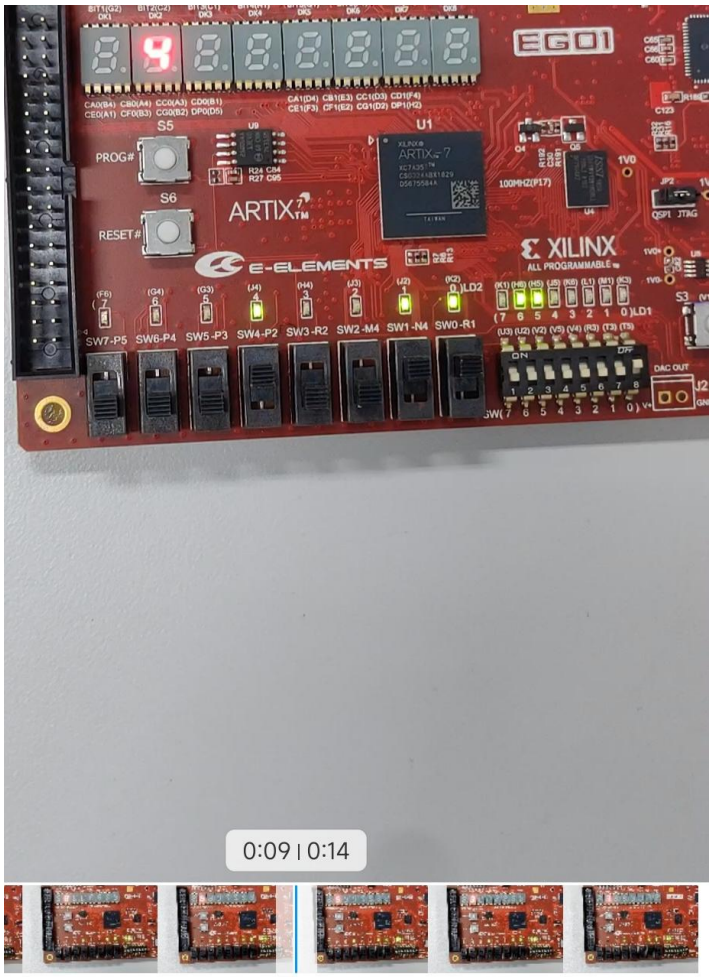

Figure 67. display “4”

Figure 68. display “3”

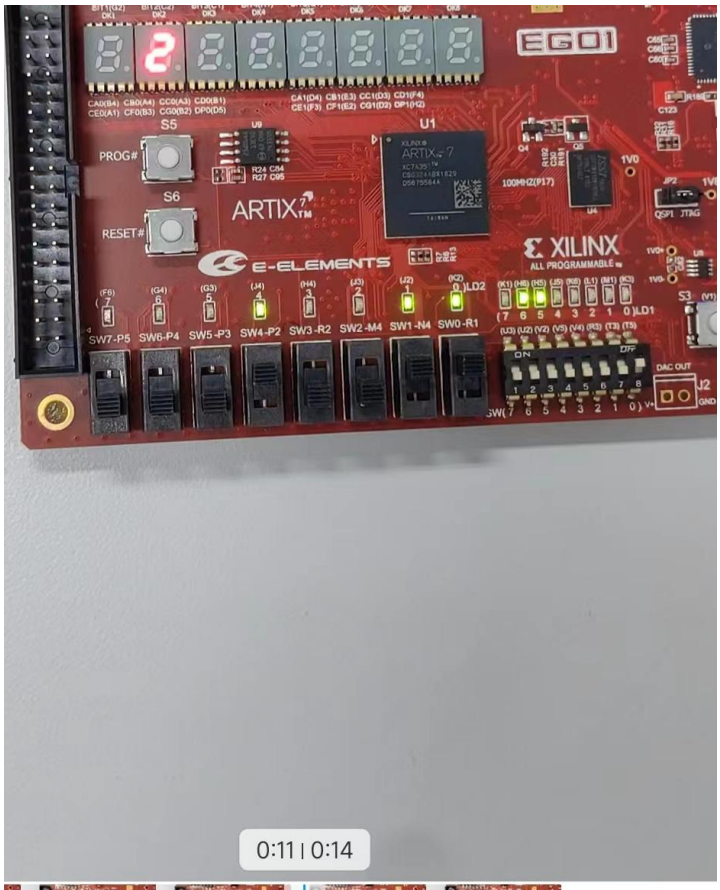

Figure 69. display “2”

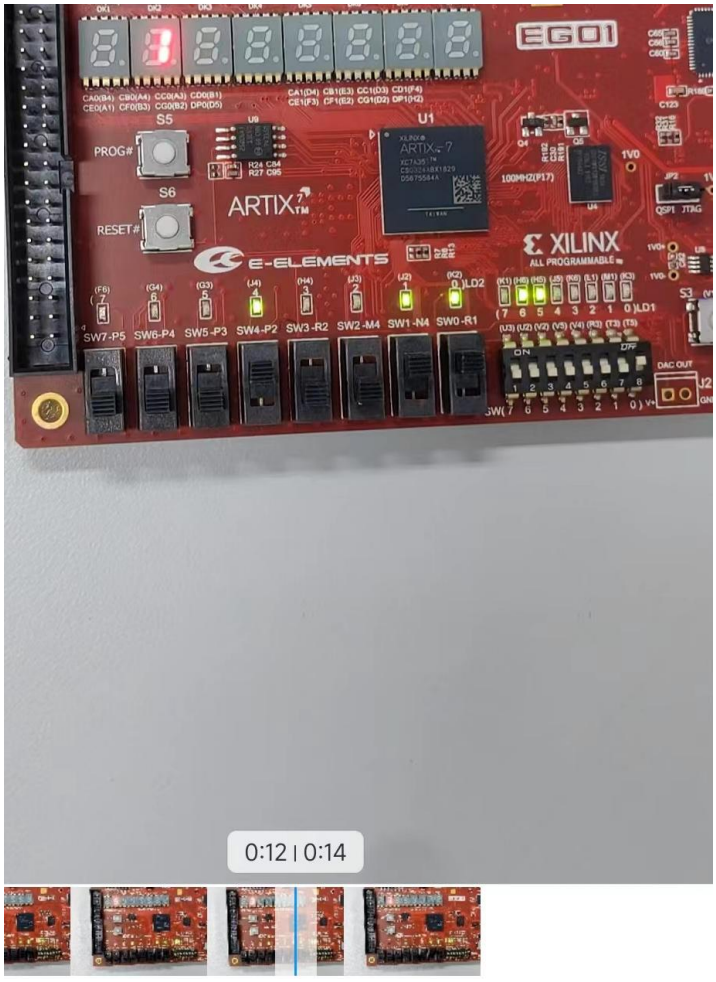

Figure 70. display “1”

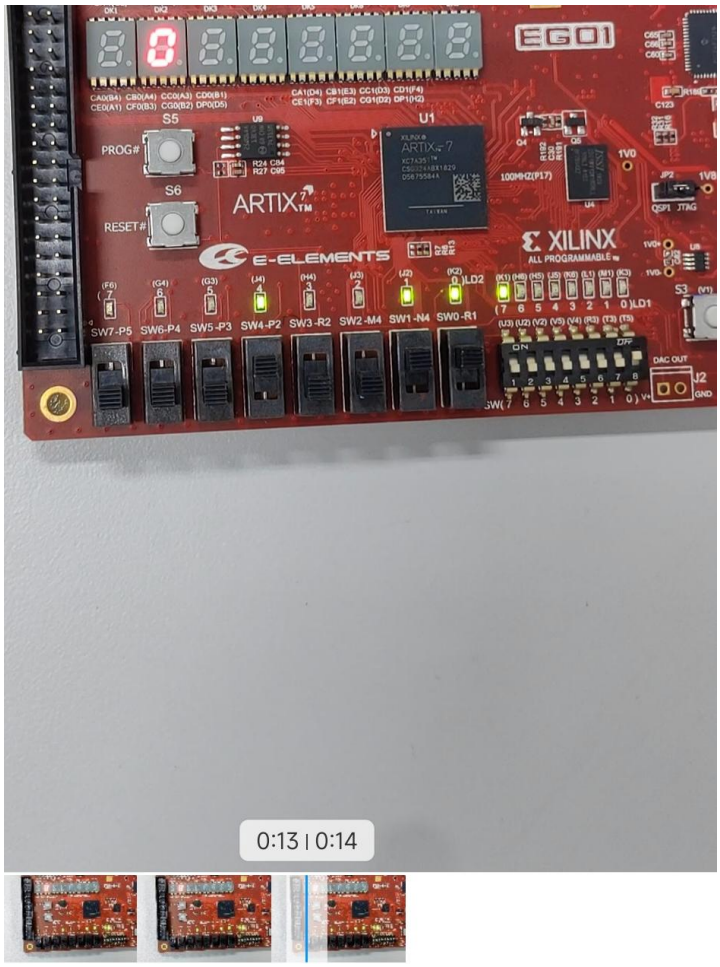

Figure 71. display “0”

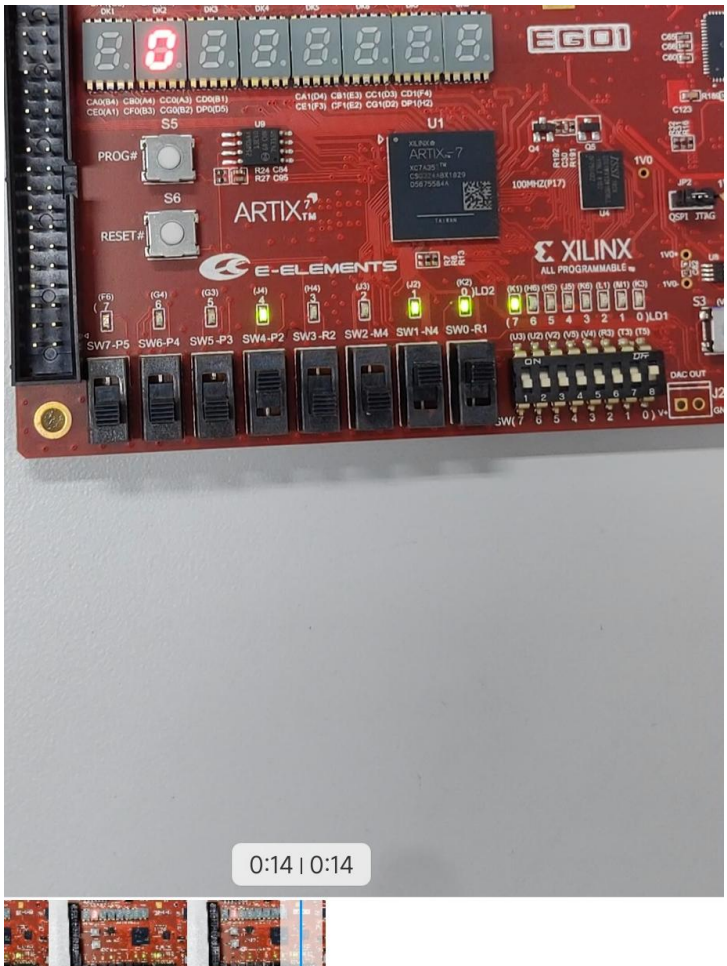

Figure 72. display “0”

2.3 Complete steps

1) Find and double click the shortcut of “Vivado” on the desktop.

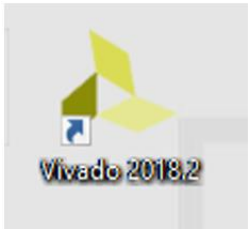

Figure 73. Step1

2) Left-click “Create Project”

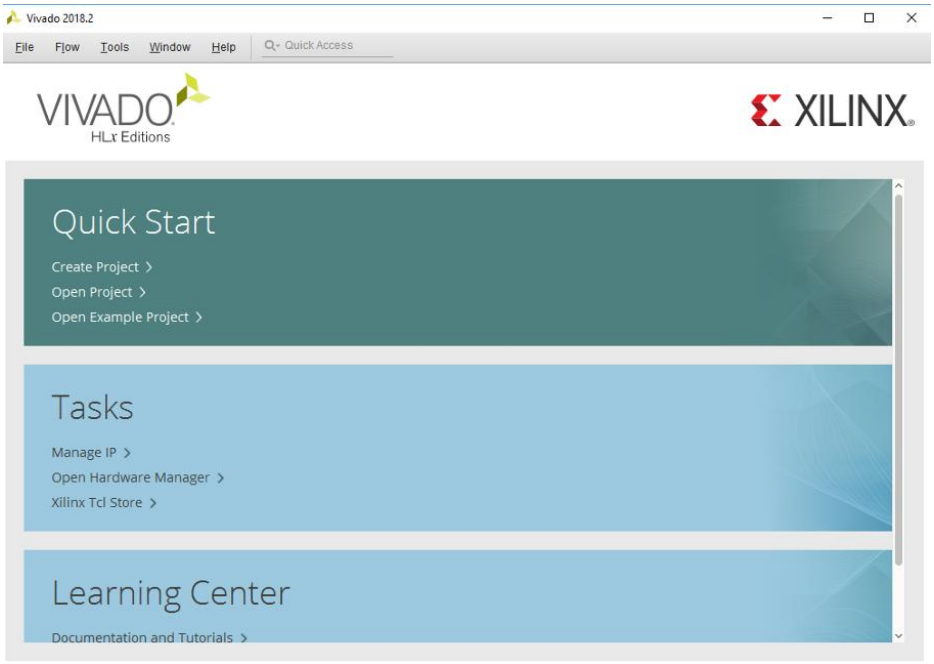

Figure 74. Step 2

3) Left-click “Next”

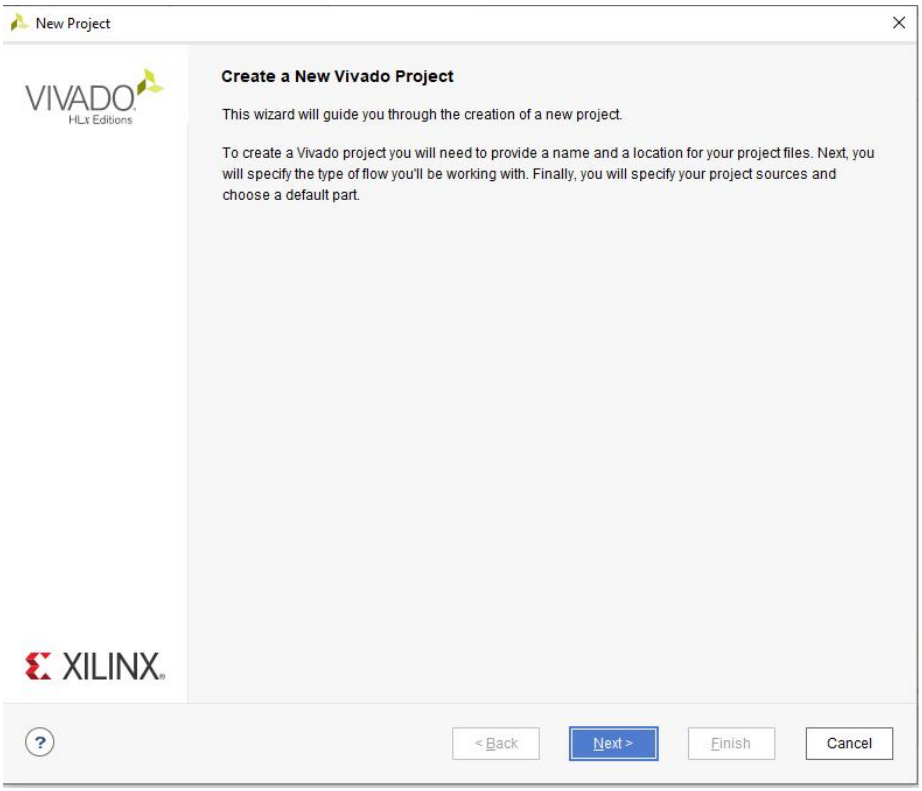

Figure 75. Step 3

4) Input the project name “myproject” => “Next”.

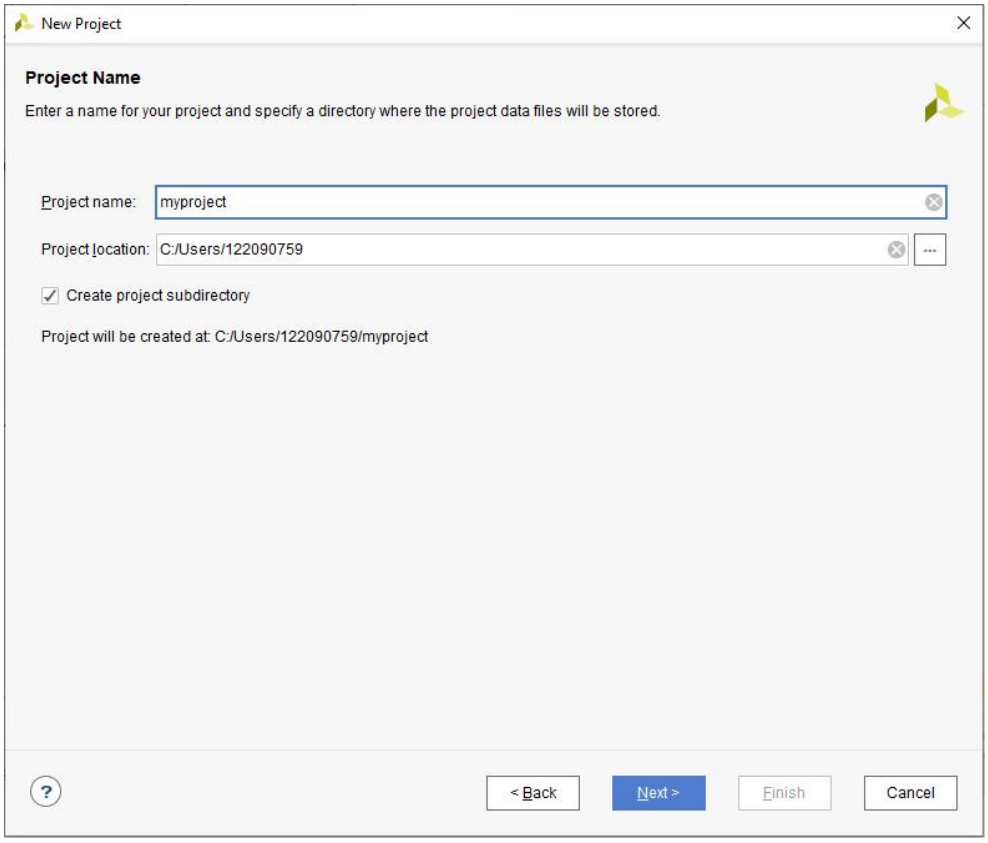

Figure 76. Step 4

5) keep the default items and click on “Next”.

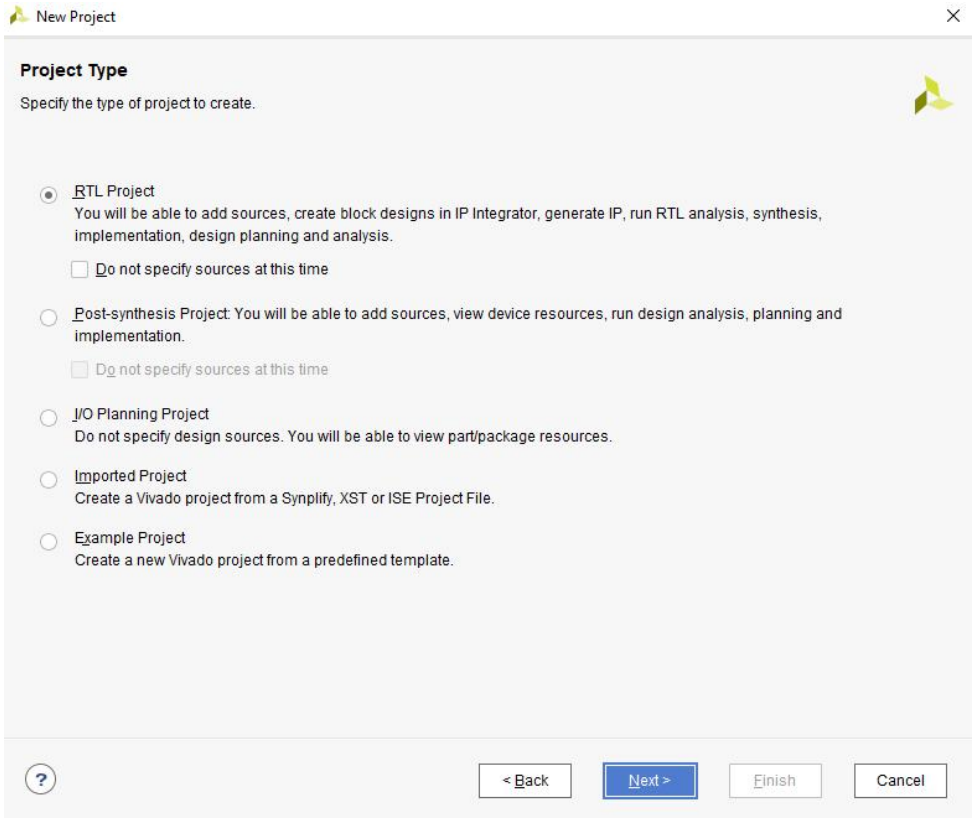

Figure 77. Step 5

6) Change the “Target language” to “VHDL”, and Click “Create File”.

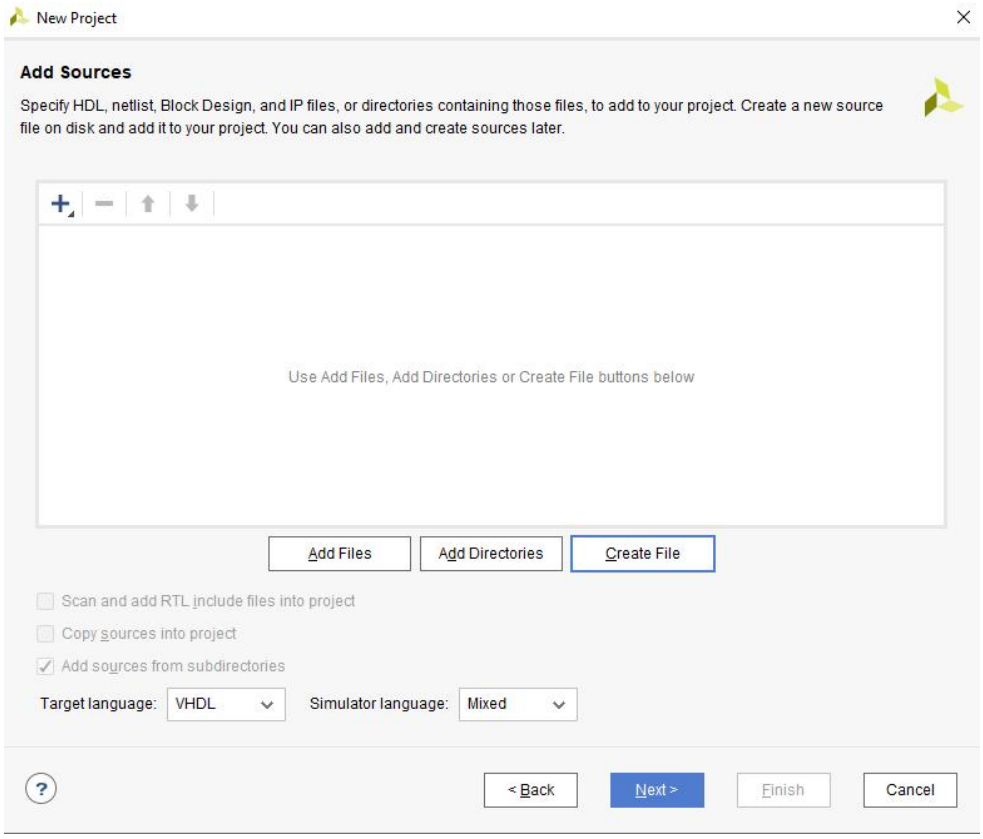

Figure 78. Step 6

7) Input the file name “part_C” => “OK” => “Next”.

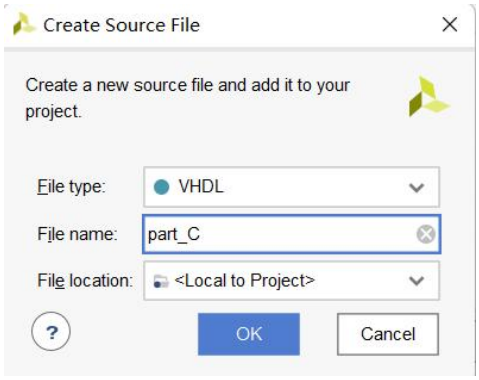

Figure 79. Step 7

8) In the “Add Existing IP (optional)” and “Add Constraint (Optional)”, both click “Next”.

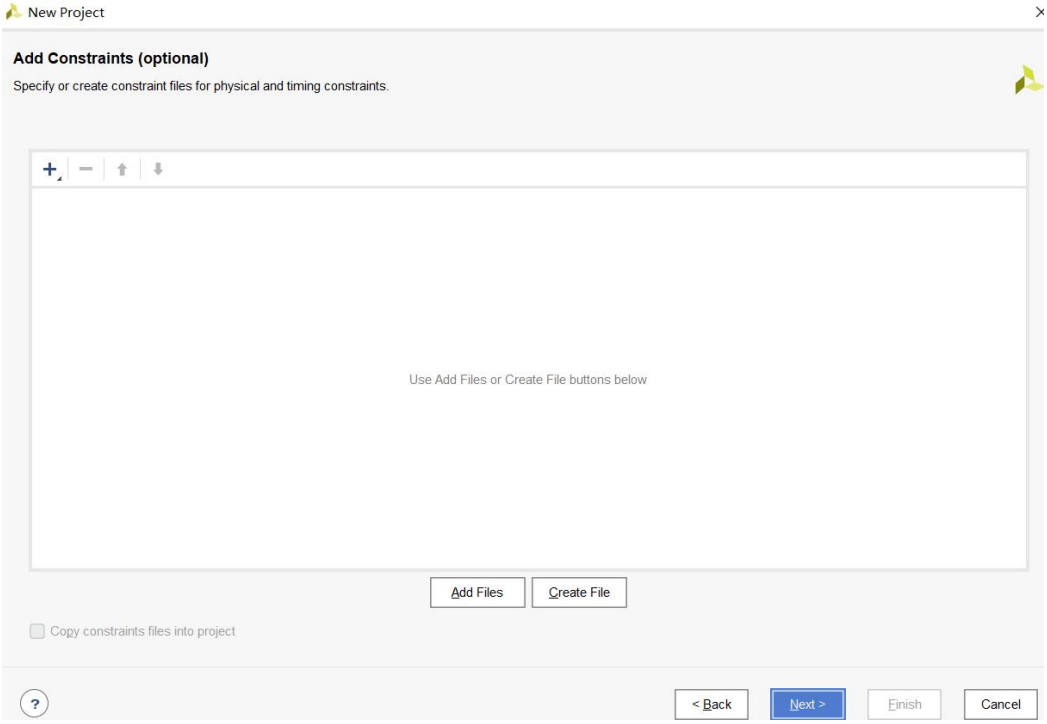

Figure 80. Step 8

9) In the “Search” textbox, input “xc7a35tcsg324-1”, and then select the corresponding chip. Click “Next” => “Finish”.

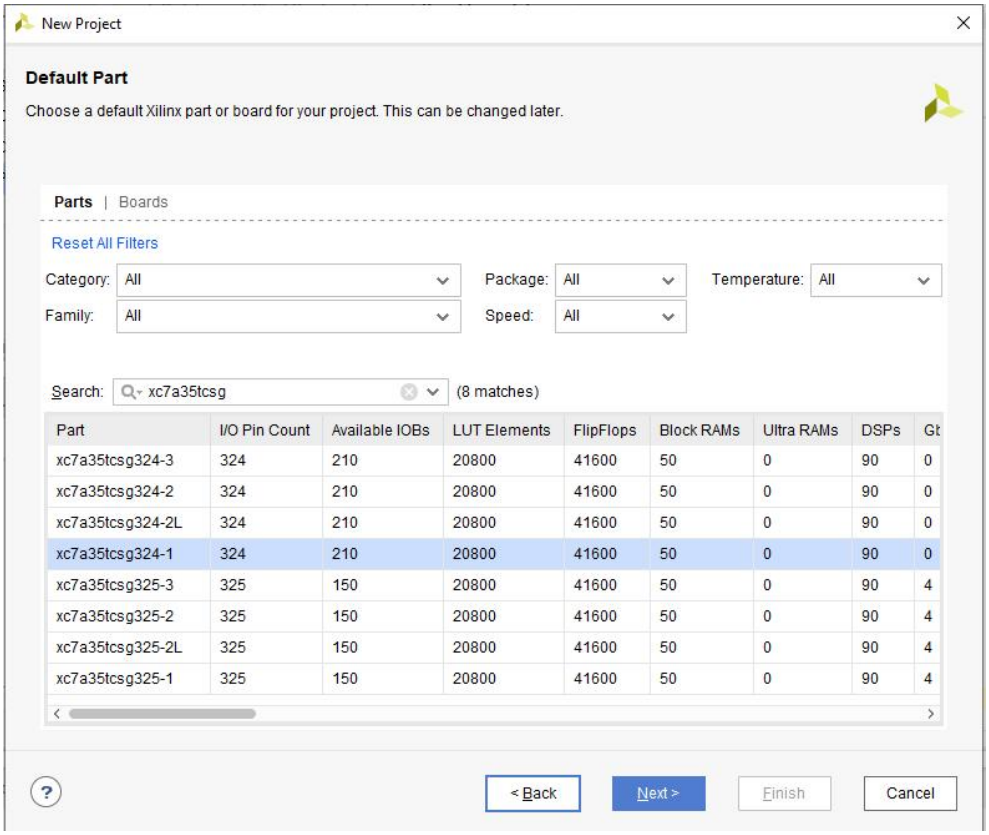

Figure 81. Step 9

10) Just click “OK” and then click “Yes”.

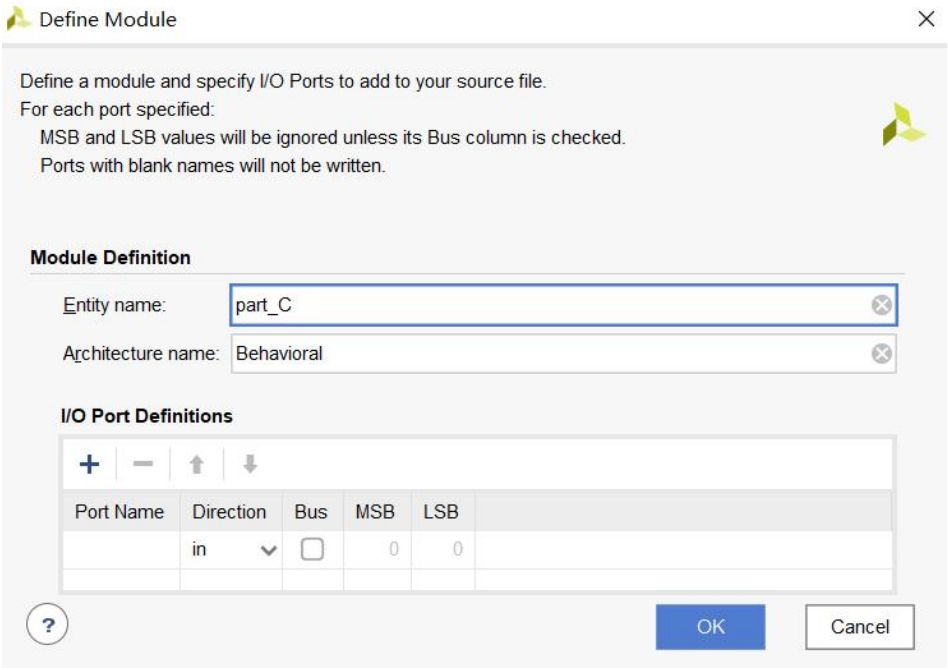

Figure 82. Step 10 (1)

Click “Yes”

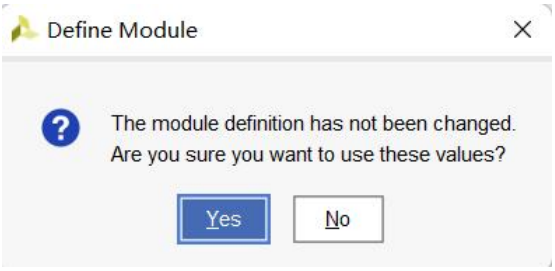

Figure 83. Step 10 (2)

11) Double click “part_C” in the “Sources” panel, you will see the automatically generated coding in the window. Modify it into the code in Figure 21. You can use “Ctrl+s” to save the coding.

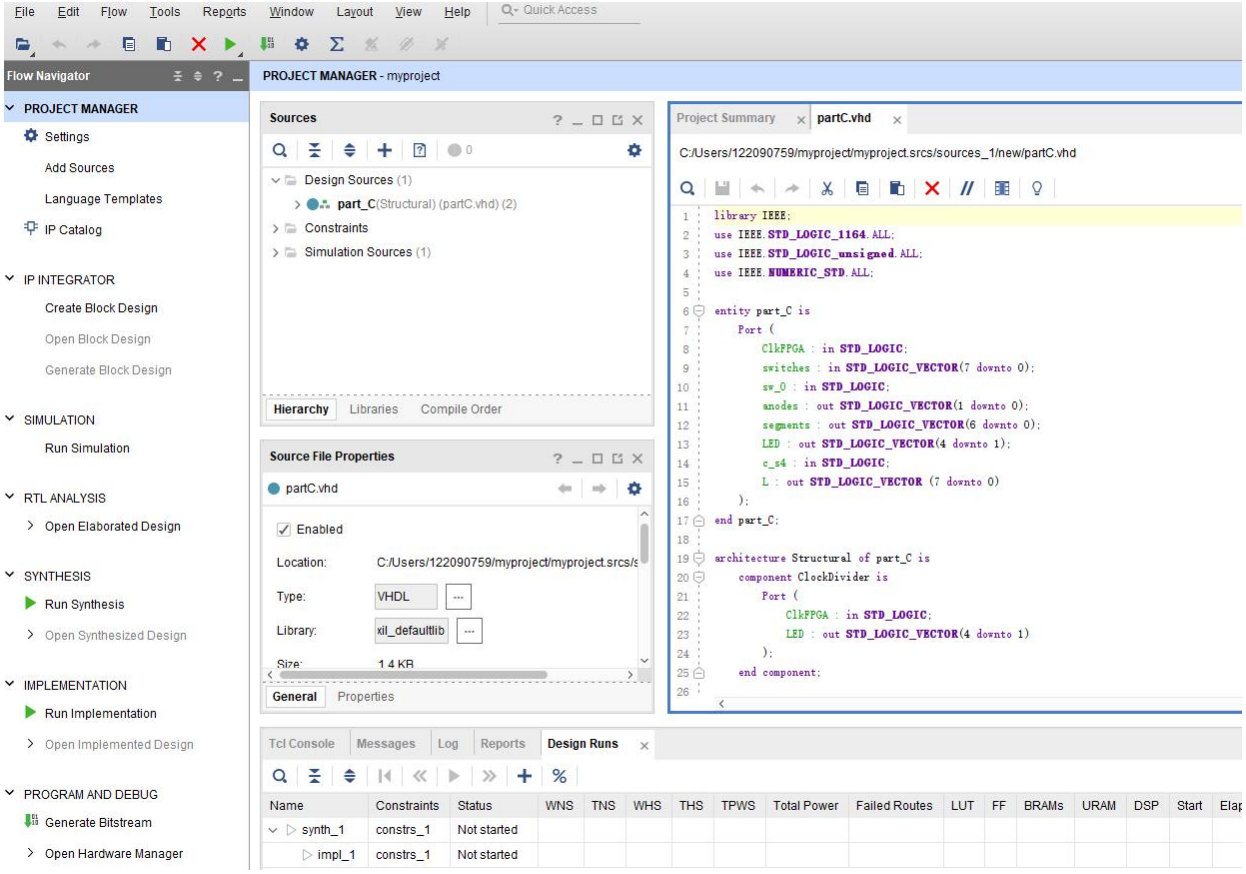

Figure 84. Step 11

12) Right-click “part_C” => left-click “Add Sources”

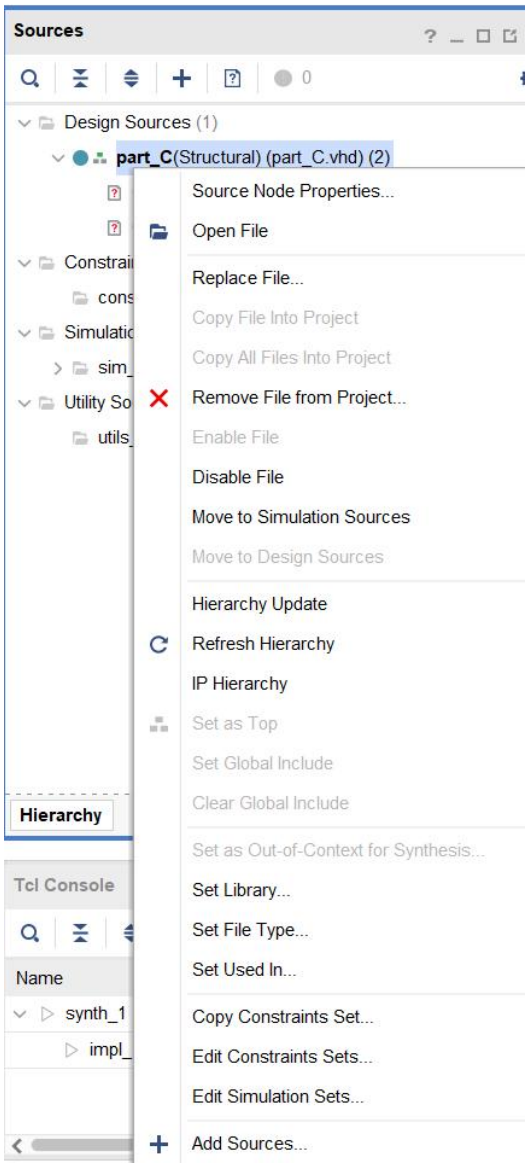

Figure 85. Step 12

13) Choose “Add or create design sources” => Click “Next”

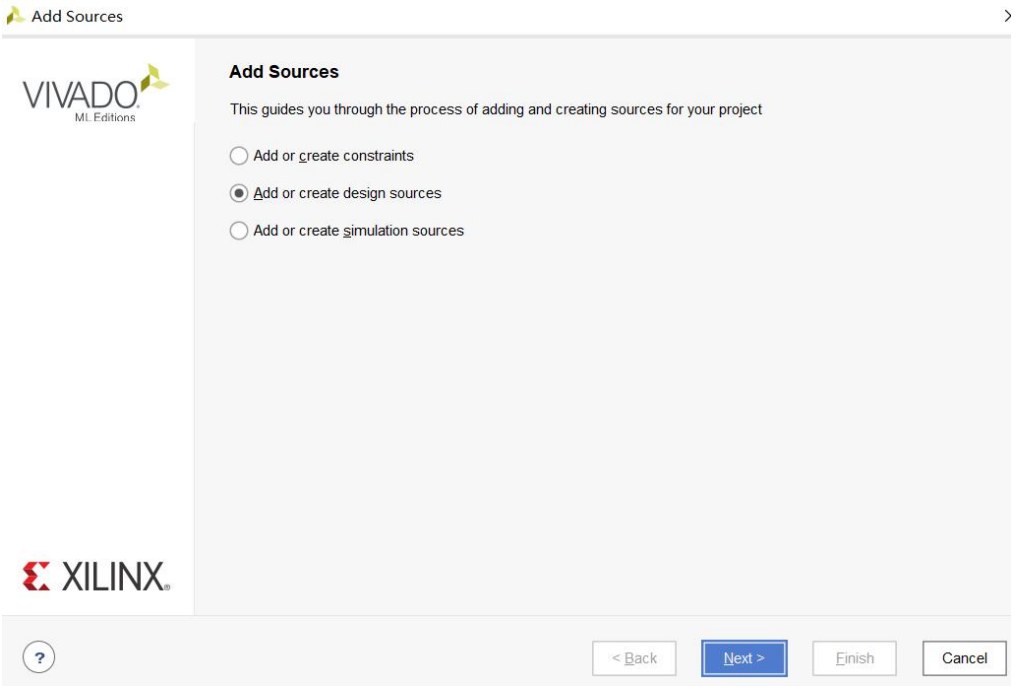

Figure 86. Step 13

14) Click “Create File” => Input the file name “CountDown” => “OK”

Click “Create File” => Input the file name “ClockDivider” => “OK”

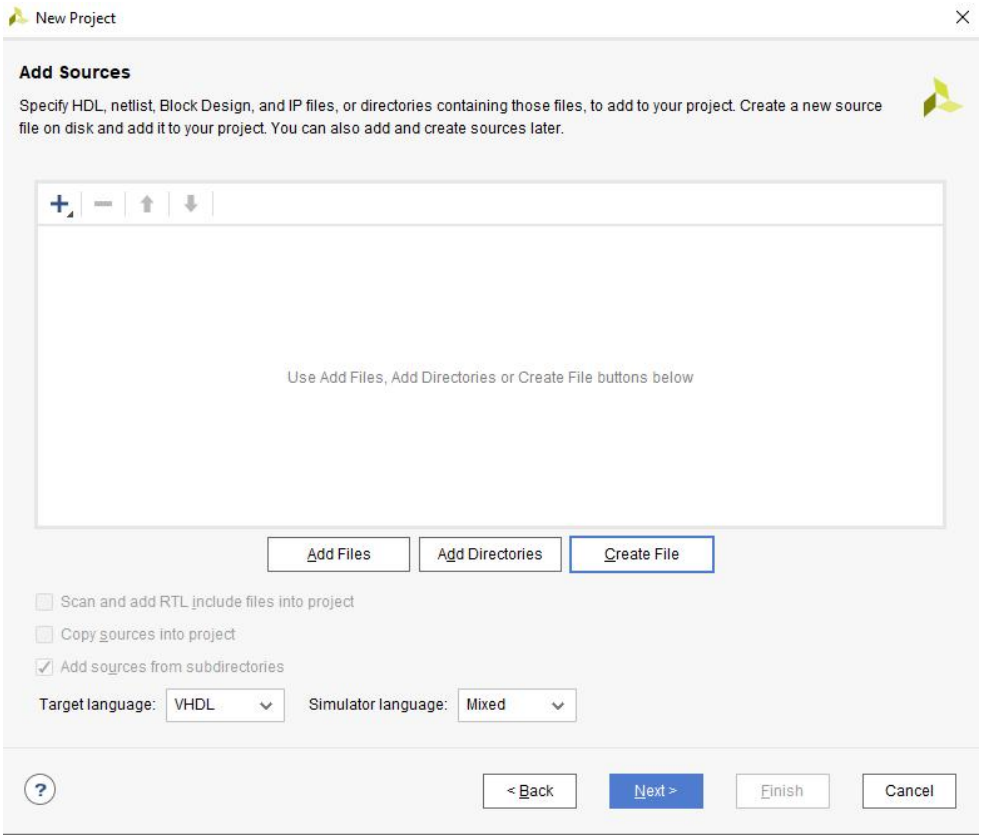

Figure 87. Step 14 (1)

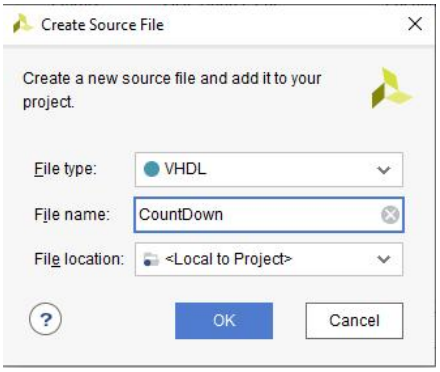

Figure 88. Step 14 (2)

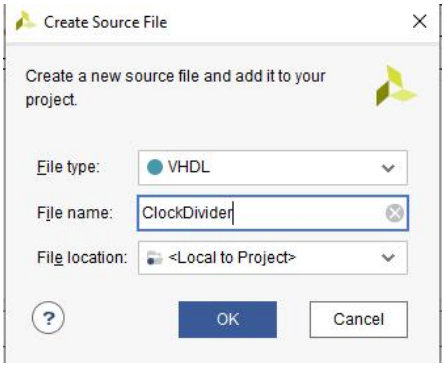

Figure 89. Step 14 (3)

15) Click both “ClockDivider” and “CountDown”

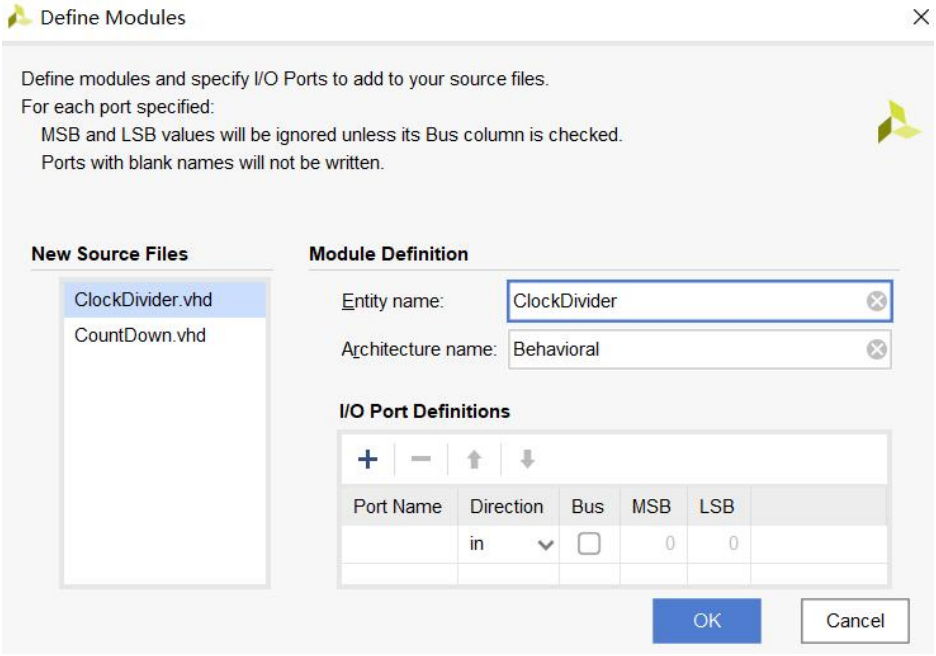

Figure 90. Step 15 (1)

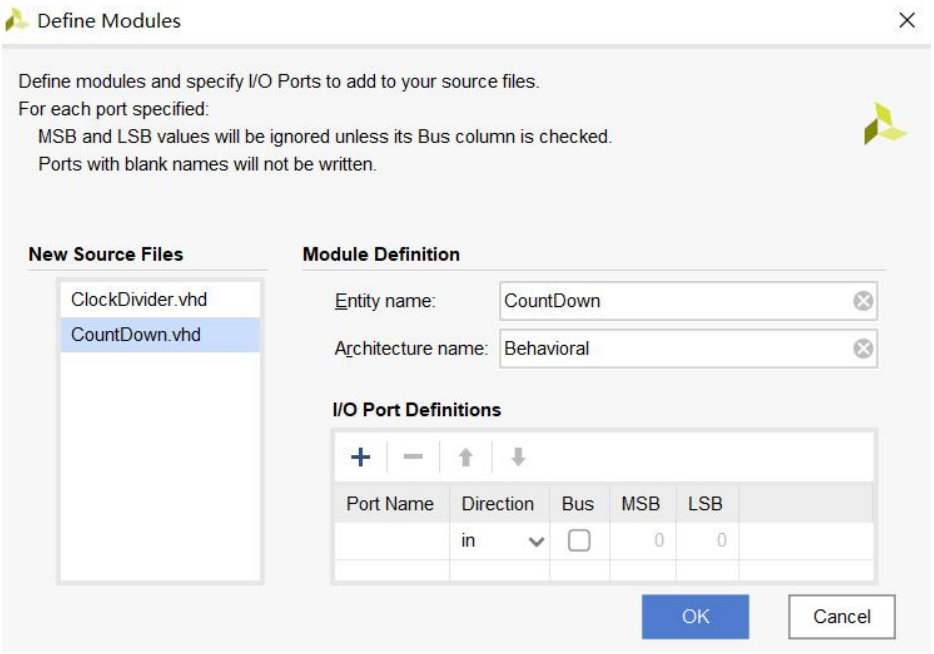

Figure 91. Step 15 (2)

Then click “OK”, click “Yes”.

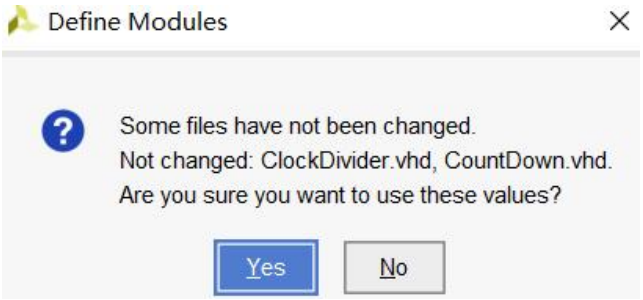

Figure 92. Step 15 (3)

Note: In this step, if you encounter this kind of dialogue box that indicating one of the files are not been visited, then you have done step 16 wrongly.

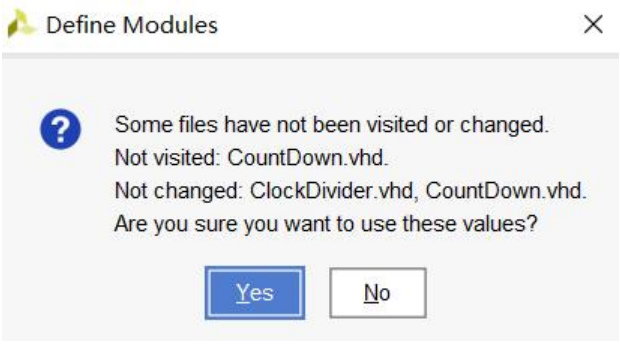

Figure 93. Step 15 (wrong)

You should see green dots instead of question marks before "ClockDivInst" and "CountdownInst". If you see question marks instead, then you have not added the two files correctly.

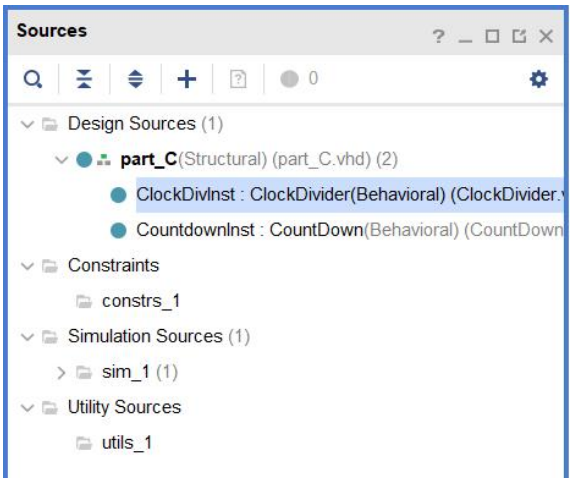

Figure 94. Check

16) Modify the ClockDivider.vhd and CountDown.vhd using code in Figure 18 and Figure 22. Use "Ctrl+s" to save the coding.

17) Click the icon for "Run synthesis".

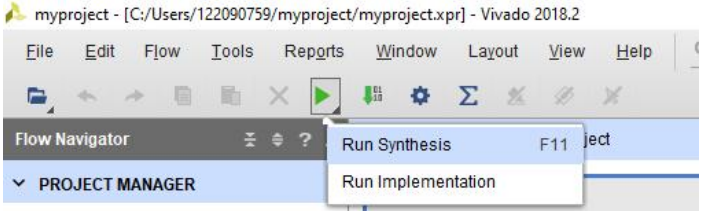

Figure 95. Step 17

18) Keep the default and Click “OK”.

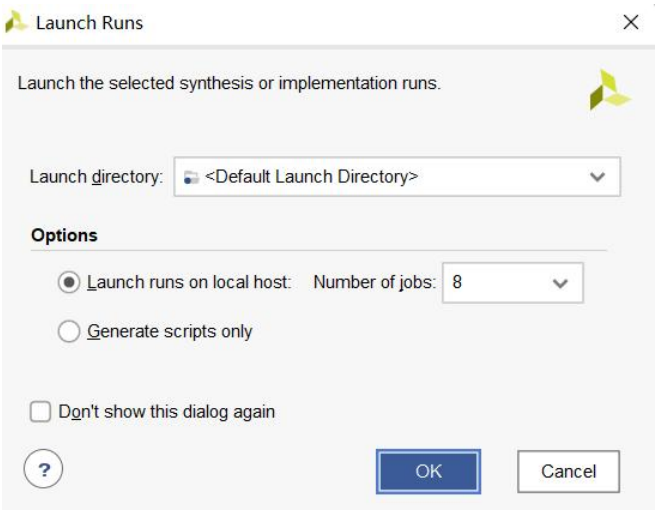

Figure 96. Step 18

There will be a progression bar on the right-up corner for a few seconds, indicating Vivado synthesis is running.

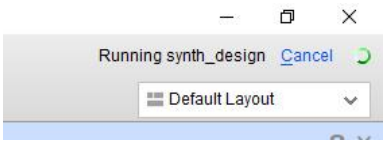

Figure 97. Synthesising

19) Then, the dialogue box below shows that the synthesis is completed. Choose “Open synthesized design” and click “OK”.

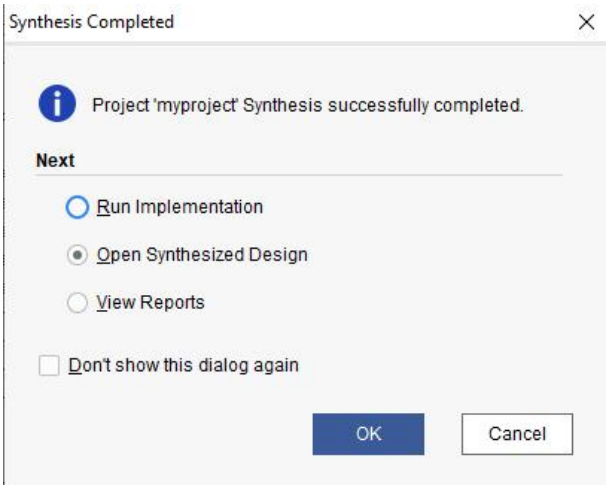

Figure 98. Step 19

20) From the dropdown menu on the right-up corner, select “I/O Planning”.

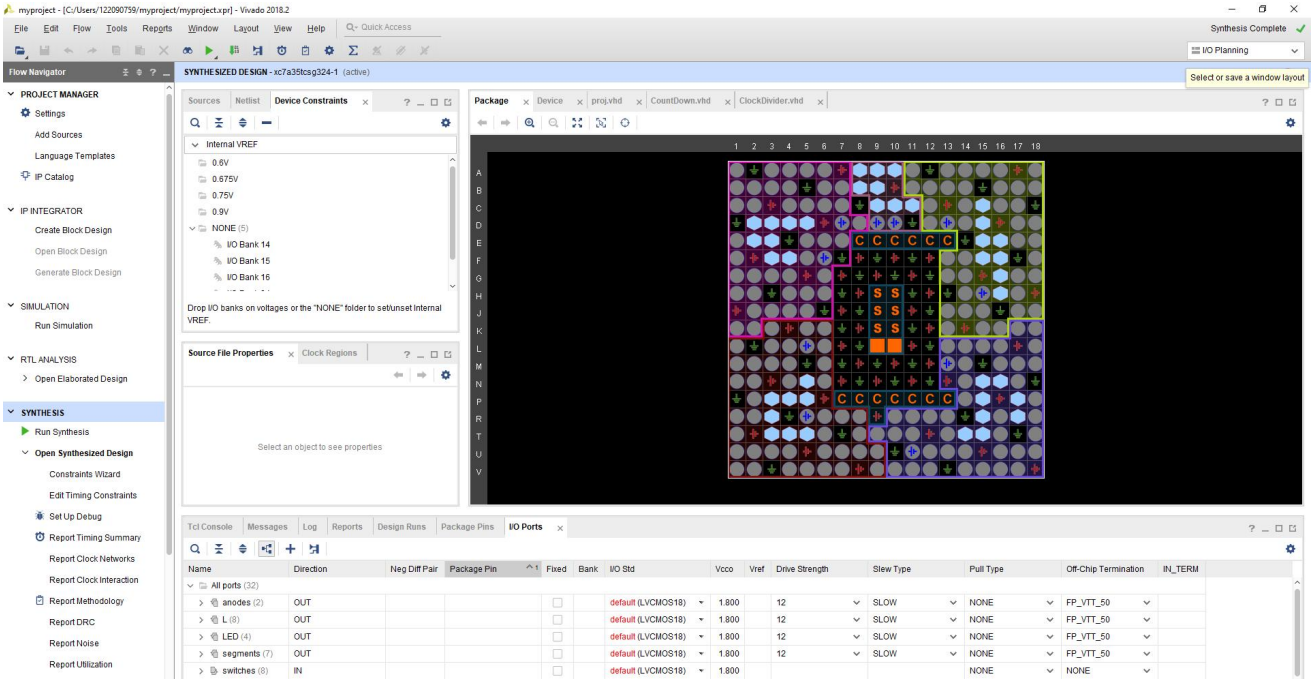

Figure 99. Step 20

21) In the Synthesized Design, assign the “Site” of each input pins and output pins. Set the “I/O Std” to “LVCMOS33*”.

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
All ports (32)													
anodes (2)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
anodes[1]	OUT		C2	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
anodes[0]	OUT		G2	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
L (8)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
L[7]	OUT		F6	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
L[5]	OUT		G3	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
L[6]	OUT		G4	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
L[3]	OUT		H4	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
L[1]	OUT		J2	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
L[2]	OUT		J3	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
L[4]	OUT		J4	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
L[0]	OUT		K2	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
LED (4)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
LED[3]	OUT		H5	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
LED[2]	OUT		H6	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
LED[4]	OUT		J5	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
LED[1]	OUT		K1	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
segments (7)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	

Figure 100. Step 21 (1)

Name	Direction	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Termination	IN_TERM
LED[1]	OUT		K1	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
segments (7)	OUT			☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
segments[2]	OUT		A1	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
segments[4]	OUT		A3	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
segments[5]	OUT		A4	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
segments[3]	OUT		B1	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
segments[0]	OUT		B2	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
segments[1]	OUT		B3	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	
segments[6]	OUT		B4	☑	35	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50	

Figure 101. Step 21 (2)

switches (8)	IN			☑	34	LVCMOS33*	3.300				NONE	NONE	
switches[2]	IN		M4	☑	34	LVCMOS33*	3.300				NONE	NONE	
switches[1]	IN		N4	☑	34	LVCMOS33*	3.300				NONE	NONE	
switches[4]	IN		P2	☑	34	LVCMOS33*	3.300				NONE	NONE	
switches[5]	IN		P3	☑	34	LVCMOS33*	3.300				NONE	NONE	
switches[6]	IN		P4	☑	34	LVCMOS33*	3.300				NONE	NONE	
switches[7]	IN		P5	☑	34	LVCMOS33*	3.300				NONE	NONE	
switches[0]	IN		R1	☑	34	LVCMOS33*	3.300				NONE	NONE	
switches[3]	IN		R2	☑	34	LVCMOS33*	3.300				NONE	NONE	
Scalar ports (3)													
ClkFPGA	IN		P17	☑	14	LVCMOS33*	3.300				NONE	NONE	
sw_0	IN		T5	☑	34	LVCMOS33*	3.300				NONE	NONE	
c_s4	IN		U4	☑	34	LVCMOS33*	3.300				NONE	NONE	

Figure 102. Step 21 (3)

22) Use “Ctrl+s” to save the constraint file. In the dialogue box, input “File name”, and click “OK”.

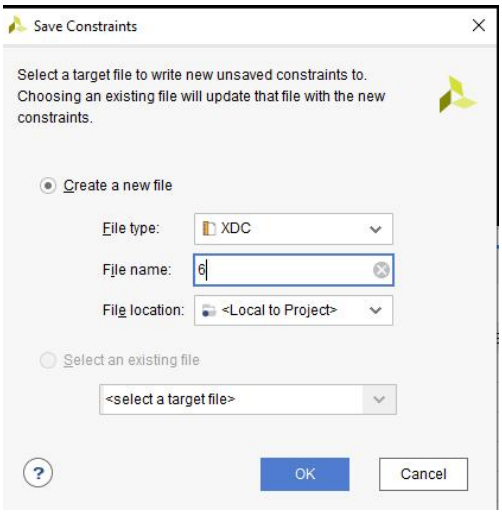

Figure 103. Step 22

23) Click the icon for “Run implementation”. It will ask you to run synthesis again. Click “OK”.

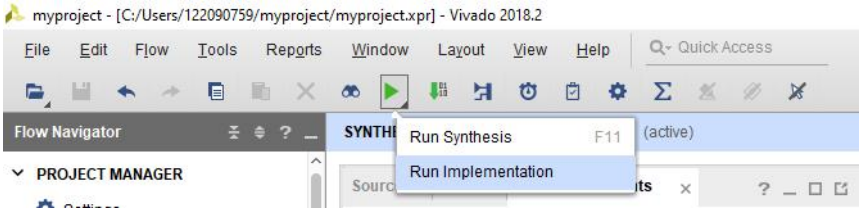

Figure 104. Step 23

Wait for the implementation done.

24) When the following dialogue box prompts out, choose “Generate Bitstream” and click “OK”.

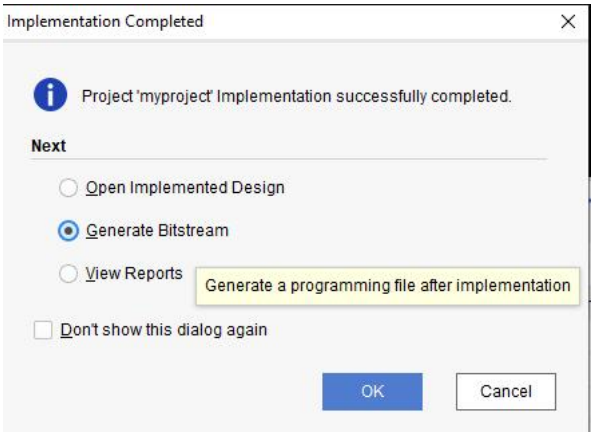

Figure 105. Step 24

It will take a few seconds, and the progress bar will be shown in the right upper corner.

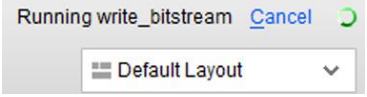

Figure 106. Generating Bitstream

25) When it is completed, select “Open Hardware Manager” and click “OK”.

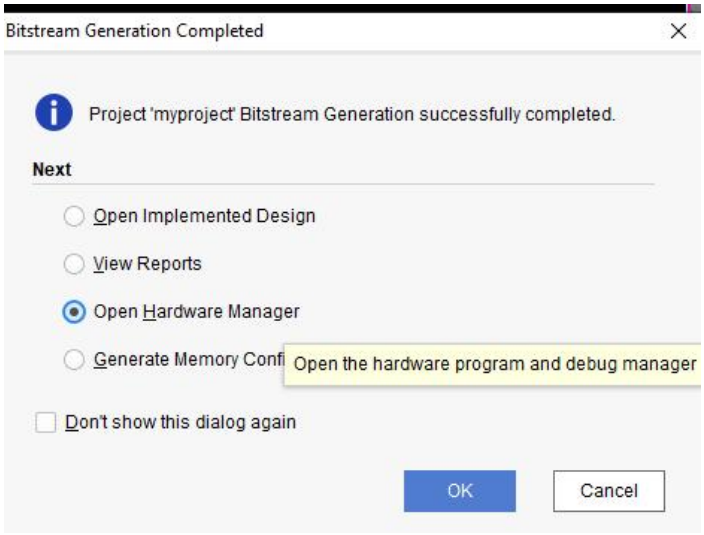

Figure 107. Step 25

26) Connect the J6 port of the development board with the computer through USB, and then turn on the switch. You will see the 7-Segments are all 0, with other 4 LEDs on the upper part of the board on.
It will take from a few seconds to a few minutes to install the driver.

27) Click the “Open target” in the “Hardware Manager”, and then click “Auto Connect”.

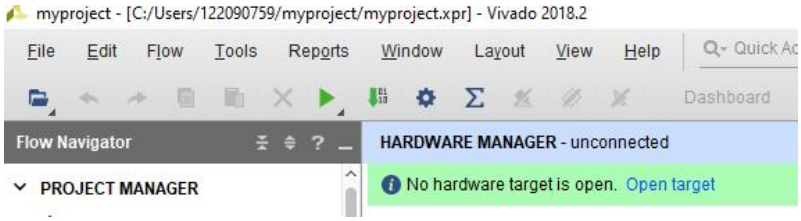

Figure 108. Step 27 (1)

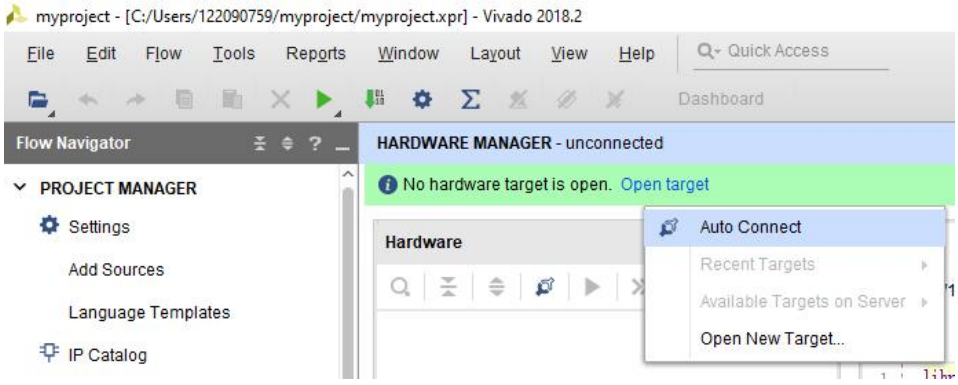

Figure 109. Step 27 (2)

When you see the interface below, it indicates that the hardware connection is successful.

Figure 110. Successful hardware connection (1)

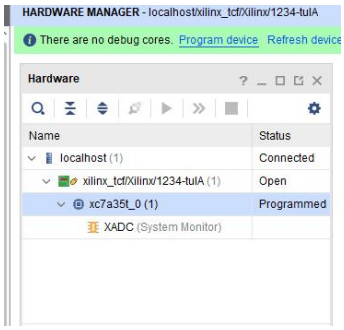

Figure 111. Successful hardware connection (2)

28) Click “Program device” to download coding the FPGA.

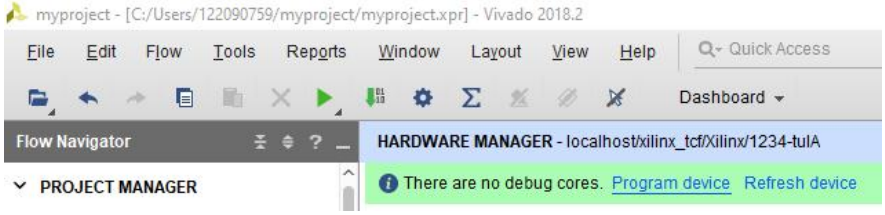

Figure 112. Step 28

29) Click “Program”.

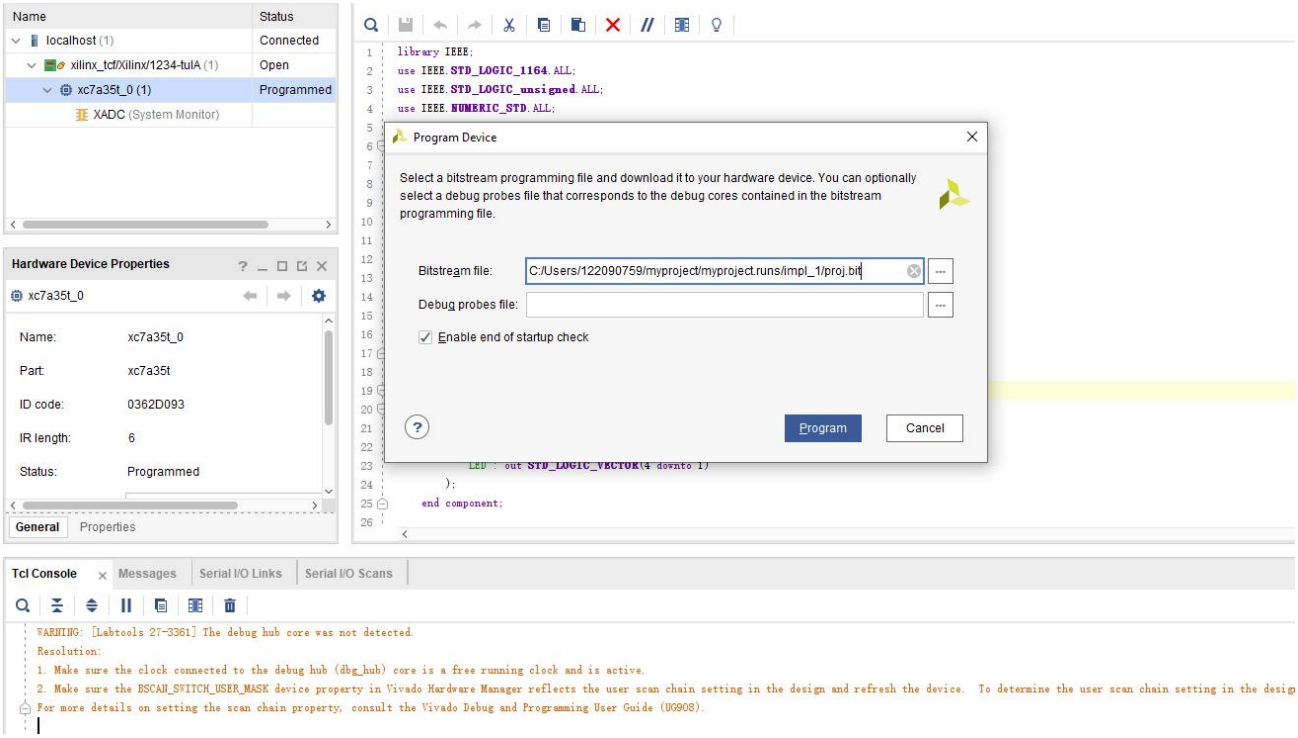

Figure 113. Step 29

30) When unplug the USB or turn off the development board, always remember to close “Hardware Manager” first.

2.4 Questions

Design the code of a count-down timer for 2 hours with the system clock of 100MHz. When the start-button is pressed, the timer starts. When the 2-hour period is completed, counter stops and an LED is turned on.

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

entity timer is
  port(
        clk: in  std_logic;            -- 100MHZ
        start_btn: in  std_logic;      -- Start button input
        LED: out std_logic             -- LED output
    );
end timer;

architecture Behavioral of timer is
    constant TimerDurationSec : integer := 7200;
    constant ClockFrequencyHz : integer := 100000000;
    signal counter : integer range 0 to TimerDurationSec := TimerDurationSec;
    signal n : integer range 0 to ClockFrequencyHz - 1 := 0;

begin
    process(clk)
    begin
        if rising_edge(clk) then
            if start_btn = '1' then
                if n = ClockFrequencyHz-1 then  --1 second
                    if counter > 0 then  -- Timer is running
                        counter <= counter - 1;
                    else
                        counter <= 0;
                    end if;
                    n <= 0;
                else
                    n <= n+1;
                end if;
            end if;
        end if;
    end process;

    LED <= '1' when counter = 0 else '0';
end Behavioral;
```

Figure 114. Code of countdown timer

3. Conclusion

In this lab, I

- Programme Xilinx FPGA for AND gate and XOR gate.
- Programme Xilinx FPGA to divide high-frequency clocks into low frequency clocks.
- Programme Xilinx FPGA for multiple 7-segment outputs simultaneously displayed.
- Programme Xilinx FPGA for a count-down timer.

I learned:

- The basics of the VHDL language and Vivado.
- How to show the 2-digits simultaneously in two 7-segment displays. This can be done by iteratively showing different values of onto the two 7-segments, and switching from them at a relatively high frequency that our eyes can not identify.
- How to generate lower frequency signals from higher frequency signals. Use integer counters to divide the high frequency clock signal. Each counter increments on every clock cycle. When a counter reaches its respective divisor value, toggles the corresponding led_state signal and resets the counter to zero.