LAB NOTE

Subject: Digital Design Principles

Topic: Magnitude Comparator

Student: Minh Quan Tran

July 12th, 2024

Table of Contents

1. Objectives 4
2. Theory and Design 6
2.1 Theory 6
2.2 Requirement 6
2.3 Solution 6
3. VHDL and Verilog 6
3.1 VHDL code 6
3.2 Verilog code 9
4. Pin Planner 14
4.1 Input and Output 14
5. Result 16

TABLE OF FIGURES

Figure 2-1: Octal Decoder's Truth Table 6
Figure 4-1: SCEMA5F31C6N board 14
Figure 4-2: SW0 and SW1 Pin No 15
Figure 4-3: LEDR0's Pin No 15
Figure 4-4: HEX's Pin No 15
Figure 5-1: When input = ‘000’. 16
Figure 5-2: When input = ‘111’ 16
Figure 5-3: When input ‘101’ 17
Figure 5-4: Input ‘101’ with Complement mode 17
Figure 5-5: Input ‘101’ with Multiline mode 18
Figure 5-6: Result for adding them up 18

1. Objectives

- Design a Magnitude Comparator using Quartus software.
- Write a VHDL and Verilog code from the design.
- Push the code and run to the SCEMA5F31C6N board.

2. Theory and Design

2.1 Theory

Magnitude Comparator is a circuit that compare 2 inputs A and B and give out either 3 output ‘100’, ‘010’ and ‘001’ which should represent Greater Than, Equal, Less Than.

2.2 Requirement

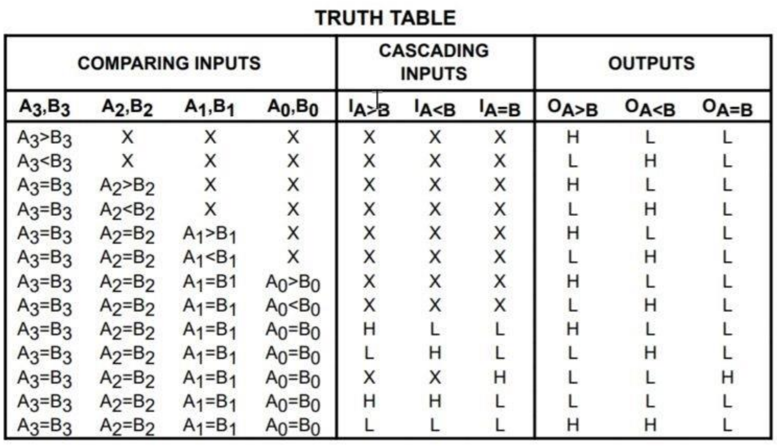
- Use VHDL and Verilog to implement a Magnitude Comparator and display to LED [0:2].
- Introduction to the VHDL Case statement.
- **Above and beyond:** Adding mode: Clock count up from 00:00:00 to 23:59:59 with 2 speed normal speed and 100x speed.

2.3 Solution

To design this system, first need to identify input and output:

- Input will be from SW0 to SW7.
- Output will be LED[2:0].

Then identify what the output will be, from the 2 inputs,



TRUTH TABLE

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A_3,B_3	A_2,B_2	A_1,B_1	A_0,B_0	$I_{A>B}$	$I_{A<B}$	$I_{A=B}$	$O_{A>B}$	$O_{A<B}$	$O_{A=B}$
$A_3>B_3$	X	X	X	X	X	X	H	L	L
$A_3<B_3$	X	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2>B_2$	X	X	X	X	X	H	L	L
$A_3=B_3$	$A_2<B_2$	X	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1>B_1$	X	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1<B_1$	X	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	A_1=B1	$A_0>B_0$	X	X	X	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0<B_0$	X	X	X	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	L	L	H	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	H	L	L	H	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	X	X	H	L	L	H
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	H	H	L	L	L	L
$A_3=B_3$	$A_2=B_2$	$A_1=B_1$	$A_0=B_0$	L	L	L	H	H	L

Figure 2-1: Magnitude Comparator’s Truth Table

3. VHDL and Verilog

3.1 VHDL code

3.1.1 Top entity

```
library ieee;
use ieee.std_logic_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity MTran_Lab7_VHDL_MagnitudeComparator is
port
(
    -- Inputs
    CLOCK_50 : in  std_logic;
    A        : in  std_logic_vector (3 downto 0);
    B        : in  std_logic_vector (3 downto 0);
    mode     : in  std_logic;

    -- Outputs
    LED   : out std_logic_vector (2 downto 0);
    HEX0  : out std_logic_vector (6 downto 0);
    HEX1  : out std_logic_vector (6 downto 0);
    HEX2  : out std_logic_vector (6 downto 0);
    HEX3  : out std_logic_vector (6 downto 0);
    HEX4  : out std_logic_vector (6 downto 0);
    HEX5  : out std_logic_vector (6 downto 0)

);
end entity;

architecture Behavioral of MTran_Lab7_VHDL_MagnitudeComparator is

-- Module declaration
component MTran_Lab7_VHDL_Timer is
port
(
    -- Inputs
    Clk     : in std_logic;
    Enable  : in std_logic;

    -- Outputs
    Seconds : inout integer;
    Minutes : inout integer;
    Hours   : inout integer
);
end component;

-- Singal declaration
signal intA          : integer := 0;
signal intB          : integer := 0;

signal Enable        : std_logic;

signal Seconds       : integer := 0;
signal Minutes       : integer := 0;
signal Hours         : integer := 0;


-- Function
function SevenSegmentDisplay(Number : integer) return std_logic_vector is
```

```
begin
    case (Number) is
    when 0       => return "1000000";
    when 1        => return "1111001";
    when 2        => return "0100100";
    when 3         => return "0110000";
    when 4        => return "0011001";
    when 5         => return "0010010";
    when 6        => return "0000010";
    when 7      => return "1111000";
    when 8      => return "0000000";
    when 9      => return "0010000";
    when others => return "1111111";
    end case;
end SevenSegmentDisplay;


-- Main code
begin
    -- Module instatiate
    module_timer: MTran_Lab7_VHDL_Timer
    port map
    (
        Clk        => CLOCK_50,
        Enable   => Enable,
        Seconds => Seconds,
        Minutes => Minutes,
        Hours    => Hours
    );

    -- Process
    process(CLOCK_50, Seconds, Minutes, Hours, A, B) is
    begin
        if mode = '0' then
            -- Disable Timer
            Enable <= '0';

            -- Reset HEX2
            HEX2 <= "1111111";

            -- Convert A and B from std_logic to integer
            intA <= to_integer(unsigned(A));
            intB <= to_integer(unsigned(B));

            -- Display A and B in decimal

            -- A
            HEX0 <= SevenSegmentDisplay(intA mod 10);
            HEX1 <= SevenSegmentDisplay(intA / 10);

            -- B
            HEX4 <= SevenSegmentDisplay(intB mod 10);
            HEX5 <= SevenSegmentDisplay(intB / 10);

            -- Compare A and B
            if intA > intB then
                LED <= "100";
                HEX3 <= "0100111";
            elsif intA = intB then
                LED <= "010";
                HEX3 <= "0110111";
            else
```

```
                    LED <= "001";
                    HEX3 <= "0110011";
                end if;

            else
                Enable <= '1';
                LED <= "000";
                -- Display Second
                HEX0 <= SevenSegmentDisplay(Seconds mod 10);
                HEX1 <= SevenSegmentDisplay(Seconds / 10);

                -- Display Minute
                HEX2 <= SevenSegmentDisplay(Minutes mod 10);
                HEX3 <= SevenSegmentDisplay(Minutes / 10);

                -- Display Hour
                HEX4 <= SevenSegmentDisplay(Hours mod 10);
                HEX5 <= SevenSegmentDisplay(Hours / 10);


            end if;

        end process;

end architecture;
```

3.1.2 Component Timer

```
library ieee;
use ieee.std_logic_1164.ALL;

entity MTran_Lab7_VHDL_Timer is
port
(
    -- Inputs
    Clk     : in    std_logic;
    Enable  : in    std_logic;

    -- Outputs
    Seconds : inout integer := 0;
    Minutes : inout integer := 0;
    Hours   : inout integer := 0

);
end entity;

architecture RTL of MTran_Lab7_VHDL_Timer is

-- Signal Declaration
signal Ticks                : integer := 0;

-- Procedure
procedure Increment( signal   Counter   : inout integer;
                     constant MaxValue  : in    integer;
                     variable ResetFlag : out   boolean) is
begin
    if Counter = MaxValue - 1 then
        -- Reset Counter
        Counter  <= 0;
        ResetFlag := True;
    else
        Counter <= Counter + 1;
        ResetFlag := False;
    end if;
end procedure;

begin

    process(Clk, Enable) is

    -- Variable Declaring
    variable SecondsFlag      : boolean := false;
    variable MinutesFlag      : boolean := false;
    variable HoursFlag        : boolean := false;

    constant MaxSeconds       : integer := 60;
    constant MaxMinutes       : integer := 60;
    constant MaxHours         : integer := 24;

    constant ClockFrequency   : integer := 50000000;

    -- Main
    begin

    if Enable = '1' then
        if rising_edge(Clk) then

            -- Increment Ticks
            Increment(Ticks, ClockFrequency, SecondsFlag);

            -- Increment Seconds whenever Ticks reach Clock Frequency
            if SecondsFlag then
                Increment(Seconds, MaxSeconds, MinutesFlag);
            end if;

            -- Increment Minute whenever 60 seconds
            if SecondsFlag and MinutesFlag then
                Increment(Minutes, MaxMinutes, HoursFlag);
            end if;

            -- Increment Hours whenever 60 minutes
            if SecondsFlag and MinutesFlag and HoursFlag then
                Increment(Hours, MaxHours, HoursFlag);
            end if;
        end if;
    end if;

    end process;
end architecture;
```

3.2 Verilog code

3.2.1 Top Module

```
module MTran_Lab7_Verilog_MagnitudeComparator
(
    input wire CLOCK_50,
    input wire [3:0] A,
    input wire [3:0] B,
    input mode,
     input speed,
    output reg [2:0] LED,
    output reg [6:0] HEX0,
    output reg [6:0] HEX1,
    output reg [6:0] HEX2,
    output reg [6:0] HEX3,
    output reg [6:0] HEX4,
    output reg [6:0] HEX5
);

// Signals declaration
reg [31:0] intA;
reg [31:0] intB;

reg Enable;

wire [31:0] Seconds1;
wire [31:0] Minutes1;
wire [31:0] Hours1;

wire [31:0] Seconds2;
wire [31:0] Minutes2;
wire [31:0] Hours2;

// Module Instantiation
MTran_Lab7_Verilog_Timer
#(
     .MAXSECONDS(60),
     .MAXMINUTES(60),
     .MAXHOURS(24),
     .CLKFREQUENCY(500000) // 500kHz => Clock 10 times faster
)

Tim1
(
    .Clk(CLOCK_50),
    .Enable(Enable),
    .Seconds(Seconds1),
    .Minutes(Minutes1),
    .Hours(Hours1)
);

MTran_Lab7_Verilog_Timer
#(
     .MAXSECONDS(60),
     .MAXMINUTES(60),
     .MAXHOURS(24),
     .CLKFREQUENCY(50000000) // 50 MHz => Normal Clk
)
Tim2
(
    .Clk(CLOCK_50),
    .Enable(Enable),
    .Seconds(Seconds2),
    .Minutes(Minutes2),
    .Hours(Hours2)
);

// Function for Seven Segment Display
function [6:0] SevenSegmentDisplay;
    input [3:0] Number;
    begin
        case (Number)
            4'h0: SevenSegmentDisplay = 7'b1000000;
            4'h1: SevenSegmentDisplay = 7'b1111001;
            4'h2: SevenSegmentDisplay = 7'b0100100;
            4'h3: SevenSegmentDisplay = 7'b0110000;
            4'h4: SevenSegmentDisplay = 7'b0011001;
            4'h5: SevenSegmentDisplay = 7'b0010010;
            4'h6: SevenSegmentDisplay = 7'b0000010;
            4'h7: SevenSegmentDisplay = 7'b1111000;
            4'h8: SevenSegmentDisplay = 7'b0000000;
            4'h9: SevenSegmentDisplay = 7'b0010000;
            default: SevenSegmentDisplay = 7'b1111111;
        endcase
    end
endfunction



always @(posedge CLOCK_50)
begin
    if (mode == 1'b0)
     begin
        // Disable Timer
        Enable = 1'b0;

        // Reset HEX2
        HEX2 = 7'b1111111;

        // Convert A and B from std_logic to integer
        intA = A;
        intB = B;

        // Display A and B in decimal
        // A
        HEX0 = SevenSegmentDisplay(intA % 10);
        HEX1 = SevenSegmentDisplay(intA / 10);

        // B
        HEX4 = SevenSegmentDisplay(intB % 10);
        HEX5 = SevenSegmentDisplay(intB / 10);

        // Compare A and B
        if (intA > intB)
              begin
                    LED  = 3'b100;
                    HEX3 = 7'b0100111;
              end
          else if (intA == intB)
              begin
                    LED  = 3'b010;
                    HEX3 = 7'b0110111;
              end
          else
              begin
                    LED  = 3'b001;
                    HEX3 = 7'b0110011;
              end

    end
     else
     begin
        Enable = 1'b1;
        LED    = 3'b000;

          if (speed == 1'b0) //normal mode
          begin
```

```
                // Display Second
                HEX0   = SevenSegmentDisplay(Seconds2 % 10);
                HEX1   = SevenSegmentDisplay(Seconds2 / 10);

                // Display Minute
                HEX2   = SevenSegmentDisplay(Minutes2 % 10);
                HEX3   = SevenSegmentDisplay(Minutes2 / 10);

                // Display Hour
                HEX4   = SevenSegmentDisplay(Hours2 % 10);
                HEX5   = SevenSegmentDisplay(Hours2 / 10);
              end
              else if (speed == 1'b1)
              begin
                  // Display Second
                HEX0   = SevenSegmentDisplay(Seconds1 % 10);
                HEX1   = SevenSegmentDisplay(Seconds1 / 10);

                // Display Minute
                HEX2   = SevenSegmentDisplay(Minutes1 % 10);
                HEX3   = SevenSegmentDisplay(Minutes1 / 10);

                // Display Hour
                HEX4   = SevenSegmentDisplay(Hours1 % 10);
                HEX5   = SevenSegmentDisplay(Hours1 / 10);
              end
    end
end

endmodule
```

3.2.2 Module Timer

```
module MTran_Lab7_Verilog_Timer

#(
parameter MAXSECONDS       = 60,
parameter MAXMINUTES       = 60,
parameter MAXHOURS         = 24,
parameter CLKFREQUENCY     = 50000000
)

(
    input wire Clk,
    input wire Enable,
    inout reg [31:0] Seconds,
    inout reg [31:0] Minutes,
    inout reg [31:0] Hours
);

// Signal Declaration
reg [31:0] Ticks = 0;

// Main Logic
always @(posedge Clk)
begin
    if (Enable)
     begin
        // Increment Ticks
      Ticks = Ticks + 1;
        if (Ticks == CLKFREQUENCY - 1)
            begin
```

```
                Ticks   = 0;
                Seconds = Seconds + 1;
            end

        if (Seconds == MAXSECONDS)
            begin
                Seconds = 0;
                Minutes = Minutes + 1;
            end

        if (Minutes == MAXMINUTES)
            begin
                Minutes = 0;
                Hours   = Hours + 1;
            end

        if (Hours == MAXHOURS)
            begin
                Hours   = 0;
            end
   end
end

endmodule
```

4. Pin Planner

4.1 Input and Output

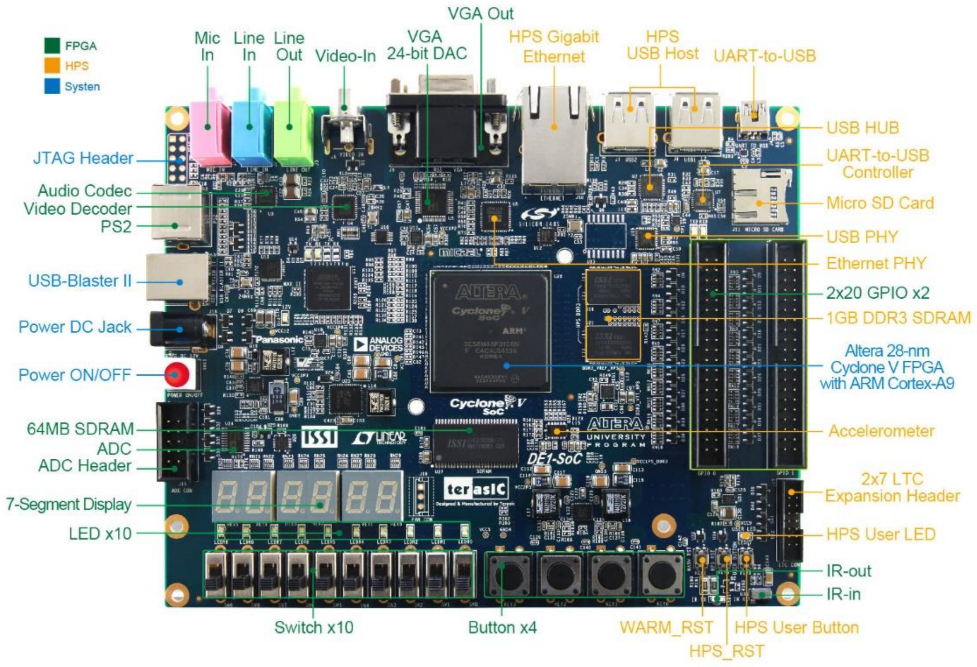


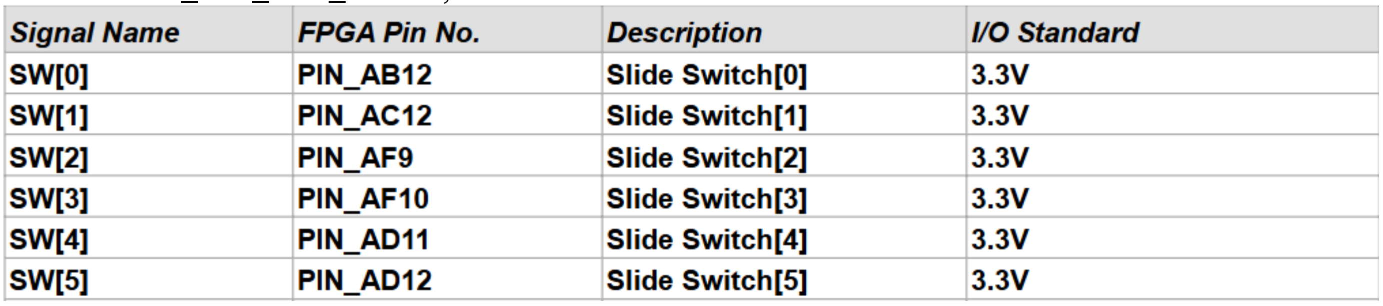
Figure 4-1: SCEMA5F31C6N board

Assigning:

SW[0]:	SW0
SW[1]:	SW1
SW[2]:	SW2
Mode[0]:	SW7
Mode[1]:	SW8
EN:	SW9
LED[0]:	LEDR0
LED[1]:	LEDR1
LED[2]:	LEDR2
LED[3]:	LEDR3
LED[4]:	LEDR4
LED[5]:	LEDR5
LED[6]:	LEDR6
LED[7]:	LEDR7
HEX0[0]:	HEX0[0]
HEX0[1]:	HEX0[1]
HEX0[2]:	HEX0[2]
HEX0[3]:	HEX0[3]
HEX0[4]:	HEX0[4]
HEX0[5]:	HEX0[5]
HEX0[6]:	HEX0[6]

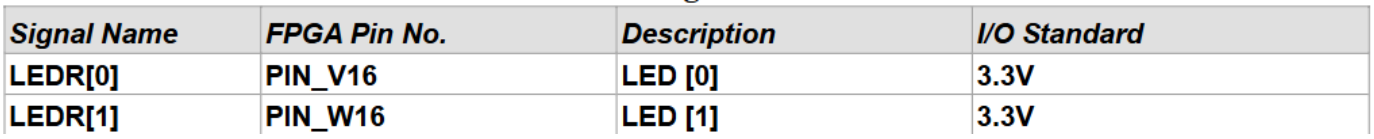
4. Pin Planner

From the DE1_SoC_User_Manual,



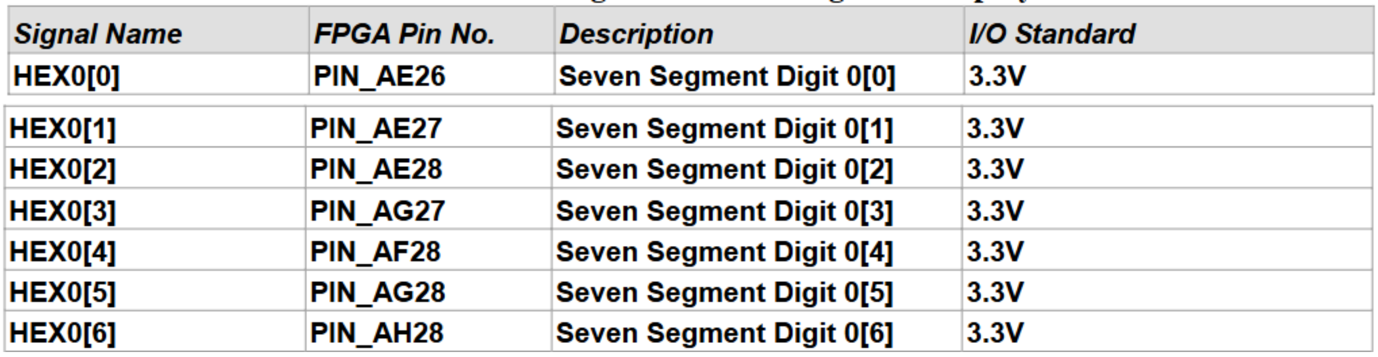
Signal Name	FPGA Pin No.	Description	I/O Standard
SW[0]	PIN_AB12	Slide Switch[0]	3.3V
SW[1]	PIN_AC12	Slide Switch[1]	3.3V
SW[2]	PIN_AF9	Slide Switch[2]	3.3V
SW[3]	PIN_AF10	Slide Switch[3]	3.3V
SW[4]	PIN_AD11	Slide Switch[4]	3.3V
SW[5]	PIN_AD12	Slide Switch[5]	3.3V

Figure 4-2: SW0 and SW1 Pin No



Signal Name	FPGA Pin No.	Description	I/O Standard
LEDR[0]	PIN_V16	LED [0]	3.3V
LEDR[1]	PIN_W16	LED [1]	3.3V

Figure 4-3: LEDR0's Pin No



Signal Name	FPGA Pin No.	Description	I/O Standard
HEX0[0]	PIN_AE26	Seven Segment Digit 0[0]	3.3V
HEX0[1]	PIN_AE27	Seven Segment Digit 0[1]	3.3V
HEX0[2]	PIN_AE28	Seven Segment Digit 0[2]	3.3V
HEX0[3]	PIN_AG27	Seven Segment Digit 0[3]	3.3V
HEX0[4]	PIN_AF28	Seven Segment Digit 0[4]	3.3V
HEX0[5]	PIN_AG28	Seven Segment Digit 0[5]	3.3V
HEX0[6]	PIN_AH28	Seven Segment Digit 0[6]	3.3V

Figure 4-4: HEX's Pin No

5. Result

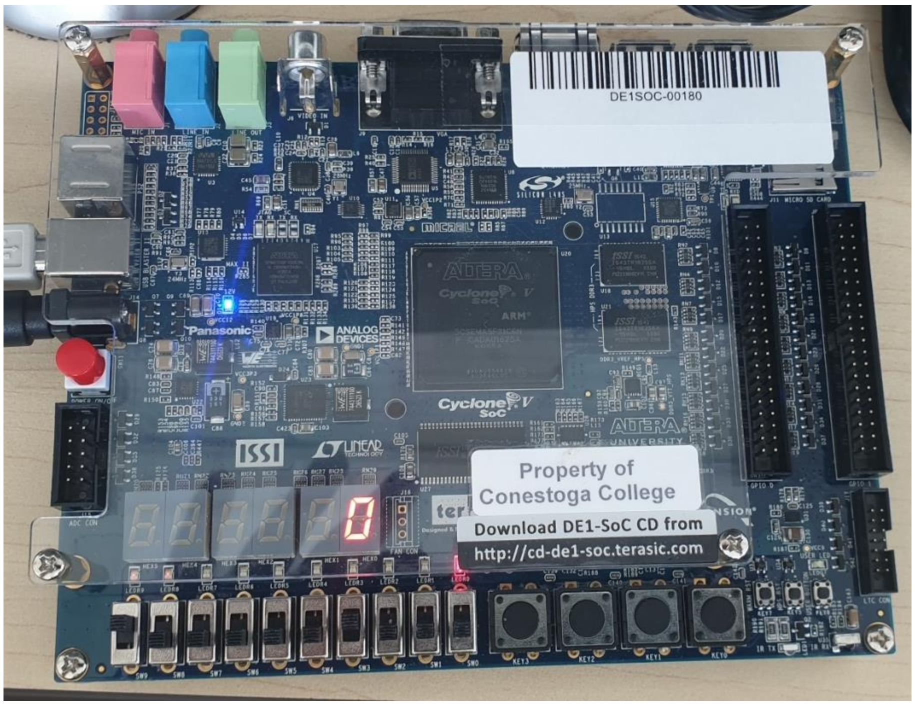


Figure 5-1: When input = ‘000’.

SW2 = 0, SW1 = 0, SW0 = 0 which should be 0, **LED0** asserted and the 7 segment LED displaying **0**.

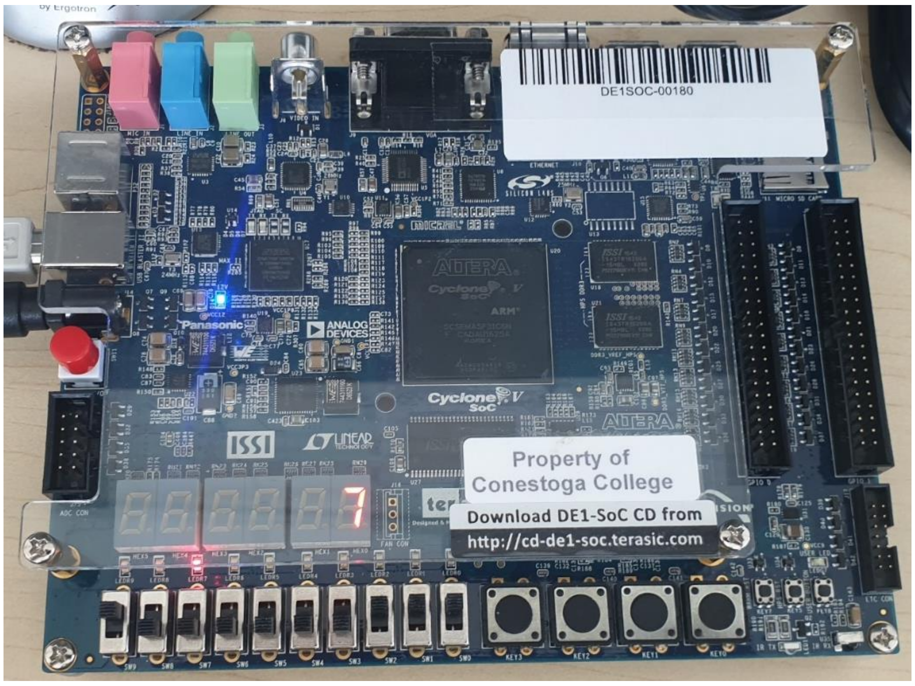


Figure 5-2: When input = ‘111’

SW2 = 1, SW1 = 1, SW0 = 1 which should be 7, **LED7** asserted and the 7 segment LED displaying **7**.

5. Result

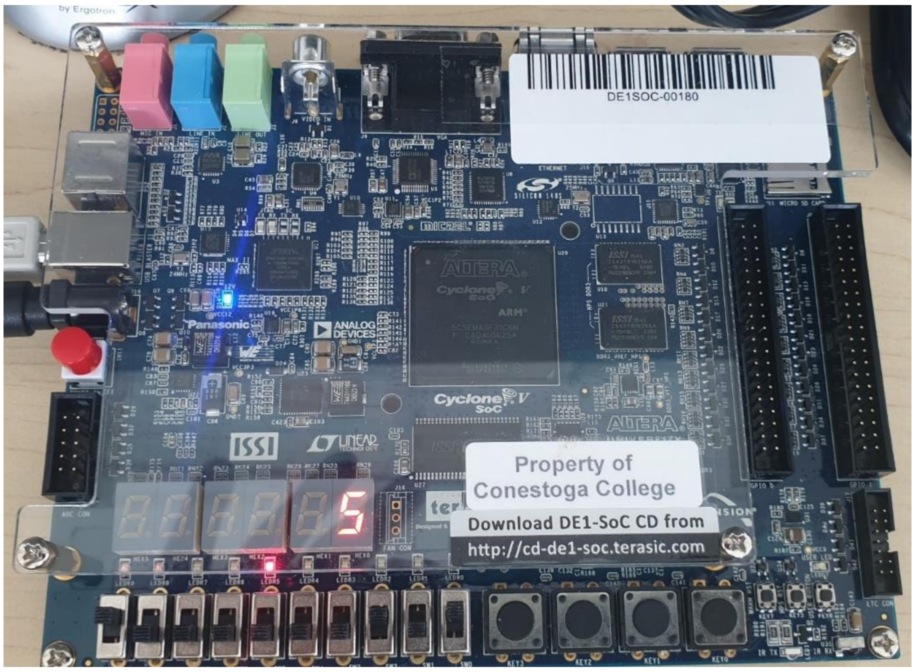


Figure 5-3: When input '101'

SW2 = 1, SW1 = 0, SW0 = 1 which should be 5, **LED5** asserted and the 7 segment LED displaying **5**.

Different mode:

- **Complement mode:**

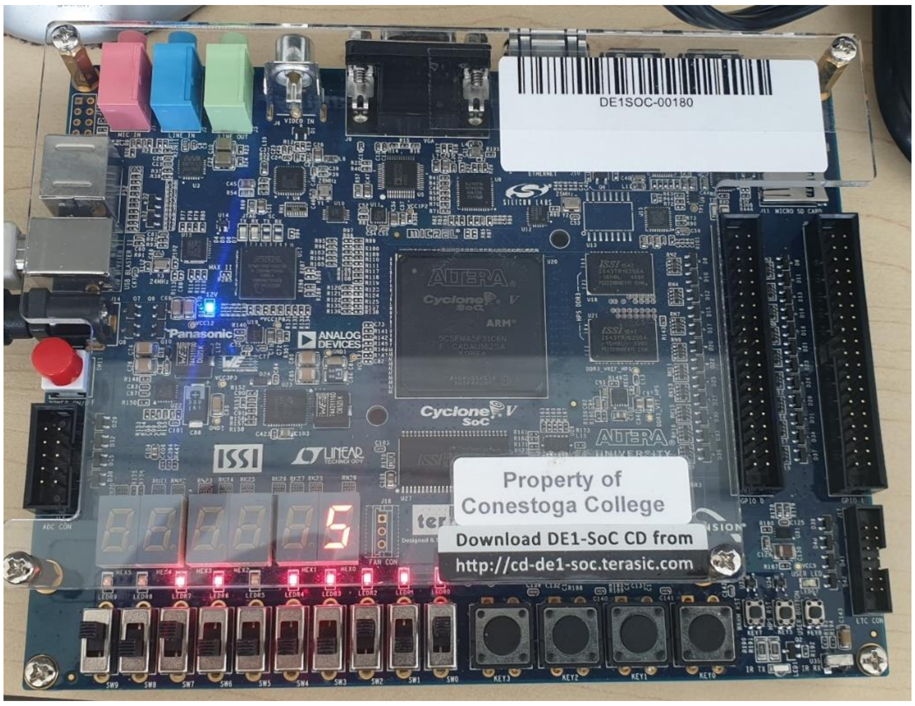


Figure 5-4: Input '101' with Complement mode

In this mode, rather than LED5 assert, all other LED except LED5 asserted. To activate this mode SW [8:7] = '01'.

5. Result

- **Multiline Mode:**

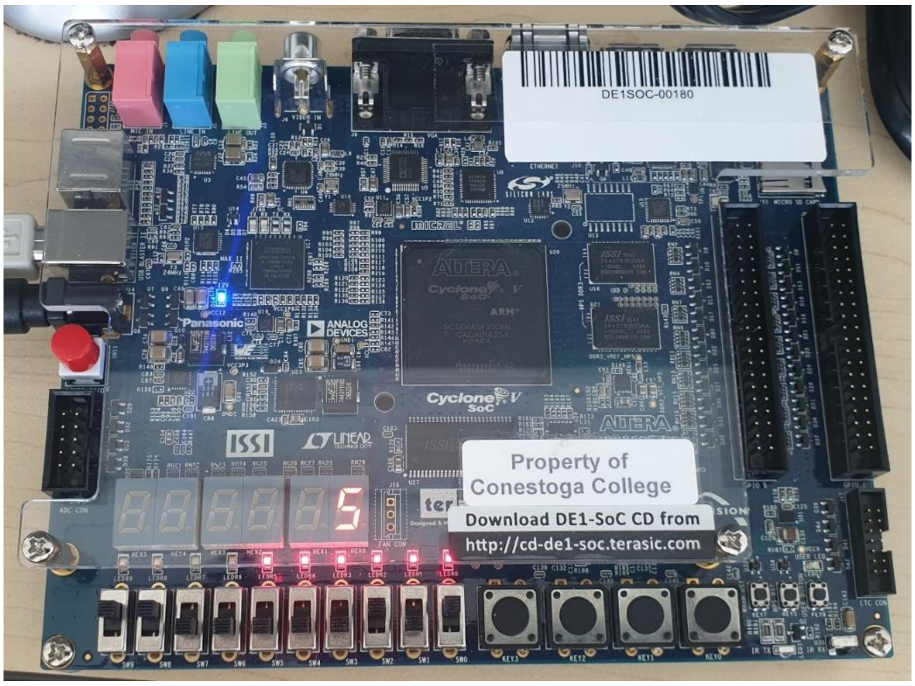


Figure 5-5: Input ‘101’ with Multiline mode

In this mode, when input is ‘101’ which is **5**, LED0 to LED5 lit up. To activate this mode SW [8:7] = ‘10’.

- **Complement Multiline Mode:**

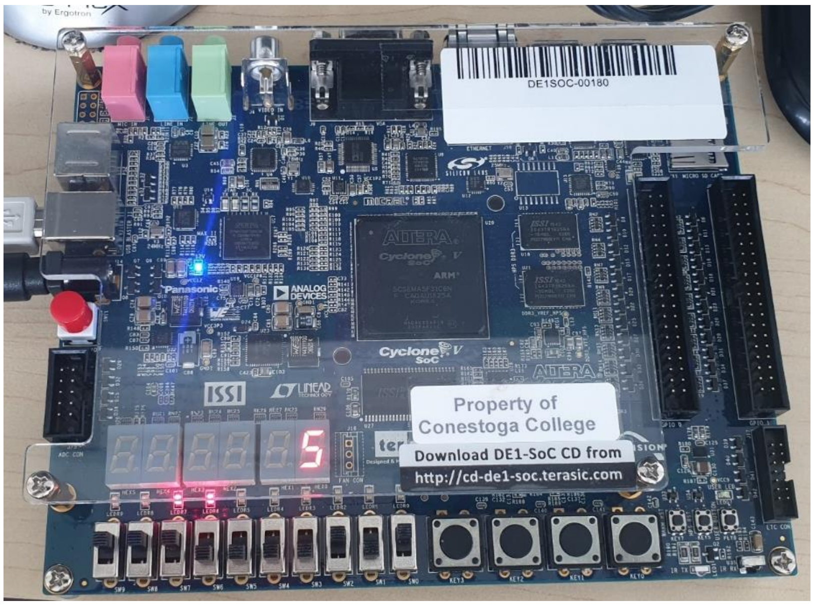


Figure 5-6: Result for adding them up

In this mode, it take the output of Multiline Mode and complemented it. To activate this mode SW [8:7] = ‘11’

REFERENCES