COL215 HW Assignment 2 - 4-Digit 7-Segment Display

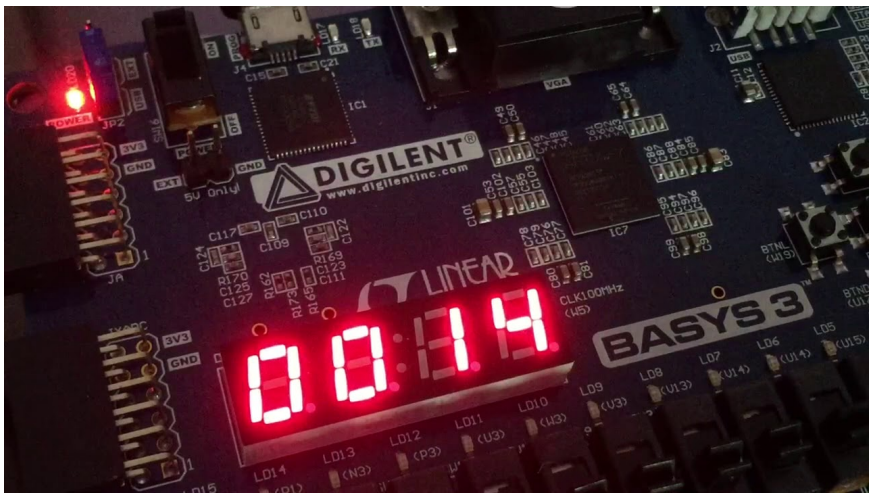
Submission By :

Yash Rawat **2023CS50334**

Priyanshi Gupta **2023CS10106**

Department of Computer Science and Engineering
Indian Institute of Technology, Delhi

September 01, 2024



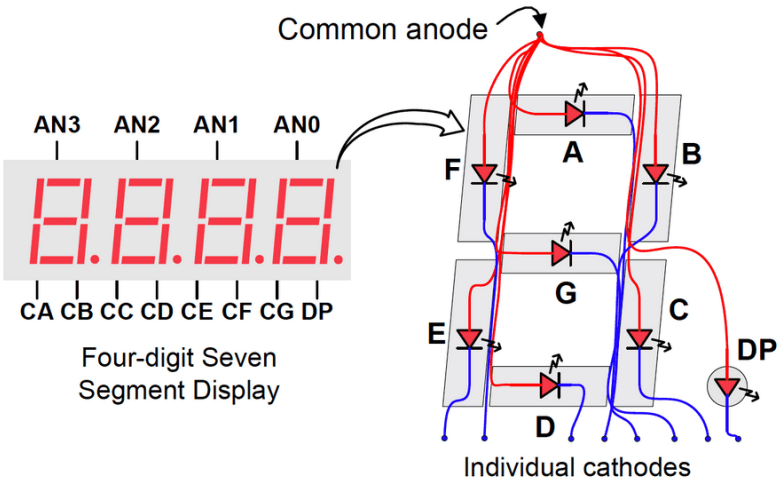


Figure 18. Common anode circuit node.

1 Assignment Problem Statement

Design a combinational circuit that takes a single 4-bit hexadecimal or decimal digit input from the switches and produces a 7-bit output for the seven-segment display of Basys 3 FPGA board. Extend the design to create a circuit that drives all 4 displays for displaying 4 digits together.

1.1 Overview of Assignment :

The assignment requires us to design three modules with the following inheritance logic :

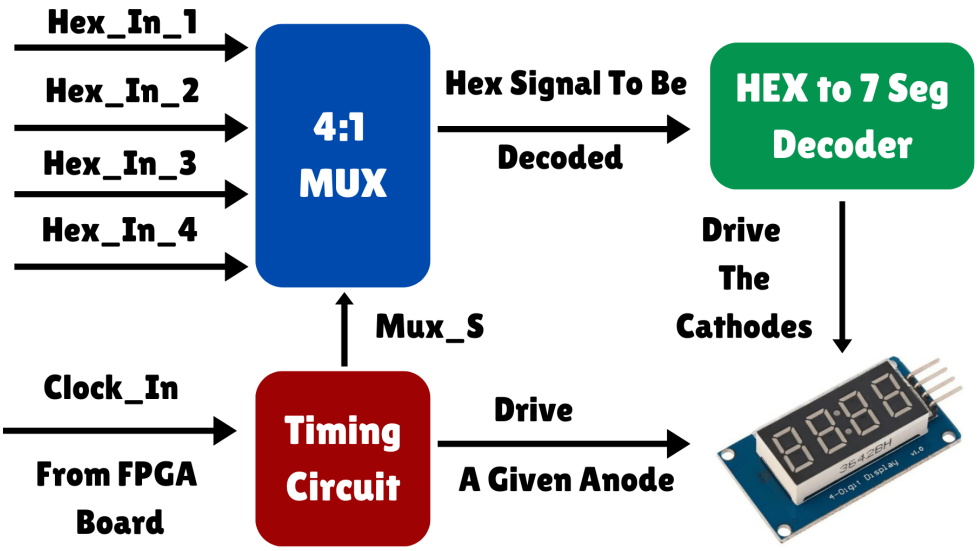


Figure 1: Overview of Design of Assignment

We will also define a wrapper VHD File - 7Seg_Display to implement all of the logic and take respective inputs for all the components.The 4:1 MUX accepts the 4 Hexadecimal input signals to be displayed on 7 Segment Display. The timing circuit governs the anode to be displayed as well as provides the corresponding selection bits to MUX. Finally the decoder module decodes the 7-bits using minimised combinational logic to drive the cathodes of display. The in-depth explanation of all the three components and their inheritance is explained later.

2 Design Decisions of Overall Project

2.1 Hexadecimal Digits

Since we are implementing display of 4 hexadecimal digits, we require 16 input bits from the board (accepted from the 16 switches available on board) to represent the input's to . We use the following symbols for representation on 7 Segment display :

Figure 2: Caption

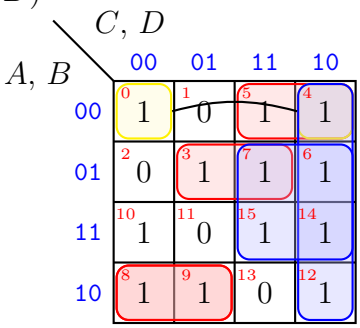
2.2 Truth Table - For BCD to 7 Segment Display

Digits & Input Bits					7 Segment Display							Min-Terms: $f(A, B, C, D)$
Digit	A	B	C	D	a	b	c	d	e	f	g	Min-Term Index
0	0	0	0	0	1	1	1	1	1	1	0	m_0
1	0	0	0	1	0	1	1	0	0	0	0	m_1
2	0	0	1	0	1	1	0	1	1	0	1	m_2
3	0	0	1	1	1	1	1	1	0	0	1	m_3
4	0	1	0	0	0	1	1	0	0	1	1	m_4
5	0	1	0	1	1	0	1	1	0	1	1	m_5
6	0	1	1	0	1	0	1	1	1	1	1	m_6
7	0	1	1	1	1	1	1	0	0	0	0	m_7
8	1	0	0	0	1	1	1	1	1	1	1	m_8
9	1	0	0	1	1	1	1	0	0	1	1	m_9
A	1	0	1	0	1	1	1	0	1	1	1	m_{10}
b	1	0	1	1	0	0	1	1	1	1	1	m_{11}
C	1	1	0	0	1	0	0	1	1	1	0	m_{12}
d	1	1	0	1	0	1	1	1	1	0	1	m_{13}
E	1	1	1	0	1	0	0	1	1	1	1	m_{14}
F	1	1	1	1	1	0	0	0	1	1	1	m_{15}

2.3 Minimizing the Combinational Logic using K-Maps

2.3.1 Segment a

$f(A, B, C, D)$

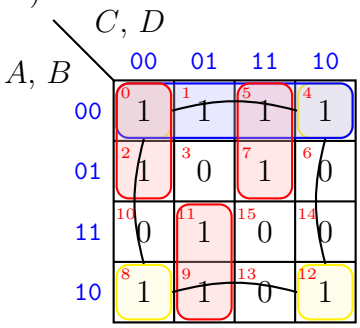


The reduced formula for segment a using the mentioned K-Map reduction is as following :

$$f(A, B, C, D) = A^{'}B^{'}D^{'} + A^{'}BD + AC^{'}D^{'}$$
$$+AB^{'}C^{'} + A^{'}C + BC + CD^{'}$$

2.3.2 Segment b

$f(A, B, C, D)$

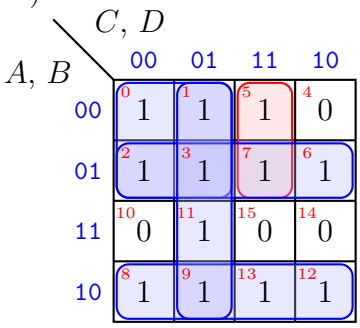


The reduced formula for segment b using the mentioned K-Map reduction is as following :

$$f(A, B, C, D) = A^{'}C^{'}D^{'} + A^{'}CD +$$
$$AC^{'}D + B^{'}D^{'} + A^{'}B^{'}$$

2.3.3 Segment c

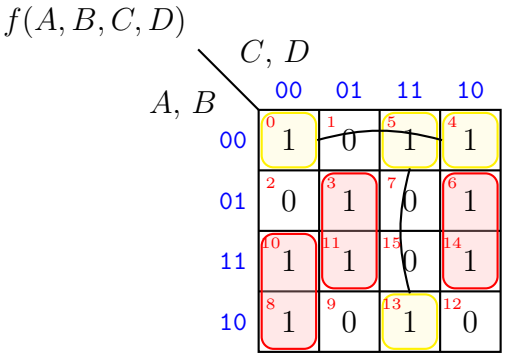
$f(A, B, C, D)$



The reduced formula for segment c using the mentioned K-Map reduction is as following :

$$f(A, B, C, D) = A^{'}CD + A^{'}C' +$$
$$A^{'}B + C^{'}D + AB^{'}$$

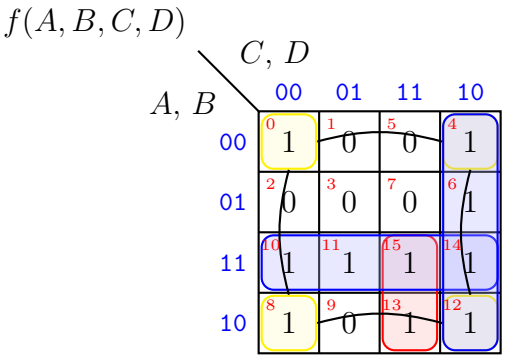
2.3.4 Segment d



The reduced formula for segment d using the mentioned K-Map reduction is as following :

$$f(A,B,C,D) = A'B'D' + B'CD+$$
$$BC'D + BCD' + A'C'D'$$

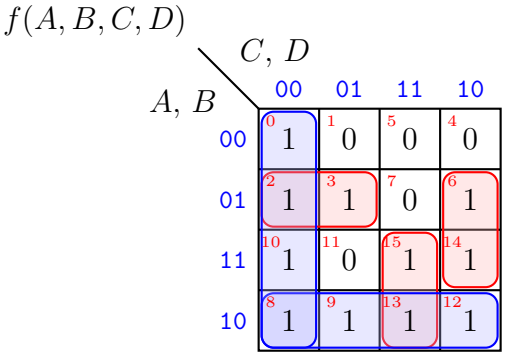
2.3.5 Segment e



The reduced formula for segment e using the mentioned K-Map reduction is as following :

$$f(A,B,C,D) = B'C'D' + AB+$$
$$CD' + AC$$

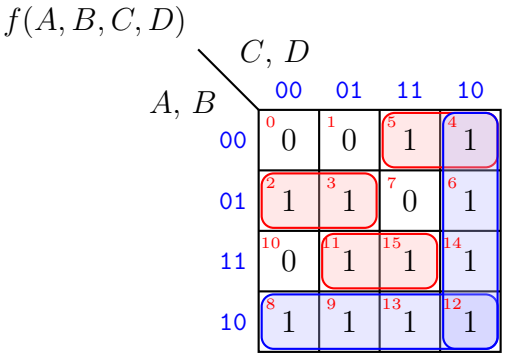
2.3.6 Segment f



The reduced formula for segment f using the mentioned K-Map reduction is as following :

$$f(A,B,C,D) = A'BC' + BCD'+$$
$$AB' + AC + C'D'$$

2.3.7 Segment g



The reduced formula for segment g using the mentioned K-Map reduction is as following :

$$f(A,B,C,D) = A'B'C + A'BC'$$

$$CD' + AB' + AD$$

3 Design Decisions for Modules

3.1 Hexadecimal-To-7-Segment-Display

Decoder is a combinational module which accepts the 4 input bits from the **MUX_4BIT** to be displayed on a seven segment display. Then the decoder implements the reduced combinational logic and sets the respective states of cathodes to drive the output on the seven segment display.

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

entity seven_seg_decoder_hex is
    port (
        dec_in : in std_logic_vector(3 downto 0);
        dec_out : out std_logic_vector(6 downto 0)
    );
end seven_seg_decoder_hex;

architecture Behavioral of seven_seg_decoder_hex is
    signal A, B, C, D : std_logic;

-- Basys 3 board uses Active Low Pins hence the values are inverted
-- Logic from actual reduced expression using K-map
-- (Sum of Min terms method)
begin
    Seg_Decoder_HEX : process(dec_in)
    begin
        D <= dec_in(0);
        C <= dec_in(1);
        B <= dec_in(2);
        A <= dec_in(3);
        dec_out(0) <= -- Logic of a
        dec_out(1) <= -- Logic of b
        dec_out(2) <= -- Logic of c
        dec_out(3) <= -- Logic of d
```

```
        dec_out(4) <= -- Logic of e
        dec_out(5) <= -- Logic of f
        dec_out(6) <= -- Logic of g

    end process;
end Behavioral;
```

Listing 1: Decoder Code

3.2 4:1 MUX

The board is used to generate 4 hexadecimal inputs, one for each seven segment display. Since the cathodes of all the 4 displays are common, the timer circuit provides select bits : *dec_in* which also corresponds to the anode, i.e. one of the four 7 Segment displays which are being driven. The implemented MUX_Process chooses the hexadecimal input to be displayed using case statement, which is passed onto the decoder module to be decoded into 7 segment bits.

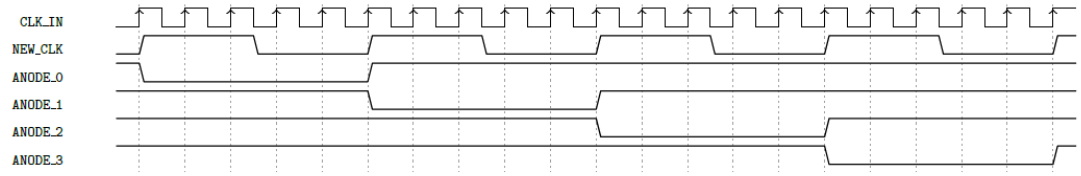
```
library ieee;
use ieee.std_logic_1164.all;

entity MUX_4BIT is
    port(
        mux_s : in std_logic_vector(1 downto 0);
        mux_d0 : in std_logic_vector(3 downto 0);
        mux_d1 : in std_logic_vector(3 downto 0);
        mux_d2 : in std_logic_vector(3 downto 0);
        mux_d3 : in std_logic_vector(3 downto 0);
        mux_out_to : out std_logic_vector(3 downto 0)
    );
end MUX_4BIT;
architecture Behavioral of MUX_4BIT is
begin
    MUX_Process: process(mux_s, mux_d0, mux_d1, mux_d2, mux_d3)
    begin
        case mux_s is
            when "00" =>
                mux_out_to <= mux_d0;
            when "01" =>
                mux_out_to <= mux_d1;
            when "10" =>
                mux_out_to <= mux_d2;
            when "11" =>
                mux_out_to <= mux_d3;
            when others =>
                mux_out_to <= "0000";
        end case;
    end process;
end Behavioral;
```

Listing 2: 4:1 MUX

3.3 Timing Circuit

Timing circuit is the Crux of this project. Since the cathodes of all the four 7 Segment displays are common, we need a way to cycle through the different anodes and a way to keep track which hexadecimal input is to displayed on the given anode (which is what the 4:1 MUX module implements).

This module also accepts **clk_in** from the Basys3 board, an inbuilt 100 MHz clock. For our practical purposes using such a high frequency clock is not viable since it may cause flickering issues, care is taken to implement a **new_clk** which uses a **counter** and **N** to increase the time period from 10 ns to 10.24 ms, a frequency 97.65625 Hz as given below.



Since human eye' vision is persistent at these refresh rates, the **MUX_PROC** process changes the state of **mux_select_counter** cyclically according to the **rising_edge** of **new_clk** and assigns it to **mux_select** which goes to MUX module. This part handles the cyclical change of anodes which allow us to display all 4 digits perceived at once.

Finally the **ANODE_select** process chooses the anode states for choosing which anode the digit is to be displayed upon.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_ARITH.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  entity Timing_block is
7      Port (
8          clk_in : in STD_LOGIC; -- 100 MHz input clock
9          reset : in STD_LOGIC; -- Reset signal (Resetting the
       internal signals to known states)
10         mux_select : out STD_LOGIC_VECTOR (1 downto 0); -- Signal
       for the mux
11         anodes_tout : out STD_LOGIC_VECTOR (3 downto 0) -- Anodes
       signal for display
12     );
13 end Timing_block;
14
15 architecture Behavioral of Timing_block is
```

```
    constant N : integer := 511;-- <need to select correct value>
    signal counter: integer := 0;
    signal mux_select_counter : STD_LOGIC_VECTOR (1 downto 0) := "00
   ";
    signal new_clk : STD_LOGIC := '0';

begin
--Process 1 for dividing the clock from 100 Mhz to 1Khz - 60hz
-- Gives rise to a clock with t = 10.24 ms or f = 97.65625 Hz
    CLK_PROC: process(clk_in, reset)
    begin
        if(reset = '1') then
            counter <= 0;
            new_clk <= '0';
        elsif rising_edge(clk_in) then
            if counter = N then
                counter <= 0;
                new_clk <= not new_clk;
            else
                counter <= counter + 1;
            end if;
        end if;

    end process;
    --Process 2 for mux select signal
    MUX_PROC: process(new_clk, reset)
    begin
        if(reset = '1') then
            mux_select_counter <= "00";
        elsif(rising_edge(new_clk)) then
            mux_select_counter <= mux_select_counter + 1;
        end if;
        mux_select <= mux_select_counter;
    end process;
    --Process 3 for anode signal
    ANODE_select: process(mux_select_counter,reset)
    begin
        if(reset = '1') then
            anodes_tout <= "1111";
        elsif(mux_select_counter  = "00") then
            anodes_tout <= "1110";
        elsif (mux_select_counter  = "01") then
            anodes_tout <= "1101";
        elsif (mux_select_counter  = "10") then
            anodes_tout <= "1011";
        elsif (mux_select_counter  = "11") then
            anodes_tout <= "0111";
        end if;
    end process;

end Behavioral;
```

Listing 3: Timing Circuit

Note that whenever the board detects reset, the **counter** and the **new_clk** is restored to a known default. Also all the displays/anodes are switched off to show the circuit is in a reset state.

3.4 7Seg_Display - The Final Wrapper

This instantiates all the other modules as entities and implements the necessary signals for input form the board to the modules and the output back to board.

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

entity display_seven_seg is
    Port (
        clock_in : in STD_LOGIC; -- For Timing_Block
        reset_timer : in STD_LOGIC; -- For Timing_Block
        d0: in std_logic_vector(3 downto 0); -- For MUX_Block
        d1: in std_logic_vector(3 downto 0); -- For MUX_Block
        d2: in std_logic_vector(3 downto 0); -- For MUX_Block
        d3: in std_logic_vector(3 downto 0); -- For MUX_Block
        an : out STD_LOGIC_VECTOR (1 downto 0); -- For Timing_Block
        seg : out STD_LOGIC_VECTOR (3 downto 0) -- For Timing_Block
    );
end display_seven_seg;

architecture Behavioral of display_seven_seg is

    component Timing_block is
        Port (
            clk_in : in STD_LOGIC; -- 100 MHz input clock
            reset : in STD_LOGIC; -- Reset Signal
            mux_select : out STD_LOGIC_VECTOR (1 downto 0); -- IN to
    MUX_Block
            anodes_tout : out STD_LOGIC_VECTOR (3 downto 0) --
   Anodes signal for display
        );
    end component;

    component seven_seg_decoder_hex is
        port (
            dec_in : in std_logic_vector(3 downto 0);
            dec_out : out std_logic_vector(6 downto 0)
        );
    end component;

    component MUX_4BIT is
        port(
            mux_s : in std_logic_vector(1 downto 0);
            mux_d0 : in std_logic_vector(3 downto 0);
            mux_d1 : in std_logic_vector(3 downto 0);
            mux_d2 : in std_logic_vector(3 downto 0);
            mux_d3 : in std_logic_vector(3 downto 0);
```

```
            mux_out_to : out std_logic_vector(3 downto 0)
        );
    end component;

    signal mux_sel: std_logic_vector(1 downto 0);
    signal mux_out_dec: std_logic_vector(3 downto 0);

begin
    Timer_Block : -- UUT and Port Mapping for Timing Circuit

    MUX_Block :-- UUT and Port Mapping for MUX

    Decoder_Block : --  UUT and Port Mapping for Decoder

end Behavioral;
```

4 Submission Details

4.1 Simulation Snapshots

4.1.1 Simulation of Hexadecimal Decoder

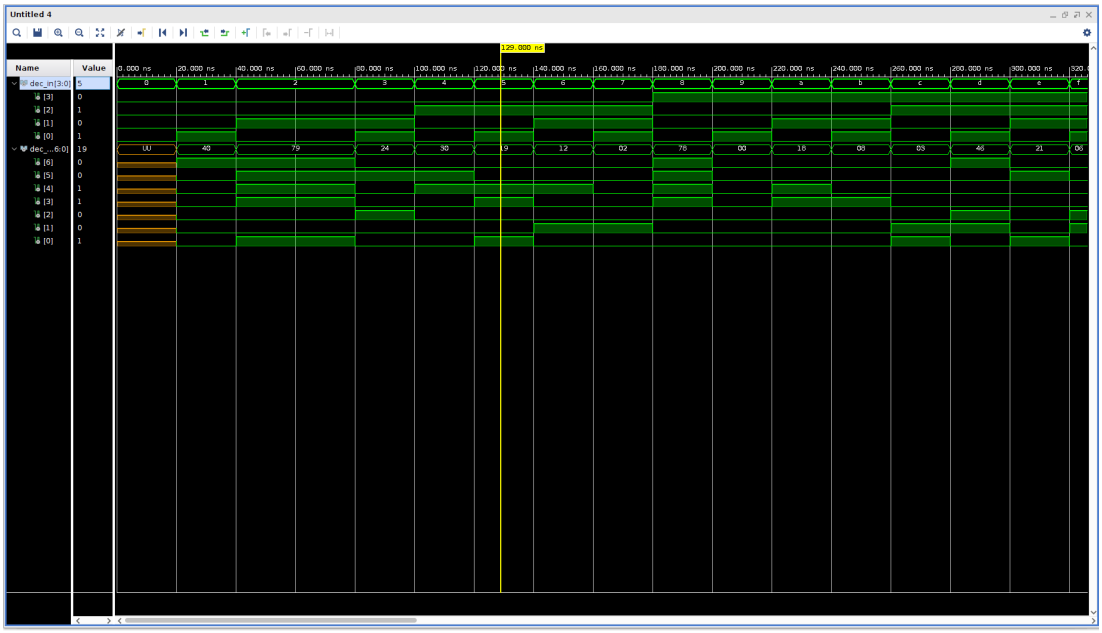


Figure 3: Simulation of Decoder

A point to note is that the output seems flipped in comparison to Truth Table. This is because the cathode pins on the Basys board work in low configuration. This holds everywhere in the simulation and the implementation.

4.1.2 Simulation of 4:1 MUX

4.1.3 Simulation of Timing Circuit

4.2 Schematic Snapshots

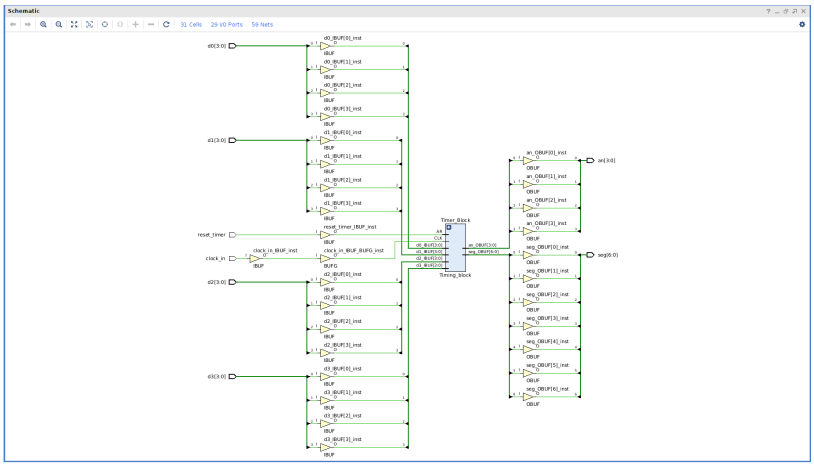


Figure 4: Schematic

4.3 Resource Utilization

1. Slice Logic

Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	52	0	0	20800	0.25
LUT as Logic	52	0	0	20800	0.25
LUT as Memory	0	0	0	9600	0.00
Slice Registers	35	0	0	41600	0.08
Register as Flip Flop	35	0	0	41600	0.08
Register as Latch	0	0	0	41600	0.00
F7 Muxes	0	0	0	16300	0.00
F8 Muxes	0	0	0	8150	0.00

* Warning! The Final LUT count, after physical optimizations and full implementation, is typically lower. Run opt_design after synthesis, if not already completed,

2. Memory

Site Type	Used	Fixed	Prohibited	Available	Util%
Block RAM Tile	0	0	0	50	0.00
RAMB36/FIFO*	0	0	0	50	0.00
RAMB18	0	0	0	100	0.00

* Note: Each Block RAM Tile only has one FIFO logic available and therefore can accommodate only one FIFO36E1 or one FIFO18E1. However, if a FIFO18E1 occupies a Bl

3. DSP

Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	90	0.00

Figure 5: Resource Utilization

4.4 Output on FPGA Board

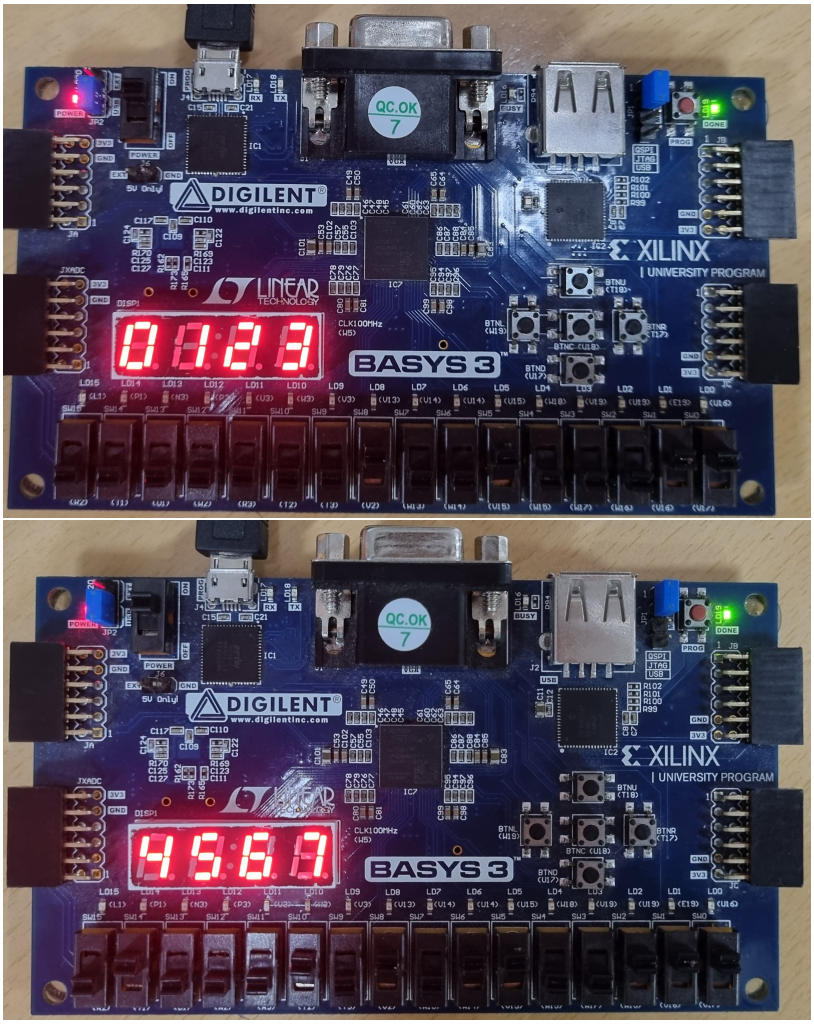


Figure 6: Basys Board output 1

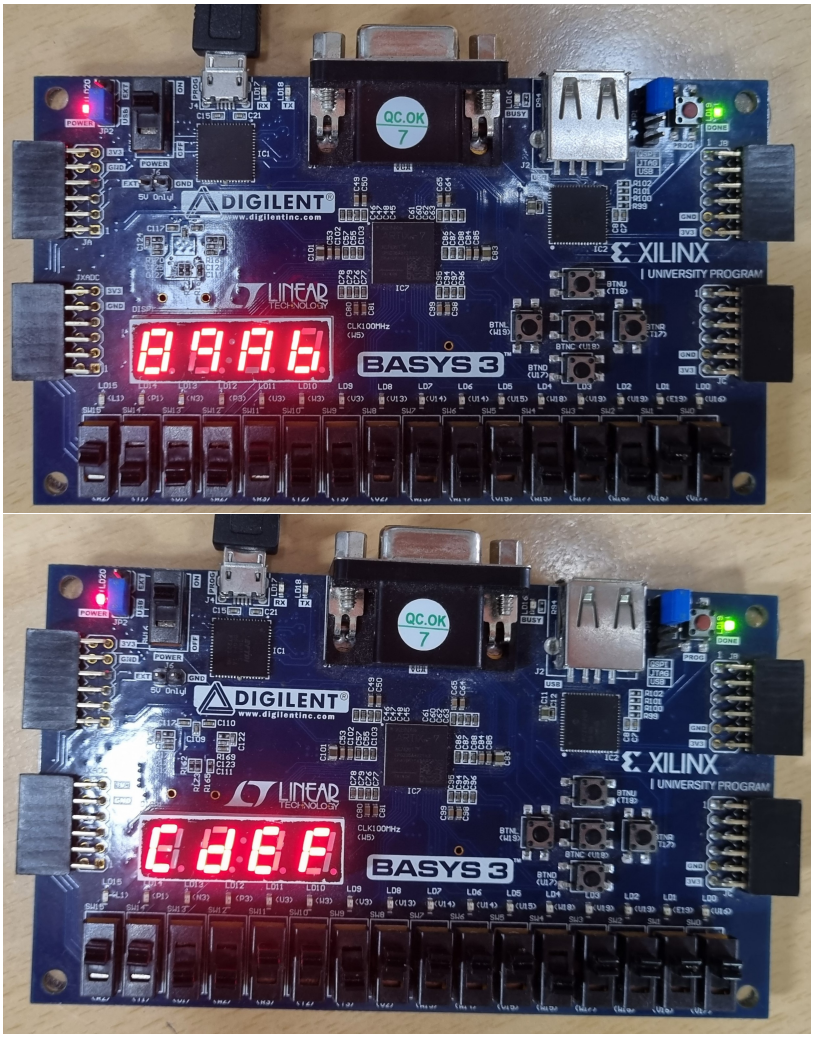


Figure 7: Basys Board output 2