Digital System Design and Implementation

Lab #1

姓名:黃鉦淳 學號:108303013

a. Verilog codes

```
module Main(Enable_G0, Enable_G1, SevenSeg_G0, SevenSeg_G1, Switch, LED);
input [7:0]Switch;
output [3:0]Enable_G0, Enable_G1;
output [7:0]SevenSeg_G0, SevenSeg_G1;
output [15:0]LED;

reg [2:0]Seg7_idx;
reg [3:0]Seg7_num;
reg [15:0]LED_num;
wire [1:0]SW_Positon, SW_Spacing, SW_Length;
wire SW_Triple, SW_LeftOr_Right;

SevenSegment
Seg7x8(.Seg7_idx(Seg7_idx), .Seg7_num(Seg7_num), .Enable_G0(Enable_G0), .Enable_G1(Enable_G1), .SevenSeg_G0(SevenSeg_G0), .SevenSeg_G1(SevenSeg_G1));
Switch_DIP
Switch8(.Switch(Switch), .SW_Positon(SW_Positon), .SW_Spacing(SW_Spacing), .SW_Length(SW_Length), .SW_Triple(SW_Triple), .SW_LeftOr_Right(SW_LeftOr_Right));
SMD_LED LED16(.LED(LED), .LED_num(LED_num));

integer len,group,tmp;
always @(SW_Triple or SW_Length or SW_Positon or SW_Spacing)
begin
    LED_num = 16'b0;
    for (group=0;group<(SW_Triple+2);group=group+1)
    begin
```

```
            for (len=0;len<SW_Length;len=len+1)
            begin
                tmp = SW_Positon+group*(SW_Length+SW_Spacing)+len;
                if(tmp<16)
                    LED_num[tmp+:1]=1'b1;
            end
        end
end

always @(SW_LeftOr_Right or SW_Positon or SW_Spacing)
begin
    if(SW_LeftOr_Right)
    begin
        Seg7_idx=0;Seg7_num=SW_Positon*5;
    end
    else
    begin
        Seg7_idx=4;Seg7_num=SW_Spacing+10;
    end
end
endmodule

module SevenSegment(Seg7_idx, Seg7_num, Enable_G0, Enable_G1, SevenSeg_G0,
SevenSeg_G1);
input [2:0]Seg7_idx;
input [3:0]Seg7_num;
output reg [3:0]Enable_G0, Enable_G1;
output reg [7:0]SevenSeg_G0, SevenSeg_G1;
reg [7:0]SevenSeg;

always @(Seg7_idx or SevenSeg)
begin
    Enable_G0=4'b0000;
    Enable_G1=4'b0000;
    case(Seg7_idx)
        0:Enable_G0=4'b0001;
        1:Enable_G0=4'b0010;
        2:Enable_G0=4'b0100;
        3:Enable_G0=4'b1000;
        4:Enable_G1=4'b0001;
        5:Enable_G1=4'b0010;
        6:Enable_G1=4'b0100;
        7:Enable_G1=4'b1000;
    endcase

    if(Seg7_idx<4)
    begin
        SevenSeg_G0=SevenSeg;SevenSeg_G1=8'b0000_0000;
    end
    else
    begin
        SevenSeg_G0=8'b0000_0000;SevenSeg_G1=SevenSeg;
    end
end

always @(Seg7_num)
begin
    case(Seg7_num)
        4'h0: SevenSeg=8'b0011_1111;
        4'h1: SevenSeg=8'b0000_0110;
        4'h2: SevenSeg=8'b0101_1011;
        4'h3: SevenSeg=8'b0100_1111;
        4'h4: SevenSeg=8'b0110_0110;
        4'h5: SevenSeg=8'b0110_1101;
        4'h6: SevenSeg=8'b0111_1101;
        4'h7: SevenSeg=8'b0000_0111;
        4'h8: SevenSeg=8'b0111_1111;
        4'h9: SevenSeg=8'b0110_1111;
        4'hA: SevenSeg=8'b0101_1111;
        4'hB: SevenSeg=8'b0111_1100;
        4'hC: SevenSeg=8'b0101_1000;
        4'hD: SevenSeg=8'b0101_1110;
        4'hE: SevenSeg=8'b0111_1001;
        4'hF: SevenSeg=8'b0111_0001;
    endcase
end
```

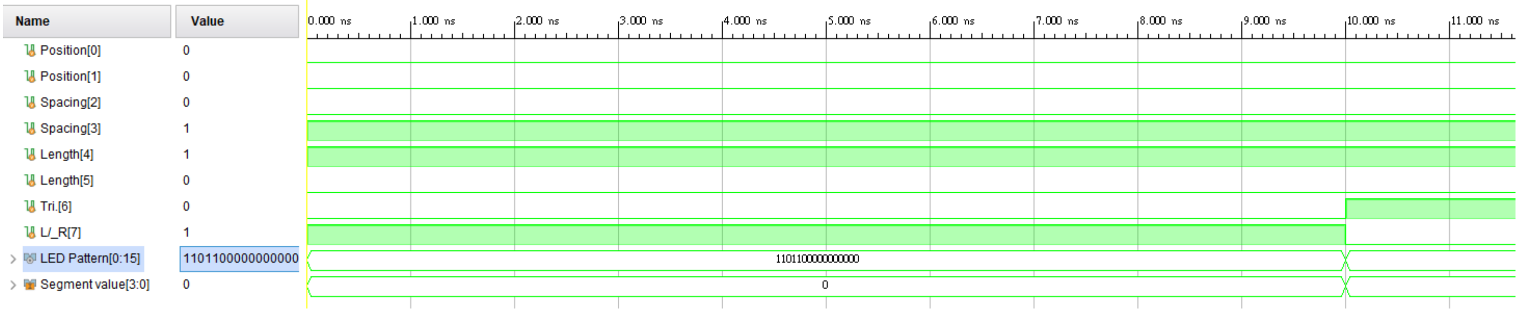
```
endmodule

module Switch_DIP(Switch, SW_Positon, SW_Spacing, SW_Length, SW_Triple,
SW_LeftOr_Right);
input [7:0]Switch;
output reg [1:0]SW_Positon, SW_Spacing, SW_Length;
output reg SW_Triple, SW_LeftOr_Right;

always @(Switch)
begin
	SW_Positon = {Switch[0],Switch[1]};
	SW_Spacing = {Switch[2],Switch[3]};
	SW_Length = {Switch[4],Switch[5]};
	SW_Triple = Switch[6];
	SW_LeftOr_Right = Switch[7];
end
endmodule

module SMD_LED(LED, LED_num);
input [15:0]LED_num;
output [15:0]LED;
assign LED=LED_num;
endmodule
```

b. Test bench

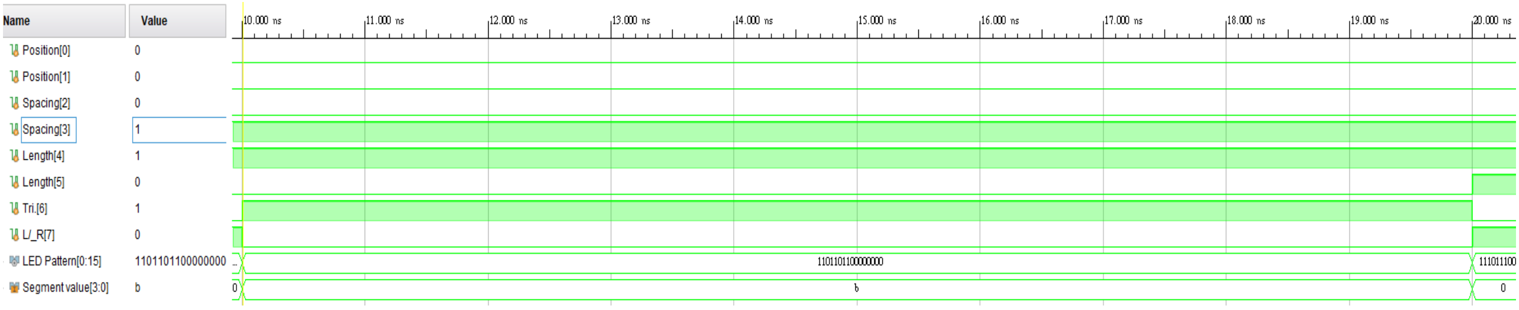
```
`timescale 1ns / 1ps

module Main_tb();
reg [7:0]Switch;
wire [15:0]LED;
wire [3:0]Enable_G0, Enable_G1;
wire [7:0]SevenSeg_G0, SevenSeg_G1;
Main main(Enable_G0, Enable_G1, SevenSeg_G0, SevenSeg_G1, Switch, LED);

initial begin
	// 108303013
	// (a) 0x01 -> 0b0001, Spacing = 2
```

```
        // (b) 0x03 -> 0b0011, Spacing = 2
        // (c) 0x0F -> 0b1111, Spacing = 1
        // Switch[0:1,4:5], Switch[2,3]
        // Switch = {Lor_R[7], Triple[6], Length[4:5], Spacing[2:3], Position[0:1]}
        // Setting (a)
        Switch = {1'b1, 1'b0, 2'b01, 2'd2, 2'b00}; #10;
        Switch = {1'b0, 1'b1, 2'b01, 2'd2, 2'b00}; #10;
        // Setting (b)
        Switch = {1'b1, 1'b0, 2'b11, 2'd2, 2'b00}; #10;
        Switch = {1'b0, 1'b1, 2'b11, 2'd2, 2'b00}; #10;
        // Setting (c)
        Switch = {1'b1, 1'b0, 2'b11, 2'd1, 2'b11}; #10;
        Switch = {1'b0, 1'b1, 2'b11, 2'd1, 2'b11}; #10;
        $finish;
    end
endmodule
```

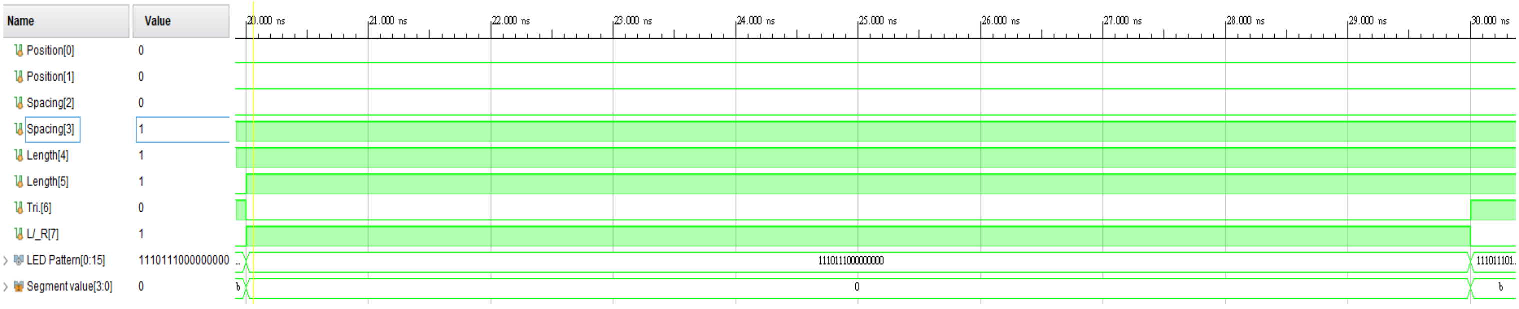
c. Behavior simulation results



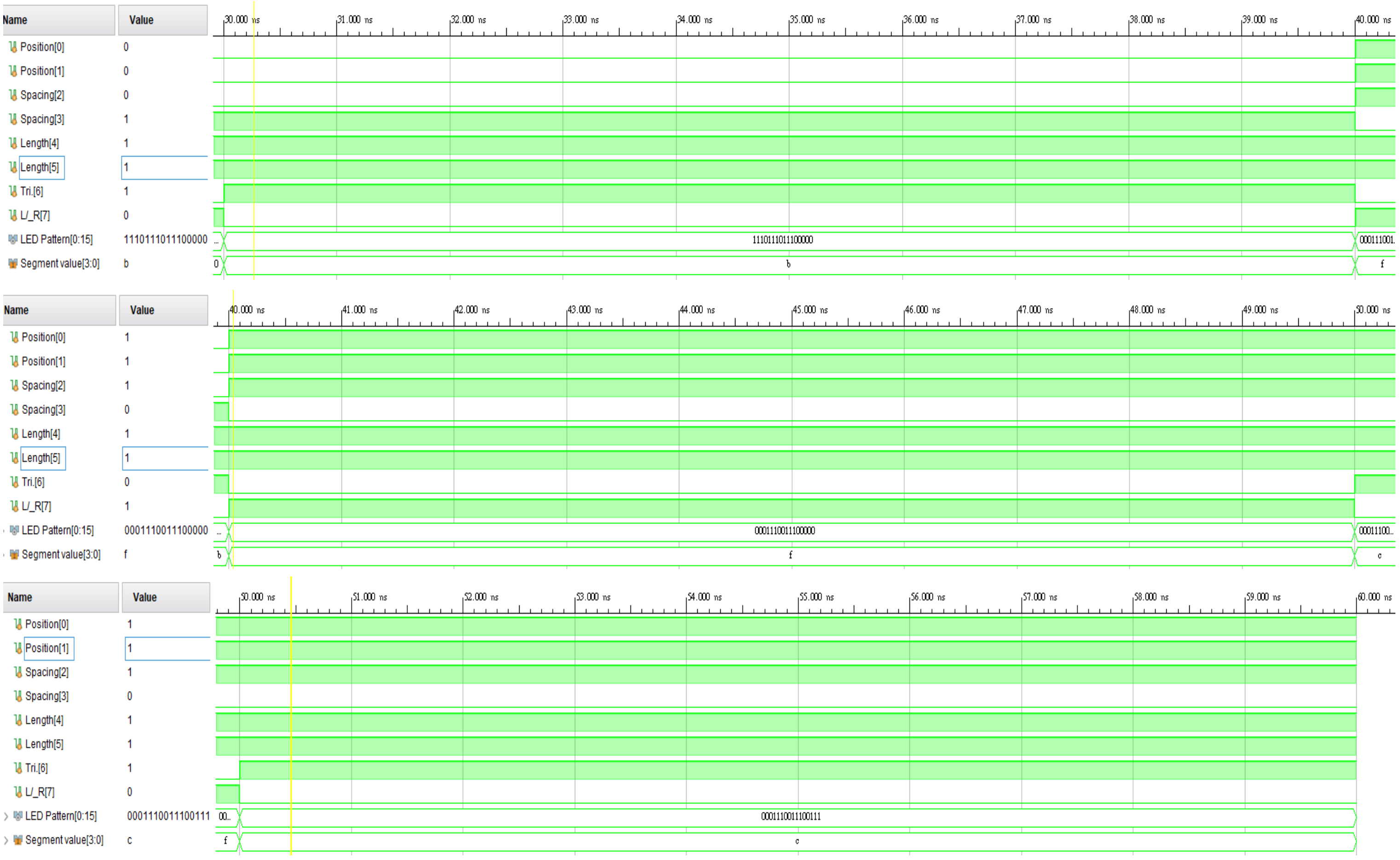
Name	Value
Position[0]	0
Position[1]	0
Spacing[2]	0
Spacing[3]	1
Length[4]	1
Length[5]	0
Tri.[6]	0
L/_R[7]	1
LED Pattern[0:15]	1101100000000000
Segment value[3:0]	0



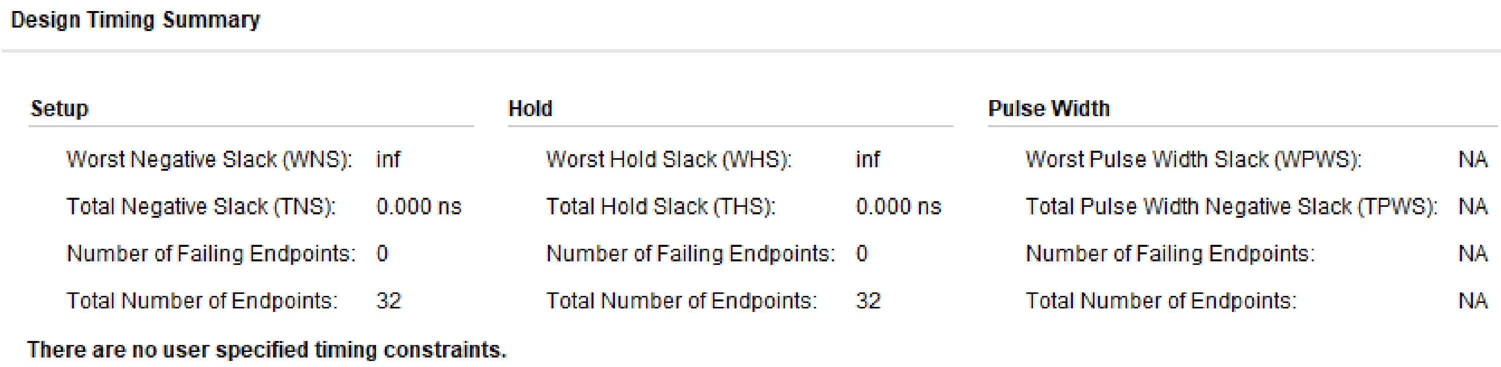
Name	Value
Position[0]	0
Position[1]	0
Spacing[2]	0
Spacing[3]	1
Length[4]	1
Length[5]	0
Tri.[6]	1
L/_R[7]	0
LED Pattern[0:15]	1101101100000000
Segment value[3:0]	b



Name	Value
Position[0]	0
Position[1]	0
Spacing[2]	0
Spacing[3]	1
Length[4]	1
Length[5]	1
Tri.[6]	0
L/_R[7]	1
LED Pattern[0:15]	1110111000000000
Segment value[3:0]	0



d. Synthesis timing report



Design Timing Summary

Setup

Worst Negative Slack (WNS): inf

Total Negative Slack (TNS): 0.000 ns

Number of Failing Endpoints: 0

Total Number of Endpoints: 32

Hold

Worst Hold Slack (WHS): inf

Total Hold Slack (THS): 0.000 ns

Number of Failing Endpoints: 0

Total Number of Endpoints: 32

Pulse Width

Worst Pulse Width Slack (WPWS): NA

Total Pulse Width Negative Slack (TPWS): NA

Number of Failing Endpoints: NA

Total Number of Endpoints: NA

There are no user specified timing constraints.