CS – 223 Digital Design

Section – 2

Lab – 05

Erdem Ege Eroğlu

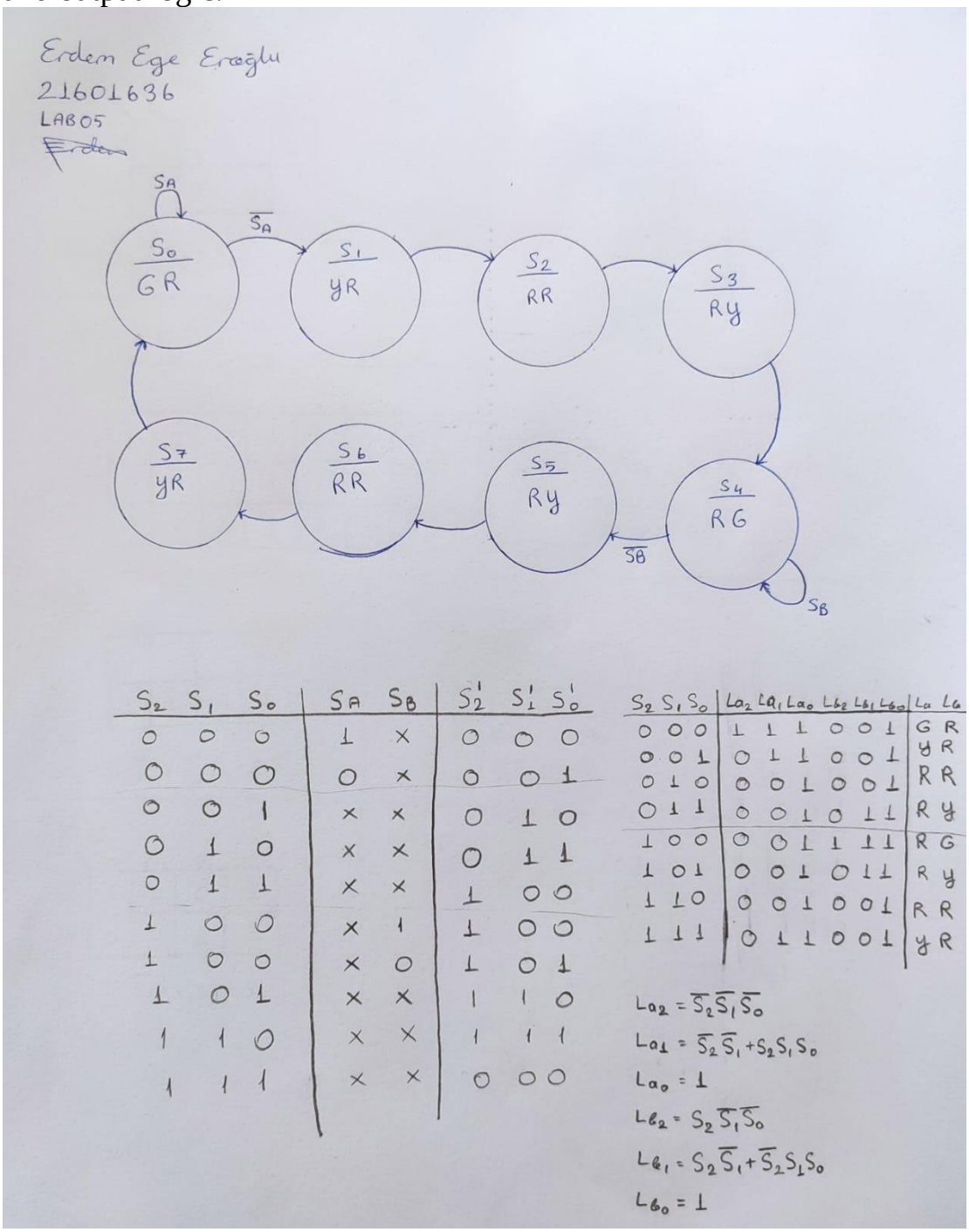
21601636

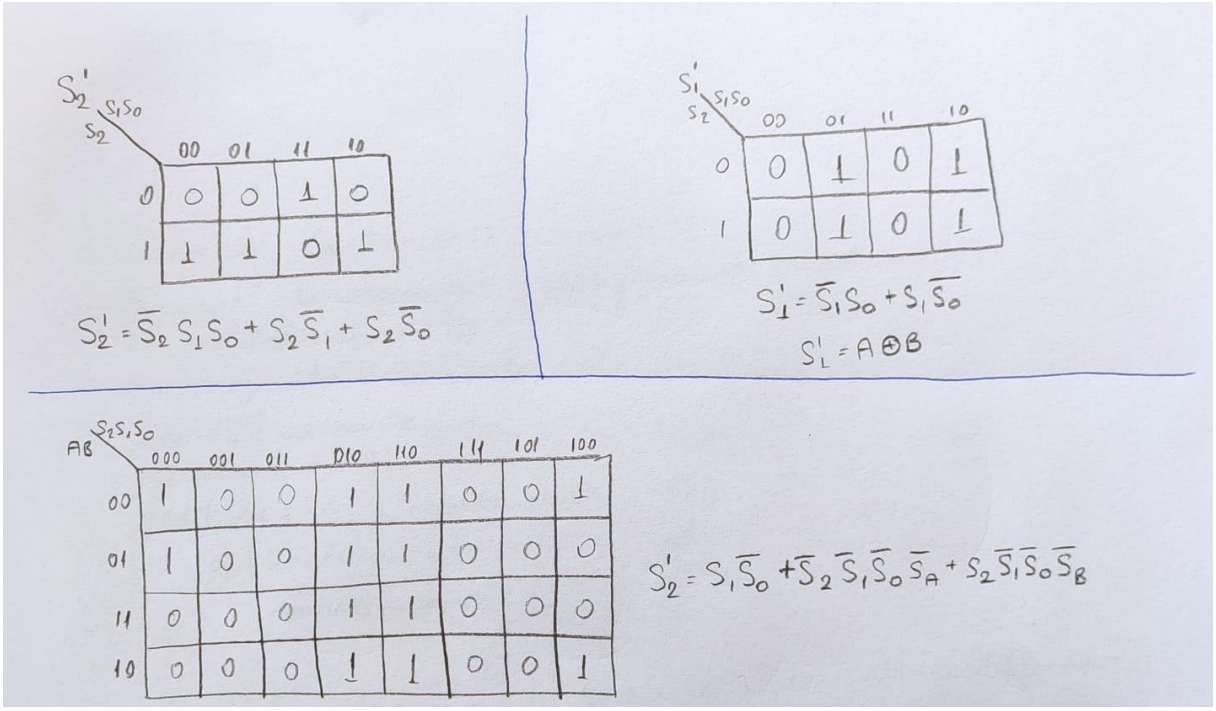
01.05.2020

(b) How many flip-flops do you need to implement this circuit?

3 flip-flops are necessary. Because 3 flip-flop can implement 2^3 = 8 states.

(c) Sketch the state transition diagram, write the state transition table and output table with binary encodings, and write the Boolean equations for the next-state and output logic.





(d) Write a SystemVerilog module for the FSM you designed in part (c) and a testbench for it.

```
`timescale 1ns / 1ps
module trafficLightFsm(input logic clk,
                       input logic rst,
                       input logic Sa, Sb,
                       output logic[2:0] La, Lb);
   parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b100,S5 = 3'b101,S6 = 3'b110, S7 = 3'b111;
   parameter Green = 3'b111, Yellow = 3'b011, Red = 3'b001;
   logic[2:0] currS, nextS;

   always_ff @(posedge clk)
   if(rst)   currS <= S0;
   else    currS <= nextS;
```

```
always_comb
    case(currS)
    S0:
      if(Sa)
          nextS = S0;
      else
          nextS = S1;

    S1:
      nextS = S2;

    S2:
      nextS = S3;

    S3:
      nextS = S4;

    S4:
      if(Sb)
          nextS = S4;
      else
          nextS = S5;

    S5:
      nextS = S6;

    S6:
      nextS = S7;

    S7:
      nextS = S0;
```

```
    default:
    nextS = S0;
endcase

always_ff @(currS)
   case(currS)
      S0:
         begin
            La <= Green;
            Lb <= Red;
         end
      S1:
         begin
            La <= Yellow;
            Lb <= Red;
         end
      S2:
         begin
            La <= Red;
            Lb <= Red;
         end
      S3:
         begin
            La <= Red;
            Lb <= Yellow;
         end
      S4:
         begin
            La <= Red;
            Lb <= Green;
         end
```

```
        S5:
            begin
                La <= Red;
                Lb <= Yellow;
            end
        S6:
            begin
                La <= Red;
                Lb <= Red;
            end
        S7:
            begin
                La <= Yellow;
                Lb <= Red;
            end
    endcase

endmodule


module trafficTestbench();
    logic clk, reset, Sa, Sb;
    logic[2:0] La, Lb;

    trafficLightFsm dut(clk, reset, Sa, Sb, La, Lb);

    always begin
        clk <= 0;
        #10;
        clk <= 1;
        #10;
```

```
    end

    initial begin
        reset <= 1;
        Sa <= 0;
        Sb <= 0;
        @(posedge clk);
        #5
        reset <= 0;
        @(posedge clk);
        #5
        Sa <= 0;
        Sb <=1;
        @(posedge clk);
        #5
        Sa <= 1;
        Sb <= 0;
        @(posedge clk);
        @(posedge clk);
        @(posedge clk);
    end
endmodule
```