LAB 4 Report

Name: Donald Maze-England

UT EID: dsm2588

Section: 15320

Checklist:

Part 1 -

i. Simulation waveform of the Flight Attendant Call System for behavioral as well as dataflow modelling
ii. K-map for minimizing the expression for next_state for dataflow modelling
iii. Boolean expression for next_state for dataflow modelling
iv. Completed design file (.v) for dataflow modelling

Part 2 -

v. State Diagram of the Rising Edge Detector
vi. Completed design files (.v) including the top module and clock divider
vii. Test-bench of the system
viii. Simulation waveform
ix. Constraints File (Just the uncommented portion)

Part 3 -

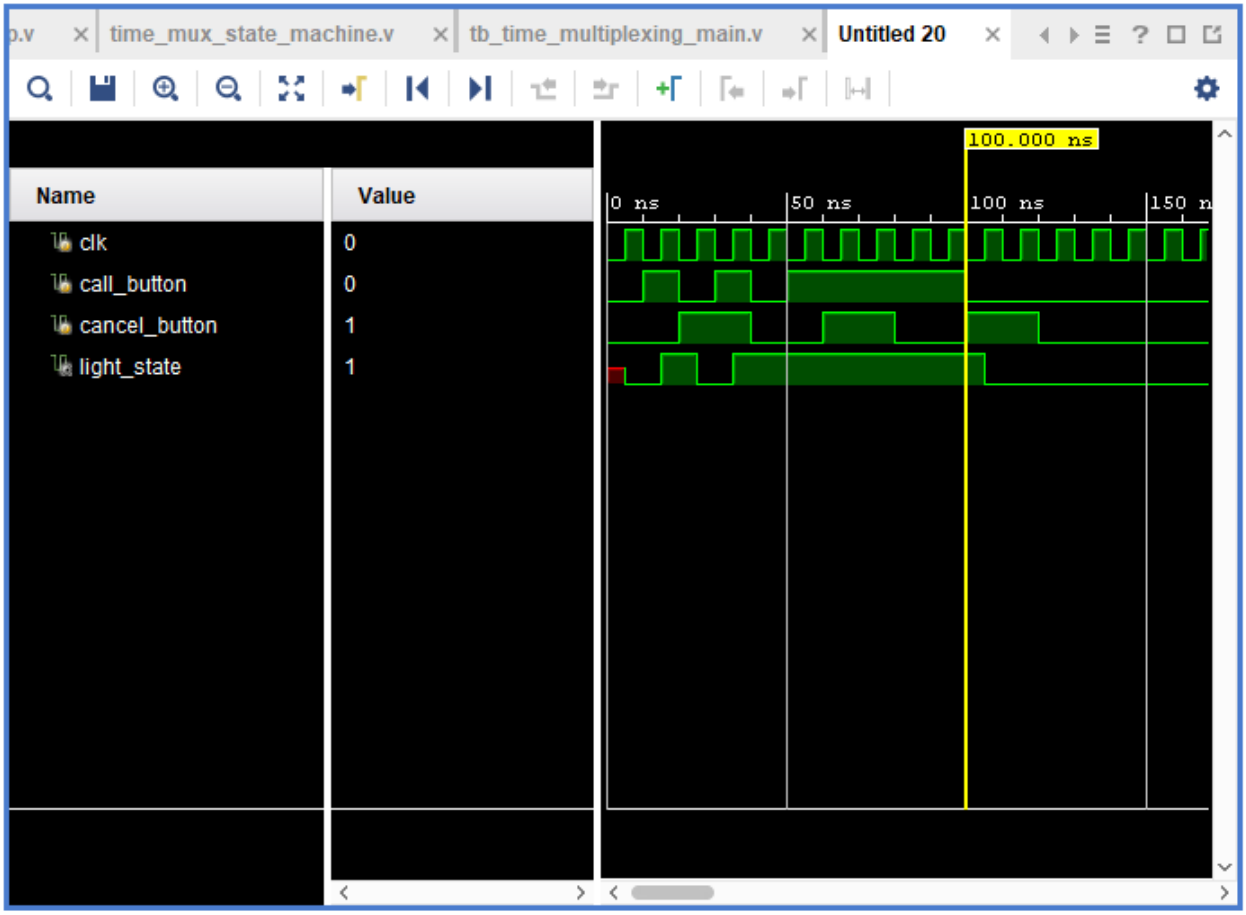
x. Completed design files (.v) of all the modules in the system
xi. Test-bench of the system
xii. Simulation waveform
xiii. Constraints File (Just the uncommented portion)

Note → *The Verilog codes and the uncommented portions of the constraint files should be copied in your lab report and the* ***actual Verilog (.v), Constraint (.xdc)***

files and Bitstream (.bit) files *need to be zipped and submitted as well on Canvas. You are not allowed to change your codes after final submission as the TAs may download the submitted codes or bitstream files from Canvas during checkouts. For the truth Table, K-maps minimizations and algebraic expressions, you are free to draw them on paper and then put the pictures in your lab report, but please make sure it is legible for the TAs to grade it properly.*

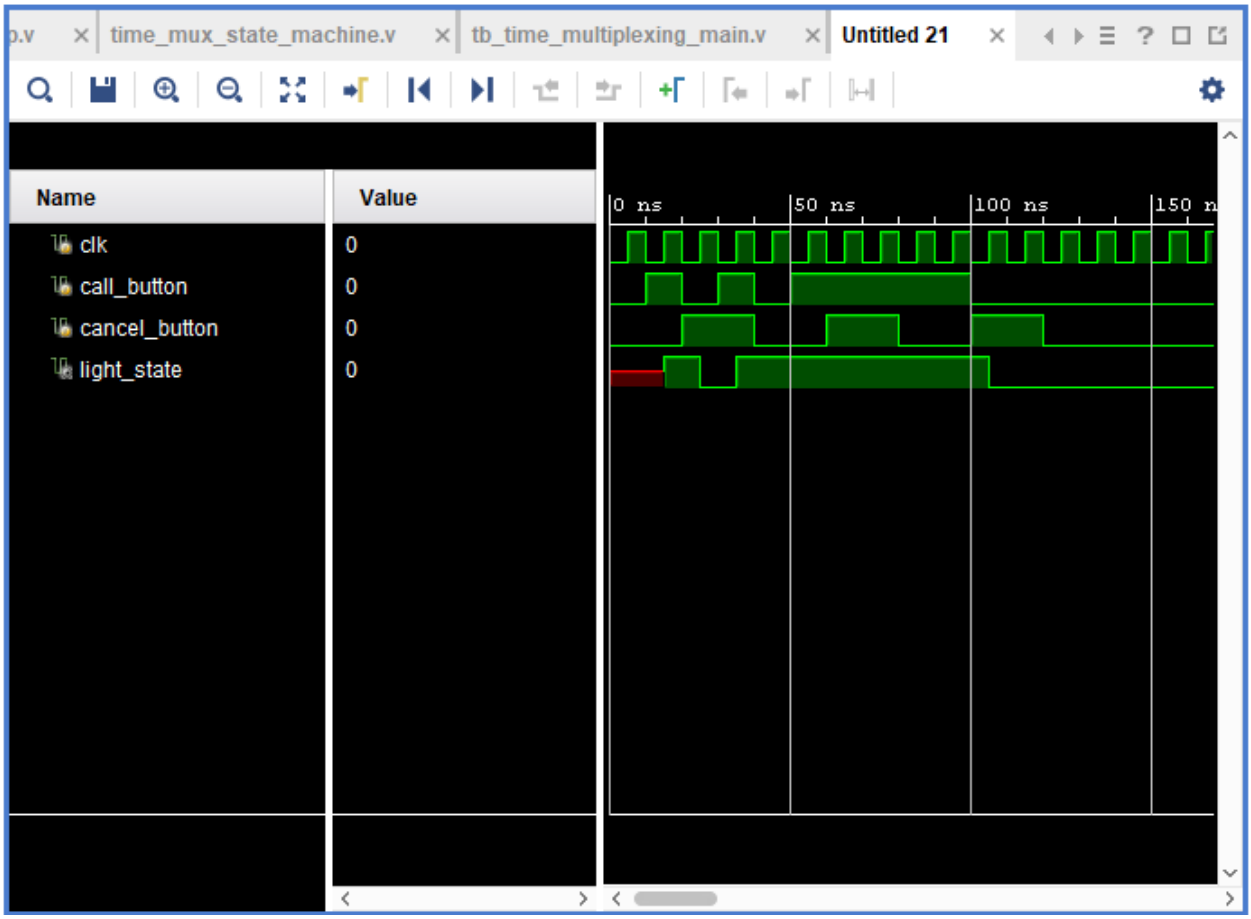
Part 1:

Flight attendant behavioural waveform:



Test Bench for part 2:

Flight Attendant Dataflow waveform:



K-Map for Minimizing Expression for Dataflow:

K Map for part 1:				
Output	Call,Cancel			
Current	00	01	11	10
0	0	0	1	1
1	1	0	1	1
Output = Call + Cancel'Current				

Boolean Expression for next_state for Dateflow Modeling:

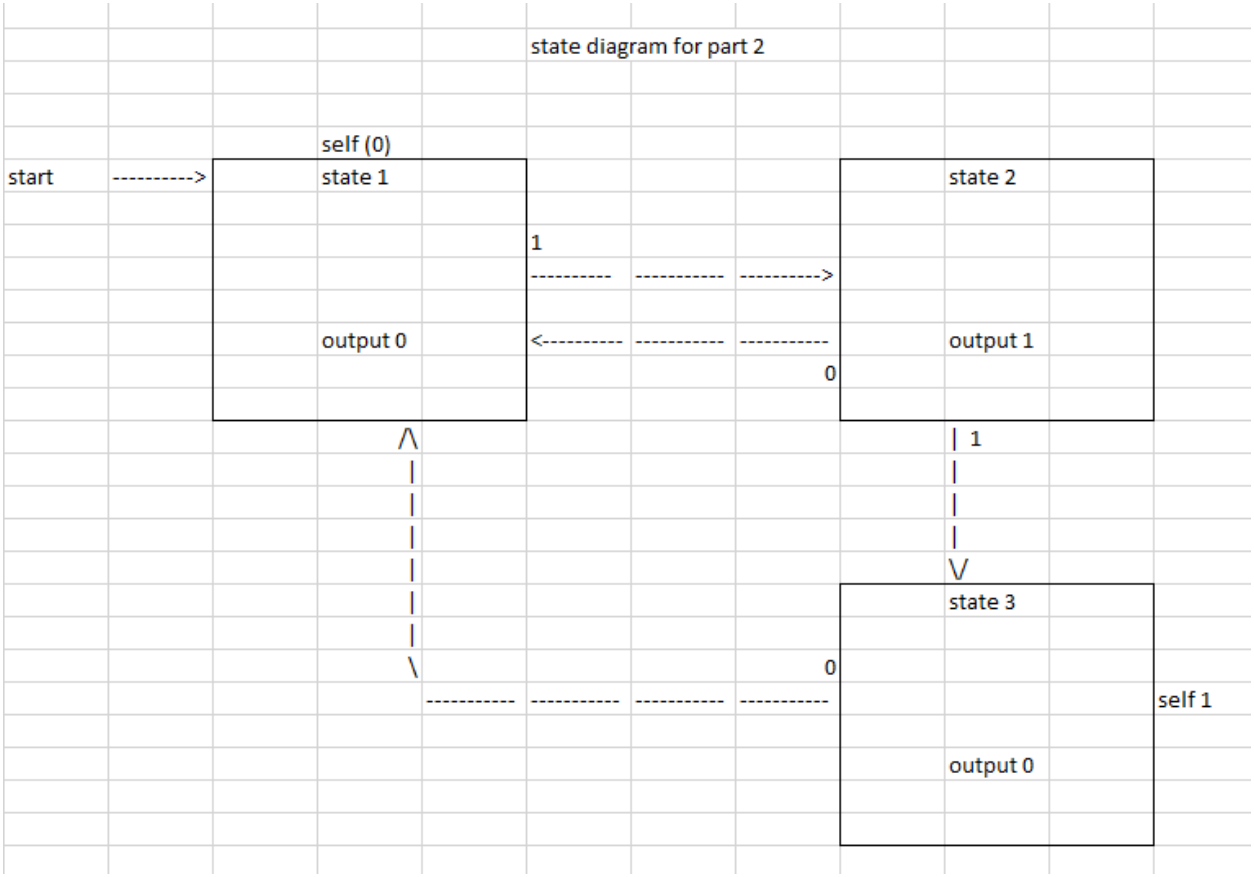
next_state = call_button | (light_state & ~cancel_button)

Completed design file (.v) for dataflow modelling:

```
`timescale 1ns / 1ps

module flight_attendent_call_system_dataflow(
    input wire clk,
    input wire call_button,
    input wire cancel_button,
    output reg light_state
    );

    wire next_state;

    assign next_state = call_button | (light_state & ~cancel_button);

    always @(posedge clk) begin
        light_state <= next_state;
    end

endmodule
```

Part 2:

State Diagram for the Rising Edge Detector:



Completed Verilog Files:

Top Module:

```
`timescale 1ns / 1ps



module rising_edge_detector(
    input clk,
    input signal,
    input reset,
    output reg outedge
    );

    wire slow_clk;

    reg [1:0] state = 2'b00;
    reg [1:0] next_state;

    clkdiv c1(clk, reset, slow_clk);



    //combinational logic
    always@(*) begin
        case (state)
```

```
2'b00 : begin
    outedge = 1'b0;
    if(~signal)
        next_state = 2'b00;
    else
        next_state = 2'b01;
end

2'b01 : begin
    outedge = 1'b1;
    if(~signal)
        next_state = 2'b00;
    else
        next_state = 2'b10;
end

2'b10 : begin
    outedge = 1'b0;
    if(~signal)
        next_state = 2'b00;
    else
        next_state = 2'b10;
end
```

```
            default : begin
                next_state = 2'b00;
                outedge = 1'b0;
                end

        endcase
    end



    //sequential logic
    always @(posedge slow_clk) begin
        if (reset)
            state <= 2'b00;
        else
            state <= next_state;
    end

endmodule
```

Clock Divider:

```
`timescale 1ns / 1ps



module clkdiv(
    input clk,
    input reset,
    output clk_out
    );

    reg [1:0] COUNT = 2'b00;

    assign clk_out = COUNT[1];

    always @(posedge clk) begin
        if (reset)
            COUNT = 0;
        else
            COUNT = COUNT+1;
        end

endmodule
```

Test Bench for part 2:

```
`timescale 1ns / 1ps



module tb_rising_edge_detector;

reg clk;
reg signal;
reg reset;


wire outedge;
wire slow_clk;

clkdiv c1(clk, reset, slow_clk);


rising_edge_detector ut(
    .clk(clk),
    .signal(signal),
    .reset(reset),
    .outedge(outedge)
);

initial begin
```

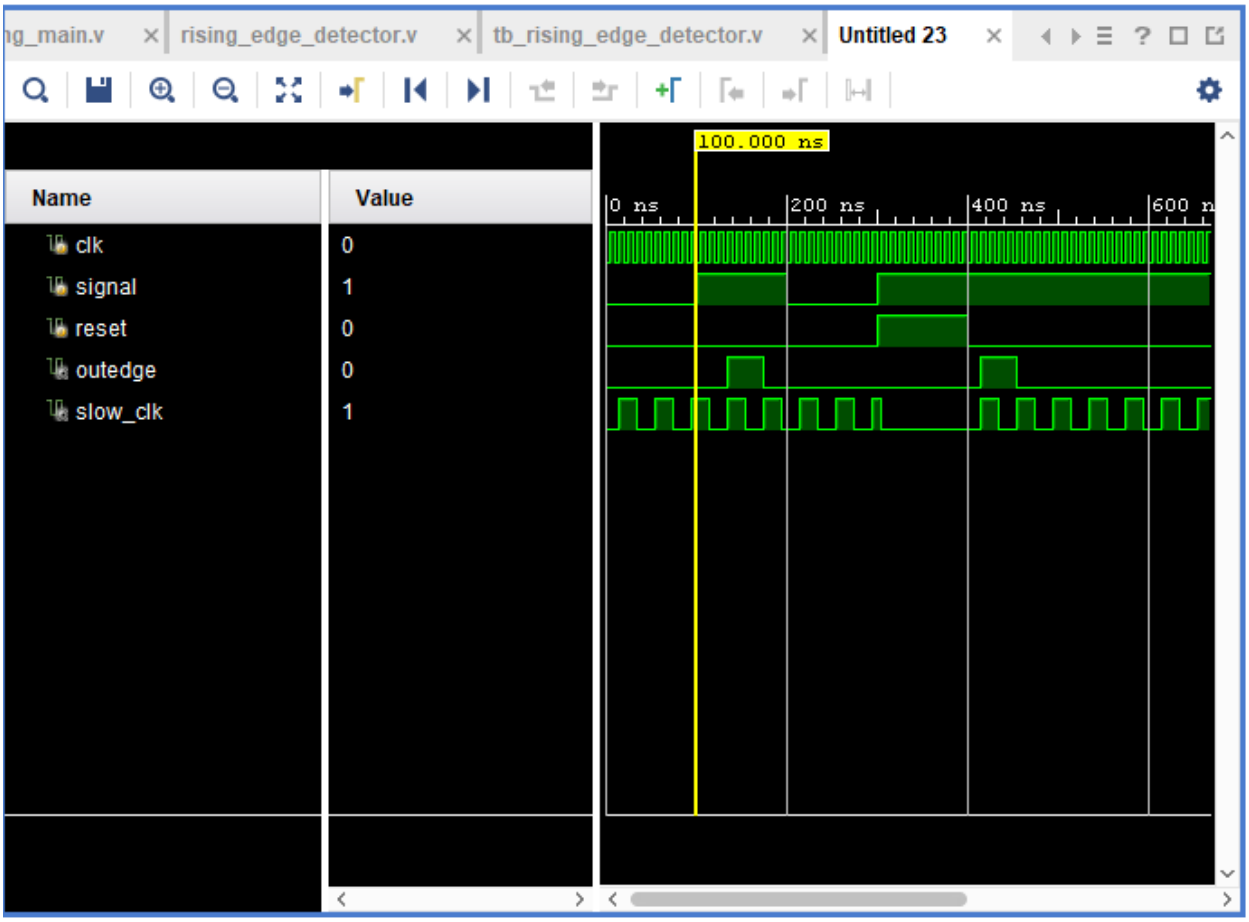
```
clk = 0;
signal = 0;
reset = 0;

#100;

signal = 1;
reset = 0;

#100

signal = 0;
reset = 0;

#100

signal = 1;
reset = 1;

#100

reset = 0;

end
```

always

#5 clk = ~clk;

endmodule

Simulation Wave form for part 2:



Constraints file for Part 2:

```
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]

set_property PACKAGE_PIN V17 [get_ports {signal}]
        set_property IOSTANDARD LVCMOS33 [get_ports {signal}]


set_property PACKAGE_PIN U16 [get_ports {outedge}]
        set_property IOSTANDARD LVCMOS33 [get_ports {outedge}]

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

Part 3:

Verilog Files:

Time_multiplexing_main:

```
`timescale 1ns / 1ps



module time_multiplexing_main(
    input clk,
    input reset,
    input [15:0] sw,
    output [3:0] an,
    output [6:0] sseg
    );

    wire [6:0] in0, in1, in2, in3;
    wire slow_clk;

    //Module instantiation of hexTo7Segment decoder
    hexTo7Segment c1 (.x(sw[3:0]), .r(in0));
    hexTo7Segment c2 (.x(sw[7:4]), .r(in1));
    hexTo7Segment c3 (.x(sw[11:8]), .r(in2));
    hexTo7Segment c4 (.x(sw[15:12]), .r(in3));

    //module instantiation of the clock divider
```

```
    clk_div_disp c5 (.clk(clk), .reset(reset), .clk_out(slow_clk));


    //module instantiation of the multiplexer
    time_mux_state_machine c6(
    .clk (slow_clk),
    .reset (reset),
    .in0 (in0),
    .in1 (in1),
    .in2 (in2),
    .in3 (in3),
    .an (an),
    .sseg (sseg)
    );
endmodule
```

hexTo7Segment:

```
`timescale 1ns / 1ps



module hexTo7Segment(
    input [3:0] x,
    output reg [6:0] r
    );
    always @(*)
        case (x)
            4'b0000 : r = 7'b0000001;
            4'b0001 : r = 7'b1001111;
            4'b0010 : r = 7'b0010010;
            4'b0011 : r = 7'b0000110;
            4'b0100 : r = 7'b1001100;
            4'b0101 : r = 7'b0100100;
            4'b0110 : r = 7'b0100000;
            4'b0111 : r = 7'b0001111;
            4'b1000 : r = 7'b0000000;
            4'b1001 : r = 7'b0001100;
            4'b1010 : r = 7'b0001000;
            4'b1011 : r = 7'b1100000;
            4'b1100 : r = 7'b0110001;
            4'b1101 : r = 7'b1000010;
            4'b1110 : r = 7'b0110000;
```

```
        4'b1111 : r = 7'b0111000;
    endcase


endmodule
```

clk_div_disp:

```
`timescale 1ns / 1ps


module clk_div_disp(
    input clk,
    input reset,
    output clk_out
    );

    reg [1:0] COUNT = 2'b00;

    assign clk_out = COUNT[1];

    always @(posedge clk) begin
        if (reset)
            COUNT = 0;
        else
            COUNT = COUNT+1;
        end

endmodule
```

time_mux_state_machine:

```
`timescale 1ns / 1ps



module time_mux_state_machine(
    input clk,
    input reset,
    input [6:0] in0,
    input [6:0] in1,
    input [6:0] in2,
    input [6:0] in3,
    output reg [3:0] an,
    output reg [6:0] sseg
    );

    reg [1:0] state = 2'b00;
    reg [1:0] next_state;



    always @(*) begin
        case (state)
            2'b00: next_state = 2'b01;
            2'b01: next_state = 2'b10;
            2'b10: next_state = 2'b11;
```

```
        2'b11: next_state = 2'b00;
    endcase
end

always @(*) begin
    case (state)
        2'b00: sseg = in0;
        2'b01: sseg = in1;
        2'b10: sseg = in2;
        2'b11: sseg = in3;
    endcase

    case(state)
        2'b00: an = 4'b1110;
        2'b01: an = 4'b1101;
        2'b10: an = 4'b1011;
        2'b11: an = 4'b0111;
    endcase
end

always @(posedge clk or posedge reset) begin
    if(reset)
        state<= 2'b00;
    else
        state <= next_state;
```

```
    end


endmodule
```

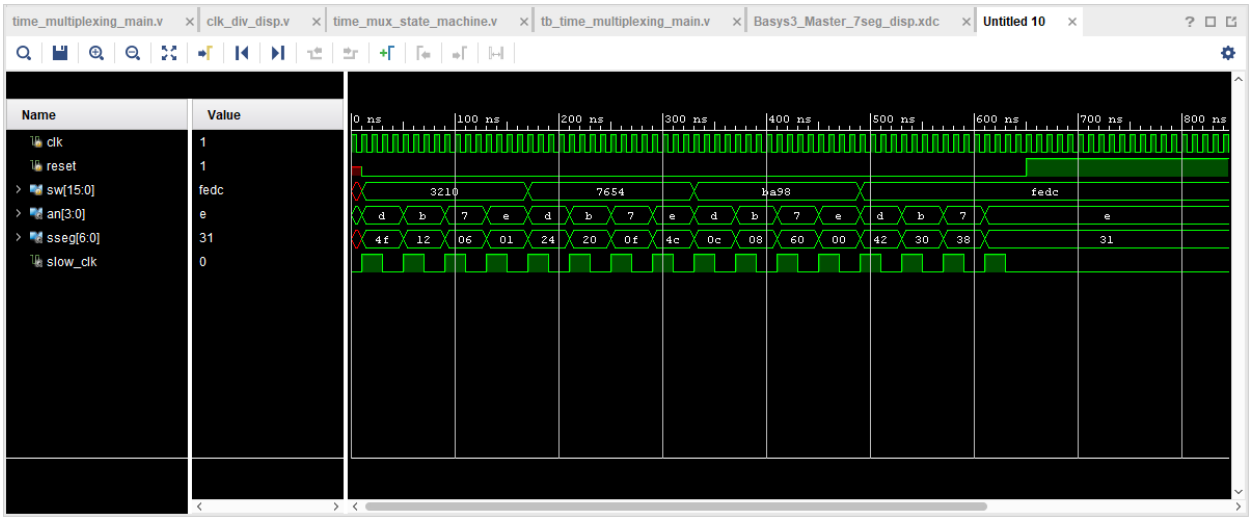
Testbench for part 3:

```
`timescale 1ns / 1ps



module tb_time_multiplexing_main;


reg clk;
reg reset;
reg [15:0] sw;
wire [3:0] an;
wire [6:0] sseg;

wire slow_clk;



time_multiplexing_main u7(
    .clk(clk),
    .reset(reset),
    .sw(sw),
    .an(an),
    .sseg(sseg)
);

    //module instantiation of the clock divider
    clk_div_disp u5 (.clk(clk), .reset(reset), .clk_out(slow_clk));
```

```
initial begin

clk = 1;

#10

reset = 0;
sw = 16'b0011001000010000;

#160;

sw = 16'b0111011001010100;

#160;

sw = 16'b1011101010011000;

#160;

sw = 16'b1111111011011100;

#160;

reset = 1;
```

```
#160;

end

always

    #5 clk = ~clk;


endmodule
```

Simulation Waveform for part 3:



Constraints File for Part 3:

```
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]

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]}]

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 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]}]

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