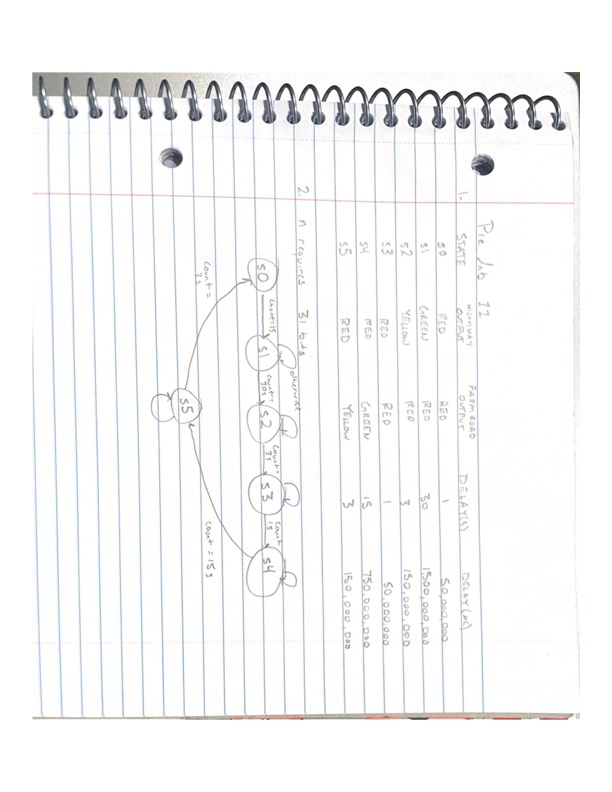
Joaquin Salas

11-27-22

Pre-Lab #11



`define one\_sec 50000000

`define three\_sec 150000000

`define thirty\_sec 1500000000

`define fifteen\_sec 750000000

module tlc\_fsm(

output reg [2:0] state, //output for debugging

output reg RstCount, //use an always block

/\*another always block for these as well\*/

output reg [1:0] highwaySignal, farmSignal;

input wire [30:0] Count, //use n(31) computed earlier

input wire Clk, Rst //clock and reset

);

//defining states

parameter s0 = 3'b000;

parameter s1 = 3'b001;

parameter s2 = 3'b010;

parameter s3 = 3'b011;

parameter s4 = 3'b100;

parameter s5 = 3'b101;

/\*intermediate nets\*/

reg [2:0] nextState;

reg [1:0] state;

//defining colors

parameter green = 2'b00;

parameter yellow = 2'b01;

parameter red = 2'b10;

/\*describing next state logic\*/

always@(state or Count)

case(state)

//Srst: nextState = s0;

s0: begin

if(Count == one\_sec) //if count reached

nextState = s1; //transition

else //otherwise remain in current state

nextState = s0;

end

s1: begin

if(Count == thirty\_sec) //if this count is reached

nextState = s2; //transition

else //remain in current state

end

s2: begin

if(Count == three\_sec) //if count reached

nextState = s3; //transition

else //otherwise

nextState = s2; //remain in current state

end

s3: begin

if(Count == one\_sec) //if count reached

nextState = s4; //transition

else //otherwise

nextState = s3; //remain in current state

end

s4: begin

if(Count == fifteen\_sec) //if count reached

nextState = s5; //transition

else //otherwise

nextState = s4; //remain in current state

end

s5: begin

if(Count == three\_sec) //if count reached

nextState = s0; //transition

else //otherwise

nextState = s5; //remain in current state

end

default: //avoid latches

nextState = s0;

endcase

/\*describe output logic\*/

always@(state or Count)

case(state)

s0: begin

highwaySignal = red;

farmSignal = red;

if(Count == one\_sec) //if count reached

RstCount = 1; //reset counter

else //otherwise

RstCount = 0; //let counter run

end

s1: begin

highwaySignal = green;

farmSignal = red;

if(Count == thirty\_sec) //if count reached

RstCount = 1; //reset counter

else //otherwise

RstCount = 0; //let counter run

end

s2: begin

highwaySignal = yellow;

farmSignal = red;

if(Count == three\_sec) //if count reached

RstCount = 1; //reset counter

else //otherwise

RstCount = 0; //let counter run

end

s3: begin

highwaySignal = red;

farmSignal = red;

if(Count == one\_sec) //if count reached

RstCount = 1; //reset counter

else //otherwise

RstCount = 0; //let counter run

end

s4: begin

highwaySignal = red;

farmSignal = green;

if(Count == fifteen\_sec) //if count reached

RstCount = 1; //reset counter

else //otherwise

RstCount = 0; //let counter run

end

s5: begin

highwaySignal = red;

farmSignal = yellow;

if(Count == three\_sec) //if count reached

RstCount = 1; //reset counter

else //otherwise

RstCount = 0; //let counter run

end

default: begin

highwaySignal = red;

farmSignal = red;

RstCount = 1;

end

endcase

/\*behavior for input clock\*/

always@(posedge Clk)

case(state)

if(Rst)

state <= s0;

else

state <= nextState;

endcase

endmodule