Mustafa Cankan BALCI

22101761

Section 1

22 November 2022

Lab 5: Seven-Segment Display

Purpose

This experiment will aim to use the 4-bit seven segment display on BASYS 3 for showing the hexadecimal numbers simultaneously.

Design Specifications

The seven segment display contains 7 light emitting diodes(LED). The seven segment display has two types: anode and cathode. BASYS 3 has a common anode 4-digit display, which is combination of four 1 digit seven segment display. 4-digit seven segment display contains four extra pins to determine which digit is going to show number. The type of BASYS 3 board's 4-digit seven segment display is anode, so it has 4 anode pins.

A four bit counter is designed for determining the hexadecimal value which is going to show in seven segment. The clock of the 4-bit counter is 100 MHz, which is the default internal clock speed of BASYS 3.

The lab module contains 2 inputs and 10 outputs. The inputs are clock signal and reset. The reset is assigned to V17 button, and the clock signal is defined to W5 pin. The outputs consists of 4 outputs for anode pins and 6 outputs for cathode pin. The anode output pins are U2, U4, V4 and W4. The cathode output pins are W7, W6, U8, V8, U5, V5 and U7.

Methodology

The hierarchal design is used to create the experiment setup. Initially, seven segment decoder is designed for one bit according to table 1 in the appendix. Afterwards, 4-digit seven segment display is designed. The counter is made to generate 10.5 ms refresh period, so the selective inputs of a 4-to-1 multiplexer are selected by the result of the counter. The 4-to-1 multiplexer is created for choosing which display lights up. The seven segment decoder is used as a low level module to representing the values in seven segment display.

Finally, the top level 4-bit counter is designed. First of all, the 4-digit seven segment display is added as a low module inside the module. The are two counters are designed inside the module. All counters are positive edge triggered, and have reset input. When the reset is 1, the counters are assigned to 0. The clock of the counters are 100 MHz, which is built-in frequency in BASYS 3. One of the counters is used to generate 1 second enable the clock. Another counter is used to counting in hexadecimal base. This counter works when the one second is passed before the last value.

Results

The RTL schematic of 4-bit counter and seven segment display is same as expected. The clock pin of all counters are connected to clock_100 pin, which is 100 MHz. There is a decoder and multiplexer in the 4-digit Seven Segment display.

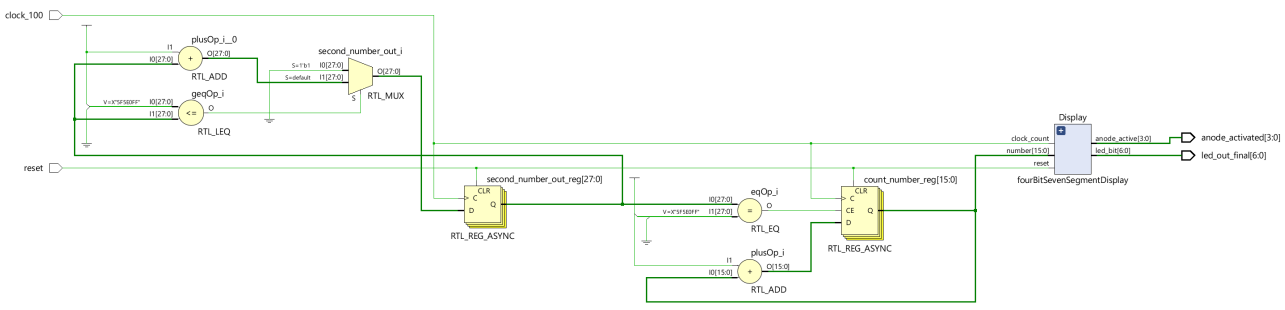

Figure 1 RTL Design of Lab Module

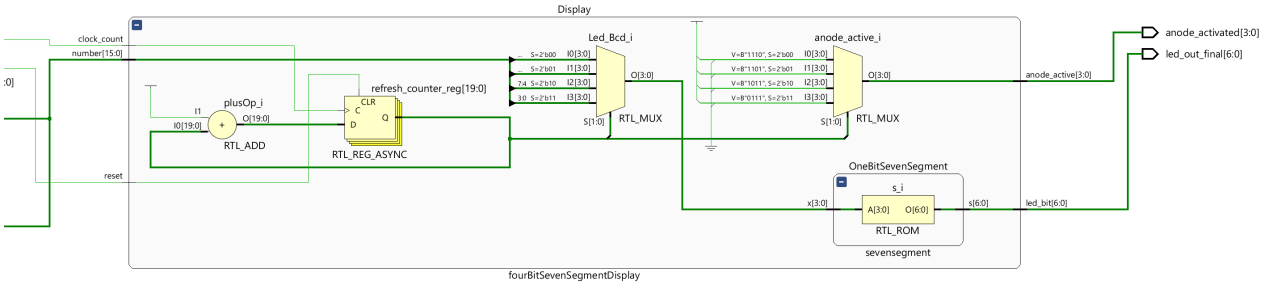

Figure 2 RTL Design of 4-digit Seven Segment Display

The module's test bench results give the expected result. The figure 3 is one of the example timing diagrams in the simulation. For instance, Anode active pin at "0111" condition changes into 1001111 to 0010010 between 3ms to 4 ms.

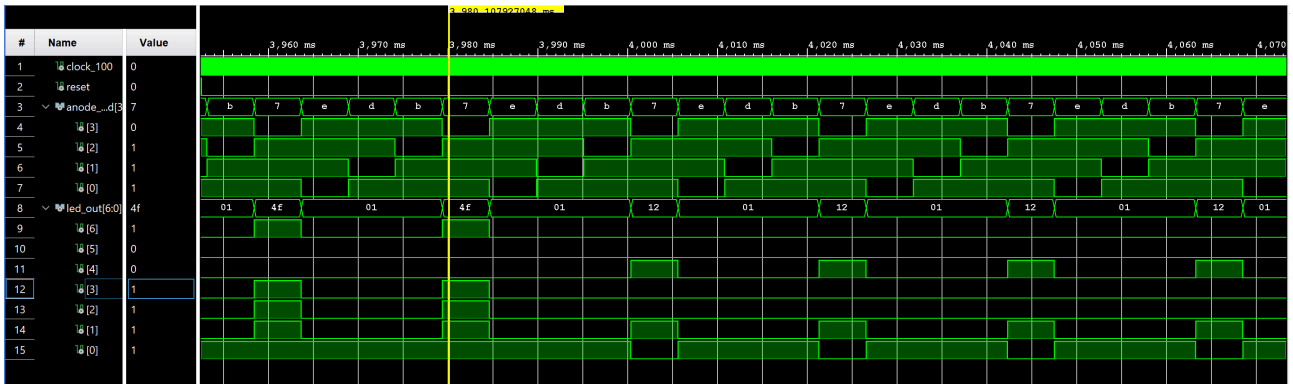

Figure 3 Test bench Graph between 3,950 ms to 4,070 ms

The application of the lab module on BASYS 3 is the same as expected. For instance, there is only one zero is displayed in the 4-digit seven segment display, when the reset button is pulled. Also, the one of the example state is presented in figure 4 in the appendix. There are also example photos of different states for a experiment setup BASYS 3 implementation.

Conclusion

In this lab, 4-bit counter will designed to show the hexadecimal entries in the 4-digit seven segment display of BASYS 3. The clock frequency is 100 MHz since it is internal frequency of BASYS 3. It will extended to 450 MHz. It is possible to create slower clock signal from 100 MHz.

However, any arbitrary frequency lower than that of the internal clock can not be created. It should be 2 division of internal value. For instance, you can create 50 MHz from 100Hz via using clock divider. You create counter its counts 2 bit number. When result reaches at 10, the counter resets to 00.

Appendix

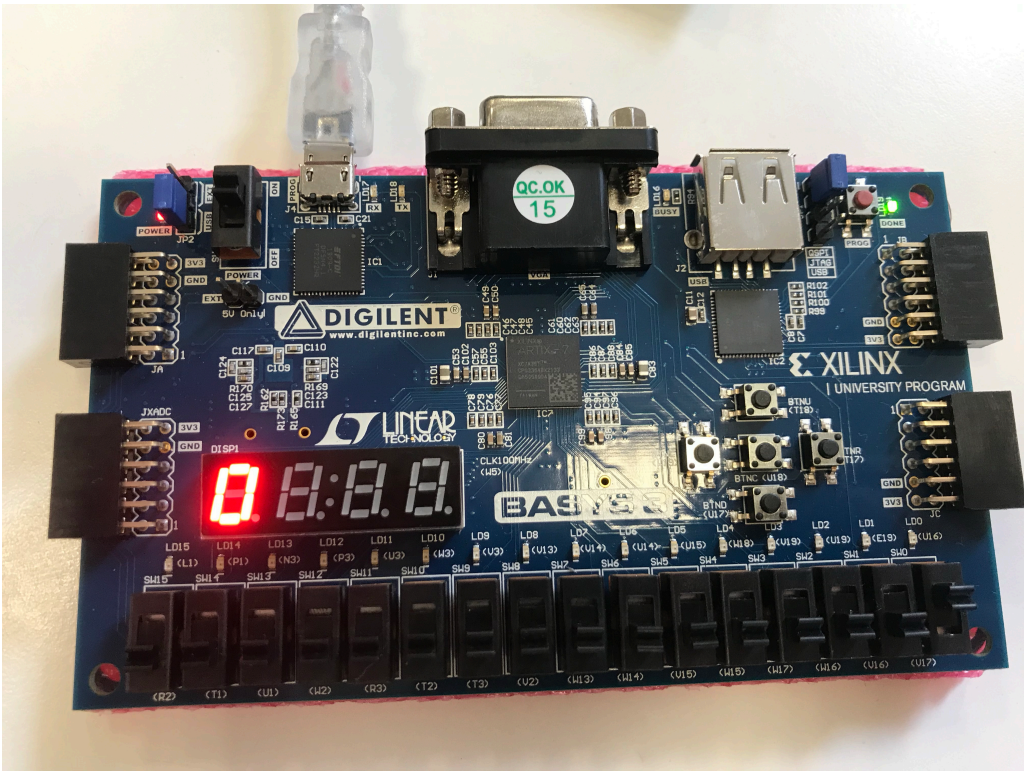

Figure 4 V17 pulled up, the result is 0

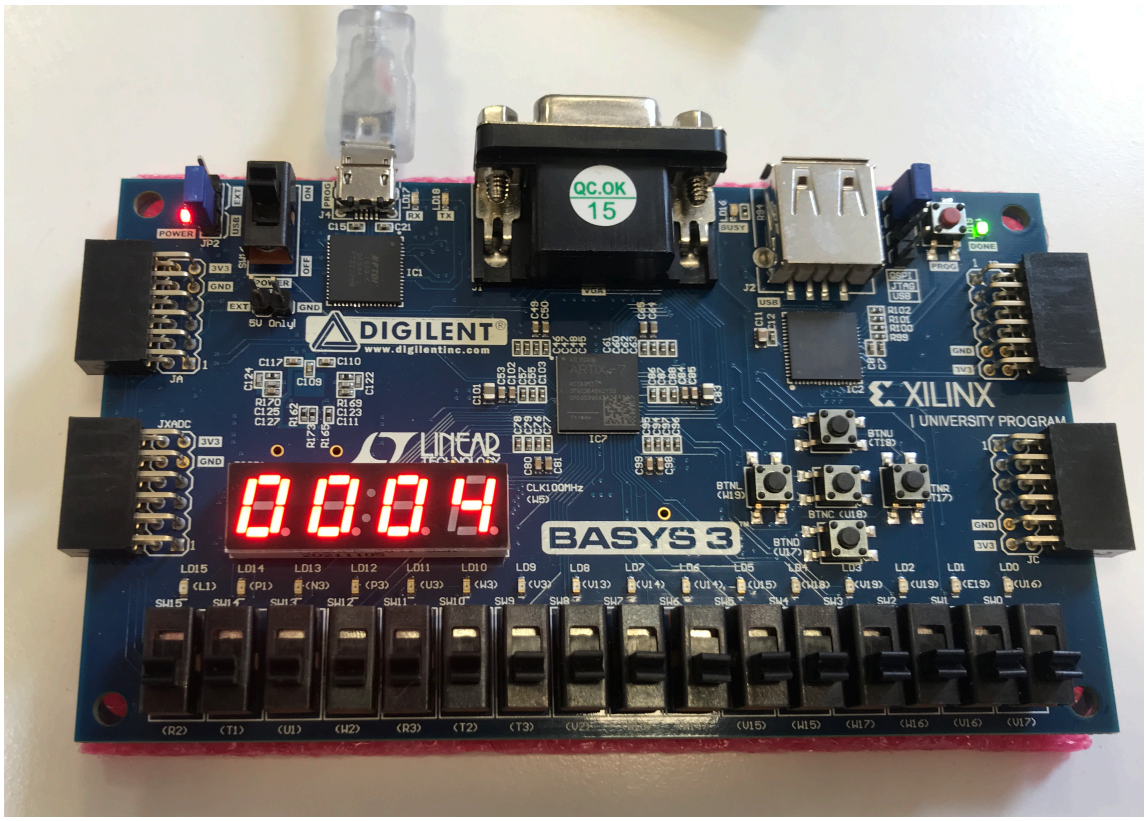

Figure 5 At 4 second the result is 4

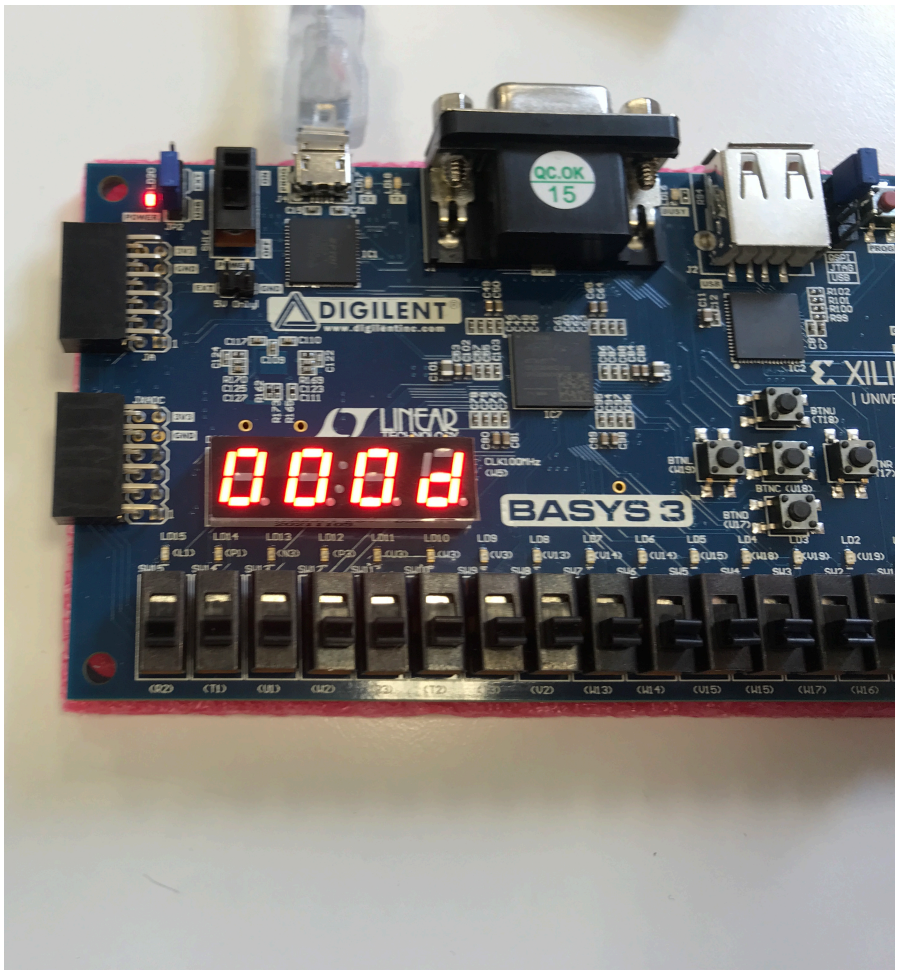

Figure 6 At 1101 state in 13 second the result is d

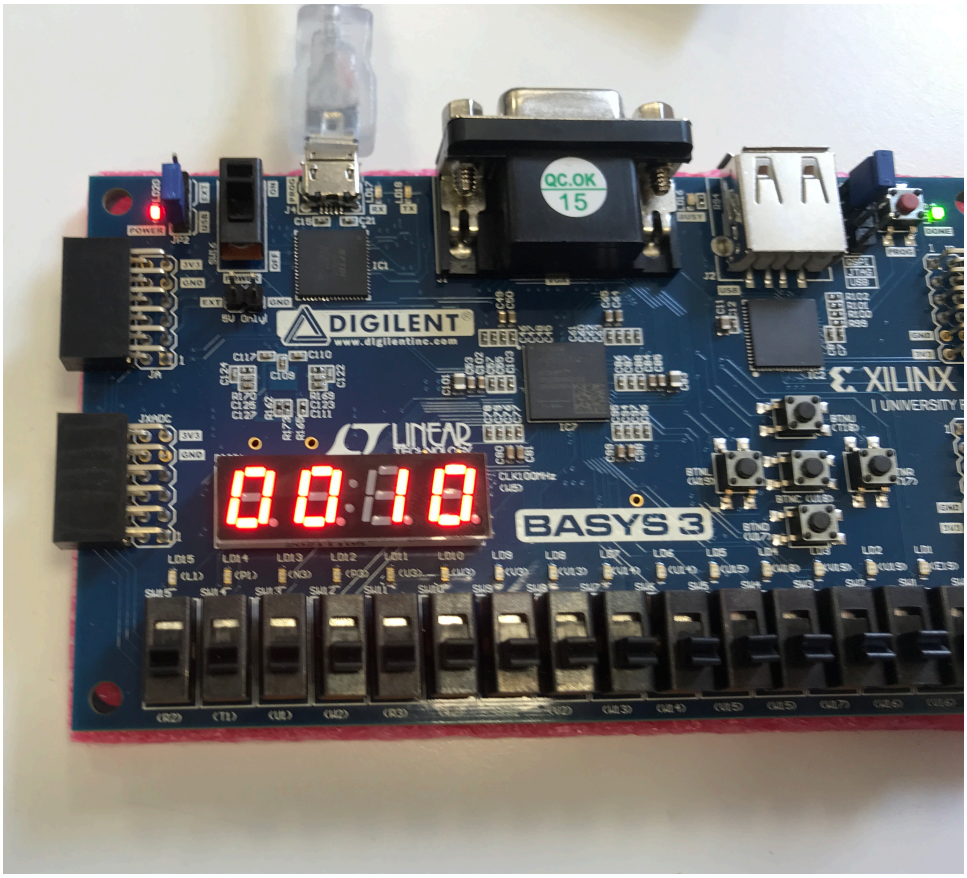

Figure 7 In 17 second the result is 10 in hexadecimal base

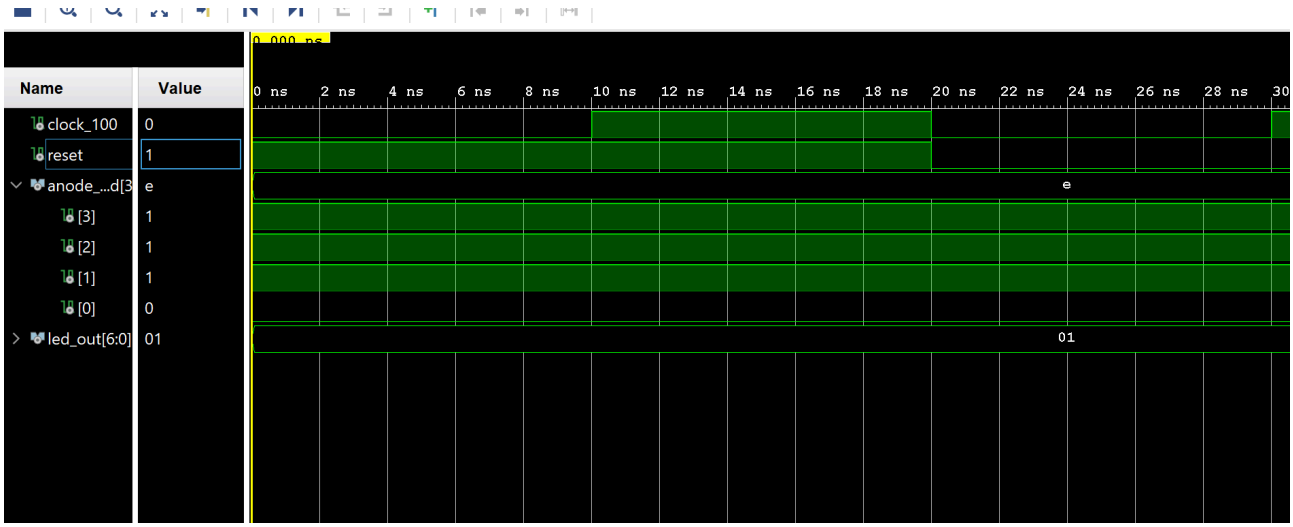

Figure 8 Test bench Graph when reset is 1

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity sevensegment is
    Port ( x : in STD_LOGIC_VECTOR (3 downto 0);
           s : out STD_LOGIC_VECTOR (6 downto 0));
end sevensegment;
architecture Behavioral of sevensegment is
begin
process(x)
begin
    case x is
        when "0000" => s <= "0000001";
        when "0001" => s <= "1001111";
        when "0010" => s <= "0010010";
        when "0011" => s <= "0000110";
        when "0100" => s <= "1001100";
        when "0101" => s <= "0100100";
        when "0110" => s <= "0100000";
        when "0111" => s <= "0001111";
        when "1000" => s <= "0000000";
        when "1001" => s <= "0000100";
        when "1010" => s <= "0000010";
        when "1011" => s <= "1100000";
        when "1100" => s <= "0110001";
        when "1101" => s <= "1000010";
        when "1110" => s <= "0110000";
```

```
        when others => s <= "0111000";
    end case;
 end process;
end Behavioral;
```

Code 1: Design Code of 1-Digit 7 Segment Display

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

entity fourBitSevenSegmentDisplay is
    Port (
          number : in std_logic_vector(15 downto 0);
          reset: in std_logic;
          anode_active : out STD_LOGIC_VECTOR (3 downto 0);
          clock_count: in std_logic;
          led_bit : out STD_LOGIC_VECTOR (6 downto 0)
          );
end fourBitSevenSegmentDisplay;

architecture Behavioral of fourBitSevenSegmentDisplay is

signal Led_Bcd : STD_LOGIC_VECTOR(3 downto 0);

Component sevensegment is
    Port(
    x : in STD_LOGIC_VECTOR (3 downto 0);
    s : out STD_LOGIC_VECTOR (6 downto 0)
    );
```

```
    end Component;



signal refresh_counter: STD_LOGIC_VECTOR (19 downto 0);
Signal sel : STD_LOGIC_VECTOR (1 downto 0);
begin
process(clock_count, reset)
begin
    if(reset = '1') Then
        refresh_counter <= (others => '0' );
     elsif(rising_edge(clock_count)) then
        refresh_counter <= refresh_counter + 1;
      end if;
end process;
sel <= refresh_counter(19 downto 18);

process(sel)
begin
    case sel is
        when "00" => anode_active <= "1110";
        -- activate first bit
        Led_Bcd <= number(15 downto 12);
        when "01" => anode_active <= "1101";
        -- activate seconde bit
        Led_Bcd <= number(11 downto 8);
        when "10" => anode_active <= "1011";
        -- activate first bit
        Led_Bcd <= number(7 downto 4);
```

```
        when "11" => anode_active <= "0111";
        -- activate first bit
        Led_Bcd <= number(3 downto 0);
        when others => anode_active <= "1111";
    end case;
 end process;
OneBitSevenSegment: sevensegment Port Map(Led_Bcd,led_bit);
end Behavioral;
```

Code 2: Design Code of 4-Digit 7 Segment Display

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


entity fourbitcounter is
    Port ( clock_100 : in STD_LOGIC;
           reset : in STD_LOGIC;
           anode_activated : out STD_LOGIC_VECTOR (3 downto 0);
           led_out_final : out STD_LOGIC_VECTOR (6 downto 0));
end fourbitcounter;
architecture Behavioral of fourbitcounter is
signal second_number_out : std_logic_vector(27 downto 0);
signal s_enable : std_logic;
signal count_number: std_logic_vector(15 downto 0);
signal Led_activating_counter: std_logic_vector(1 downto 0);
Component fourBitSevenSegmentDisplay is
    Port (
           number : in std_logic_vector(15 downto 0);
```

```
        reset: in std_logic;
        anode_active : out STD_LOGIC_VECTOR (3 downto 0);
        clock_count: in std_logic;
        led_bit : out STD_LOGIC_VECTOR (6 downto 0)
        );
 end Component;
begin
Display: fourBitSevenSegmentDisplay
    Port Map(
        number => count_number,
        reset => reset,
        anode_active => anode_activated,
        clock_count => clock_100,
        led_bit => led_out_final);
--Counting Process
process(clock_100, reset)
begin
    if(reset='1') Then
        second_number_out <= (others => '0');
    elsif(rising_edge(clock_100)) Then
        if(second_number_out>=x"5F5E0FF") then
            second_number_out <= (others => '0');
         else
            second_number_out<= second_number_out + "0000001";
         end if;
        end if;
```

```
end Process;
s_enable <= '1' when second_number_out=x"5F5E0FF" else '0';


process(clock_100, reset)
begin
    if(reset='1') Then
        count_number <= (others => '0');
    elsif(rising_edge(clock_100)) Then
        if(s_enable = '1') then
                count_number <= count_number + x"0001";
         end if;
    end if;
end process;
end Behavioral;
```

Code 3: Design Code of Lab Module

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity counter_testbench is
end counter_testbench;
architecture Behavioral of counter_testbench is
Component fourbitcounter is
    Port ( clock_100 : in STD_LOGIC;
           reset : in STD_LOGIC;
           anode_activated : out STD_LOGIC_VECTOR (3 downto 0);
           led_out_final : out STD_LOGIC_VECTOR (6 downto 0));
```

```
end Component;
-- Inputs
Signal clock_100: std_logic := '0' ;
Signal reset: std_logic := '0';
-- Outputs
Signal anode_activated: std_logic_vector(3 downto 0);
Signal led_out: std_logic_vector(6 downto 0);
begin
uut: fourbitcounter Port Map(
    clock_100 => clock_100,
    reset => reset,
    anode_activated => anode_activated,
    led_out_final => led_out
);
clk_process: process
begin
    clock_100 <= '0';
        wait for 10ns;
    clock_100 <= '1';
        wait for 10ns;
end Process;
stim_proc: process
begin
    reset <= '1';
        wait for 20ns;
    reset <= '0';
```

```
        wait;
end Process;
end Behavioral;
```

Code 4: Test bench Code of Experiment Setup