

Todo:

Replace code with 200000 for counter

Replace testbench for part 4 and verilog design code

Create circuit designs for each part

Explanations

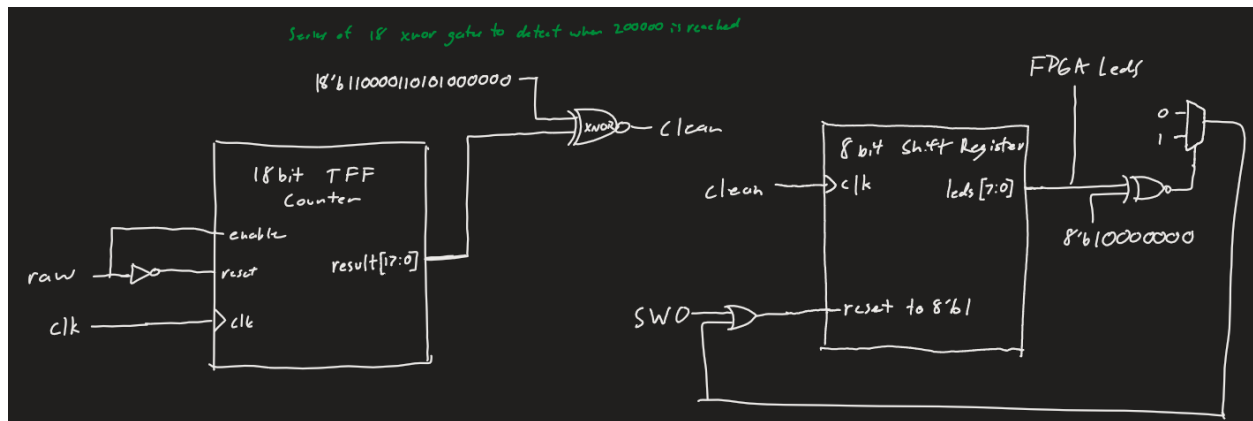
Make note of change counter for debouncing in simulation waveform

Mark Mitri and Ryan Forelli
ECE 128

9/27/2021
Lab 5

Part 1: Debouncer using Behavioral modeling

Design:



Code:

```
`timescale 1ns / 1ps

module debouncer ( input raw,
    input clk,
    output reg clean,
    output reg [17:0] counter);
    wire TC;
    always @ (posedge clk) begin
        if(~raw)
            counter <= 18'b000000000000000000;
        else
            counter <= counter + 18'b000000000000000001;
    end
```

```

assign TC = (counter==18'b110000110101000000);
always @ (posedge clk) begin
if(~raw)
    clean <= 1'b0;
else if (TC)
    clean <= 1'b1;
end
endmodule

module debouncing (input btn1, reset, clk, output reg[7:0] leds,
output clean, output [2:0] counter);

    debouncer d1(btn1, clk, clean, counter);

    always @ (posedge clean) begin
        if(!reset)
            begin
                if(leds == 8'b10000000)
                    leds = 8'b00000001;
                else
                    leds = leds << 8'b00000001;
            end
        else
            leds = 8'b00000001;
        end
    end
endmodule

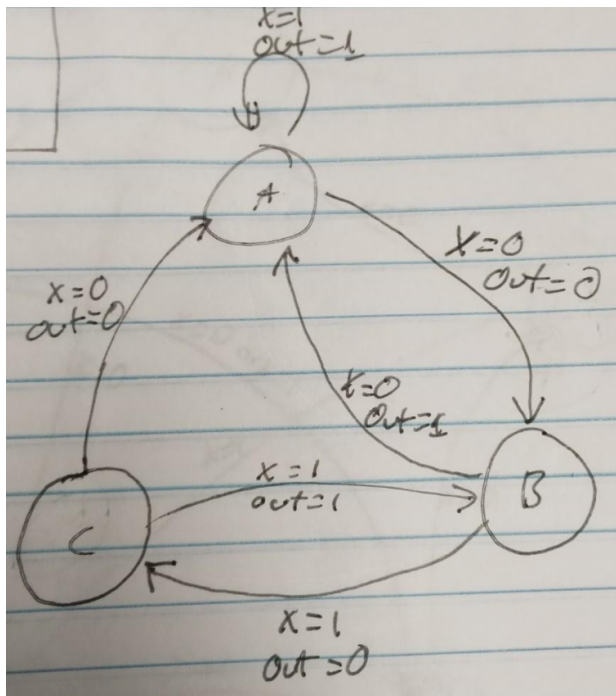
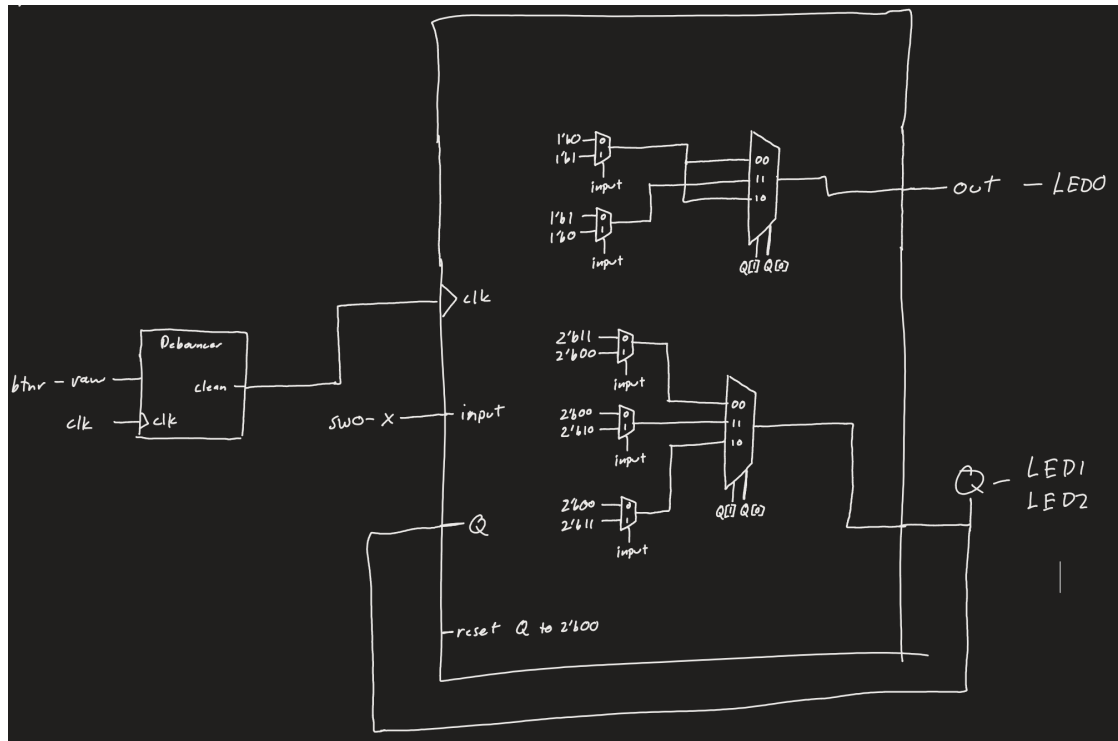
```

Signature:

L-b S Part 1 TA Y.Z

Part 2: Mealy Machine using Behavioral modeling

Design:



Code:

```
`timescale 1ns / 1ps

module debouncer ( input raw,
    input clk,
    output reg clean);
    reg [17:0] counter;
    wire TC;
    always @ (posedge clk) begin
        if(~raw)
            counter <= 18'b000000000000000000;
        else
            counter <= counter + 18'b000000000000000001;
    end

    assign TC = (counter==18'b110000110101000000);
    always @ (posedge clk) begin
        if(~raw)
            clean <= 1'b0;
        else if (TC)
            clean <= 1'b1;
        end
    endmodule

module fsm(input x, btnr, clk, output reg out, output reg[1:0] Q);
    wire debounced_btnr;
    debouncer debounced_button1(btnr, clk, debounced_btnr);

    always @ (posedge debounced_btnr) begin
        case(Q)
            3'b00: begin //If state is A
                if (!x) begin
                    Q <= 2'b11; //Set state to B
                    out = 0;
                end
            else begin
                Q <= 2'b00; //Set state to A
                out = 1;
            end
        endcase
    end
endmodule
```

```

        end
    end
    3'b11: begin //If state is B
        if (!x) begin
            Q <= 2'b00; //Set state to A
            out=1;
        end
        else begin
            Q <= 2'b10; //Set state to C
            out=0;
        end
    end
    3'b10: begin //If state is C
        if (!x) begin
            Q <= 2'b00; //Set state to A
            out = 0;
        end
        else begin
            Q <= 2'b11; //Set state to B
            out = 1;
        end
    end
    default: begin
        Q <= 2'b00;
    end
endcase
end
endmodule

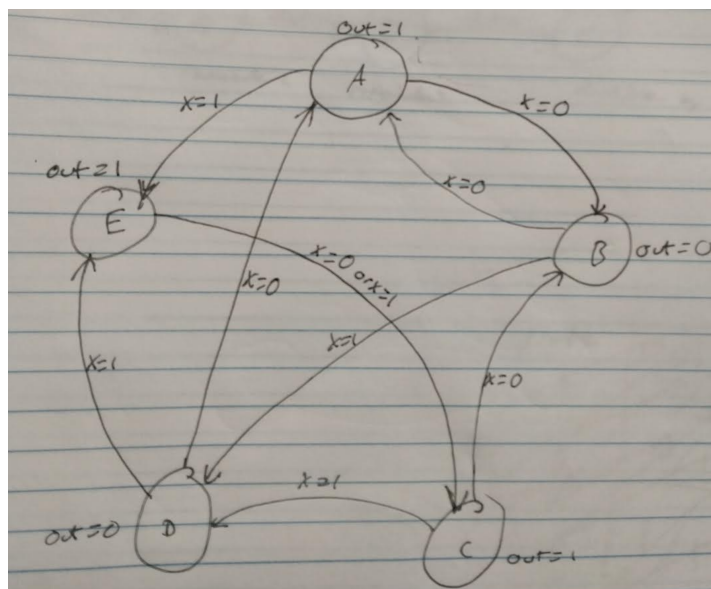
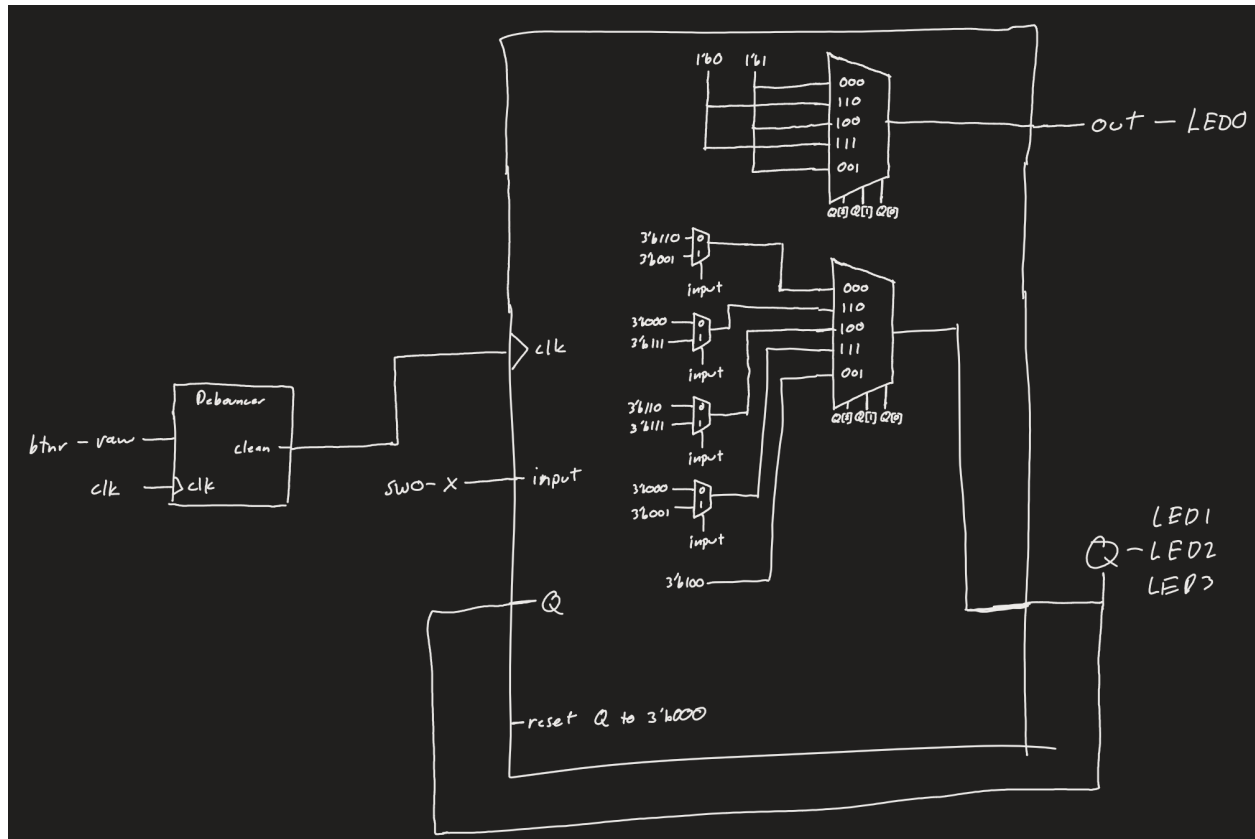
```

Signature:

Lab 5 Part 2 TA Y.Z

Part 3: Moore Machine using Behavioral modeling

Design:



Code:

```

`timescale 1ns / 1ps
module debouncer ( input raw,
    input clk,
    output reg clean);
    reg [17:0] counter;
    wire TC;
    always @ (posedge clk) begin
        if(~raw)
            counter <= 18'b000000000000000000;
        else
            counter <= counter + 18'b000000000000000001;
    end
    assign TC = (counter==18'b110000110101000000);
    always @ (posedge clk) begin
        if(~raw)
            clean <= 1'b0;
        else if (TC)
            clean <= 1'b1;
    end
endmodule

module fsm(input x, btnr, clk, output out, output reg[2:0] Q);
    wire debounced_btnr;
    debouncer debounced_button1(btnr, clk, debounced_btnr);

    assign out = (~Q[0]&~Q[1]&~Q[2])| (Q[0]&~Q[1]&~Q[2]) |
    (~Q[0]&~Q[1]&Q[2]);

    always @ (posedge debounced_btnr) begin
        case(Q)
            3'b000: begin //If state is A
                if (!x) begin
                    Q <= 3'b110; //Set state to B
                end
            end
            else begin
                Q <= 3'b001; //Set state to E
            end
        end
        3'b110: begin //If state is B

```

```

        if (!x) begin
            Q <= 3'b000; //Set state to A
        end
        else begin
            Q <= 3'b111; //Set state to D
        end
    end
    3'b100: begin //If state is C
        if (!x) begin
            Q <= 3'b110; //Set state to B
        end
        else begin
            Q <= 3'b111; //Set state to D
        end
    end
    3'b111: begin //If state is D
        if (!x) begin
            Q <= 3'b000; //Set state to A
        end
        else begin
            Q <= 3'b001; //Set state to E
        end
    end
    3'b001: begin //If state is E
        Q <= 3'b100; //Set state to C
    end
    default: begin
        Q <= 3'b000;
    end
endcase
end
endmodule

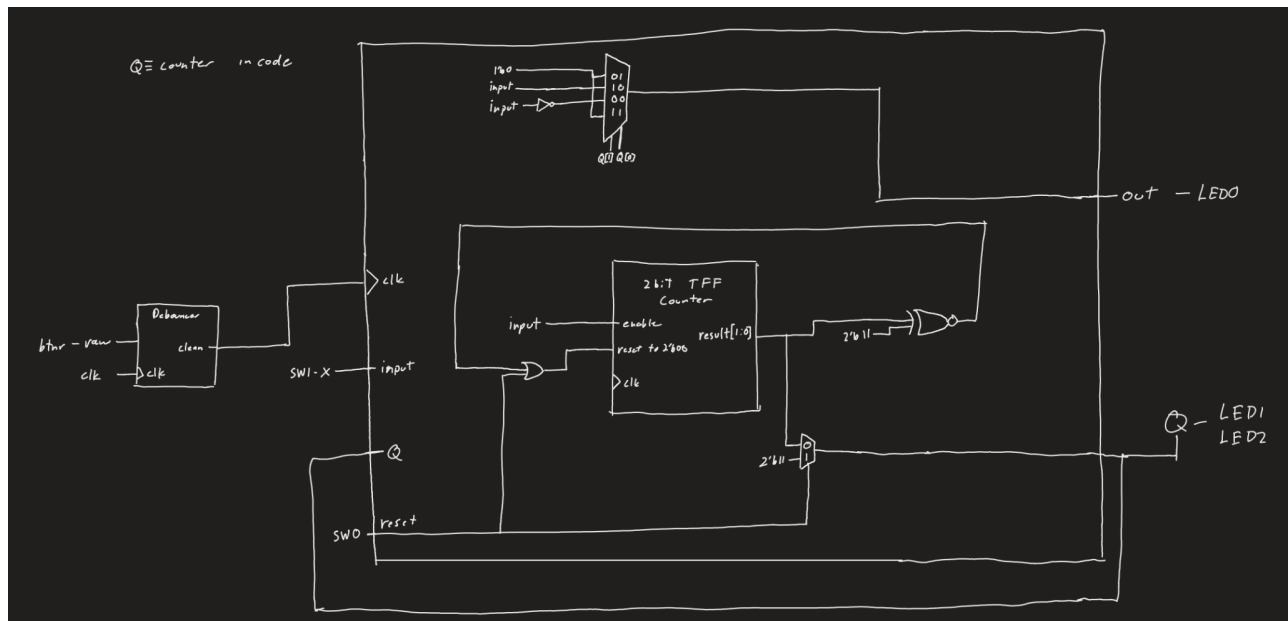
```

Signature:

Lab 5 Part 3 TA Y.Z

Part 4: Mealy Sequence detector using Behavioral modeling

Design:



Code:

```
`timescale 1ns / 1ps

module debouncer ( input raw,
    input clk,
    output reg clean);
    reg [17:0] counter;
    wire TC;
    always @ (posedge clk) begin
        if(~raw)
            counter <= 18'b000000000000000000;
        else
            counter <= (counter + 18'b000000000000000001);
    end
    assign TC = (counter==18'b110000110101000000);
    always @ (posedge clk) begin
        if(~raw)
            clean <= 1'b0;
```

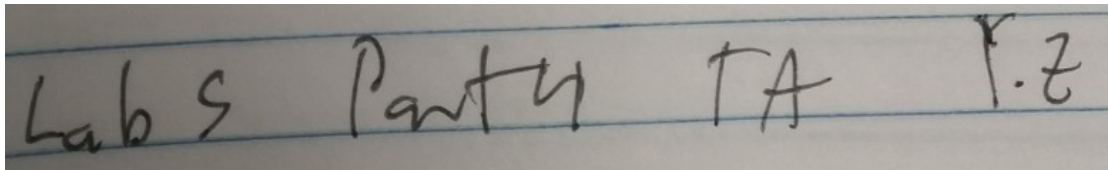
```

    else if (TC)
        clean <= 1'b1;
    end
endmodule

module mealy_sequence_detector(input x, btnr, reset, clk, output reg
out, output reg[1:0] counter, output clean);
    debouncer debounced_button1(btnr, clk, clean);
    always @ (posedge clean) begin
        if(!reset) begin
            if(counter == 2'b11) counter = 2'b00;
            if(x) begin
                counter = counter + 2'b01;
                if(counter == 2'b11) begin
                    out = 1'b1;
                    counter = 2'b00;
                end
            end
        end
        else begin
            out = 1'b0;
            counter = 2'b11;
        end
    end
end
endmodule

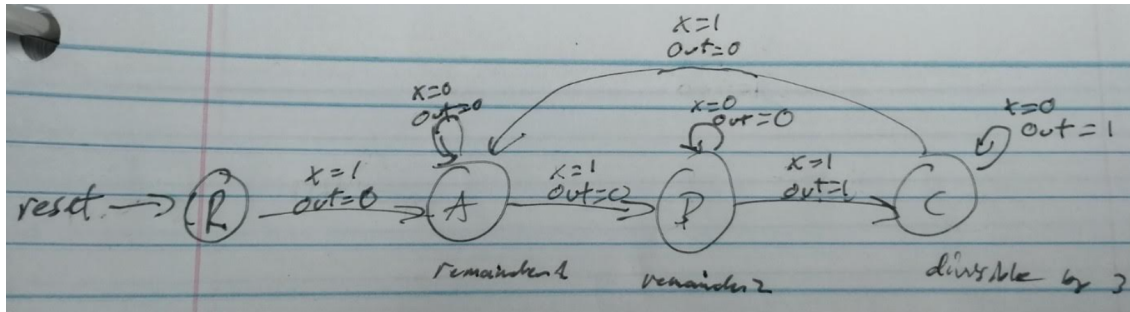
```

Signature:



Lab 9 Part 4 TA R.Z.

State Transition Diagram:



State Assignment:

| state assignment | | |
|------------------|------|------|
| state | Q[1] | Q[0] |
| R | 1 | 1 |
| A | 0 | 1 |
| B | 1 | 0 |
| C | 0 | 0 |

Testbench:

Note: Counter in design code was changed to 10 instead of 200000 for waveform readability.

```

`timescale 1ns / 1ps

module fsm_moore_tb;
    reg x, btr, reset, clk;
    wire out, clean;
    wire[2:0] state;

    mealy_sequence_detector seq1(x, btr, reset, clk, out, state,
clean);
    always begin
        #1 clk = !clk;
    end

    initial begin

```

```

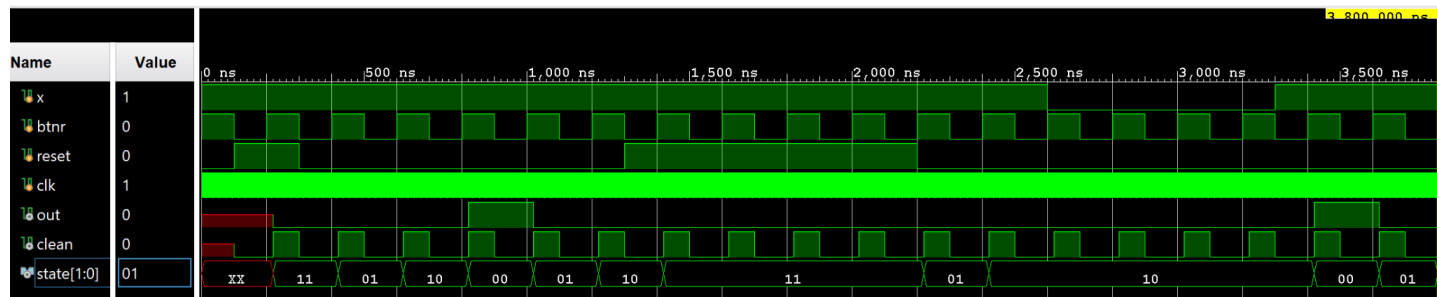
    clk = 0;
    x = 1;
    reset = 0;
    btnr = 1; #100
    reset = 1;
    btnr = 0; #100 btnr = 1; #100
    reset = 0;
    btnr = 0; #100 btnr = 1; #100 btnr = 0; #100 btnr = 1; #100 btnr
= 0; #100 btnr = 1; #100 btnr = 0; #100 btnr = 1; #100 btnr = 0; #100
btnr = 1; #100
    reset = 1; btnr = 0; #100
    btnr = 1; #100 btnr = 0; #100 btnr = 1; #100 btnr = 0; #100 btnr
= 1; #100 btnr = 0; #100 btnr = 1; #100 btnr = 0; #100
    reset = 0;
    btnr = 1; #100 btnr = 0; #100 btnr = 1; #100 btnr = 0; #100
    x = 0;
    btnr = 1; #100 btnr = 0; #100 btnr = 1; #100 btnr = 0; #100 btnr
= 1; #100 btnr = 0; #100 btnr = 1; #100
    x = 1;
    btnr = 0; #100 btnr = 1; #100 btnr = 0; #100 btnr = 1; #100 btnr
= 0; #100
    $finish; //end the simulation
end
endmodule

```

Functionalities to be tested:

1. Reset sets state to 2'b11 and output to 1'b0.
2. Sequence detector counter begins counting up when input is 1'b1.
3. Debouncing counter counts to 10 before propagating the clean signal.
4. When the number of 1's inputted is divisible by 3, output is 1'b1.
5. When input is 1'b0, the fsm maintains its current state.

Simulation waveform:



The simulation waveform behavior is expected. Any time btnr is set high, 10 clock cycles pass (200000 in practice) before the signal is propagated and clean is set high. While reset is high, the state is set to 2'b11 on the next positive clean signal edge at t=220ns. When reset is set low and input x is set high, the state begins changing on each positive clean signal edge. When the number of 1's inputted reaches a number that is divisible by 3 at t=820ns, the state is set to 2'b00 and the output is set high. After the next positive clean signal edge while input is still set high, output is set low as the number of 1's inputted is no longer divisible by 3. When reset is set high again at t=1300ns, the state is set to 2'b11 on the next positive clean signal edge at t=1420ns. After reset is set low again, the input is set low at t=2600ns which results in the state to remain unchanged as no 1's are inputted during this time until the input is set high again at t=3300ns. This behavior is expected.