Final project

Student：Shi Cheng

Student ID：X673894

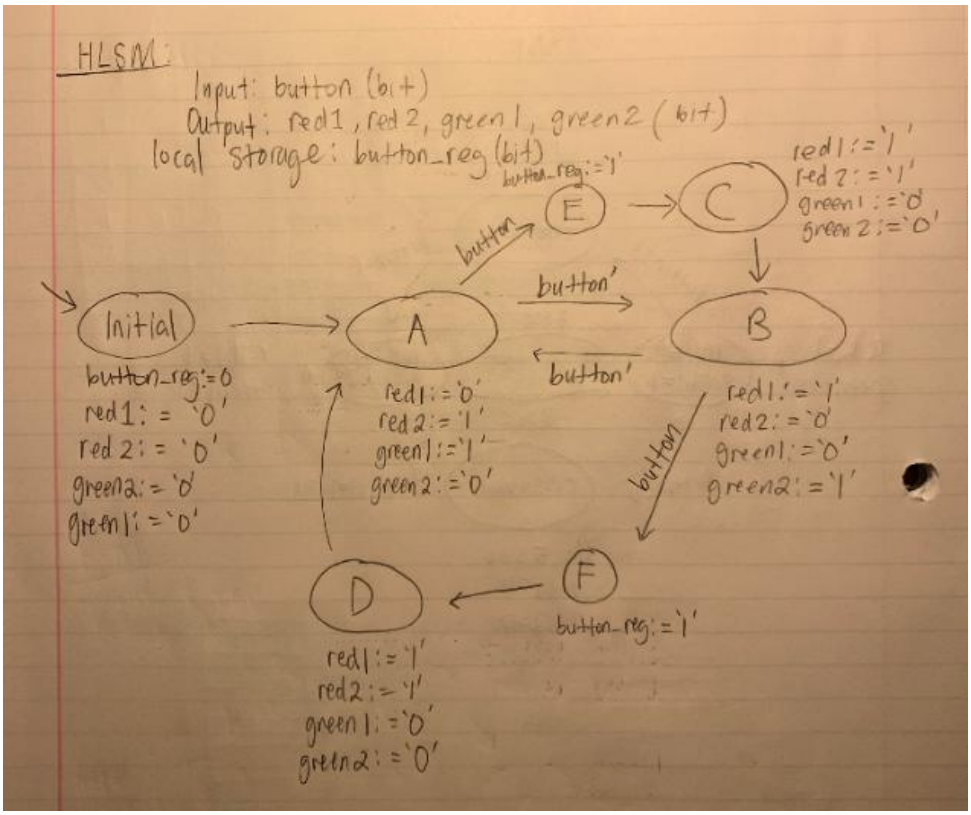
Course and section：EE 120A Section21

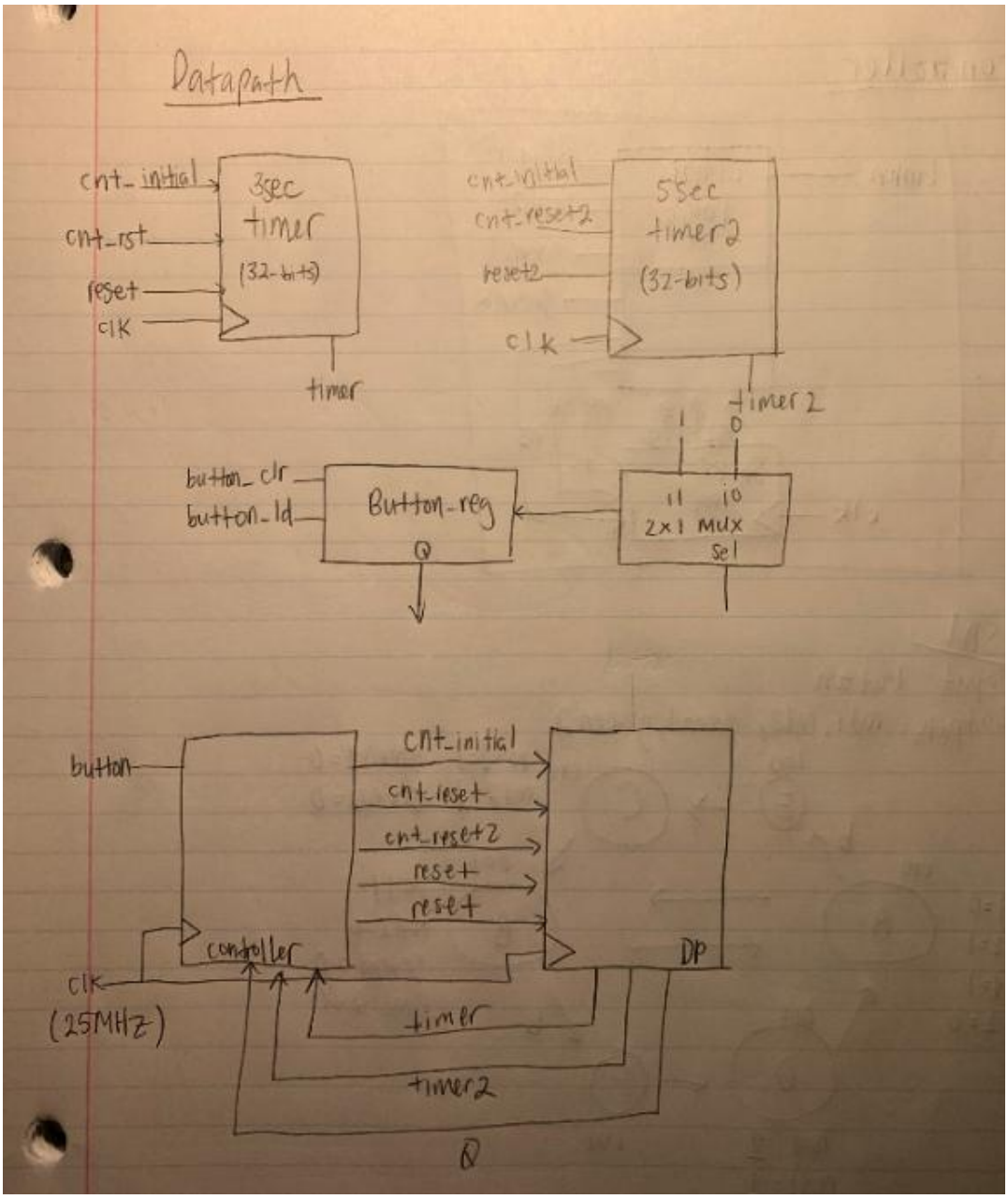
Partner: Yilun Liang

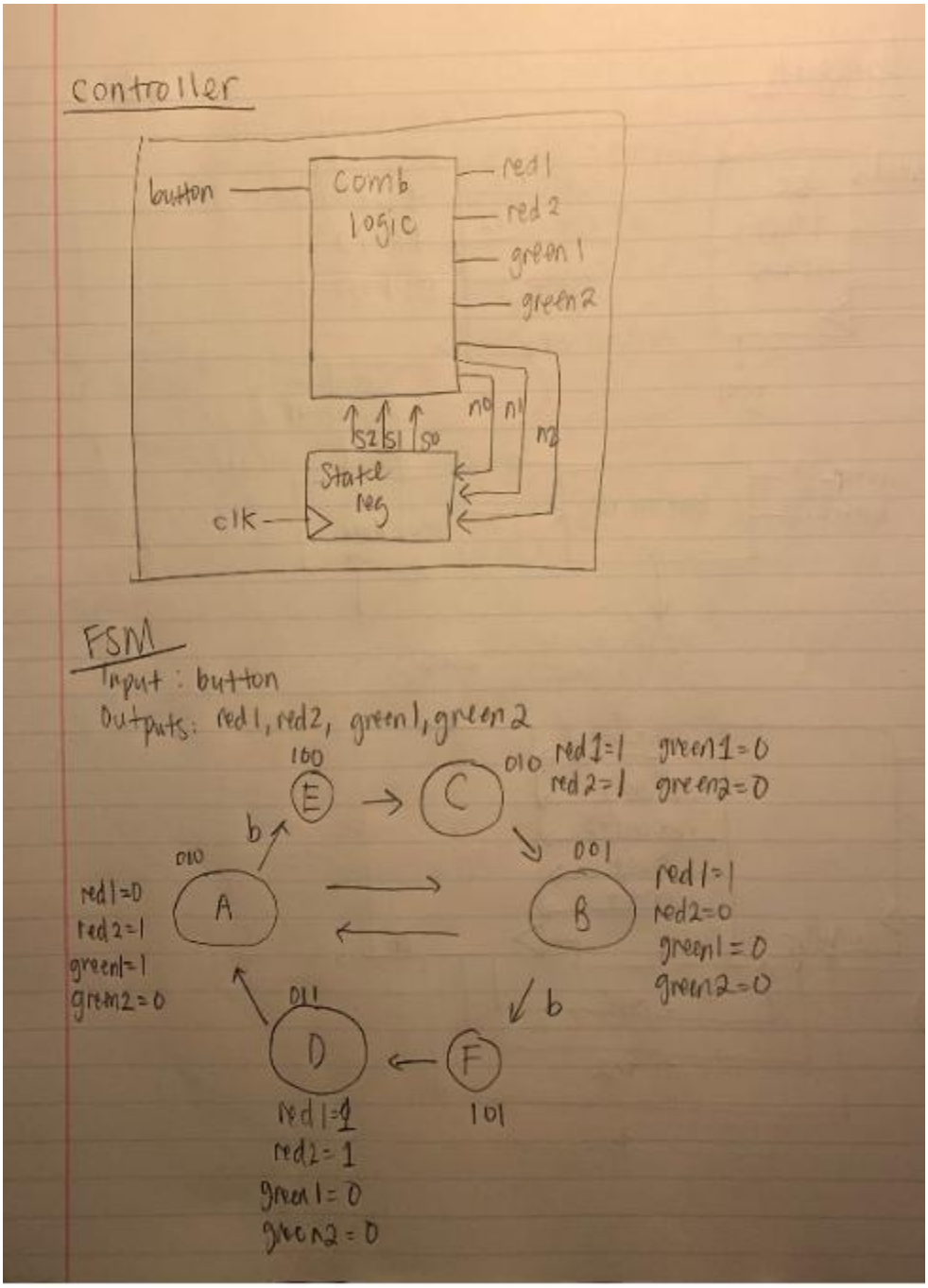
Purpose

The purpose is to make a four-way intersection traffic light using RTL design and Verilog. Some design constraint were that the traffic light only have two light for each way. The constraint is no yellow light and turn signals. The traffic lights is ON for 3 seconds. We need to also add a crosswalk button for pedestrians to cross the road. When you push the crosswalk button, all the traffic light must turn red after three seconds then it allows the pedestrians to cross. The crosswalk is ON for 5 seconds. We used 25MHz to calculate the ticks for the timer.

Circuit Design




Datapath
cnt_initial
cnt_rst
reset
clk
3sec timer
(32-bits)
timer
cnt_initial
cnt_reset2
reset2
clk
5sec timer2
(32-bits)
timer2
1
0
button_clr
button_ld
Button_reg
Q
I1
I0
2x1 MUX
Sel
button
cnt_initial
cnt_reset
cnt_reset2
reset
reset
controller
DP
clk
(25MHz)
timer
timer2
Q



https://youtu.be/ImMyQ0gpg90

```
module traffic#(
   parameter NBITS=32)
   (
   input clk,
   input button,
   input button2,
   output reg red1,
   output reg red2,
   output reg green1,
   output reg green2
    );

   wire timer;
   wire timer2;
   reg reset=1;
   reg reset2=1;
   wire [NBITS-1:0] cnt_ini;
   wire [NBITS-1:0] cnt_rst1;
   wire [NBITS-1:0] cnt_rst2;
   reg [2:0]current_state= 3'b000;
   reg [2:0]next_state=3'b000;
   reg button_reg = 1'b0;
   reg button2_reg = 1'b0;

always @(posedge clk) begin
current_state = next_state;
end
```

```
assign cnt_ini = 32'h0000;
assign cnt_rst1 = 32'h47868C0;
assign cnt_rst2 = 32'h7735940;

localparam A = 3'b000;
localparam B = 3'b001;
localparam C = 3'b010;
localparam D = 3'b011;
localparam E = 3'b100;
localparam F = 3'b101;
localparam G = 3'b110;
localparam H = 3'b111;

always @(posedge clk) begin
case (current_state)
A: begin
   reset= 0;
   red1 = 0;
   green1 = 1;
   red2 = 1;
   green2 =0;
   if (button == 1)
   begin
   button_reg = 1;
   reset=0;
   next_state = E;
   end
   else if (timer == 1)
   begin
```

```
    reset=1;
    next_state = B;
    end
end
    //else
    //next_state = A;
    //end

B: begin
    reset=0;
    red1 = 1;
    green1 = 0;
    red2 = 0;
    green2 = 1;
    if (button2 == 1)
    begin
    button2_reg = 1;
    reset=0;
    next_state = F;
    end
    else if (timer == 1)
    begin
    reset=1;
    next_state = A;
    end
end

C:begin
    reset2 = 0;
    red1 = 1;
    green1 = 0;
    red2 = 1;
    green2 = 0;
    if (timer2 == 1)
    begin
    reset = 1;
    reset2 = 1;
    next_state = B;
    end
    //else
    //begin
    //next_state=C;
    //end
end

D:begin
    reset2 = 0;
    red1 = 1;
    green1 = 0;
    red2 = 1;
    green2 = 0;
    if (timer2 ==1)
    begin
    reset = 1;
    reset2 = 1;
```

```
    next_state = A;
    end
end

E:begin
    reset = 0;
    red1 = 0;
    green1 = 1;
    red2 = 1;
    green2 = 0;
    if (timer == 1 && button_reg == 1)
    begin
    reset = 1;
    reset2 = 1;
    next_state = C;
    end
end

F:begin
    reset = 0;
    red1 = 1;
    green1 = 0;
    red2 = 0;
    green2 = 1;
    if (timer == 1 && button2_reg == 1)
    begin
    reset = 1;
    reset2 = 1;
    next_state = D;

    end
end

G:begin
    reset = 0;
    red1=1;
    green1=0;
    red2 = 0;
    green2=1;
    if (button == 1)
    begin
    button_reg = 1;
    reset=0;
    next_state = E;
    end
end

H: begin
    reset = 0;
    red1 = 0;
    green1 = 1;
    red2 = 1;
    green2 = 0;
    if (button2 == 1)
    begin
    button2_reg = 1;
    reset=0;
    next_state = F;
    end
```

```
end

   default: begin
   red1 = 0;
   green1 = 1;
   red2 =1;
   green2 =0;
   next_state = B ;
   end

   endcase
end

TIMER_ST #( .NBITS(NBITS) ) timerst (
.timer(timer),
.clk(clk),
.reset(reset) ,
.cnt_ini(cnt_ini),
.cnt_rst1(cnt_rst1)
);

TIMER_ST #( .NBITS(NBITS) ) timerst2 (
.timer(timer2),
.clk(clk),
.reset(reset2) ,
.cnt_ini(cnt_ini),
.cnt_rst1(cnt_rst2)
);
endmodule

module TIMER_ST #(
    parameter NBITS = 32
)
(
 output wire timer ,
 input wire clk ,
 input wire reset,
 input [NBITS-1:0] cnt_ini ,
 input [NBITS-1:0] cnt_rst1
);
wire [NBITS-1:0] q ;
wire [NBITS-1:0] qnext ;
// Compute the next value
adder #( .NBITS(NBITS ) )
 c1 (.q(q),
.cnt_ini(cnt_ini),
.cnt_rst1(cnt_rst1),
.nextq(qnext),
.tick(timer) );
// Save the next state
floprl #( .NBITS(NBITS ) )
 c2 (.clk(clk),
.reset(reset),
.cnt_ini(cnt_ini),
.nextq(qnext),
.q(q) );
endmodule
```

```
module adder#( parameter NBITS = 16 )(
input [NBITS-2:0] q ,
input [NBITS-2:0] cnt_ini ,
input [NBITS-2:0] cnt_rst1 ,
output[NBITS-2:0] nextq,
output tick
);
wire same ;
wire[NBITS-2:0] inextq;

addergen_st #(.NBITS(NBITS))
nextval ( .r(inextq), // Next value
 .cout(), // Carry out - Don't use
 .a(q), // Current value
 .b(16'b0000_0001), // Plus One
 .cin(16'b0000_0000) ) ; // No carry in

comparatorgen_st #(.NBITS(NBITS))
comparator (
.r(same) ,
.a(inextq),
.b(cnt_rst1) );

assign tick = (same) ? 'd1 : 'd0 ;
assign nextq = (same) ? cnt_ini : inextq ;
endmodule
```

```
module addergen_st #( parameter NBITS = 16 )(
output wire[NBITS-1:0] r ,
output wire cout ,
input wire[NBITS-1:0] a ,
input wire[NBITS-1:0] b ,
input wire cin ) ;
wire [NBITS:0] carry;
assign carry[0]= cin ;
genvar k ;
generate
for (k=0; k < NBITS; k = k+1)
begin : blk
 fulladder_st FA (
 .r(r[k]),
 .cout(carry[k+1]),
 .a(a[k]),
 .b(b[k]),
 .cin(carry[k]) ) ;
end
endgenerate
assign cout = carry[NBITS] ;
endmodule
```

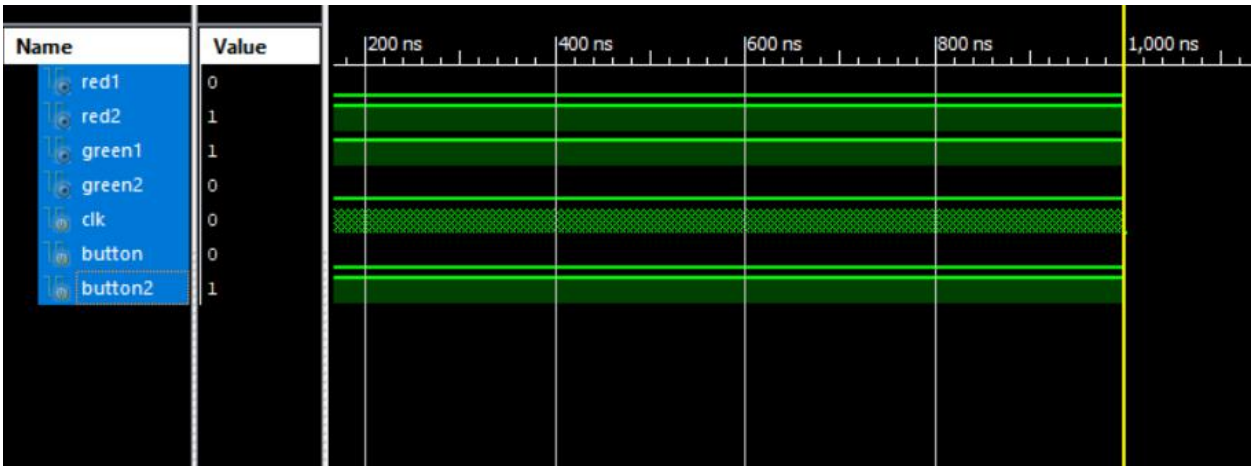
```
module fulladder_st(
output wire r,
output wire cout,
input wire a,
input wire b,
input wire cin
) ;
assign r = (a ^ b) ^ (cin) ;
assign cout = (a & b) | ( a & cin ) | ( b & cin ) ;
endmodule


module comparatorgen_st #( parameter NBITS = 16 )(
output wire r ,
input wire[NBITS-1:0] a ,
input wire[NBITS-1:0] b );
wire [NBITS-1:0] iresult ;
genvar k ;
generate
for (k=0; k < NBITS; k = k+1)
begin : blk
 xor cl (iresult[k], a[k], b[k] ) ;
end
endgenerate
// Reduction plus negation
assign r = ~(|iresult);
endmodule

module floprl#( parameter NBITS = 16 )(
 input clk,
input reset,
input [NBITS-2:0] cnt_ini,
input [NBITS-2:0] nextq,
output[NBITS-2:0] q
);
reg [NBITS-2:0] iq=0 ;
always @(posedge clk) begin
 if (reset) begin
iq <= cnt_ini ;
 end
 else begin
iq <= nextq;
 end
end
assign q = iq ;
endmodule
```

UCF:

```
// Inputs
NET "clk" LOC = "B8" ;
NET "button" LOC = "G12" ;
NET "button2" LOC = "M4";
// Outputs
NET "red1" LOC = "G1" ;
NET "green1" LOC = "P4" ;
NET "red2" LOC = "N4" ;
NET "green2" LOC = "N5" ;
```

Testbench:

```
module testbench;

   // Inputs
   reg clk;
   reg button;
   reg button2;

   // Outputs
   wire red1;
   wire red2;
   wire green1;
   wire green2;

   // Instantiate the Unit Under Test (UUT)
   traffic uut (
      .clk(clk),
      .button(button),
      .button2(button2),
      .red1(red1),
      .red2(red2),
      .green1(green1),
      .green2(green2)
   );

      always begin
      #2 clk = ~clk;
      end
   initial begin
      clk = 0;

       // Initialize Inputs
       button = 1;
       #3;

       button2 = 1;
       button = 0;
       #3;
    end

endmodule
```



Problems and Technical Issues

Problems and technical issues encountered were we try to add something where if someone or a pedestrian continues to push the crosswalk button, it will stay in that state. We couldn't get it to work, so we decided to scrap it. Another problem was that we wanted to add the extra button where the traffic lights become red whichever buttons is pressed, but instead we added the extra crosswalk button for their specific traffic light. We had a problem with the waveform from the test bench that looks incorrect because of the timer module.

Conclusion

Our project worked according to our specifications we designed our traffic light system. We were able to design the traffic light using our HLSM design and controller. The HLSM had six states and the data path design had two timer, one for the crosswalk and the traffic light. Some ways to improve the system is to add maybe yellow light and traffic light for the four way and not just two traffic lights. The most difficult part of the project is the coding.

TA problem

How many timers do we use in the project?

Ans:We use two timers.One is for timing 3s,the other one is for timing 5s.