Course Code - CS223

Course Name - Digital Design

Section – 001

Lab Number – 05

Name Surname – Moin Khan

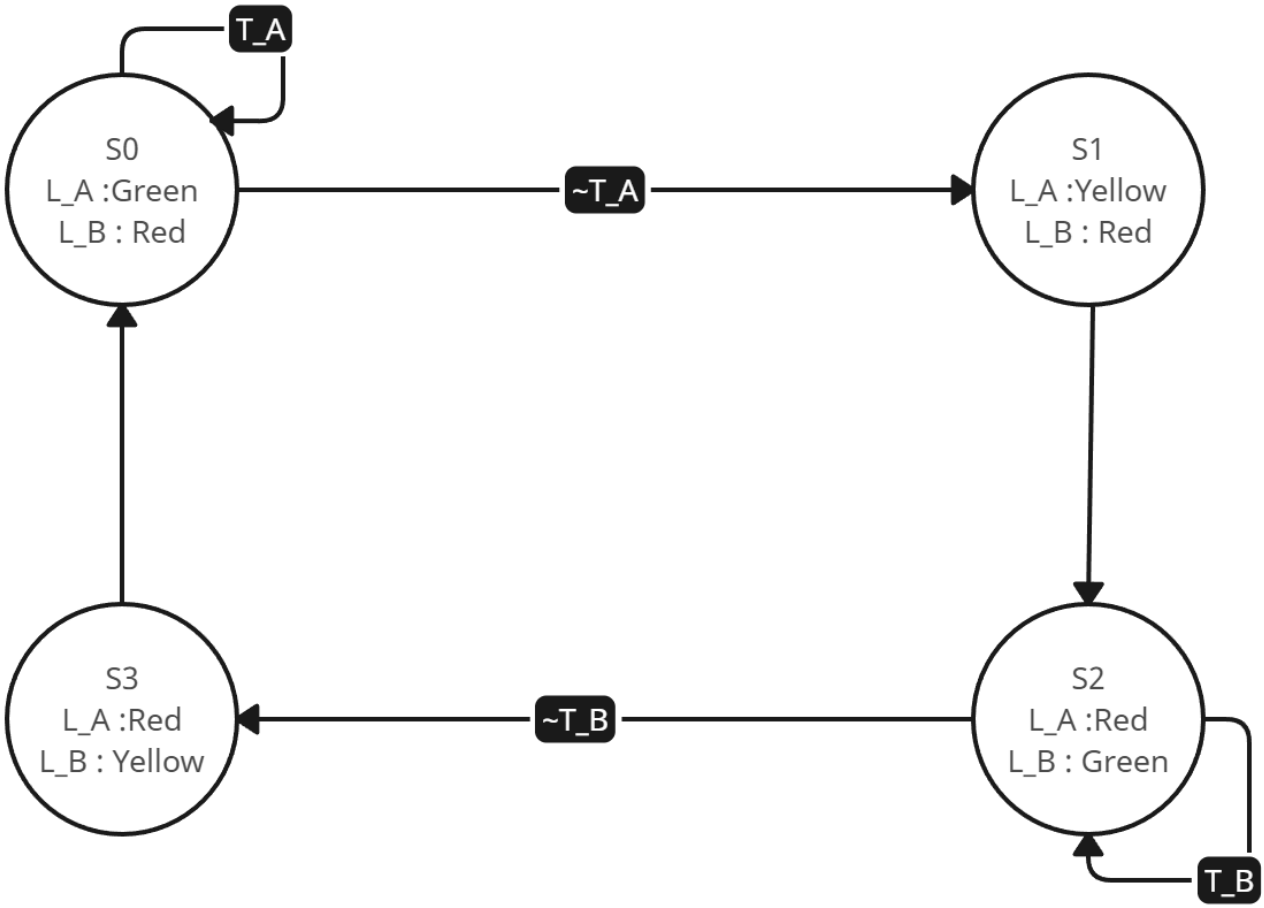
Student Id – 22101287

Date – 27/11/2023

1. Number of Flip-Flops

2 flip-flops are required for this FSM. Since we will have 4 states, we will have 2-bits (S_1 and S_0) to represent those states. To build a 2-bit register, 2 flip-flops will be needed.

2. State Transition Diagram



Current State S	Inputs T_A	T_B	Next State S'
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0

State Transition Table

State	Encoding
S0	00
S1	01
S2	10
S3	11

State Encoding

Current State S_1	S_0	Inputs T_A	T_B	Current State S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

Encoded State Transition Table

2.2 Next State Equation

$$S'_1 = S_1 \oplus S_0$$

$$S'_0 = \overline{S_1}\,\overline{S_0}\,\overline{T_A} + S_1\overline{S_0}\,\overline{T_B}$$

State	L_A	L_B
S0	Green	Red
S1	Yellow	Red
S2	Red	Green
S3	Red	Yellow

Sate Output Table

Output	Encoding
Green	00
Yellow	01
Red	10

Output Encoding

Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

Output Table with Encoding

2.3 Output Logic Equation

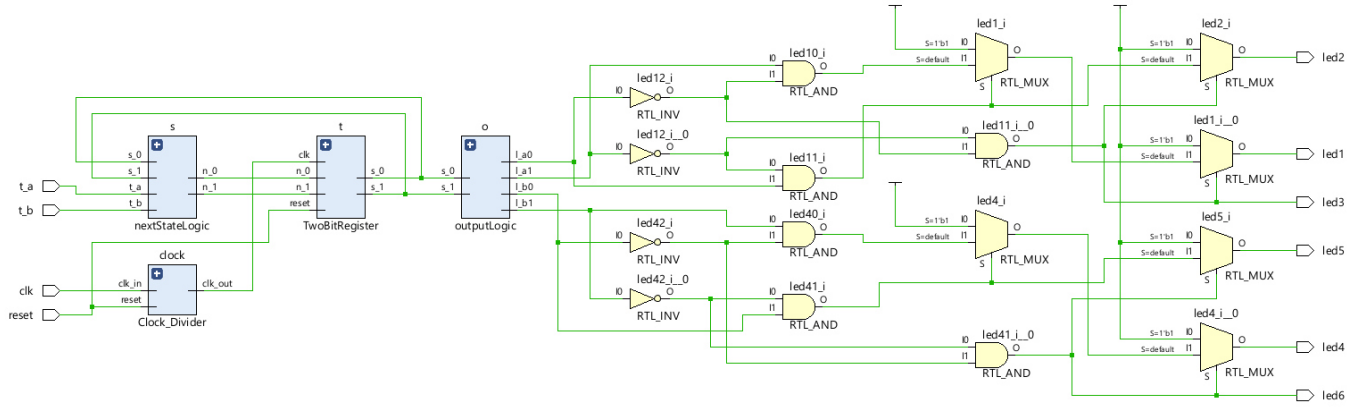
$$L_{A1} = S_1$$

$$L_{A0} = \overline{S_1} S_0$$

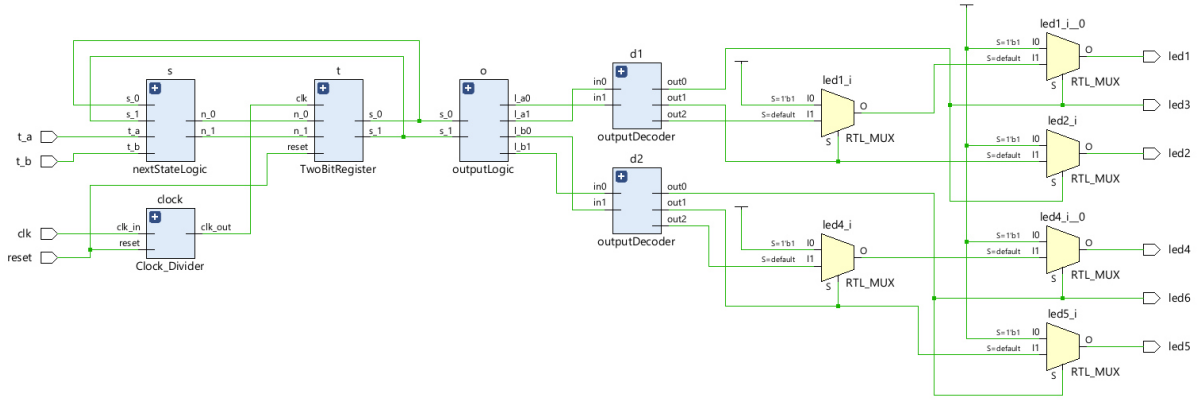
$$L_{B1} = \overline{S_1}$$

$$L_{B1} = S_1 S_0$$

2.4 Schematics for Finite State Machine



3. Redesign Using Decoder



Code:

```
`timescale 1ns / 1ps


module Clock_Divider(
  input wire clk_in,
```

```
  input wire reset,
  output reg clk_out
);

  reg [29:0] counter = 0;

  always @(posedge clk_in or posedge reset) begin
    if (reset) begin
      counter <= 0;
      clk_out <= 0;
    end else begin
      if (counter == 149999999) begin  // Assuming a 100 MHz original clock  299999999
        counter <= 0;
        clk_out <= ~clk_out;
      end else begin
        counter <= counter + 1;
      end
    end
  end

endmodule
```

```
`timescale 1ns / 1ps


module nextStateLogic(input wire s_1,s_0,t_a,t_b, output
wire n_1,n_0);
assign n_1 = s_1^s_0;
assign n_0 = ((~s_1)&(~s_0)&(~t_a)) |
((s_1)&(~s_0)&(~t_b));
endmodule



`timescale 1ns / 1ps


module outputLogic(input wire s_1,s_0, output wire
l_a1,l_a0,l_b1,l_b0);
assign l_a1 = s_1;
assign l_a0 = (~s_1)&s_0;
assign l_b1 = ~s_1;
assign l_b0 = s_1&s_0;
endmodule



`timescale 1s / 1ps
module TL_Controller(input wire t_a,t_b,clk,reset,output
reg led1,led2,led3,led4,led5,led6);
```

```
wire s1,s0;
wire n1,n0;
reg slowClk;
reg l_a1,l_a0,l_b1,l_b0;
TwoBitRegister t(slowClk,reset,n1,n0,s1,s0);
nextStateLogic s(s1,s0,t_a,t_b,n1,n0);
outputLogic o(s1,s0,l_a1,l_a0,l_b1,l_b0);
Clock_Divider clock(clk,reset,slowClk);
always_comb begin
        if (~l_a1 && ~l_a0) begin
            led1 = 1'b1;
            led2 = 1'b1;
            led3 = 1'b1;
        end
        else if (~l_a1 && l_a0) begin
            led1 = 1'b1;
            led2 = 1'b1;
            led3 = 1'b0;
        end
        else if (l_a1 && ~l_a0) begin
            led1 = 1'b1;
            led2 = 1'b0;
            led3 = 1'b0;
```

```
    end
    else begin
        led1 = 1'b0;
        led2 = 1'b0;
        led3 = 1'b0;
    end
end
always_comb begin
    if (~l_b1 && ~l_b0) begin
        led4 = 1'b1;
        led5 = 1'b1;
        led6 = 1'b1;
    end
    else if (~l_b1 && l_b0) begin
        led4 = 1'b1;
        led5 = 1'b1;
        led6 = 1'b0;
    end
    else if (l_b1 && ~l_b0) begin
        led4 = 1'b1;
        led5 = 1'b0;
        led6 = 1'b0;
    end
```

```
        else begin
            led4 = 1'b0;
            led5 = 1'b0;
            led6 = 1'b0;
        end
    end
endmodule


module outputDecoder (
    input wire in0,
    input wire in1,
    output reg out0,
    output reg out1,
    output reg out2);

    // Decoder logic using an if-else statement
    always_comb begin
        if (in0 == 1'b0 && in1 == 1'b0) begin
            out0 = 1'b1;
            out1 = 1'b0;
            out2 = 1'b0;;
        end
```

```
        else if (in0 == 1'b0 && in1 == 1'b1) begin
            out0 = 1'b0;
            out1 = 1'b1;
            out2 = 1'b0;
        end
        else if (in0 == 1'b1 && in1 == 1'b0) begin
            out0 = 1'b0;
            out1 = 1'b0;
            out2 = 1'b1;
        end
        else begin
            // Handle any other input combinations if needed
            out0 = 1'b0;
            out1 = 1'b0;
            out2 = 1'b0;
        end
    end

endmodule

`timescale 1s / 1ps
module TL_ControllerD(input wire
t_a,t_b,clk,reset,output reg led1,led2,led3,led4,led5,led6);
```

```
wire s1,s0;
wire n1,n0;
reg slowClk;
reg l_a1,l_a0,l_b1,l_b0;
reg La0,La1,La2,Lb0,Lb1,Lb2;
TwoBitRegister t(slowClk,reset,n1,n0,s1,s0);
nextStateLogic s(s1,s0,t_a,t_b,n1,n0);
outputLogic o(s1,s0,l_a1,l_a0,l_b1,l_b0);
Clock_Divider clock(clk,reset,slowClk);
outputDecoder d1(l_a1,l_a0,La0,La1,La2);
outputDecoder d2(l_b1,l_b0,Lb0,Lb1,Lb2);

always_comb begin
        if (La0) begin
            led1 = 1'b1;
            led2 = 1'b1;
            led3 = 1'b1;
        end
        else if (La1) begin
            led1 = 1'b1;
            led2 = 1'b1;
            led3 = 1'b0;
        end
```

```
    else if (La2) begin
        led1 = 1'b1;
        led2 = 1'b0;
        led3 = 1'b0;
    end
    else begin
        led1 = 1'b0;
        led2 = 1'b0;
        led3 = 1'b0;
    end
end
always_comb begin
    if (Lb0) begin
        led4 = 1'b1;
        led5 = 1'b1;
        led6 = 1'b1;
    end
    else if (Lb1) begin
        led4 = 1'b1;
        led5 = 1'b1;
        led6 = 1'b0;
    end
    else if (Lb2) begin
```

```
            led4 = 1'b1;
            led5 = 1'b0;
            led6 = 1'b0;
        end
        else begin
            led4 = 1'b0;
            led5 = 1'b0;
            led6 = 1'b0;
        end
    end
endmodule
```