Digital System Design and Implementation

Lab #2

姓名:黃鉦淳 學號:108303013

a. Verilog codes

```
module Main(Enable_G0, Enable_G1, SevenSeg_G0, SevenSeg_G1, Switch_PIN, LED_PIN, clk_100MHz, Reset);
input clk_100MHz, Reset;
input [7:0]Switch_PIN;
output [3:0]Enable_G0, Enable_G1;
output [7:0]SevenSeg_G0, SevenSeg_G1;
output [15:0]LED_PIN;

reg [2:0]Seg7_idx;
reg [3:0]Seg7_num;
reg [15:0]LED_num;

wire Speed, Keep, clk;
wire [3:0]Pattern;
wire [1:0]Times;

reg [2:0]group;
reg [1:0]counter_times;
reg [7:0]segment_value;

SevenSegment Seg7x8(Seg7_idx, Seg7_num, Enable_G0, Enable_G1, SevenSeg_G0, SevenSeg_G1, clk_100MHz, Reset);
Switch_PIN Switch_PIN8(Switch_PIN, Speed, Pattern, Keep, Times, clk_100MHz, Reset);
LED LED16(LED_PIN, LED_num);
CLK_DIV CLK_Div(Speed, clk, clk_100MHz, Reset);
```

```
// LED
always @(posedge clk or negedge Reset)
begin
	if(!Reset)
	begin
		group = 1;
		counter_times = 0;
		LED_num = 0;
	end
	else
	begin
		if(group == 5)
		begin
			group = 1;
			counter_times = counter_times + 1;
		end

		if(counter_times<{Times[1],Times[0]})
		begin
			if(group == 1)
				LED_num = 0;

			if(Keep)
				LED_num = LED_num | (Pattern<<(4*(group-1)));
			else
				LED_num = Pattern<<(4*(group-1));

			group = group + 1;
		end
		else
			LED_num = 0;
	end
end

// Hardware Program
reg [15:0]counter_Seg7;
// Simulation Program
// reg counter_Seg7;
// Seven Segment
always @(posedge clk_100MHz or negedge Reset)
begin
    if(!Reset)
    begin
        counter_Seg7 = 0;
        Seg7_idx = 0; Seg7_num = 0;
    end
    else
    begin
        if(counter_times<{Times[1],Times[0]})
        begin
            if(counter_Seg7==0)
            begin
                Seg7_idx = 0; Seg7_num = {Pattern[0], Pattern[1], Pattern[2],
Pattern[3]};
            end
            // Hardware Program
            else if(counter_Seg7[15]==1)
            // Simulation Program
            // else if(counter_Seg7==1)
            begin
                Seg7_idx = 1; Seg7_num = (group-1);
            end

            segment_value = ({Pattern[0], Pattern[1], Pattern[2], Pattern[3]}) << 4
| (group-1);
        end
        else
        begin
            if(counter_Seg7==0)
            begin
                Seg7_idx = 0; Seg7_num = 0;
            end
            // Hardware Program
            else if(counter_Seg7[15]==1)
            // Simulation Program
            // else if(counter_Seg7==1)
            begin
                Seg7_idx = 1; Seg7_num = 0;
            end
            segment_value = 0;
        end

        counter_Seg7 = counter_Seg7 + 1;
    end
end
endmodule

module SevenSegment(Seg7_idx, Seg7_num, Enable_G0, Enable_G1, SevenSeg_G0,
SevenSeg_G1, clk_100MHz, Reset);
input clk_100MHz, Reset;
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 @(posedge clk_100MHz or negedge Reset)
begin
    if(!Reset)
    begin
        Enable_G0 = 0; Enable_G1 = 0;
        SevenSeg_G0 = 0; SevenSeg_G1 = 0;
    end
    else
    begin
        Enable_G0 = 0; Enable_G1 = 0;
        SevenSeg_G0 = 0; SevenSeg_G1 = 0;

        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;
```

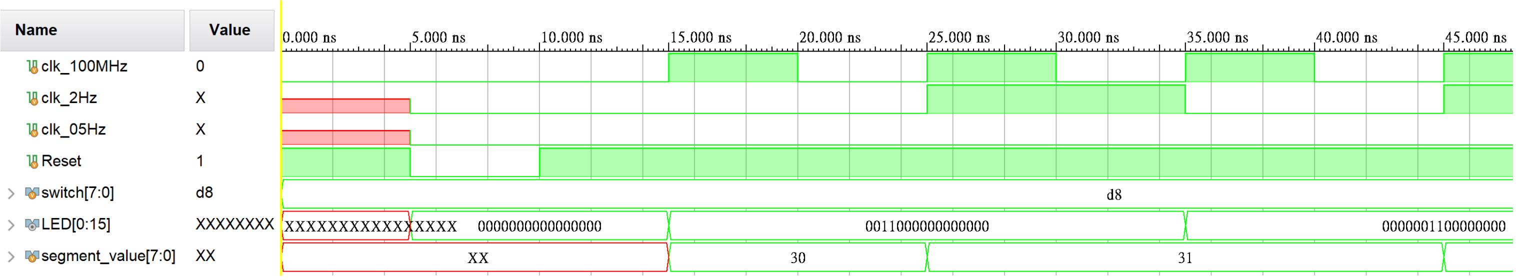
```
            4'h4: SevenSeg = 8'b0110_0110;
            4'h5: SevenSeg = 8'b0110_1101;
            4'h6: SevenSeg = 8'b0111_1101;
            4'h7: SevenSeg = 8'b0000_0111;
            4'h8: SevenSeg = 8'b0111_1111;
            4'h9: SevenSeg = 8'b0110_1111;
            4'hA: SevenSeg = 8'b0101_1111;
            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

            if(Seg7_idx < 4)
            begin
                Enable_G0 = (1 << (Seg7_idx-0));
                SevenSeg_G0 = SevenSeg;
            end
            else
            begin
                Enable_G1 = (1 << (Seg7_idx-4));
                SevenSeg_G1 = SevenSeg;
            end
        end
end
endmodule

module Switch_PIN(Switch_PIN, Speed, Pattern, Keep, Times, clk_100MHz, Reset);
input clk_100MHz, Reset;
input [7:0]Switch_PIN;
output reg Speed, Keep;
output reg [1:0] Times;
output reg [3:0] Pattern;

always @(posedge clk_100MHz or negedge Reset)
begin
    if(!Reset)
    begin
        Speed = Switch_PIN[0];
        Pattern = Switch_PIN[4:1];
        Keep = Switch_PIN[5];
        Times = {Switch_PIN[6], Switch_PIN[7]};
    end
end
endmodule

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

module CLK_DIV(Speed, clk, clk_100MHz, Reset);
input Speed, clk_100MHz, Reset;
output reg clk;
reg clk_05Hz, clk_2Hz;

// Hardware Program
reg [27:0] counter;

// Simulation Program
// reg [2:0] counter_05Hz;
// reg [1:0] counter_2Hz;
// reg [2:0] counter;

always @(posedge clk_100MHz or negedge Reset)
begin
    if(!Reset)
    begin
        counter = 0;
        clk = 0;
        // Simulation Program
        // clk_05Hz = 0;
        // clk_2Hz = 0;
        // counter_05Hz = 0;
        // counter_2Hz = 0;
    end
    else
    begin
        counter = counter + 1;

        // Hardware Program
        if((Speed && counter[27]==1) || (!Speed && counter[25]==1))
        begin
            clk = ~clk;
            counter = 0;
        end

        // Simulation Program
        // if((Speed && counter[2]==1) || (!Speed && counter[0]==1))
        // begin
        //      clk = ~clk;
        //      counter = 0;
        // end
        // if(counter_05Hz[2] == 1)
        // begin
        //      clk_05Hz = ~clk_05Hz;
        //      counter_05Hz = 0;
        // end

        // if(counter_2Hz[0] == 1)
        // begin
        //      clk_2Hz = ~clk_2Hz;
        //      counter_2Hz = 0;
        // end
        // counter_05Hz = counter_05Hz + 1;
        // counter_2Hz = counter_2Hz + 1;
    end
end
endmodule
```

b. Test bench

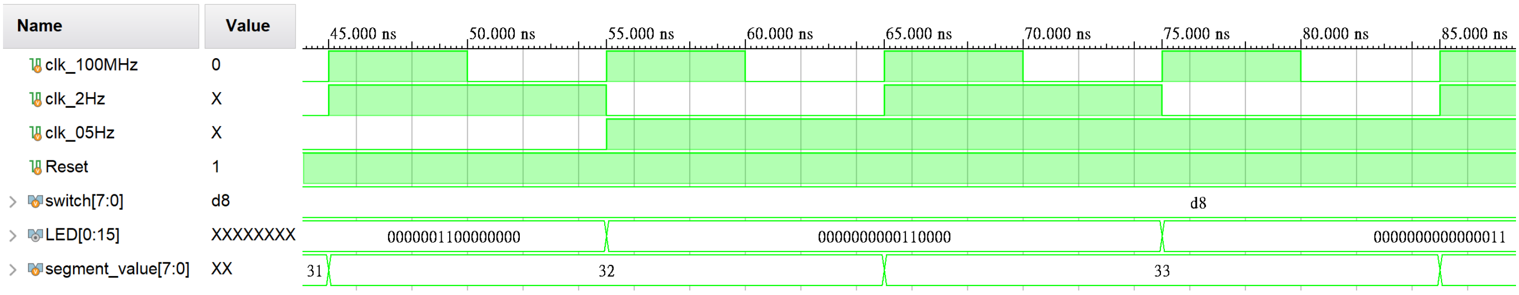
```
`timescale 1ns / 1ps
```

```
module Main_tb();
reg clk_100MHz, Reset;
reg [7:0]switch;

wire [3:0]Enable_G0, Enable_G1;
wire [7:0]SevenSeg_G0, SevenSeg_G1;
wire [15:0]LED;

Main main(Enable_G0, Enable_G1, SevenSeg_G0, SevenSeg_G1, switch, LED,
clk_100MHz, Reset);

initial begin
	// 108303013 | Pattern = 3
	// (a) Keep = 0, Times = 3
	// (b) Keep = 1, Times = 2
	// switch = {Times[7:6], Keep[5], Pattern[4:1], Speed[0]}
	clk_100MHz = 0;
	switch = 0;
	// (a)
	Reset = 1;
	switch = {2'b11, 1'd0, 4'b1100, 1'd0}; #5;
	Reset = 0; #5;
	Reset = 1;
	repeat(48+4) begin
		#5; clk_100MHz = ~clk_100MHz; // 100MHz
	end
	// (b)
	Reset = 1;
	switch = {2'b01, 1'd1, 4'b1100, 1'd0}; #5;
	Reset = 0; #5;
	Reset = 1;
	repeat(32+4) begin
		#5; clk_100MHz = ~clk_100MHz; // 100MHz
	end
	$finish;
end
endmodule
```

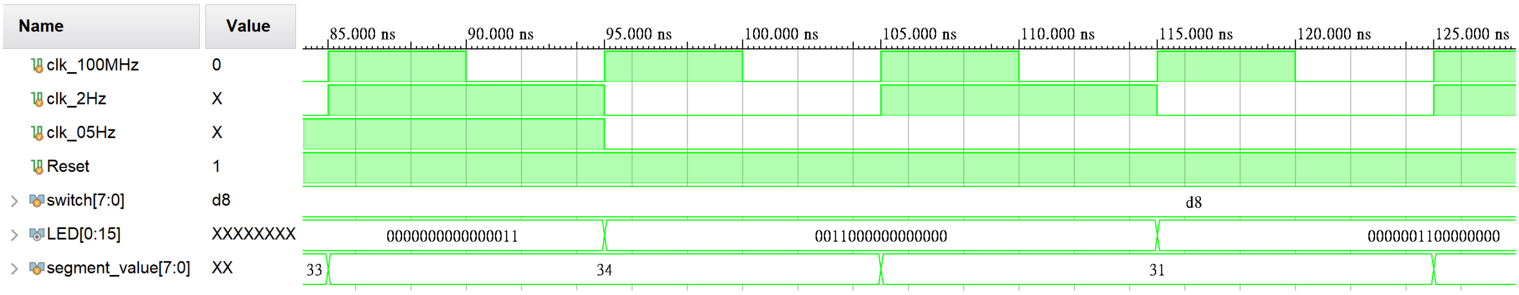
c. Behavior simulation results



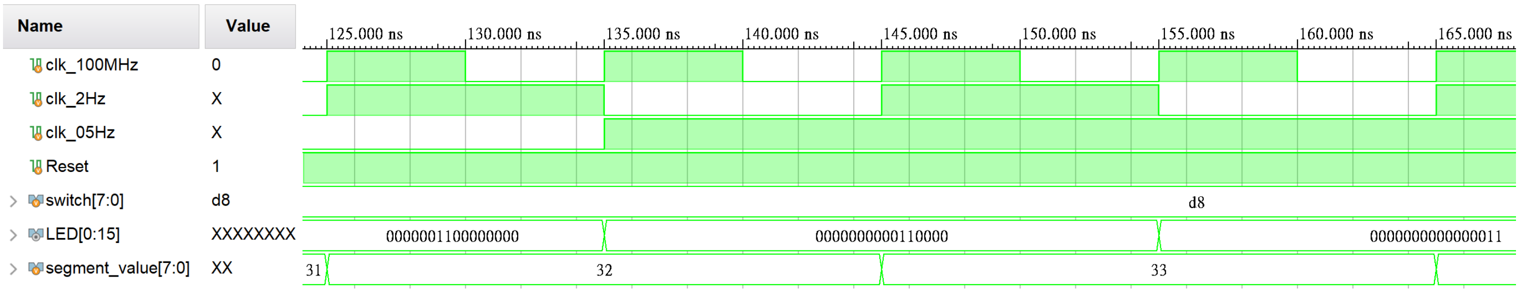
Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX



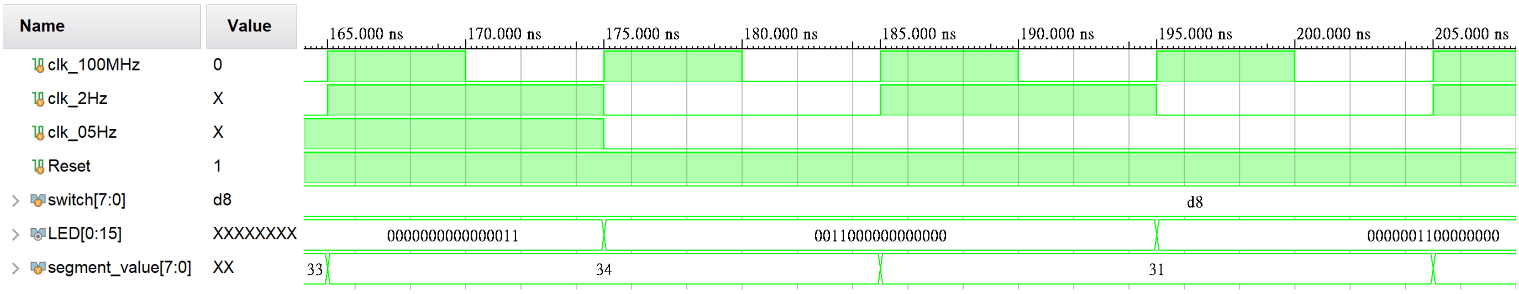
Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX



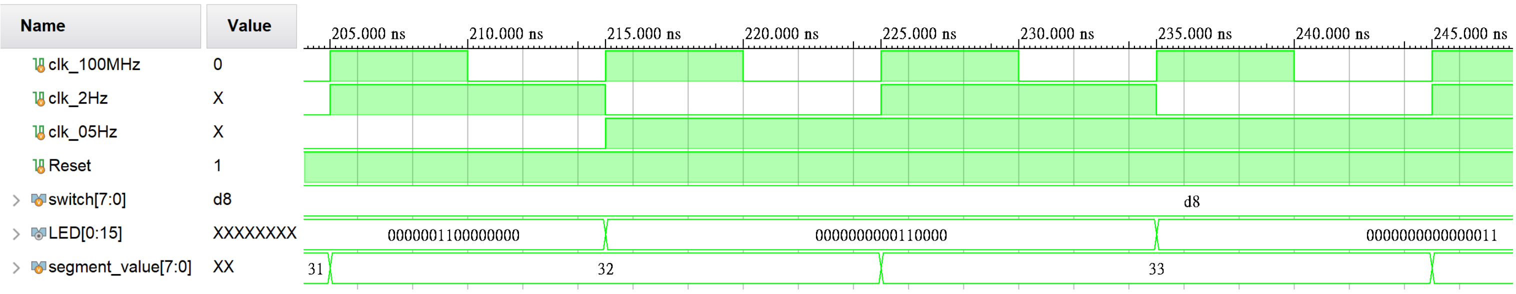
Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX



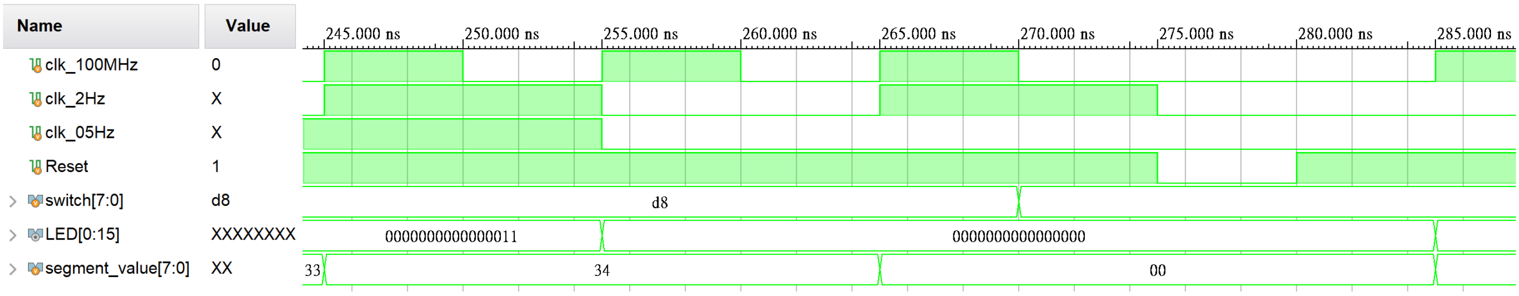
Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX



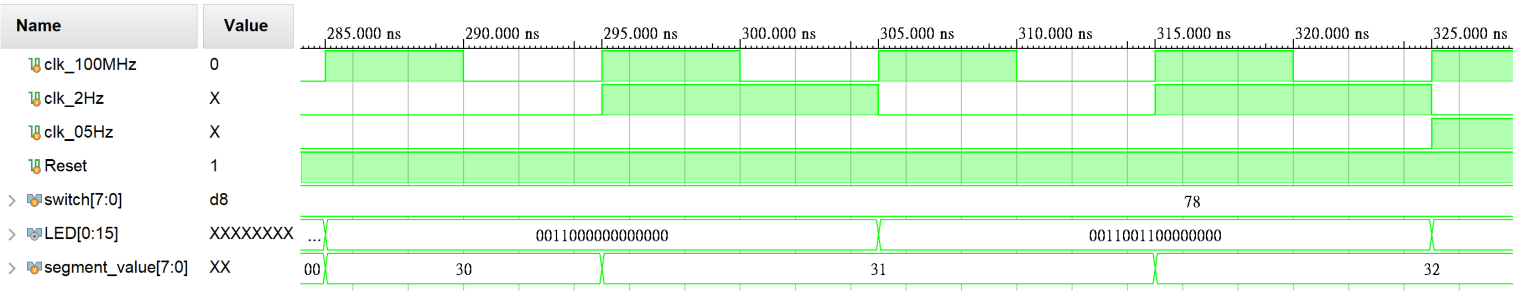
Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX



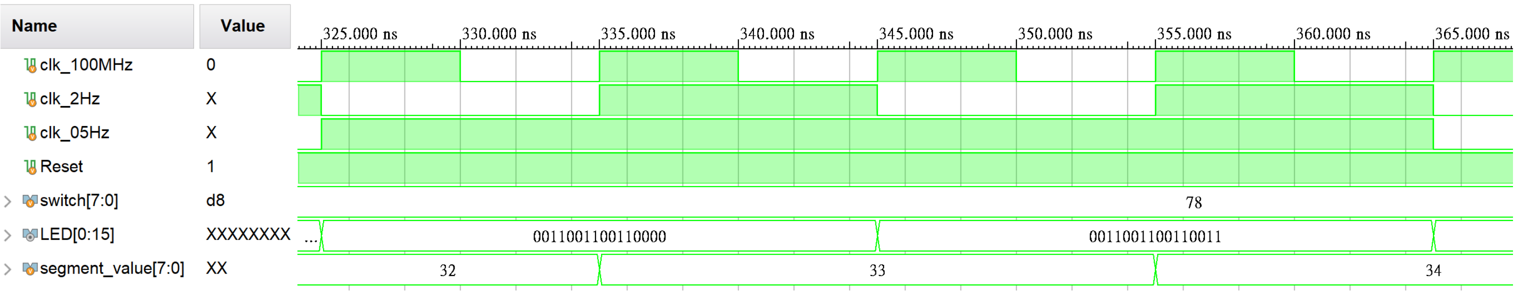
Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX



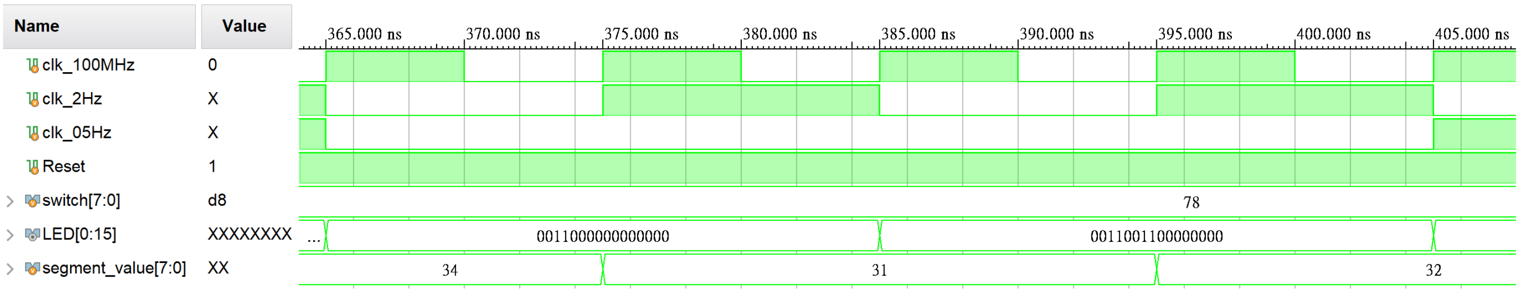
Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX



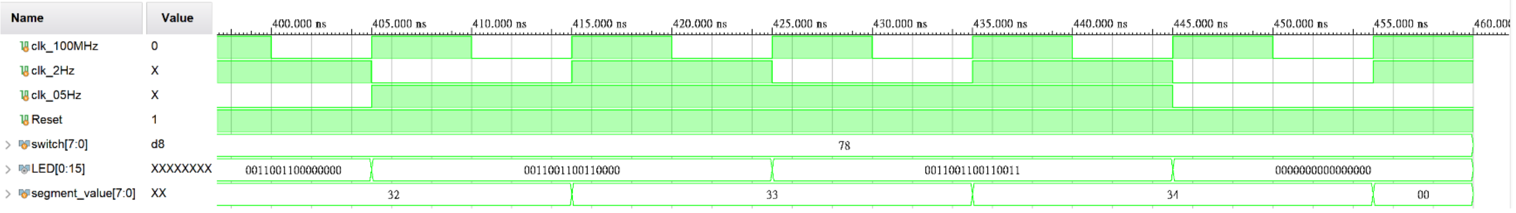
Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX



Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX



Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX



Name	Value
clk_100MHz	0
clk_2Hz	X
clk_05Hz	X
Reset	1
switch[7:0]	d8
LED[0:15]	XXXXXXXX
segment_value[7:0]	XX

變數解釋

clk_100MHz為FPGA上震盪器的時脈100MHz

clk_2Hz為100MHz除頻成2Hz的頻率，用於顯示LED的切換頻率

clk_05Hz為100MHz除頻成0.5Hz的頻率，用於顯示LED的切換頻率

switch為large switch輸入的結果

LED為顯示至FPGA上的結果

segment_value為兩顆七節管顯示的數字

Speed = switch[0]
Pattern = switch[4:1]
Keep = switch[5]
Times = switch[7:6]

Setting (a)

當switch = 0xd8(0b11011000)時，Times=3(0b11), Keep=0(0b0), Pattern=3(0b1100), Speed=0(0b0)。

我的學號為奇數，Times=3則LED顯示共三輪(由左至右)、Keep=0則LED分group依序顯示pattern，Speed=0則LED以2Hz切換。

七節管一顆燈會顯示pattern、另一顆燈會顯示group。

因此，可預期以下結果，其結果完全符合上圖(15.00~270.00 [ns])。

LED	SevenSegment
0011_0000_0000_0000	30

0000_0011_0000_0000	31
0000_0000_0011_0000	32
0000_0000_0000_0011	33
0011_0000_0000_0000	30
0000_0011_0000_0000	31
0000_0000_0011_0000	32
0000_0000_0000_0011	33
0011_0000_0000_0000	30
0000_0011_0000_0000	31
0000_0000_0011_0000	32
0000_0000_0000_0011	33

Setting (b)

當switch = 0x78(0b01111000)時，Times=2(0b01), Keep=1(0b1), Pattern=3(0b1100), Speed=0(0b0)。

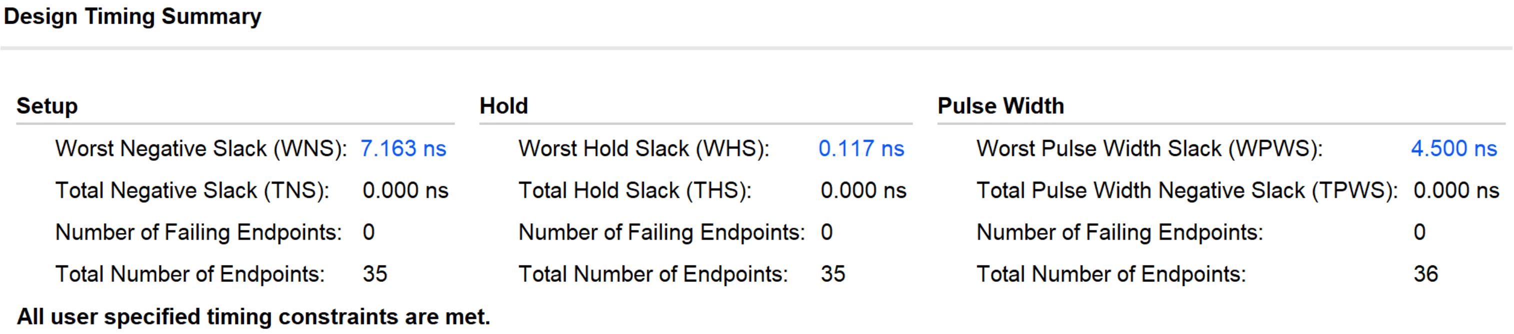
我的學號為奇數，Times=2則LED顯示共兩輪(由左至右)、Keep=1則LED分group依序顯示pattern同一輪當中不會消失，Speed=0則LED以2Hz切換。

七節管一顆燈會顯示pattern、另一顆燈會顯示group。

因此，可預期以下結果，其結果完全符合上圖(285.00~455.00 [ns])。

LED	SevenSegment
0011_0000_0000_0000	30
0011_0011_0000_0000	31
0011_0011_0011_0000	32
0011_0011_0011_0011	33
0011_0000_0000_0000	30
0011_0011_0000_0000	31
0011_0011_0011_0000	32
0011_0011_0011_0011	33

d. Synthesis timing report



Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	7.163 ns	Worst Hold Slack (WHS):	0.117 ns	Worst Pulse Width Slack (WPWS):	4.500 ns
Total Negative Slack (TNS):	0.000 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	0	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	35	Total Number of Endpoints:	35	Total Number of Endpoints:	36

All user specified timing constraints are met.