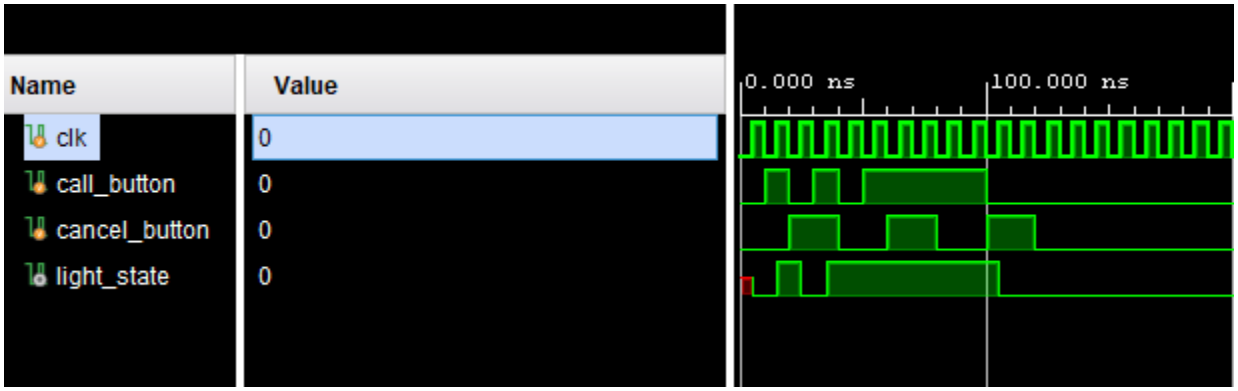
LAB 4 Report

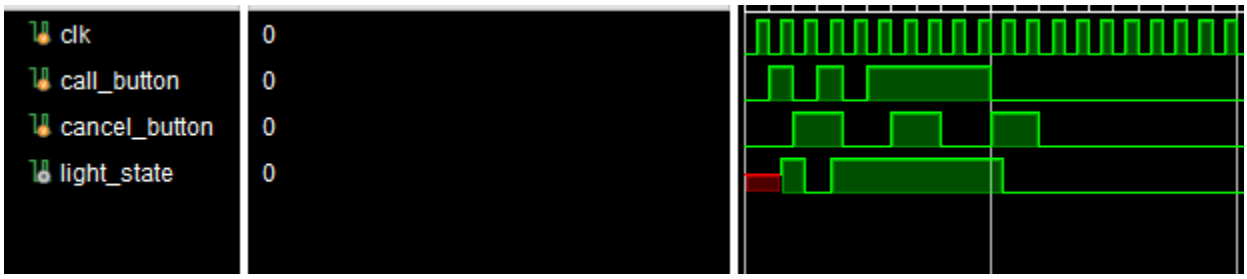
Zach Mullaney zm4822
Joanna Enyioke jee2353

Part 1:

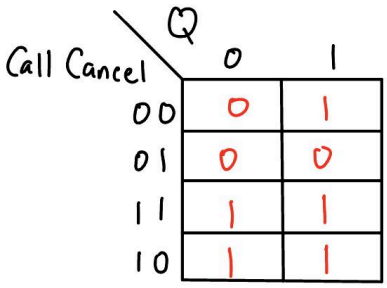
Behavioral waveform:



Dataflow waveform:



K-Map:



$D = Q\,Call'\,Cancel' + Call = Next\_state$

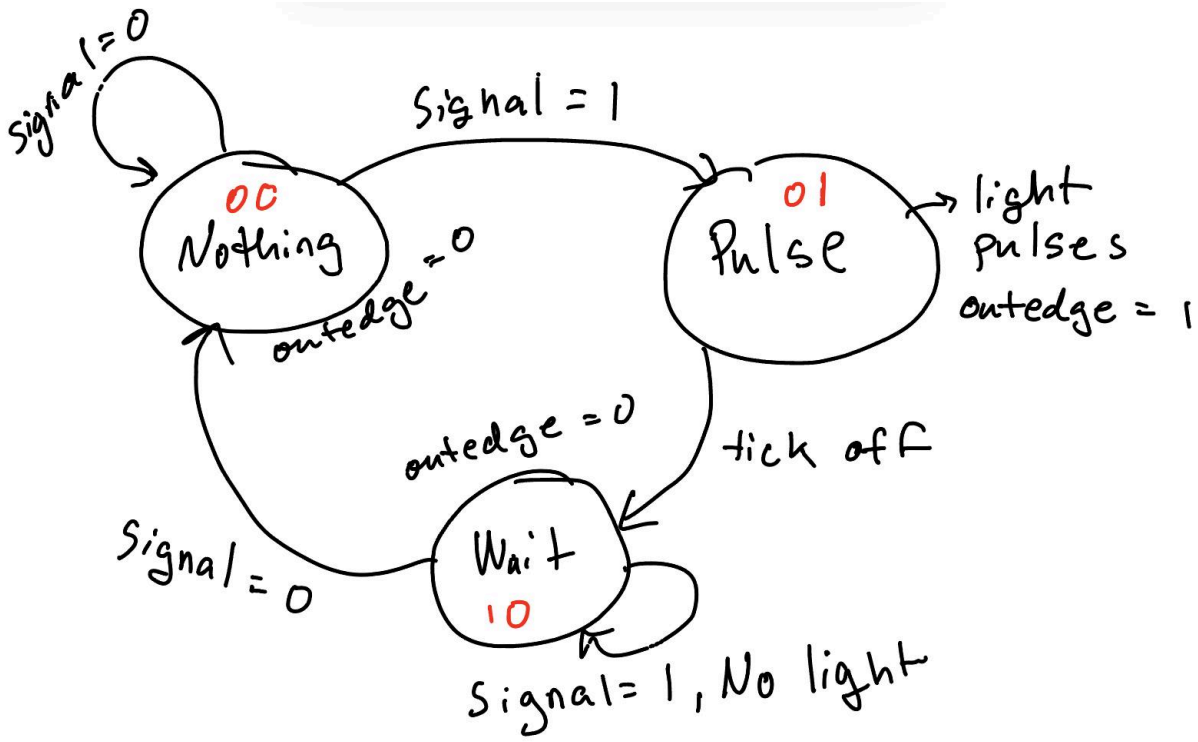
$Next\_State = ((\sim Call\_button)\,\&\,(\sim Cancel\_button)\,\&\,light\_state)) \mid call\_button$

Dataflow Design:

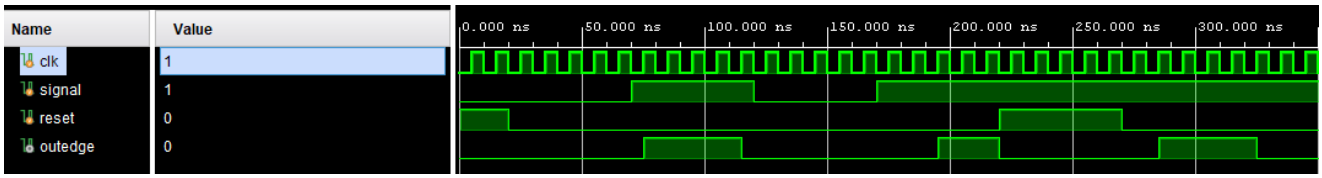
```
28  //// Dataflow
29  wire next_state ;
30
31  // Combinational block
32  assign next_state = ((~call_button)&(~cancel_button)&(light_state)) | (call_button);
33
34  // Sequential block
35  always @( posedge clk ) begin
36      light_state <= next_state ;
37  end
38
39  endmodule
```

Part 2:

State Diagram:



Waveform (used clk directly):



clkdiv.v

```
1   `timescale 1ns / 1ps
2
3   module clkdiv(
4       input clk,
5       input reset,
6       output clk_out
7   );
8
9
10      reg [26:0] COUNT;
11
12          assign clk_out=COUNT[26];
13
14          always @(posedge clk)
15          begin
16          if (reset)
17              COUNT = 0;
18          else
19              COUNT = COUNT + 1;
20          end
21
22  endmodule
```

Rising Edge Design File:

```
`timescale 1ns / 1ps

module rising_edge_det (
    input clk,       // Fast system clock
    input signal,
    input reset,
    output reg outedge
);

    wire slow_clk;
    reg prev_signal; // Stores previous signal state for edge detection

    // Clock divider instance (provides `slow_clk`)
    clkdiv cl(.clk(clk), .reset(reset), .clk_out(slow_clk));

    // Detect rising edge (0 → 1 transition) **on slow clock**
    always @(posedge slow_clk or posedge reset) begin
        if (reset)
            prev_signal <= 0;
        else
            prev_signal <= signal;
    end

    wire signal_rising = (signal == 1'b1) && (prev_signal == 1'b0); // Detect rising edge

    // Generate a 1-cycle pulse on slow_clk rising edge
    always @(posedge slow_clk or posedge reset) begin
        if (reset)
            outedge <= 0;
        else
            outedge <= signal_rising; // `outedge` is high for 1 slow_clk cycle
    end

endmodule
```

Testbench Rising Edge:

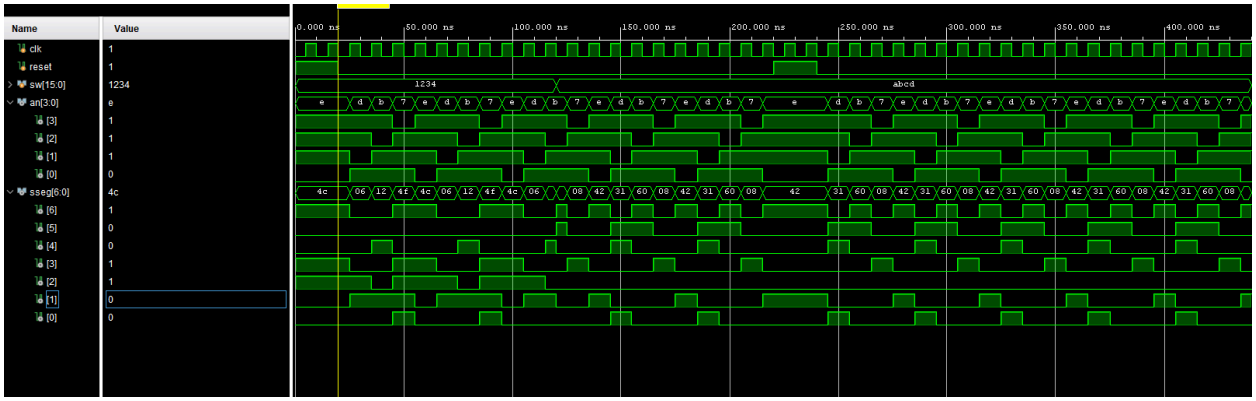
```
1  `timescale 1ns / 1ps
2
3  module tb_rising_edge_det;
4
5      reg clk;
6      reg signal;
7      reg reset;
8      wire outedge;
9
10     rising_edge_det u1 (
11         .clk(clk),
12         .signal(signal),
13         .reset(reset),
14         .outedge(outedge)
15     );
16
17     // Clock Generation
18     always #5 clk = ~clk;
19
20     // Test Sequence
21     initial begin
22         clk = 0;
23         signal = 0;
24         reset = 1;
25
26         #20 reset = 0;  // Release reset
27
28         #50 signal = 1;  // Rising edge
29         #50 signal = 0;  // Falling edge
30         #50 signal = 1;  // Another rising edge
31         #50 reset = 1;   // Reset again
32         #50 reset = 0;   // Release reset
33         #100;            // Wait to observe final behavior
34
35         $finish;
36     end
37
38 endmodule
39
```

Part 2 Constraints:

```
## SWITCH
set_property PACKAGE_PIN V17 [get_ports {signal}]
## Sets the switch to use 3.3V logic
    set_property IOSTANDARD LVCMOS33 [get_ports {signal}]

## Clock signal - Uncomment if needed (will be used in future labs)
set_property PACKAGE_PIN W5 [get_ports {clk}]
    set_property IOSTANDARD LVCMOS33 [get_ports {clk}]
    create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports {clk}]

##Buttons
set_property PACKAGE_PIN U18 [get_ports {reset}]
    set_property IOSTANDARD LVCMOS33 [get_ports {reset}]

## LEDs
## Connects the output c in our gate module to pin U16 (LED0 on-board)
set_property PACKAGE_PIN U16 [get_ports {outedge}]
### Sets the LED to use 3.3V logic
    set_property IOSTANDARD LVCMOS33 [get_ports {outedge}]
```

Part 3:

Waveform (used clk directly):



Design (time_mux_state_machine.v):

```
1  `timescale 1ns / 1ps
2
3  module time_mux_state_machine(
4      input clk,
5      input reset,
6      input [6:0] in0,
7      input [6:0] in1,
8      input [6:0] in2,
9      input [6:0] in3,
10     output reg [3:0] an,
11     output reg [6:0] sseg
12 );
13
14 reg [1:0] state;
15 reg [1:0] next_state;
16 wire clk_slow; // Slow clock signal from clkdiv
17
18 // Instantiate clkdiv
19 clkdiv my_clkdiv (
20     .clk(clk),
21     .reset(reset),
22     .clk_out(clk_slow)
23 );
24
25 always @(*) begin
26     case(state) // State transition logic
27         2'b00 : next_state = 2'b01;
28         2'b01 : next_state = 2'b10;
29         2'b10 : next_state = 2'b11;
30         2'b11 : next_state = 2'b00;
31     endcase
32 end
33
34 always @(*) begin
35     case (state) // Multiplexing logic
36         2'b00 : begin sseg = in0;     an = 4'b1110; end
37         2'b01 : begin sseg = in1;     an = 4'b1101; end
38         2'b10 : begin sseg = in2;     an = 4'b1011; end
39         2'b11 : begin sseg = in3;     an = 4'b0111; end
40         default: begin sseg = 4'b0000; an = 4'b1111; end
41     endcase
42 end
```

```
43
44      // Use slow clock for state transitions
45      always @(posedge clk_slow or posedge reset) begin
46          if (reset)
47              state <= 2'b00;
48          else
49              state <= next_state;
50      end
51
52      endmodule
```

clkdiv.v (same as Part 2):

```
1       `timescale 1ns / 1ps
2
3       module clkdiv(
4           input clk,
5           input reset,
6           output clk_out
7       );
8
9
10          reg [26:0] COUNT;
11
12              assign clk_out=COUNT[26];
13
14              always @(posedge clk)
15              begin
16              if (reset)
17                  COUNT = 0;
18              else
19                  COUNT = COUNT + 1;
20              end
21
22      endmodule
```

Testbench:

```
`timescale 1ns / 1ps

module tb_time_multiplexing_main;

    // Testbench signals
    reg clk;
    reg reset;
    reg [15:0] sw;  // Switch input
    wire [3:0] an;  // Anode output
    wire [6:0] sseg; // 7-segment display output

    // Instantiate the main multiplexing module
    time_multiplexing_main uut (
        .clk(clk),
        .reset(reset),
        .sw(sw),
        .an(an),
        .sseg(sseg)
    );

    // Clock generation
    always #5 clk = ~clk; // 10ns clock period (100MHz simulated)

    // Test stimulus
    initial begin
        // Initialize signals
        clk = 0;
        reset = 1;
        sw = 16'h1234; // Input switches set to 0x1234

        #20 reset = 0; // Release reset after 20ns

        // Wait for some clock cycles to observe transitions
        #100;

        // Change switch inputs
        sw = 16'hABCD;
        #100;

        // Activate reset again to see the effect
        reset = 1;
        #20 reset = 0;

```

```
        // Run simulation for some more time
        #200;

        // Finish simulation
        $finish;
    end

endmodule
```

hexto7segment.v:

```
`timescale 1ns / 1ps

module hexto7segment (
    input[3:0] x,
    output reg [6:0] r
);

always @(*)
    case (x)
        4'b0000 : r = 7'b0000001;   //0
        4'b0001 : r = 7'b1001111;   //1
        4'b0010 : r = 7'b0010010;   //2
        4'b0011 : r = 7'b0000110;   //3
        4'b0100 : r = 7'b1001100;   //4
        4'b0101 : r = 7'b0100100;   //5
        4'b0110 : r = 7'b0100000;   //6
        4'b0111 : r = 7'b0001111;   //7
        4'b1000 : r = 7'b0000000;   //8
        4'b1001 : r = 7'b0000100;   //9
        4'b1010 : r = 7'b0001000;   //A
        4'b1011 : r = 7'b1100000;   //b
        4'b1100 : r = 7'b0110001;   //C
        4'b1101 : r = 7'b1000010;   //d
        4'b1110 : r = 7'b0110000;   //E
        4'b1111 : r = 7'b0111000;   //F
    endcase
endmodule
```

Part 3 Constraints:

```
## Clock signal - Uncomment if needed (will be used in future labs)
set_property PACKAGE_PIN W5 [get_ports clk]
    set_property IOSTANDARD LVCMOS33 [get_ports clk]
    create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]

## Switches
set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[9]}]
set_property PACKAGE_PIN T2 [get_ports {sw[10]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[10]}]
set_property PACKAGE_PIN R3 [get_ports {sw[11]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[11]}]
set_property PACKAGE_PIN W2 [get_ports {sw[12]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[12]}]
set_property PACKAGE_PIN U1 [get_ports {sw[13]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[13]}]
set_property PACKAGE_PIN T1 [get_ports {sw[14]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[14]}]
set_property PACKAGE_PIN R2 [get_ports {sw[15]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sw[15]}]
```

```
#7 segment display
set_property PACKAGE_PIN W7 [get_ports {sseg[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sseg[6]}]
set_property PACKAGE_PIN W6 [get_ports {sseg[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sseg[5]}]
set_property PACKAGE_PIN U8 [get_ports {sseg[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sseg[4]}]
set_property PACKAGE_PIN V8 [get_ports {sseg[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sseg[3]}]
set_property PACKAGE_PIN U5 [get_ports {sseg[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sseg[2]}]
set_property PACKAGE_PIN V5 [get_ports {sseg[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sseg[1]}]
set_property PACKAGE_PIN U7 [get_ports {sseg[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {sseg[0]}]

#set_property PACKAGE_PIN V7 [get_ports dp]
#   set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {an[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {an[0]}]
set_property PACKAGE_PIN U4 [get_ports {an[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {an[1]}]
set_property PACKAGE_PIN V4 [get_ports {an[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {an[2]}]
set_property PACKAGE_PIN W4 [get_ports {an[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {an[3]}]


##Buttons
set_property PACKAGE_PIN U18 [get_ports reset]
    set_property IOSTANDARD LVCMOS33 [get_ports reset]
```

All bitstream files are located in the lab submission folder.