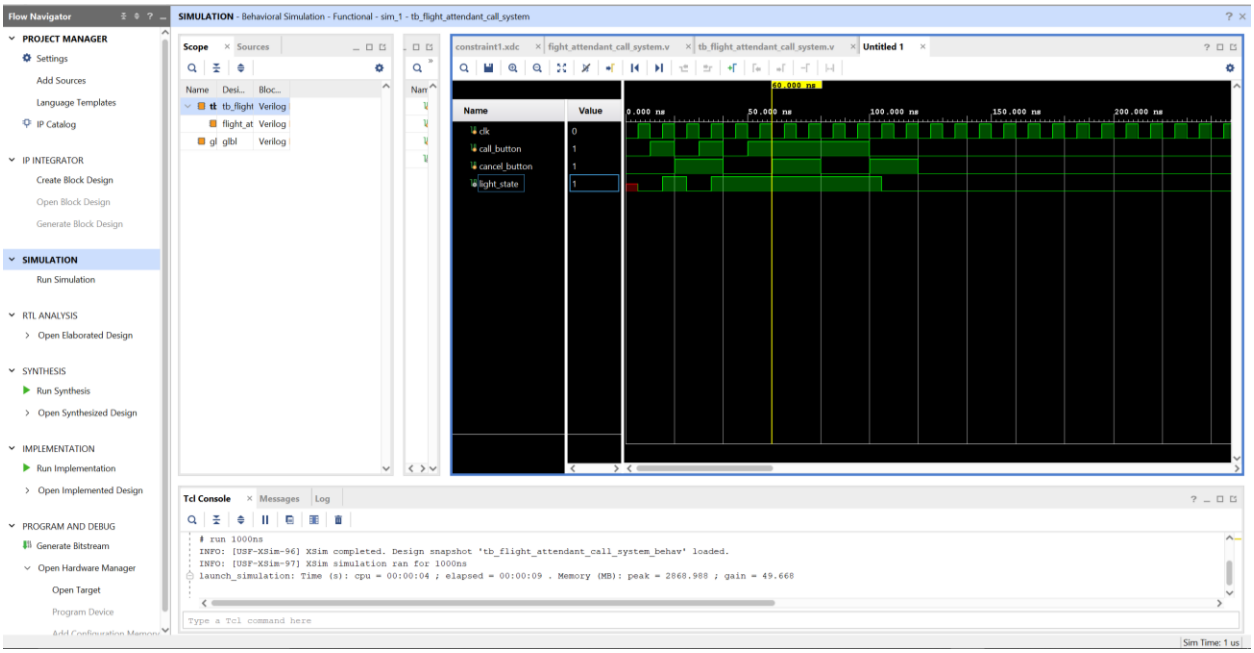
# ***EE 316 – Lab 4 Report*: Sequential Logic Design**

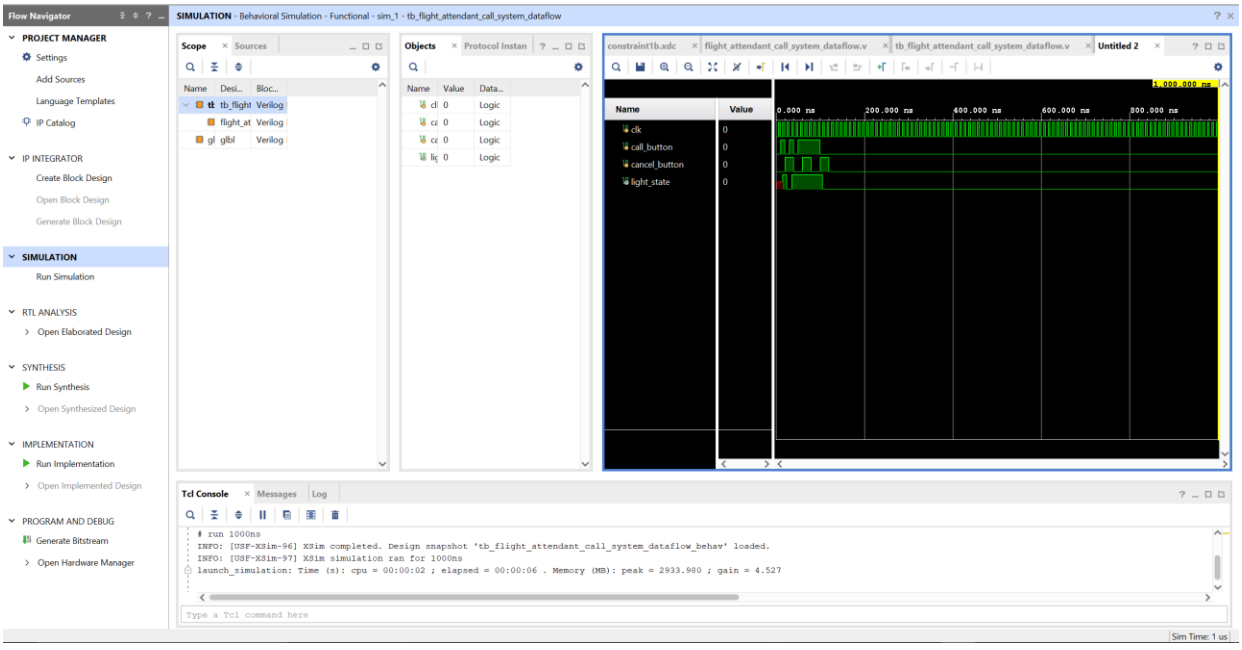
**Ayan Basu** [redacted]

## ***Part 1 (Finite State Machine)*: Design of Flight Attendant Call System**

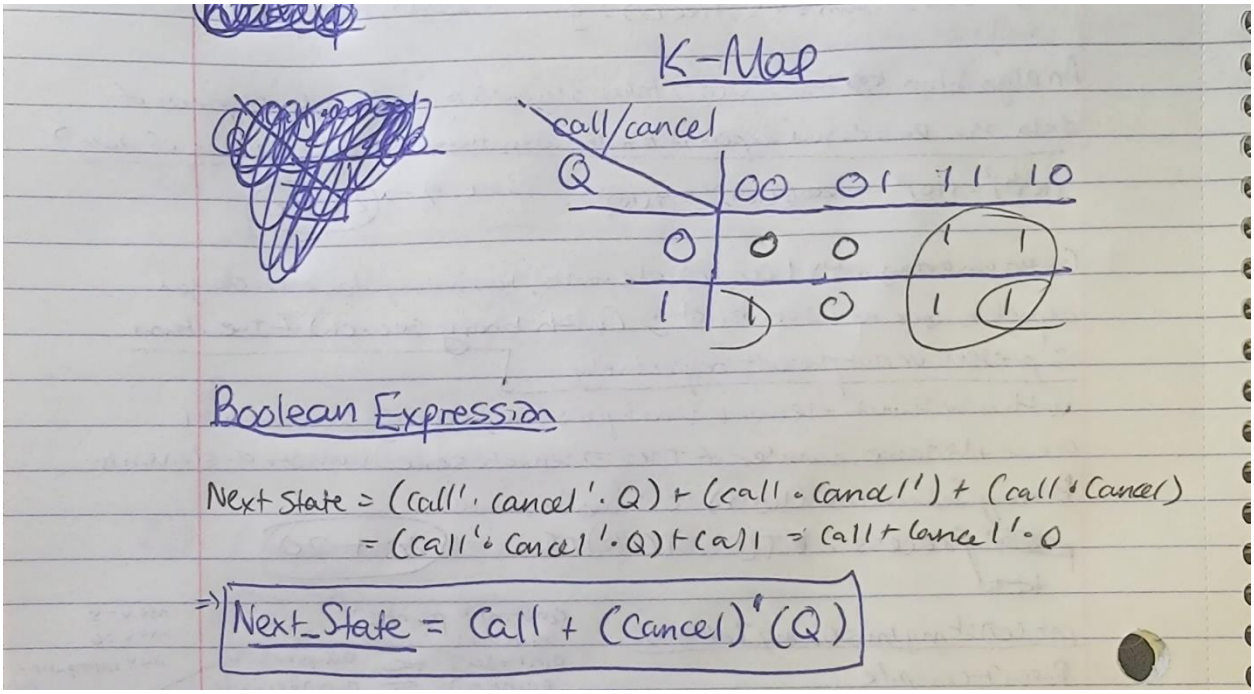
### ***Waveform Simulation – Behavioral Model***




### ***Waveform Simulation – Dataflow Model***



***K-Map & Boolean Expression***

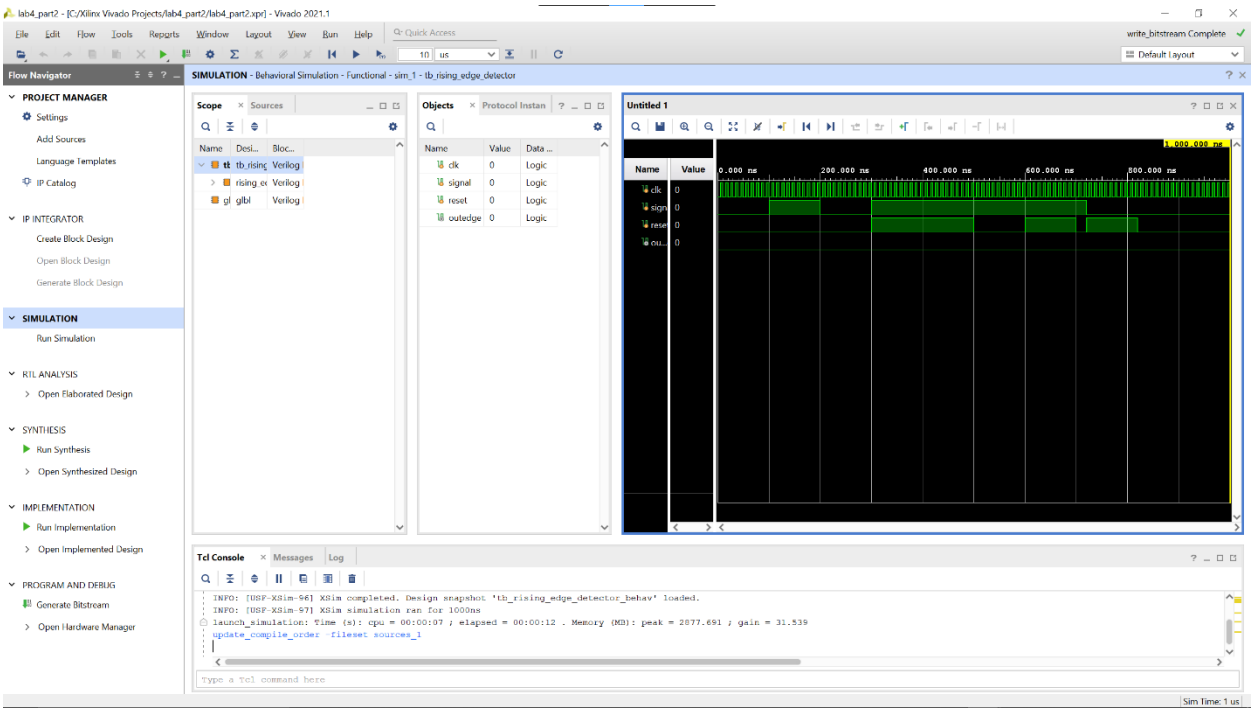



***Design File (.v) for Dataflow Modeling***

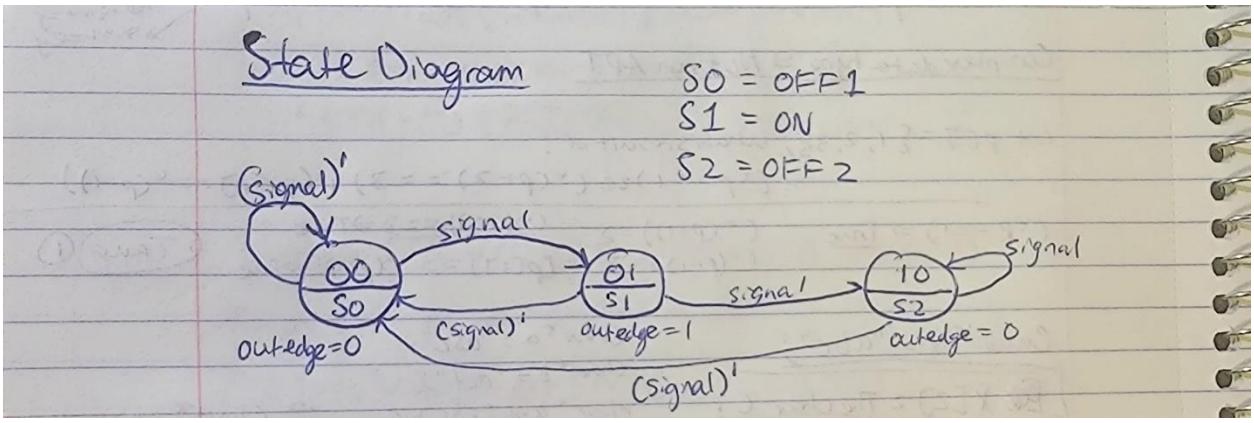
```
`timescale 1ns / 1ps
module flight_attendant_call_system_dataflow(
    input wire clk,
    input wire call_button,
    input wire cancel_button,
    output reg light_state
);
wire next_state;
assign next_state = (((~cancel_button) & light_state) | (call_button));

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

## ***Part 2 (Edge Detector)*: Design of Rising Edge Detector**

### ***Waveform Simulation***




### ***State Diagram***




### ***Design Files – Rising Edge Detector***

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

```
//
// Create Date: 10/20/2021 07:43:08 PM
// Design Name:
// Module Name: rising_edge_detector
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


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;
```

```
clkdiv cl(clk, reset, slow_clk);



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

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

   end

endmodule
```

***<u>Design Files – Clock Divider</u>***

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
```

```
// Engineer:
//
// Create Date: 10/20/2021 07:41:10 PM
// Design Name:
// Module Name: clkdiv
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module clkdiv(

    input clk,
    input reset,
    output clk_out

    );

    reg [25:0] COUNT;
```

```
    initial begin
    COUNT = 0;
    end

       assign clk_out = COUNT[25];

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

endmodule
```

***Rising Edge Detector Test-Bench System***

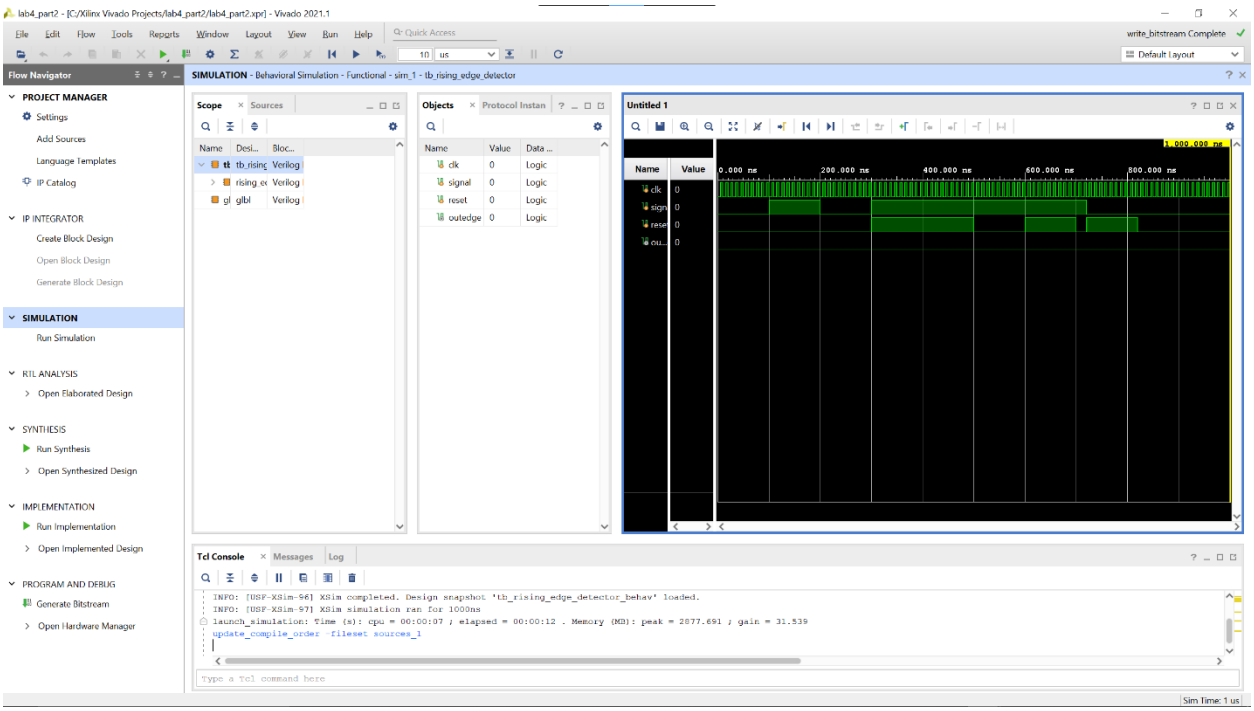
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 10/20/2021 07:44:22 PM
// Design Name:
// Module Name: tb_rising_edge_detector
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
```

```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module tb_rising_edge_detector;

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

   rising_edge_detector ul (
      .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;
signal = 1;
reset = 1;

#100;

signal = 1;
reset = 0;
```

```
        #100;

        reset = 1;

        #100;

        reset = 0;

        #20;

        signal = 0;
        reset = 1;

        #100


        signal = 0;
        reset =0;

end

    always
    #5 clk = ~clk;

endmodule
```

***<u>Waveform Simulation</u>***




***Constraints File***

## Clock signal

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 {signal}]

set_property IOSTANDARD LVCMOS33 [get_ports {signal}]

## LEDs

set_property PACKAGE_PIN U16 [get_ports {outedge}]

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 (Time Multiplexing)*: Design of Controller for Sequential Display of 4 Digits**

***Clk_Disp***

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

reg[25:0] COUNT = 0;

assign clk_out = COUNT[25];

always @(posedge clk) begin
   COUNT = COUNT + 160;
end
endmodule
```

***Time Mux State Machine***

```
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;
   endcase
end

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

always @(*) begin
   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
```

***Time Multiplexing Main***

```
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;

hex_to_7segment c1 (.x(sw[3:0]), .r(in0));
hex_to_7segment c2 (.x(sw[7:4]), .r(in1));
hex_to_7segment c3 (.x(sw[11:8]), .r(in2));
hex_to_7segment c4 (.x(sw[15:12]), .r(in3));

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

time_mux_state_machine c6(
```

```
    .clk(slow_clk),
    .reset(reset),
    .in0(in0),
    .in1(in1),
    .in2(in2),
    .in3(in3),
    .an(an),
    .sseg(sseg));

Endmodule
```

**Hextosevensegmentdisplay**

```
module hex_to_7segment(
    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
```

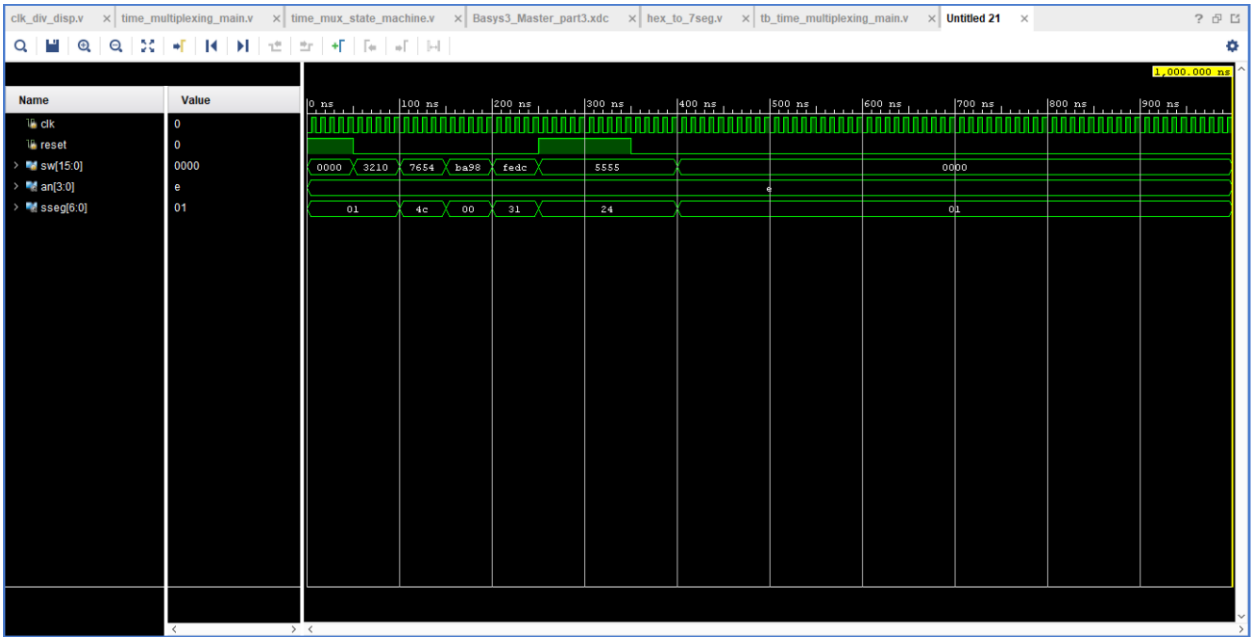
***Design Files – Multiplexing Test-Bench Files***

```
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;

#50;

reset = 0;
sw = 16'h3210;

#50;

sw = 16'h7654;

#50;

sw = 16'hBA98;

#50;

sw = 16'hFEDC;

#50;

sw = 16'h5555;
reset = 1;

#100;
```

```
reset = 0;

#50;

sw = 16'h0000;



end

always
#5 clk = ~clk;

endmodule
```

***Waveform Simulation***




***Constraints File***

## Clock signal

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 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}]
```