Practice Problem (2)

Design an electronic lock system (**lockSys**, see Figure 1) for a garage door lock. The electronic lock accepts a 3-digit user code input, one digit at a time. If the input code sequence exactly matches 010, the electronic lock is opened (**openLock** is asserted). If any part of the 3-digit code input sequence is incorrect, the **Alarm** is activated and the user is forced to restart code entry i.e. all backward arcs go back to the initial state if a "wrong" digit is entered. When **openLock** is asserted after the correct code has been entered, the lock stays open. And the lock shuts itself (**openLock** is de-asserted) if the asynchronous input signal **rst** is asserted or a 0/1 is entered after the correct input code sequence, until the correct input code is again entered.

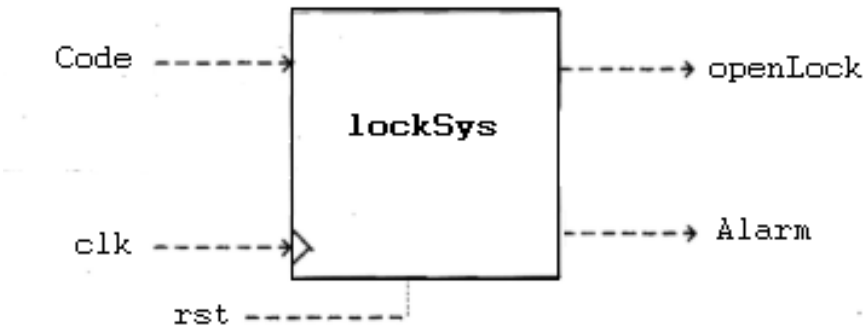


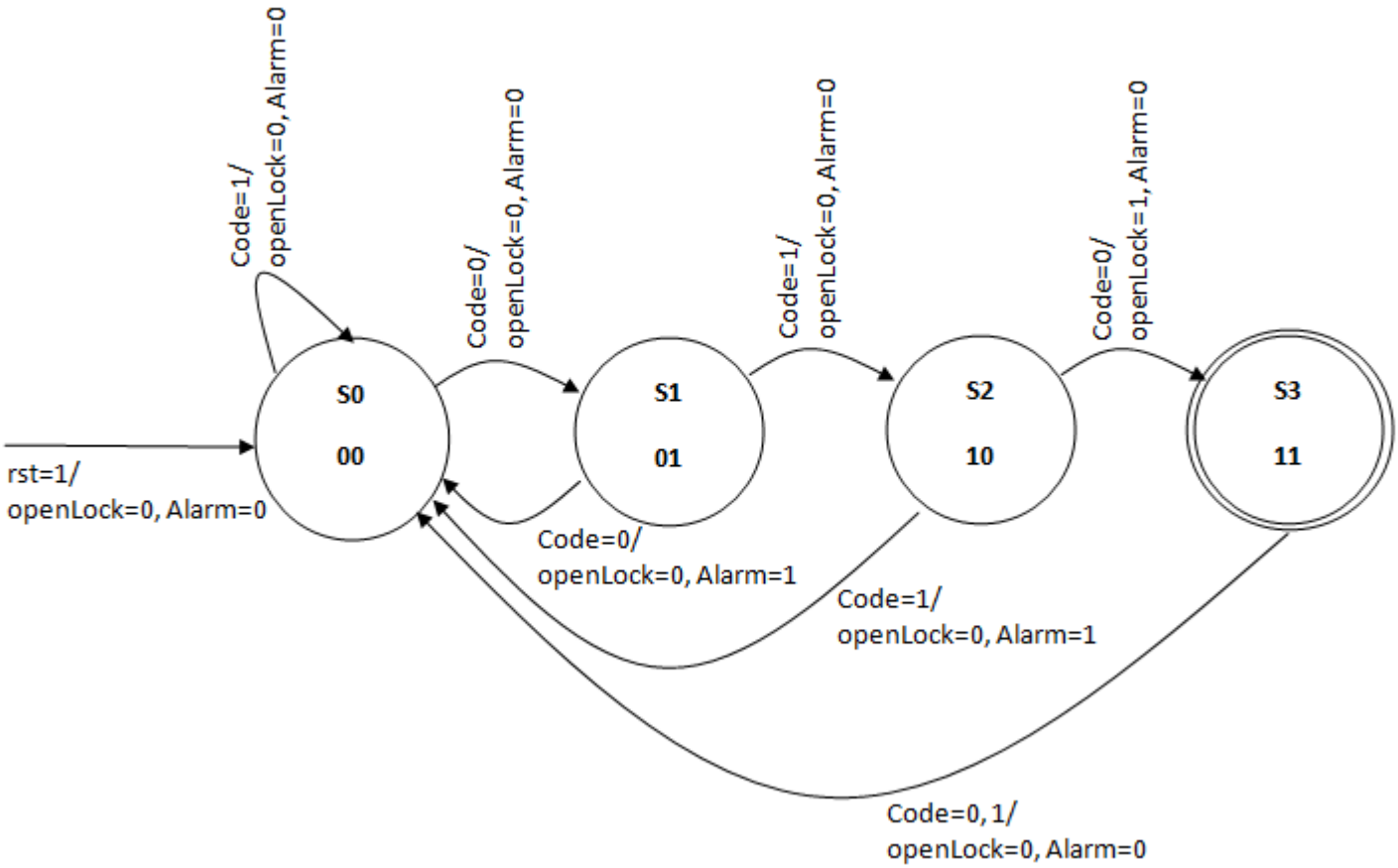
Figure 1. Electronic Lock System (lockSys)

(a) Design a Mealy/Moore FSM for **lockSys**.

(b) Implement the FSM in **(a)** in Verilog.

Solution:

(a)


Code=1/
openLock=0, Alarm=0
Code=0/
openLock=0, Alarm=0
Code=1/
openLock=0, Alarm=0
Code=0/
openLock=1, Alarm=0
S0
00
S1
01
S2
10
S3
11
rst=1/
openLock=0, Alarm=0
Code=0/
openLock=0, Alarm=1
Code=1/
openLock=0, Alarm=1
Code=0, 1/
openLock=0, Alarm=0

(b)

```
module sysLock (Code, clk, rst, openLock, Alarm);

     input Code, clk, rst;
     output openLock, Alarm;
     reg openLock, Alarm;

     parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3;
     reg [1:0] state;

     always @(posedge clk, rst)
          if (rst) begin
               openLock <= 0;
               Alarm <= 0;
               state <= S0;
          end
          else
               case (state)
                    S0: begin
                         openLock = Code?0:0;
                         Alarm = Code?0:0;
                         state = Code?S0:S1;
                    end
                    S1: begin
                         openLock = Code?0:0;
                         Alarm = Code?0:1;
                         state = Code?S2:S0;
                    end
                    S2: begin
                         openLock = Code?0:1;
                         Alarm = Code?1:0;
                         state = Code?S0:S3;
                    end
                    S3: begin
                         openLock = Code?0:0;
                         Alarm = Code?0:0;
                         state = Code?S0:S0;
                    end
               endcase

endmodule
```

```
module tst_sysLock;

      reg Code, clk, rst;
      wire openLock, Alarm;

      sysLock lock (Code, clk, rst, openLock, Alarm);

      initial   begin
            clk = 0;
            #10 Code = 1;
            #20 Code = 0;
            #20 Code = 1;
            #20 Code = 0;
            #20 Code = 1;
            #20 Code = 0;
            #20 Code = 0;
            #20 Code = 1;
            #20 Code = 0;
            #20 Code = 1;
            #20 Code = 0;
      end

      initial   begin
            rst = 1;
            #10 rst = 0;
      end

      always
            #10 clk = ~clk;

endmodule
```