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

**Ayan Basu [EID: ab73287] (Section: 17760)**

**Saptarshi Mondal [EID: sm72999] (Section: 17760)**

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

***Waveform Simulation – Behavioral Model***

Graphical user interface, application, PowerPoint

Description automatically generated

***Waveform Simulation – Dataflow Model***

**Graphical user interface, application

Description automatically generated**

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

Text, letter

Description automatically generated

***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***

Graphical user interface, application

Description automatically generated

***State Diagram***

A piece of paper with writing on it

Description automatically generated

***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

***Design Files – Clock Divider***

`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

***Waveform Simulation***

Graphical user interface, application

Description automatically generated

***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***

Graphical user interface

Description automatically generated

***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}]