LAB NOTE

Subject: Digital Design Principles

Topic: Shift Register

Student: Minh Quan Tran

July 26th, 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
3.1.1 Top entity 6
3.2 Verilog code 8
4. Pin Planner 10
4.1 Input and Output 10
5. Result 12

TABLE OF FIGURES

Figure 2-1: Shift Register's Truth Table 6
Figure 4-1: SCEMA5F31C6N board 10
Figure 4-2: SW0 and SW1 Pin No 11
Figure 4-3: LEDR0's Pin No 11
Figure 4-4: HEX's Pin No 11
Figure 5-1: When input = '000'. 12
Figure 5-2: When input = '111' 12
Figure 5-3: When input '101' 13
Figure 5-4: Input '101' with Complement mode 13
Figure 5-5: Input '101' with Multiline mode 14
Figure 5-6: Result for adding them up 14

1. Objectives

- Design a Counters 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

A shift register is a digital circuit used for storing and transferring data, consisting of a series of flip-flops connected in a chain. Data is shifted through these flip-flops sequentially, either left or right, with each clock pulse. There are various types of shift registers, including Serial-In Serial-Out (SISO), Serial-In Parallel-Out (SIPO), Parallel-In Serial-Out (PISO), and Parallel-In Parallel-Out (PIPO), each serving different data input and output requirements. Shift registers are crucial in digital systems for tasks such as data storage, data transfer, and data conversion between serial and parallel formats.

2.2 Requirement

- Use VHDL and Verilog to implement the shift and display to LED [1:4].
- Introduction to the VHDL Case statement.
- **Above and beyond:** Adding Complement mode

2.3 Solution

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

- Input will be from BTN0 to BTN3
    - BTN0 : CLK
    - BTN1 : Speed
    - BTN3 : Reset
- Output will be LED[4:1].

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

MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS						OUTPUTS			
	MR	S_1	S_0	D_{SR}	D_{SL}	P_n	Q_0	Q_1	Q_2	Q_3
Reset	L	X	X	X	X	X	L	L	L	L
Hold	H	l	l	X	X	X	q_0	q_1	q_2	q_3
Shift Left	H H	h h	l l	X X	l h	X X	q_1 q_1	q_2 q_2	q_3 q_3	L H
Shift Right	H H	l l	h h	l h	X X	X X	L H	q_0 q_0	q_1 q_1	q_2 q_2
Parallel Load	H	h	h	X	X	P_0	P_0	P_1	P_2	P_3

L = LOW Voltage Level
H = HIGH Voltage Level
X = Don't Care
l = LOW voltage level one set-up time prior to the LOW to HIGH clock transition
h = HIGH voltage level one set-up time prior to the LOW to HIGH clock transition
p_n (q_n) = Lower case letters indicate the state of the referenced input (or output) one set-up time prior to the LOW to HIGH clock transition.

Figure 2-1: Shift Register's Truth Table

3. VHDL and Verilog

3.1 VHDL code

3.1.1 Top entity

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

entity MTran_Lab9_VHDL_ShiftRegister is
port
(
        -- Inputs
        BTN   : in std_logic;
        Speed : in std_logic;
        Reset : in std_logic;
        SW    : in std_logic_vector (7 downto 0);

        -- Outputs
        LED   : out std_logic_vector (3 downto 0);
        HEX0  : out std_logic_vector (6 downto 0)

);
end entity;

architecture behavioral of MTran_Lab9_VHDL_ShiftRegister is

-- Signal Declaration
signal Counter : integer                        := 1;

signal Output  : std_logic_vector (3 downto 0) := "0000";
signal Sel     : std_logic_Vector (1 downto 0);

-- Function declaration
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;

begin

-- Process

        -- Check Sel
        process(SW(6),SW(7)) is
        begin
                Sel <= SW(7) & SW(6);
        end process;
```

```
	-- Check Speed
	process(Speed) is
	begin
		if Speed ='0' then
			if Counter < 4 then
				Counter <= Counter * 2;
			else
				Counter <= 1;
			end if;
		end if;

	end process;

	-- Main code (Shift)
	process(Sel, SW, BTN, Reset) is
	begin

	-- Reset BTN
	if Reset = '0' then
		Output  <= "0000";
	elsif rising_edge(BTN) then
		if Counter = 1 then

		case (Sel) is
		when "00" => NULL;
		when "01" => Output <= Output(2 downto 0) & SW(0);
		when "10" => Output <= SW(5) & Output(3 downto 1);
		when "11" => Output <= SW(4 downto 1);
		end case;
		elsif Counter = 2 then

		case (Sel) is
		when "00" => NULL;
		when "01" => Output <= Output(1 downto 0) & SW(0) & SW(0);
		when "10" => Output <= SW(5) & SW(5) & Output(3 downto 2);
		when "11" => Output <= SW(4 downto 1);
		end case;


		elsif Counter = 4 then
		case (Sel) is
		when "00" => NULL;
		when "01" => Output <= SW(0) & SW(0) & SW(0) & SW(0);
		when "10" => Output <= SW(5) & SW(5) & SW(5) & SW(5);
		when "11" => Output <= SW(4 downto 1);
		end case;


		end if;
	end if;
	end process;

	-- Output
	process(Output, Counter)
	begin
		LED  <= Output;
		HEX0 <= SevenSegmentDisplay(Counter);
	end process;
end Behavioral;
```

3.2 Verilog code

```
module MTran_Lab9_Verilog_ShiftRegister
(
    input wire BTN,
    input wire Speed,
    input wire Reset,
    input wire [7:0] SW,
    output reg [3:0] LED,
    output reg [6:0] HEX0
);

    // Signal Declarations
    reg [3:0] Output;
    reg [1:0] Sel;
    integer Counter = 1;

    // Seven Segment Display function
    function [6:0] SevenSegmentDisplay;
        input integer Number;
        begin
            case (Number)
                0: SevenSegmentDisplay = 7'b1000000;
                1: SevenSegmentDisplay = 7'b1111001;
                2: SevenSegmentDisplay = 7'b0100100;
                3: SevenSegmentDisplay = 7'b0110000;
                4: SevenSegmentDisplay = 7'b0011001;
                5: SevenSegmentDisplay = 7'b0010010;
                6: SevenSegmentDisplay = 7'b0000010;
                7: SevenSegmentDisplay = 7'b1111000;
                8: SevenSegmentDisplay = 7'b0000000;
                9: SevenSegmentDisplay = 7'b0010000;
                default: SevenSegmentDisplay = 7'b1111111;
            endcase
        end
    endfunction

    // Check Sel
    always @(*)
       begin
        Sel = {SW[7], SW[6]};
    end

    // Check Speed
    always @(negedge Speed)
       begin
        if (Speed == 0)
                begin
            if (Counter < 4)
                Counter = Counter * 2;
            else
                Counter = 1;
        end
    end

    // Main code (Shift)
    always @(negedge BTN or negedge Reset)
       begin
        if (Reset == 0)
            Output <= 4'b0000;
        else if (BTN == 0)
                begin
```

```
            if (Counter == 1)
                            begin
                case (Sel)
                    2'b00: Output = Output;
                    2'b01: Output = {Output[2:0], SW[0]};
                    2'b10: Output = {SW[5], Output[3:1]};
                    2'b11: Output = SW[4:1];
                endcase
            end
                            else if (Counter == 2)
                            begin
                case (Sel)
                    2'b00: Output = Output;
                    2'b01: Output = {Output[1:0], SW[0], SW[0]};
                    2'b10: Output = {SW[5], SW[5], Output[3:2]};
                    2'b11: Output = SW[4:1];
                endcase
            end
                            else if (Counter == 4)
                            begin
                case (Sel)
                    2'b00: Output = Output;
                    2'b01: Output = {SW[0], SW[0], SW[0], SW[0]};
                    2'b10: Output = {SW[5], SW[5], SW[5], SW[5]};
                    2'b11: Output = SW[4:1];
                endcase
            end
        end
    end

    // Output
    always @(*)
        begin
         LED  = Output;
         HEX0 = SevenSegmentDisplay(Counter);
    end
endmodule
```

4. Pin Planner

4.1 Input and Output

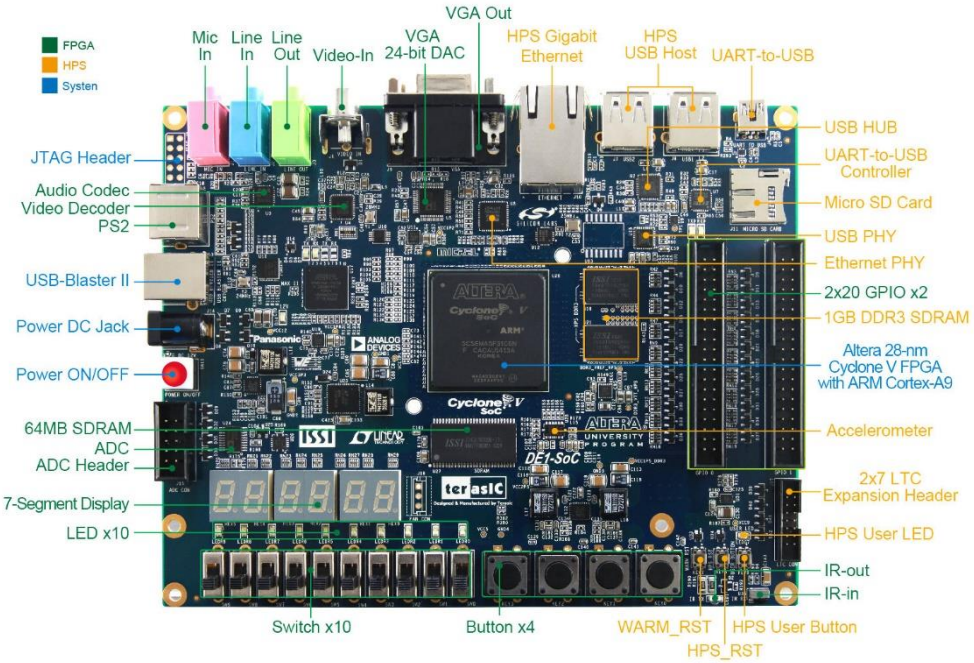

Figure 4-1: SCEMA5F31C6N board

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

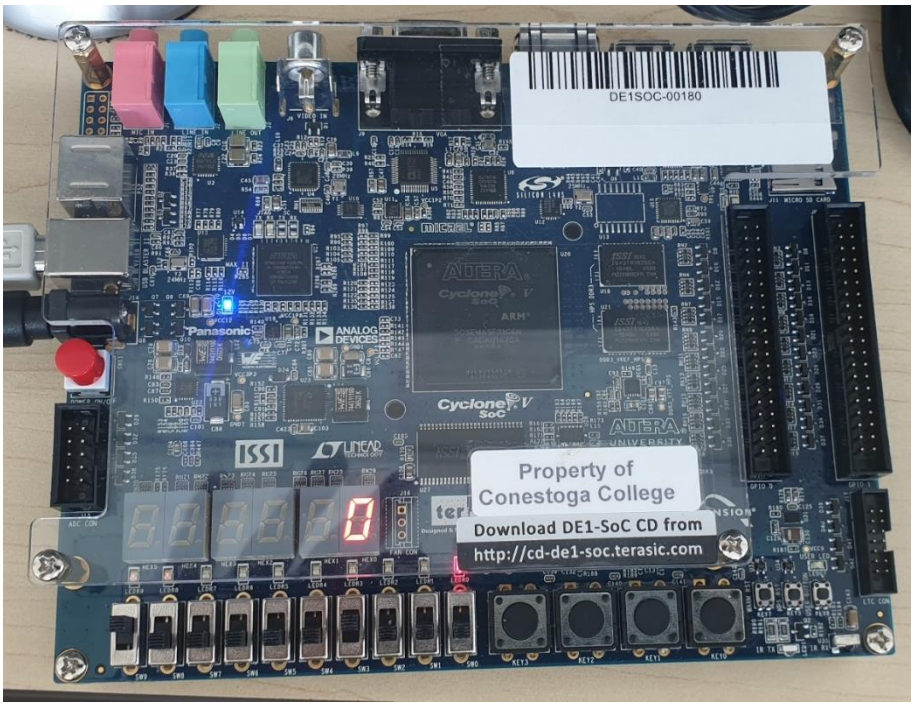

Figure 5-1: When input = ‘000’.

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

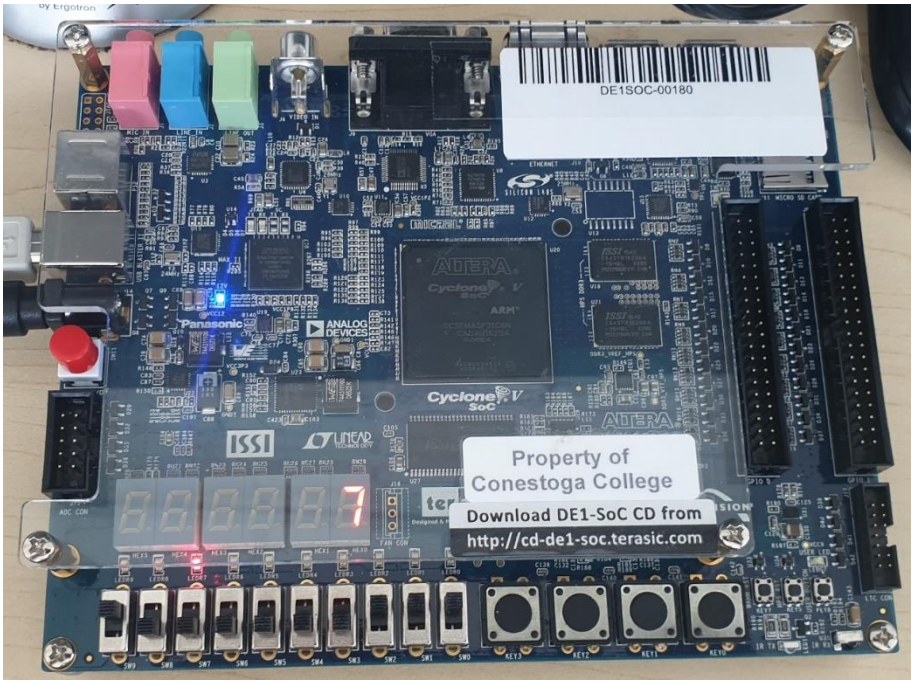

Figure 5-2: When input = ‘111’

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

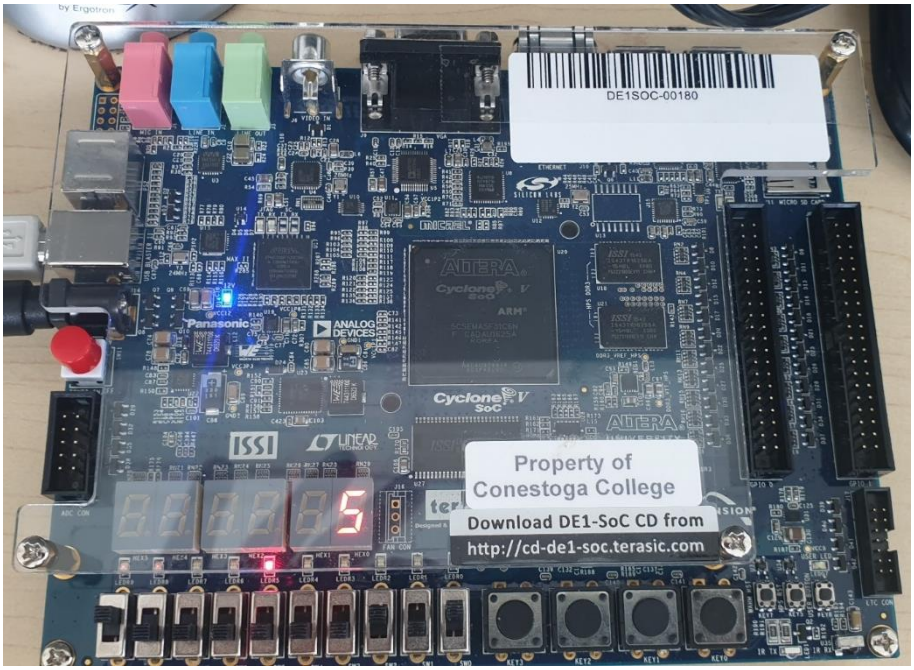

Figure 5-3: When input '101'

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

Different mode:

- **Complement mode:**

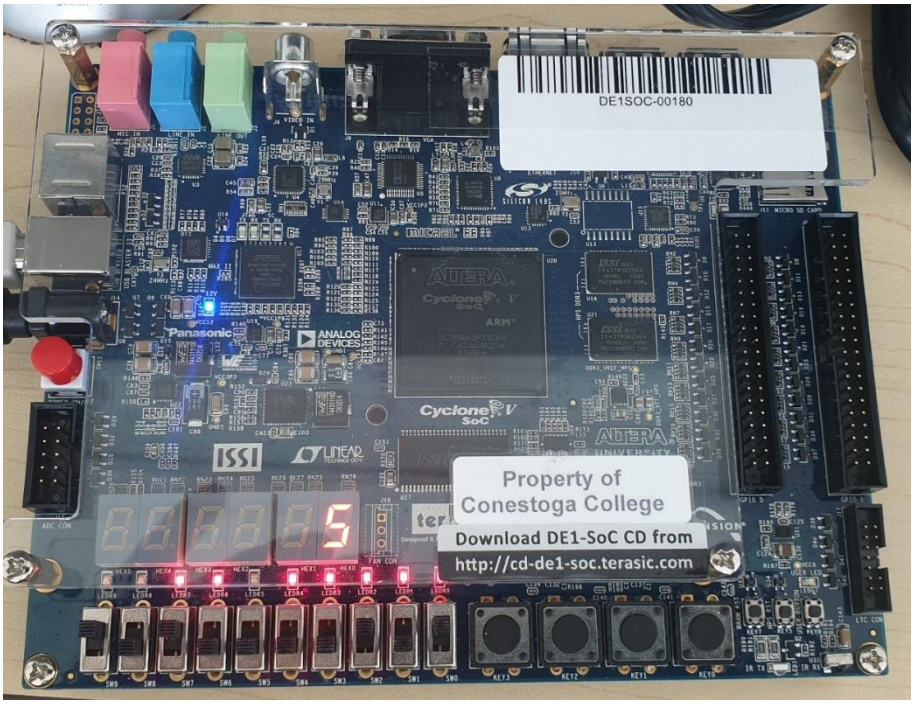

Figure 5-4: Input '101' with Complement mode

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:**

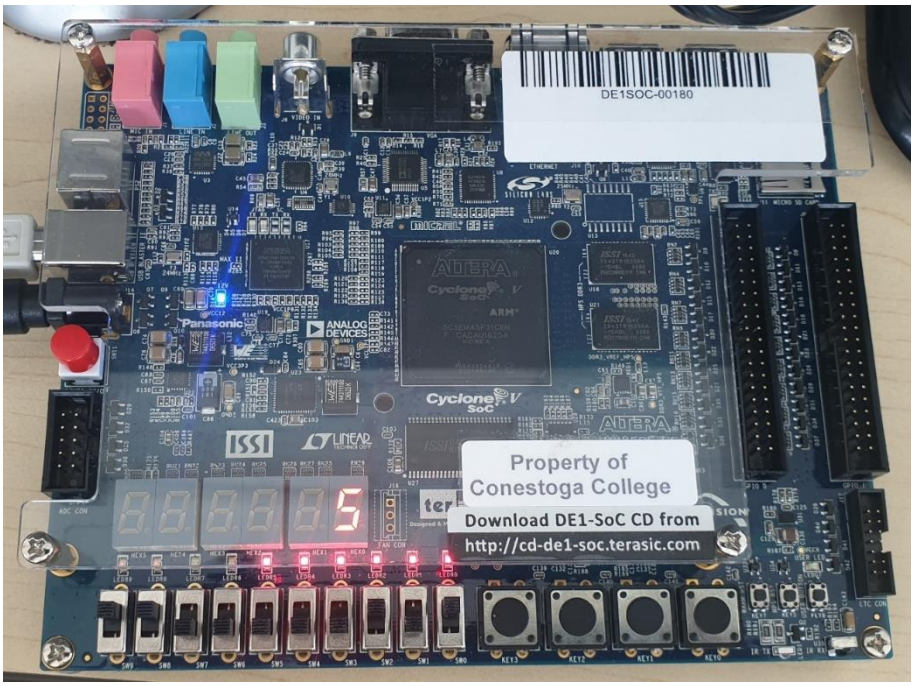

Figure 5-5: Input '101' with Multiline mode

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:**

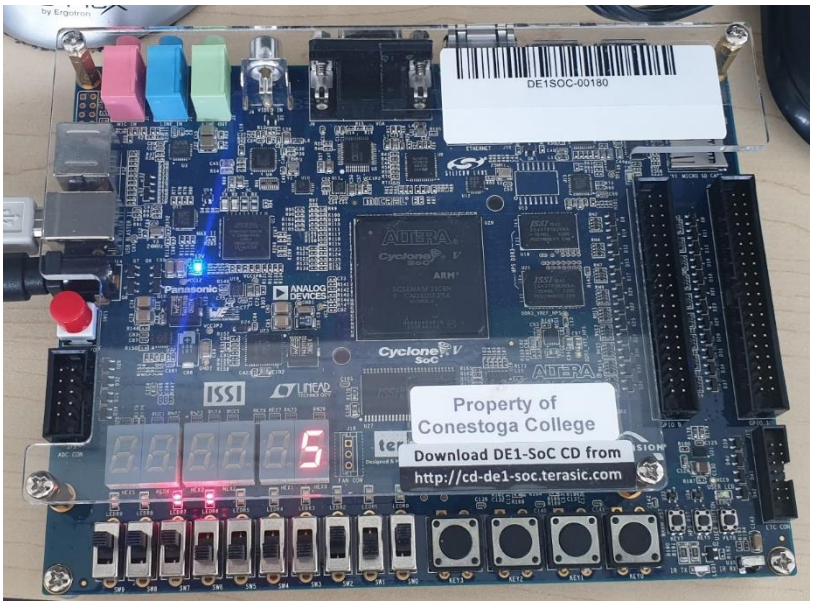

Figure 5-6: Result for adding them up

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

REFERENCES