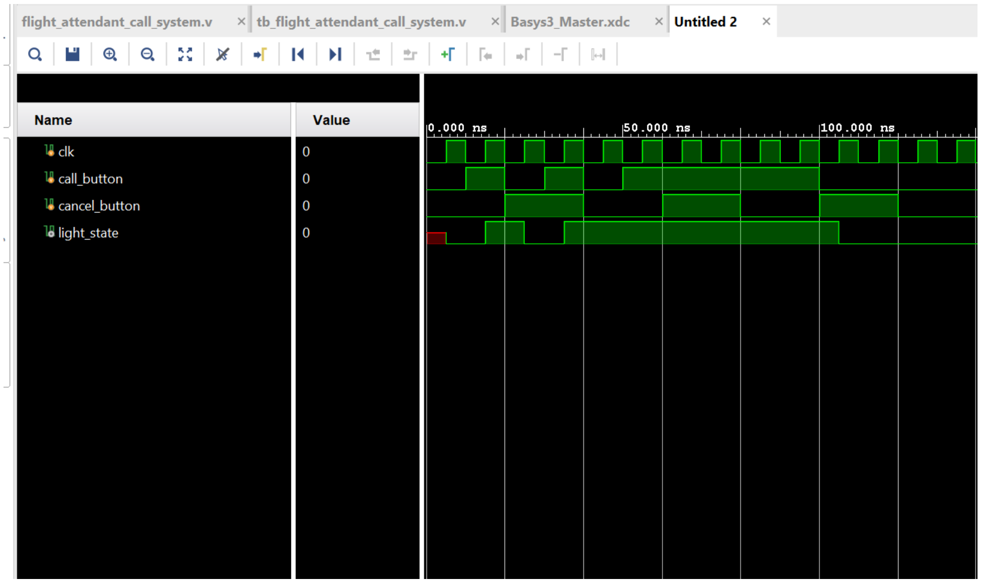
Lab 4 Report

Name: Devin Chaky **UTEID:** dmc4627 **Section:** 17015

Name: Manasvi Perisetty **UTEID:** mp49564 **Section:** 17015

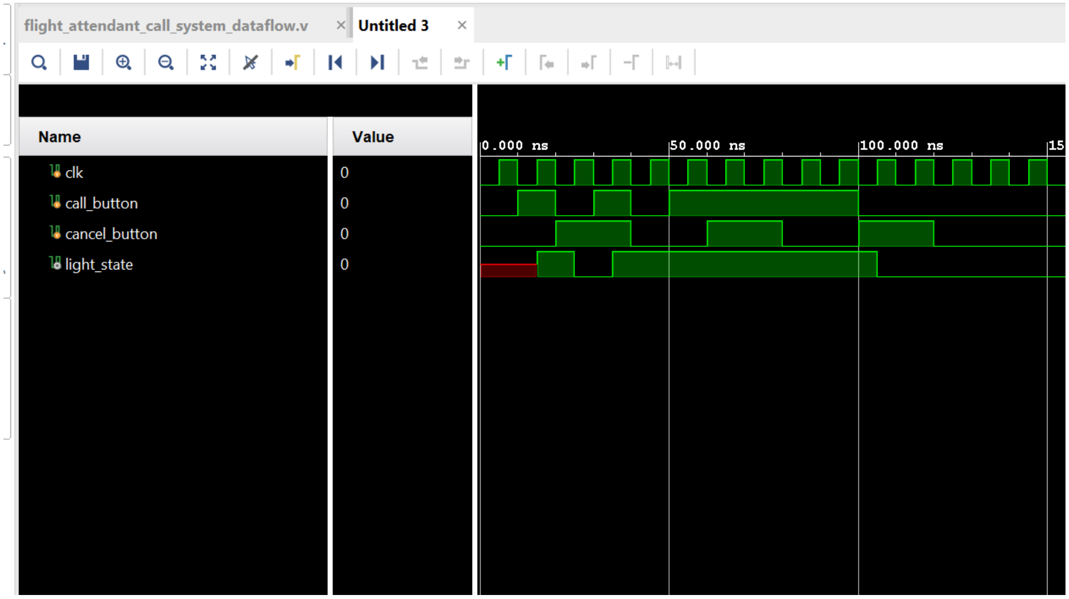
Part 1 -

i. Simulation Waveform for behavioral modeling:



Name	Value
clk	0
call_button	0
cancel_button	0
light_state	0

Simulation Waveform for dataflow modeling:



Name	Value
clk	0
call_button	0
cancel_button	0
light_state	0

ii / iii. K-map for minimizing the expression for next_state for dataflow modeling

Call	Cancel	D	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Q minimized:

call \ can D	00	01	11	10
0		1		
1	1	1	1	1

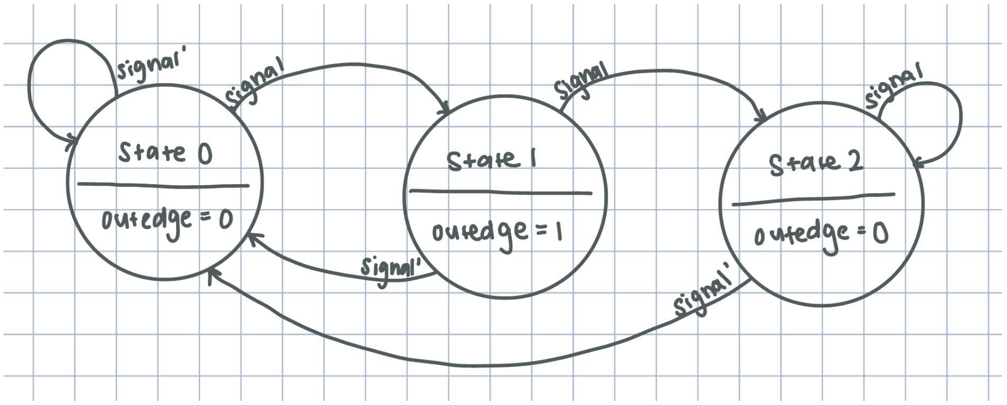
Q = call + (~cancel & D)

iv. Completed Design File for Dataflow Modeling

```
module flight_attendant_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 | (~cancel_button & light_state);

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

endmodule
```

Part 2 -

v. State Diagram for Rising Edge Detector



vi. Completed Design File for Rising Edge Detector

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

    wire slow_clk;

    reg[1:0] state;
    reg[1:0] next_state;

    clkdivide cl(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'b10;
                else
                    next_state = 2'b00;
                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 clk) begin
        if(reset)
            state <= 2'b00;
        else
            state <= next_state;
      end

endmodule

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

    reg[20:0] COUNT;

    assign clk_out=COUNT[20];

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

endmodule
```

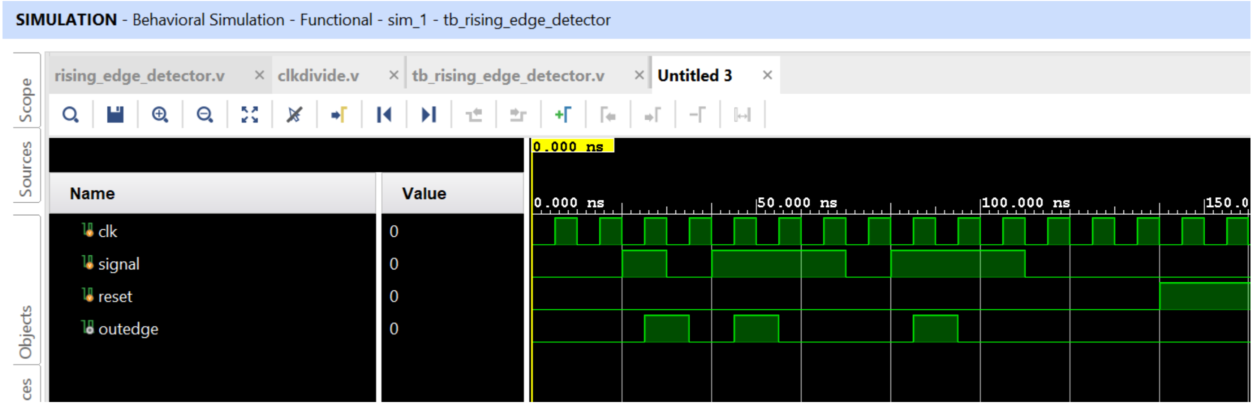
vii. Testbench Code

```
module tb_rising_edge_detector;

reg clk;
reg signal;
reg reset;
wire outedge;

rising_edge_detector uut (.clk(clk), .signal(signal), .reset(reset), .outedge(outedge));
```

```
initial
begin

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

#10;

signal = 0;

#10;

signal = 1;

#10;
signal = 0;

#10;
signal = 1;

#10;
signal = 1;

#10;
signal = 1;

#10;
signal = 0;

#10;
signal = 1;

#10;
signal =1;

#10;
signal = 1;

#10;
signal = 0;

#10;
signal = 0;

#20;
signal =0;
reset = 1;
```

```
end

always
#5 clk = ~clk;



endmodule
```

viii. Simulation Waveform for Rising Edge Detector



SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_rising_edge_detector

Name	Value
clk	0
signal	0
reset	0
outedge	0

ix. Constraints File

```
# 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
# Connects pin V17 (SW0 on the board) to input a in our gate module
set_property PACKAGE_PIN V17 [get_ports {signal}]
# Sets the switch to use 3.3V logic
        set_property IOSTANDARD LVCMOS33 [get_ports {signal}]

## 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}]

##Buttons
```

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

Part 3-

x. Completed Design Files

```
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 hextosegment 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), .slow_clk(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


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

    always@(*)
```

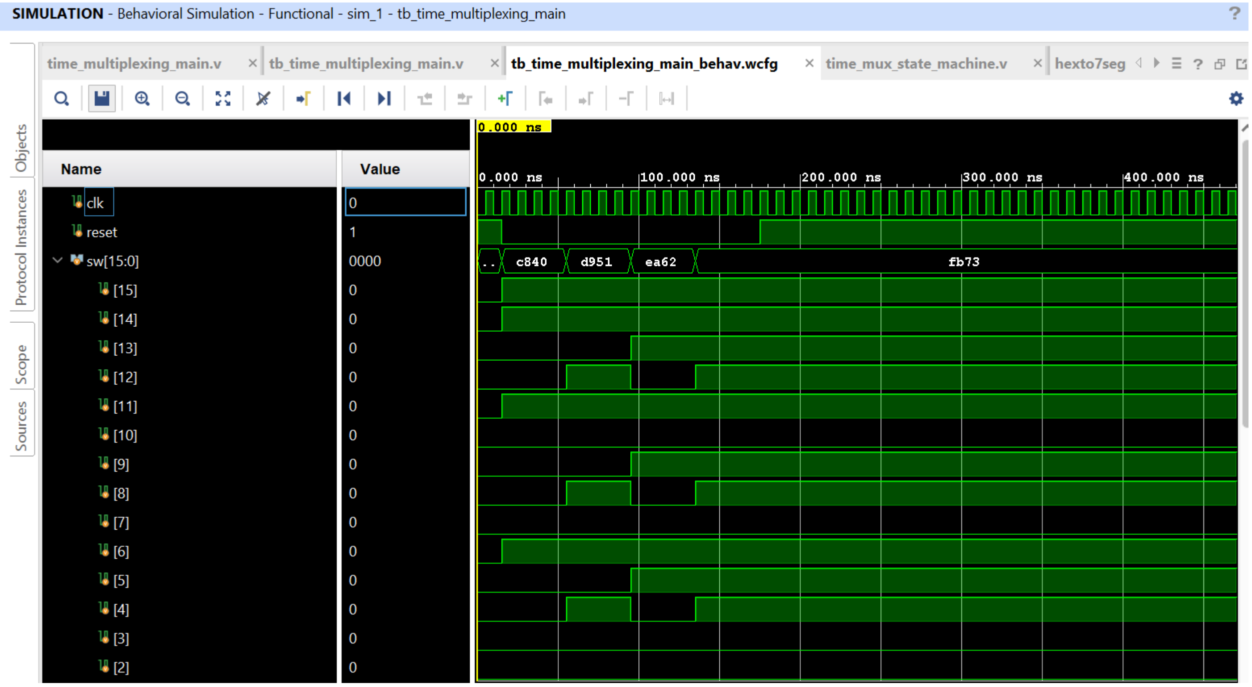
```
        case (x)
            4'b0000 : r = 7'b0000001;
            4'b0001 : r = 7'b1001111;
            4'b0010 : r = 7'b0010010;
            4'b0011 : r = 7'b0000110;
            4'b0100 : r = 7'b1001100;
            4'b0101 : r = 7'b0100100;
            4'b0110 : r = 7'b0100000;
            4'b0111 : r = 7'b0001111;
            4'b1000 : r = 7'b0000000;
            4'b1001 : r = 7'b0000100;
            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


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

    reg[25:0] COUNT;

    assign slow_clk=COUNT[25];

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

endmodule


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;
  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;
      default : next_state = 2’b00;
     endcase
   end

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

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

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

endmodule
```

xi. Testbench Code

```
module tb_time_multiplexing_main;

reg clk;
reg reset;
reg [15:0]sw;

wire [3:0]an;
wire [6:0]sseg;

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

initial
begin

clk = 0;
reset = 1;
sw = 0;

#15;

reset = 0;

sw[0] = 0;  //0

sw[6] = 1; //4

sw[11] = 1; //8

sw[14] = 1; //12
sw[15] = 1;

#40;

sw = 0;
sw[0] = 1;  //1

sw[4]=1;    // 5
sw[6]=1;

sw[8] = 1;  // 9
sw[11] = 1;

sw[12] = 1; //13
sw[14] = 1;
sw[15] = 1;
```

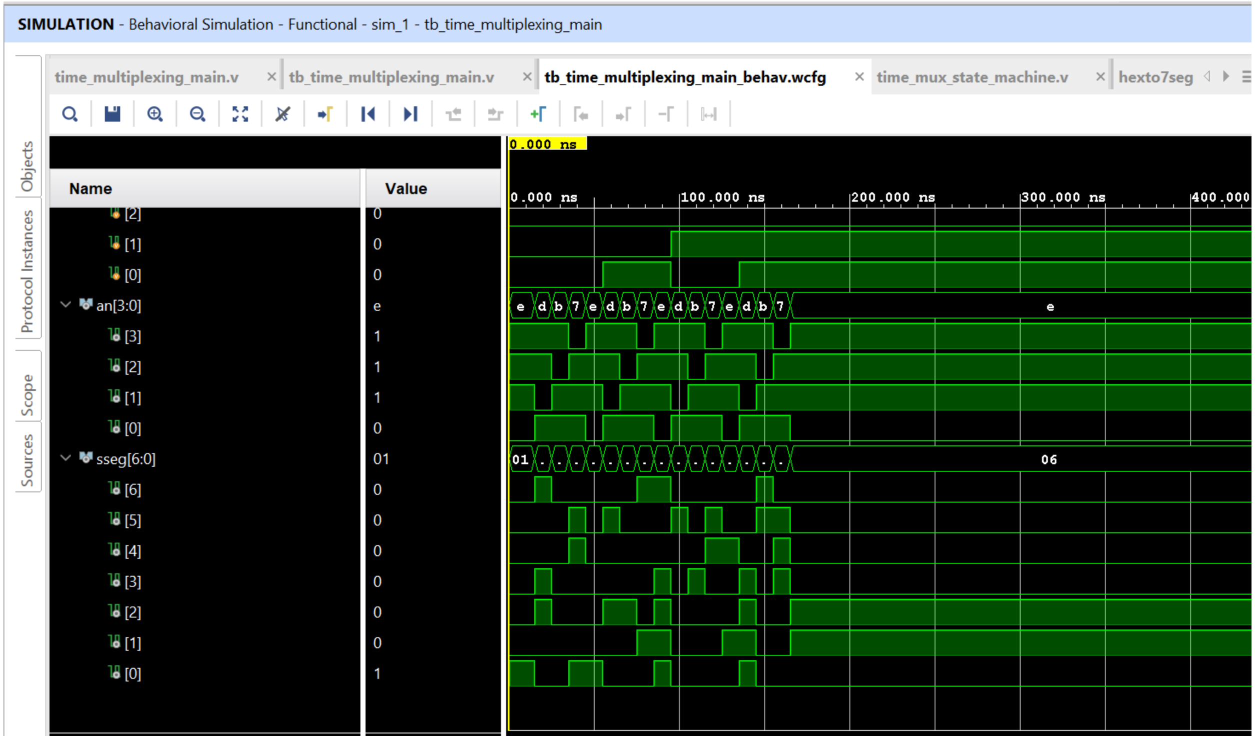
```
#40;

sw = 0;
sw[1] = 1;  //2

sw[5] =1;   // 6
sw[6] = 1;

sw[9] = 1;  // 10
sw[11] = 1;

sw[13] = 1; //14
sw[14] = 1;
sw[15] = 1;

#40;

sw = 0;
sw[0] = 1;  // 3
sw[1] = 1;

sw[4] = 1; // 7
sw[5] = 1;
sw[6] = 1;

sw[8] = 1;  //11
sw[9] = 1;
sw[11] = 1;

sw[12] = 1; // 15
sw[13] = 1;
sw[14] = 1;
sw[15] = 1;

#40;
reset = 1;

end

always
#5 clk = ~clk;


endmodule
```

xii. Simulation Waveform for BCD



SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_time_multiplexing_main

Name	Value
clk	0
reset	1
sw[15:0]	0000
[15]	0
[14]	0
[13]	0
[12]	0
[11]	0
[10]	0
[9]	0
[8]	0
[7]	0
[6]	0
[5]	0
[4]	0
[3]	0
[2]	0



SIMULATION - Behavioral Simulation - Functional - sim_1 - tb_time_multiplexing_main

Name	Value
[2]	0
[1]	0
[0]	0
an[3:0]	e
[3]	1
[2]	1
[1]	1
[0]	0
sseg[6:0]	01
[6]	0
[5]	0
[4]	0
[3]	0
[2]	0
[1]	0
[0]	1

xii. Constraints File:

```
# 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
## Connects pin V17 (SW0 on the board) to input a in our gate module
set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
# Sets the switch to use 3.3V logic
	set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]

# Connects pin V16 (SW1 on the board) to input b in our gate module
set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
# Sets the switch to use 3.3V logic
	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 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}]
```