

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

Submission By :

Yash Rawat      Priyanshi Gupta  
2023CS50334      2023CS10106

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

September 02, 2024

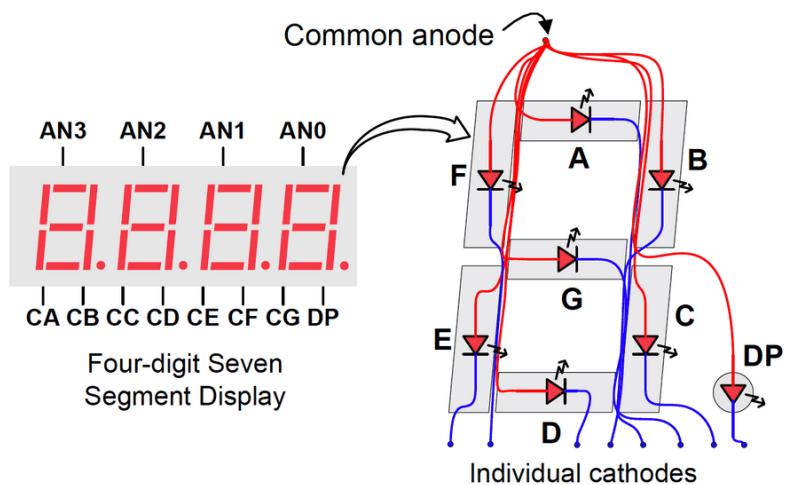
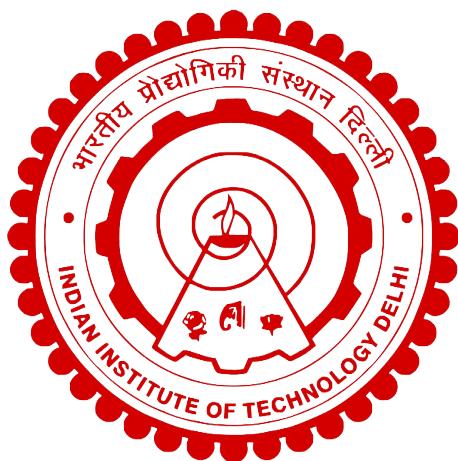


Figure 18. Common anode circuit node.

# 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 :

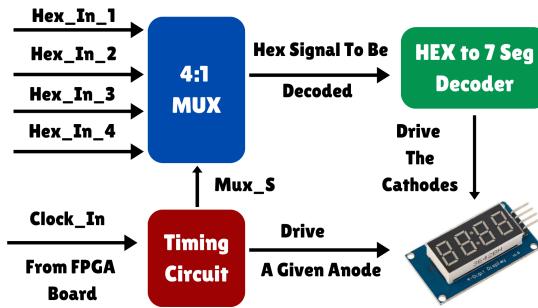


Figure 1: Overview of Design of Assignment

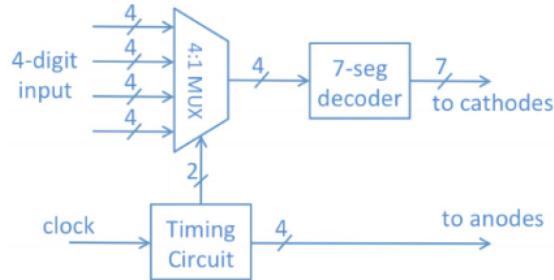


Figure 2: Circuit Diagram of Assignment

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 inputs to the MUX. We use the following symbols for representation on 7 Segment display :



Figure 3: Caption

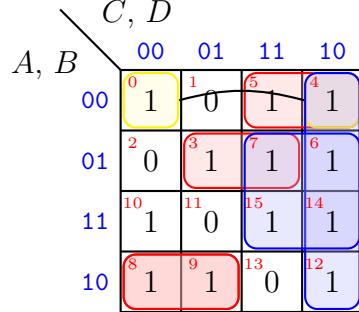
### 2.2 Truth Table - For BCD to 7 Segment Display

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		$m_9$
<i>A</i>	1	0	1	0	1	1	1	0	1	1	1		$m_{10}$
<i>b</i>	1	0	1	1	0	0	1	1	1	1	1		$m_{11}$
<i>C</i>	1	1	0	0	1	0	0	1	1	1	0		$m_{12}$
<i>d</i>	1	1	0	1	0	1	1	1	1	0	1		$m_{13}$
<i>E</i>	1	1	1	0	1	0	0	1	1	1	1		$m_{14}$
<i>F</i>	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

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

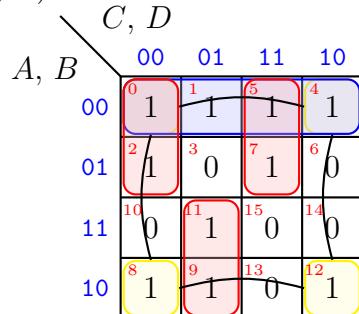


The reduced formula for segment a using the mentioned K-Map reduction is as following :

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

### 2.3.2 Segment b

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

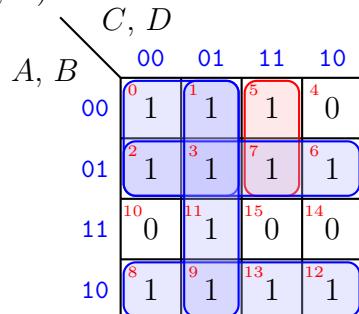


The reduced formula for segment b using the mentioned K-Map reduction is as following :

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

### 2.3.3 Segment c

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

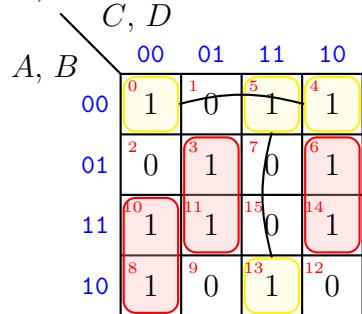


The reduced formula for segment c using the mentioned K-Map reduction is as following :

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

### 2.3.4 Segment d

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

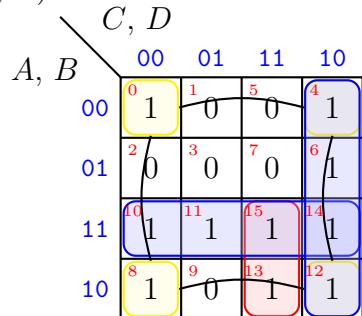


The reduced formula for segment d using the mentioned K-Map reduction is as following :

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

### 2.3.5 Segment e

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

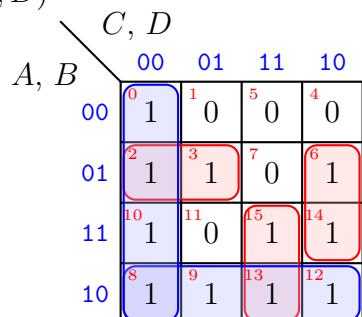


The reduced formula for segment e using the mentioned K-Map reduction is as following :

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

### 2.3.6 Segment f

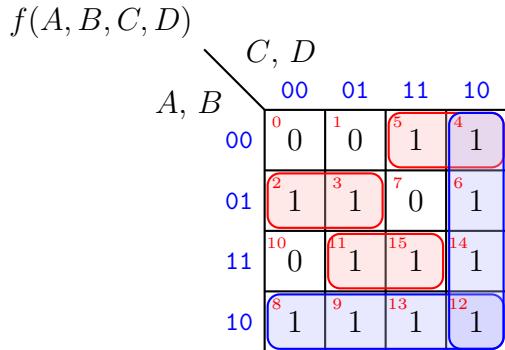
$$f(A, B, C, D)$$



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.

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity seven_seg_decoder_hex is
5     port (
6         dec_in : in std_logic_vector(3 downto 0);
7         dec_out : out std_logic_vector(6 downto 0)
8     );
9 end seven_seg_decoder_hex;
10
11 architecture Behavioral of seven_seg_decoder_hex is
12     signal A, B, C, D : std_logic;
13
14 -- Basys 3 board uses Active Low Pins hence the values are inverted
15 -- Logic from actual reduced expression using K-map
16 -- (Sum of Min terms method)
17 begin
18     Seg_Decoder_HEX : process(dec_in)
19     begin
20         D <= dec_in(0);
21         C <= dec_in(1);
22         B <= dec_in(2);
23         A <= dec_in(3);
24         dec_out(0) <= -- Logic of a
25         dec_out(1) <= -- Logic of b
26         dec_out(2) <= -- Logic of c
27         dec_out(3) <= -- Logic of d

```

```

28      dec_out(4) <= -- Logic of e
29      dec_out(5) <= -- Logic of f
30      dec_out(6) <= -- Logic of g
31
32  end process;
33 end Behavioral;
```

Listing 1: Hexadecimal to 7 Segment 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.

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity MUX_4BIT is
5   port(
6     mux_s : in std_logic_vector(1 downto 0);
7     mux_d0 : in std_logic_vector(3 downto 0);
8     mux_d1 : in std_logic_vector(3 downto 0);
9     mux_d2 : in std_logic_vector(3 downto 0);
10    mux_d3 : in std_logic_vector(3 downto 0);
11    mux_out_to : out std_logic_vector(3 downto 0)
12  );
13 end MUX_4BIT;
14 architecture Behavioral of MUX_4BIT is
15 begin
16   MUX_Process: process(mux_s, mux_d0, mux_d1, mux_d2, mux_d3)
17   begin
18     case mux_s is
19       when "00" =>
20         mux_out_to <= mux_d0;
21       when "01" =>
22         mux_out_to <= mux_d1;
23       when "10" =>
24         mux_out_to <= mux_d2;
25       when "11" =>
26         mux_out_to <= mux_d3;
27       when others =>
28         mux_out_to <= "0000";
29     end case;
30   end process;
31 end Behavioral;
```

Listing 2: 4:1 MUX Module

### 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  $10\text{ ns}$  to an appropriate value as shown in the figure below.

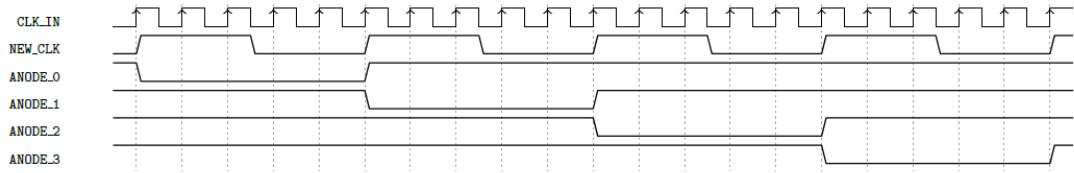


Figure 4: Clock and Anode Selection Logic to be implemented by Timing Block

Since human eye vision is persistent at implemented 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
16   constant N : integer := 2560000;-- <need to select correct value
17   >
18   signal counter: integer := 0;
19   signal mux_select_counter : STD_LOGIC_VECTOR (1 downto 0) := "00
";
20   signal new_clk : STD_LOGIC := '0';
21
22 begin
23   CLK_PROC: process(clk_in, reset)
24   begin
25     if(reset = '1') then
26       counter <= 0;
27       new_clk <= '0';
28     elsif rising_edge(clk_in) then
29       if counter = N then
30         counter <= 0;
31         new_clk <= not new_clk;
32       else
33         counter <= counter + 1;
34       end if;
35     end if;
36   end process;
37   --Process 2 for mux select signal
38   MUX_PROC: process(new_clk, reset)
39   begin
40     if(reset = '1') then
41       mux_select_counter <= "00";
42     elsif(rising_edge(new_clk)) then
43       mux_select_counter <= mux_select_counter + 1;
44     end if;
45     mux_select <= mux_select_counter;
46   end process;
47   --Process 3 for anode signal
48   ANODE_Select: process(mux_select_counter,reset)
49   begin
50     if(reset = '1') then
51       anodes_tout <= "1111";
52     elsif(mux_select_counter = "00") then
53       anodes_tout <= "1110";
54     elsif (mux_select_counter = "01") then
55       anodes_tout <= "1101";
56     elsif (mux_select_counter = "10") then
57       anodes_tout <= "1011";
58     elsif (mux_select_counter = "11") then
59       anodes_tout <= "0111";
60     end if;
61   end process;
62
63 end Behavioral;

```

Listing 3: Timing Circuit Module

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 from the board to the modules and the output back to board.

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity display_seven_seg is
5     Port (
6         clock_in : in STD_LOGIC; -- For Timing_Block
7         reset_timer : in STD_LOGIC; -- For Timing_Block
8         d0: in std_logic_vector(3 downto 0); -- For MUX_Block
9         d1: in std_logic_vector(3 downto 0); -- For MUX_Block
10        d2: in std_logic_vector(3 downto 0); -- For MUX_Block
11        d3: in std_logic_vector(3 downto 0); -- For MUX_Block
12        an : out STD_LOGIC_VECTOR (1 downto 0); -- For Timing_Block
13        seg : out STD_LOGIC_VECTOR (3 downto 0) -- For Timing_Block
14    );
15 end display_seven_seg;
16
17 architecture Behavioral of display_seven_seg is
18
19     component Timing_block is
20         Port (
21             clk_in : in STD_LOGIC; -- 100 MHz input clock
22             reset : in STD_LOGIC; -- Reset Signal
23             mux_select : out STD_LOGIC_VECTOR (1 downto 0); -- IN to
MUX_Block
24             anodes_tout : out STD_LOGIC_VECTOR (3 downto 0) ---
Anodes signal for display
25         );
26     end component;
27
28     component seven_seg_decoder_hex is
29         port (
30             dec_in : in std_logic_vector(3 downto 0);
31             dec_out : out std_logic_vector(6 downto 0)
32         );
33     end component;
34
35     component MUX_4BIT is
36         port(
37             mux_s : in std_logic_vector(1 downto 0);
38             mux_d0 : in std_logic_vector(3 downto 0);
39             mux_d1 : in std_logic_vector(3 downto 0);
40             mux_d2 : in std_logic_vector(3 downto 0);
41             mux_d3 : in std_logic_vector(3 downto 0);

```

```

42         mux_out_to : out std_logic_vector(3 downto 0)
43     );
44 end component;
45
46 signal mux_sel: std_logic_vector(1 downto 0);
47 signal mux_out_dec: std_logic_vector(3 downto 0);
48
49 begin
50     Timer_Block : -- UUT and Port Mapping for Timing Circuit
51     MUX_Block :-- UUT and Port Mapping for MUX
52     Decoder_Block : -- UUT and Port Mapping for Decoder
53 end Behavioral;

```

Listing 4: 7Seg-Display - The Wrapper Module

## 4 Submission Details

### 4.1 Simulation Snapshots

#### 4.1.1 Simulation of Hexadecimal Decoder

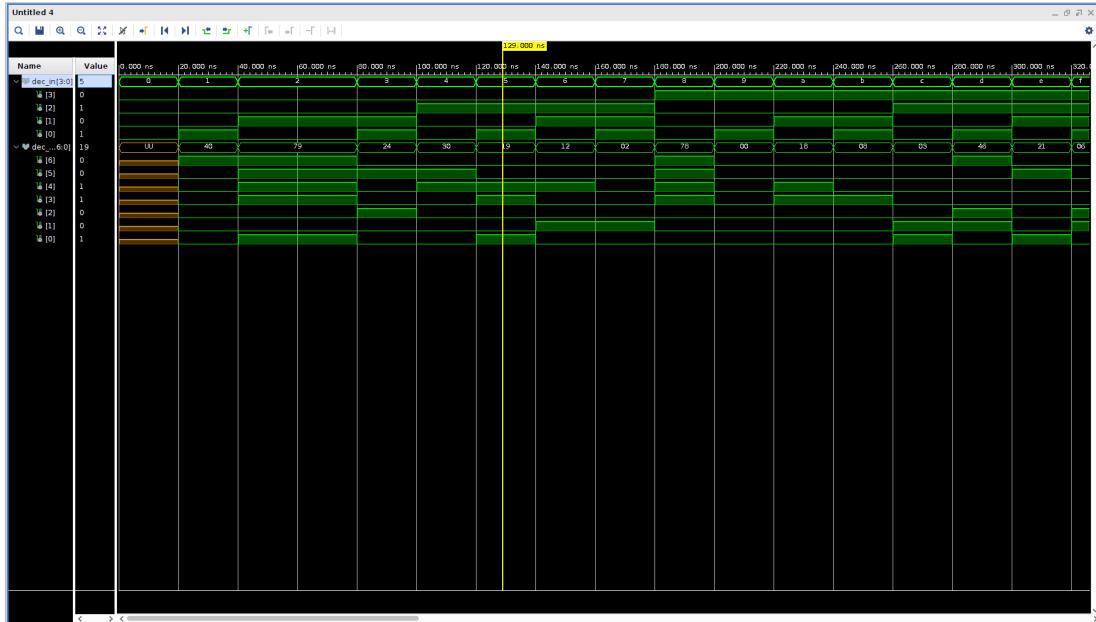


Figure 5: Simulation of Hexadecimal Decoder

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

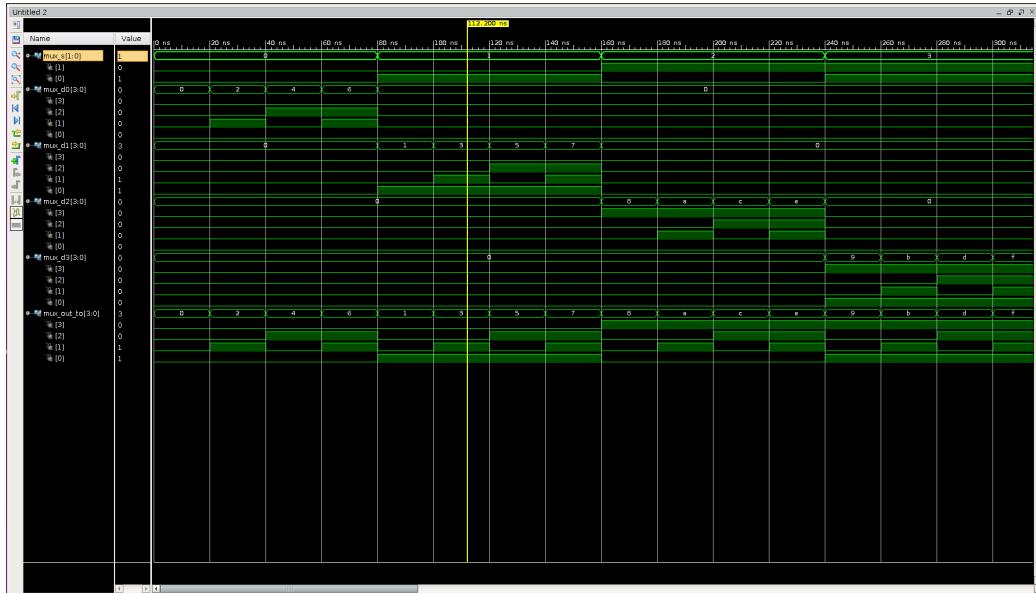


Figure 6: Simulation of 4:1 Mux Module

#### 4.1.3 Simulation of Timing Circuit



Figure 7: Schematic of - 4 Digit 7 Segment Display implementation

#### 4.1.4 Simulation of 7-Seg Display Module

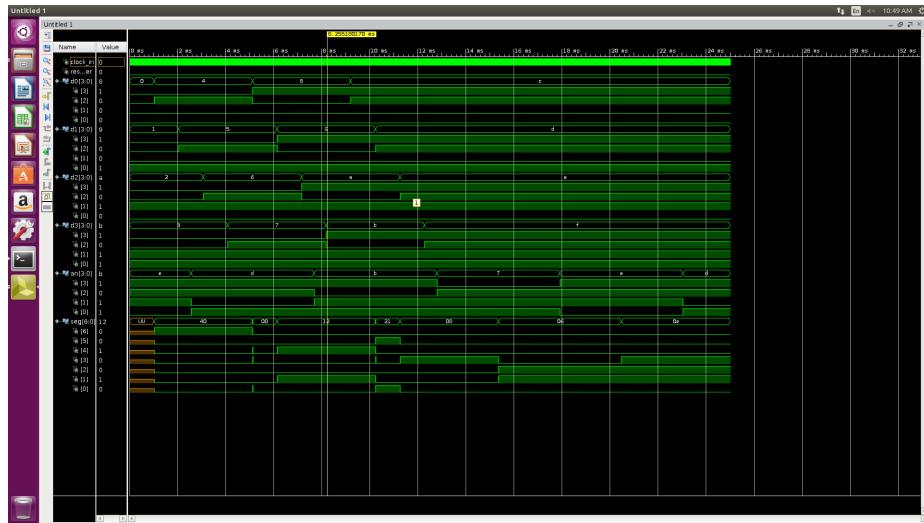


Figure 8: Schematic of - 4 Digit 7 Segment Display implementation

## 4.2 Schematic Snapshots

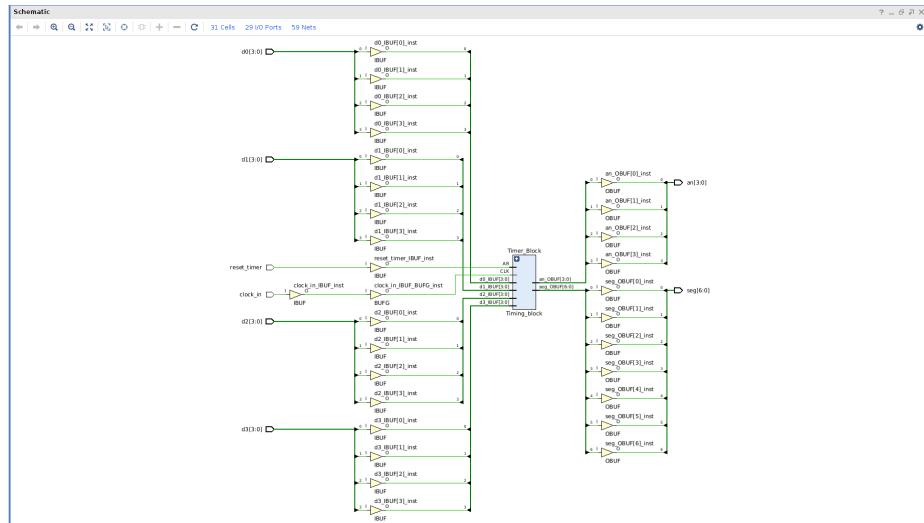


Figure 9: Schematic of - 4 Digit 7 Segment Display implementation

### 4.3 Resource Utilization Table

1. Slice logic					
Site Type	Used	Fixed	Prohibited	Available	Util%
Slice LUTs*	52	0	0	28800	0.25
LUT as Logic	52	0	0	28800	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	58	0.00
RAMB36/FIFO*	0	0	0	58	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 B1

3. DSP					
Site Type	Used	Fixed	Prohibited	Available	Util%
DSPs	0	0	0	98	0.00

Figure 10: Resource Utilization

### 4.4 Output on FPGA Board

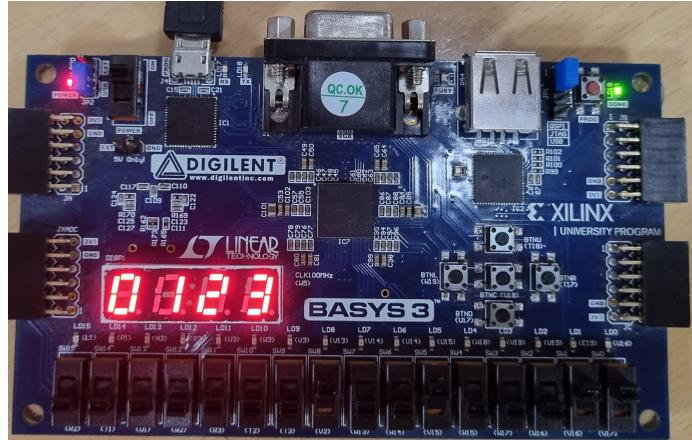


Figure 11: Basys Board Output : Digits 0 – 3

The switches in groups of 4 control one display each from left to right as is visible in the snapshots above and below. No flickering is observed in the physical implementation hence our timing circuit has been implemented as per requirements.

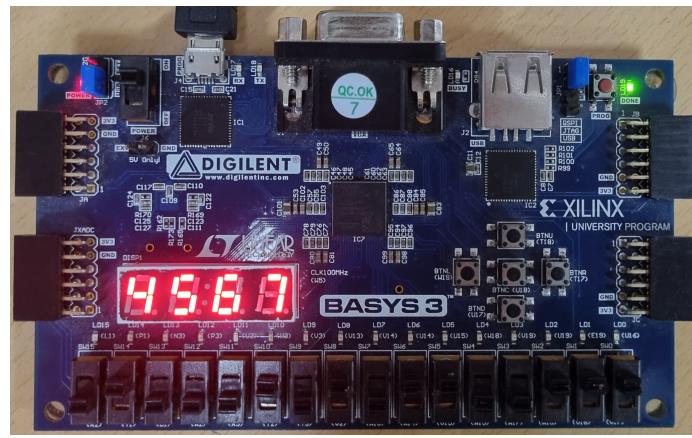


Figure 12: Basys Board output : Digits 4 – 7

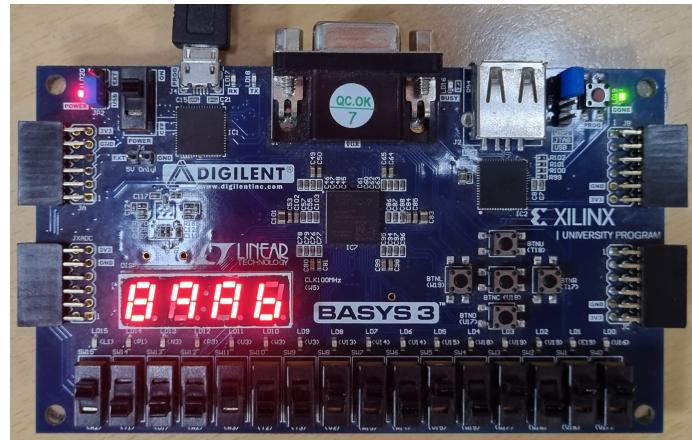


Figure 13: Basys Board output : Digits 8 – b

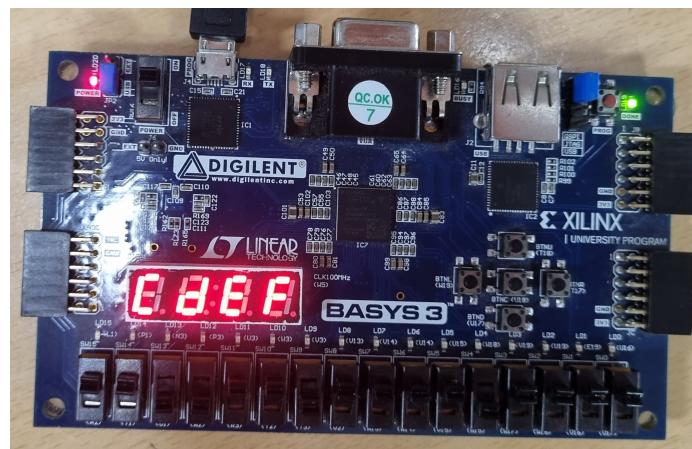


Figure 14: Basys Board output : Digits C – F