Digital Locker

Digital Design Course Project

EEL2020

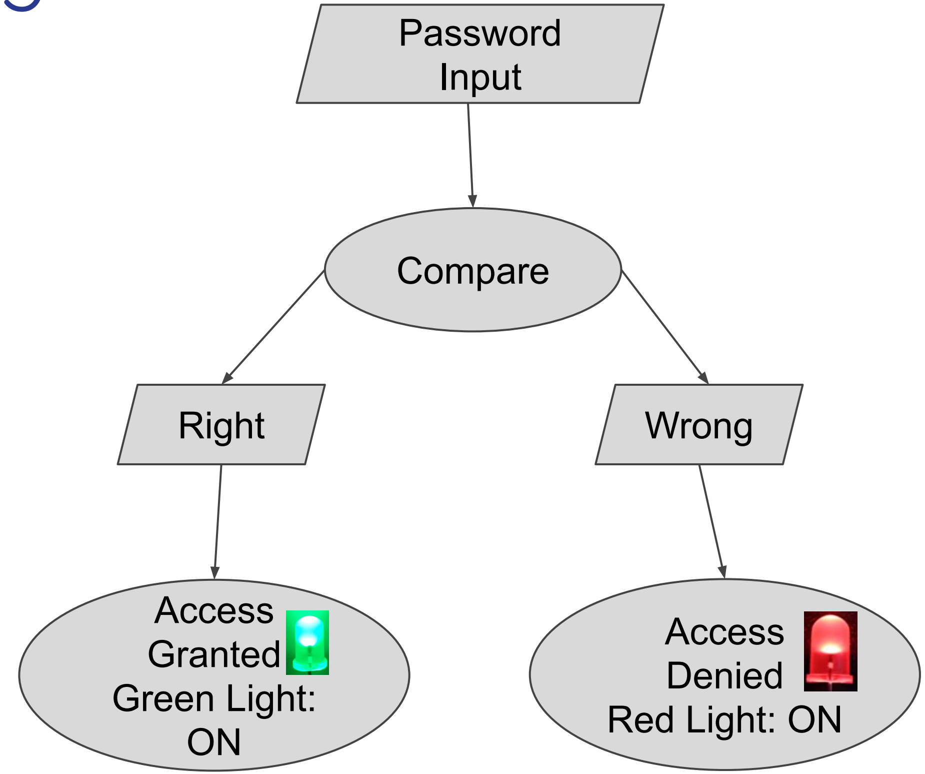
About the Project

Our project digital locker aims to build a digital lock that can be unlocked by a 4 digit password. This user can input the password and based on the input the locker will give a signal i.e., the locker led will turn green if the password is correct and locker led will turn if the password is wrong.

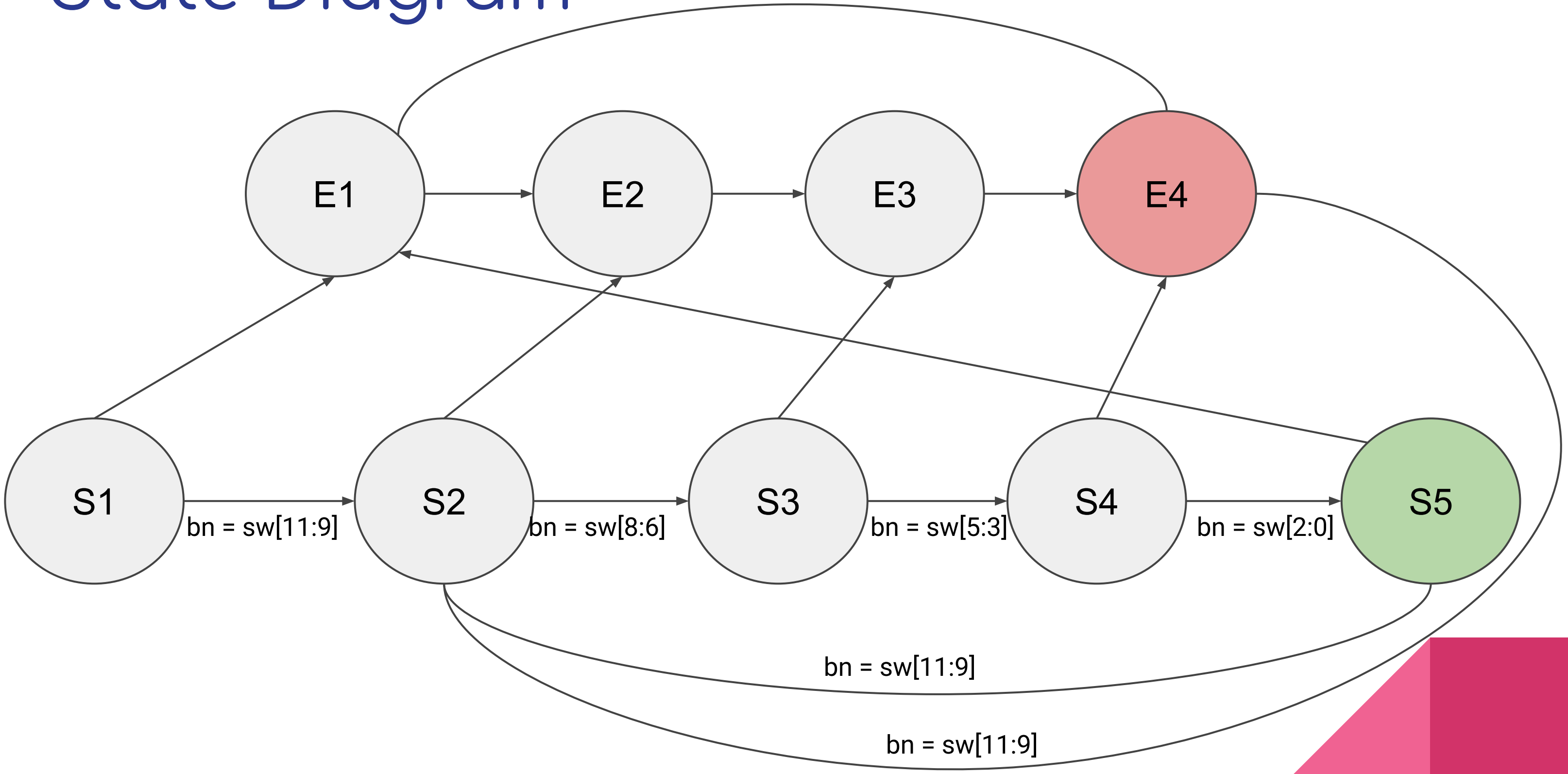
Features

1.) User can enter the password.
2.) The red light will glow if the entered password is wrong.
3.) The green light will glow if the entered password is correct.

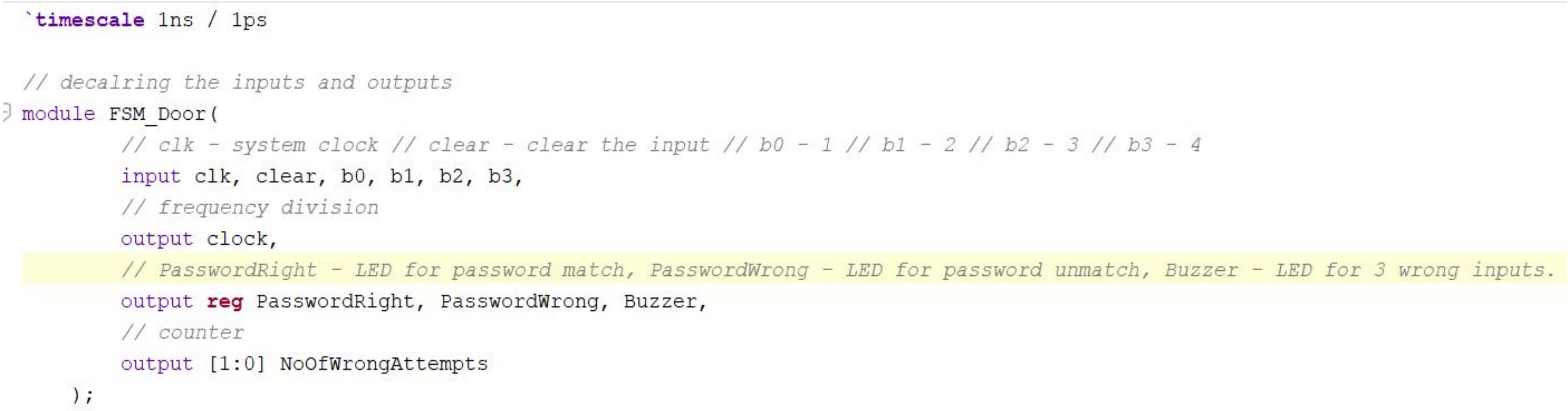
Block Diagram



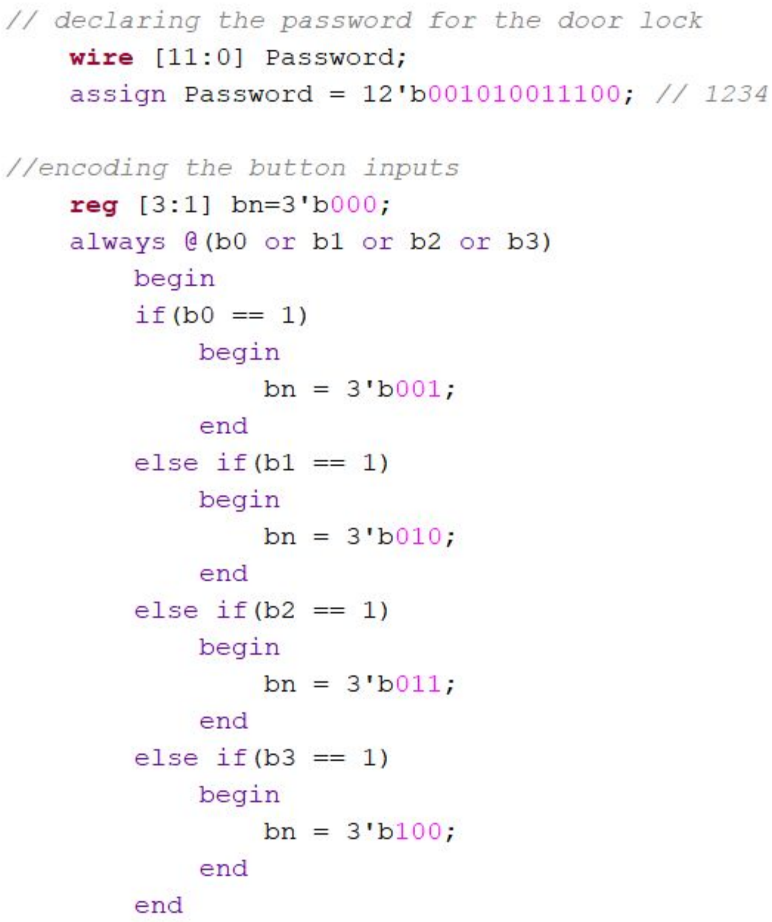
State Diagram



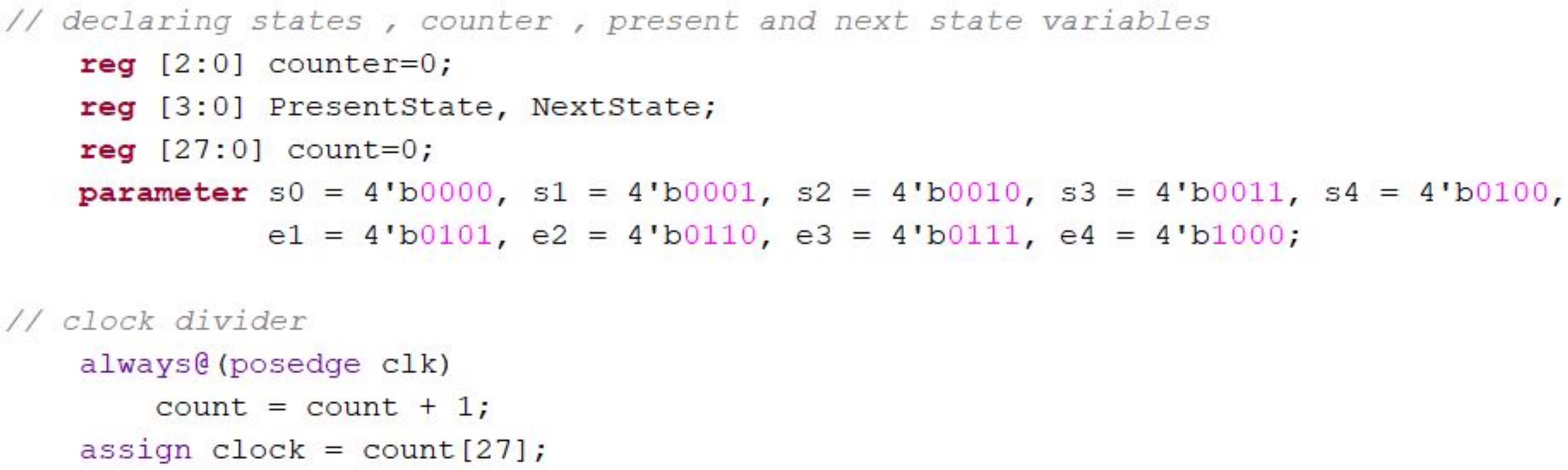
Code snippets



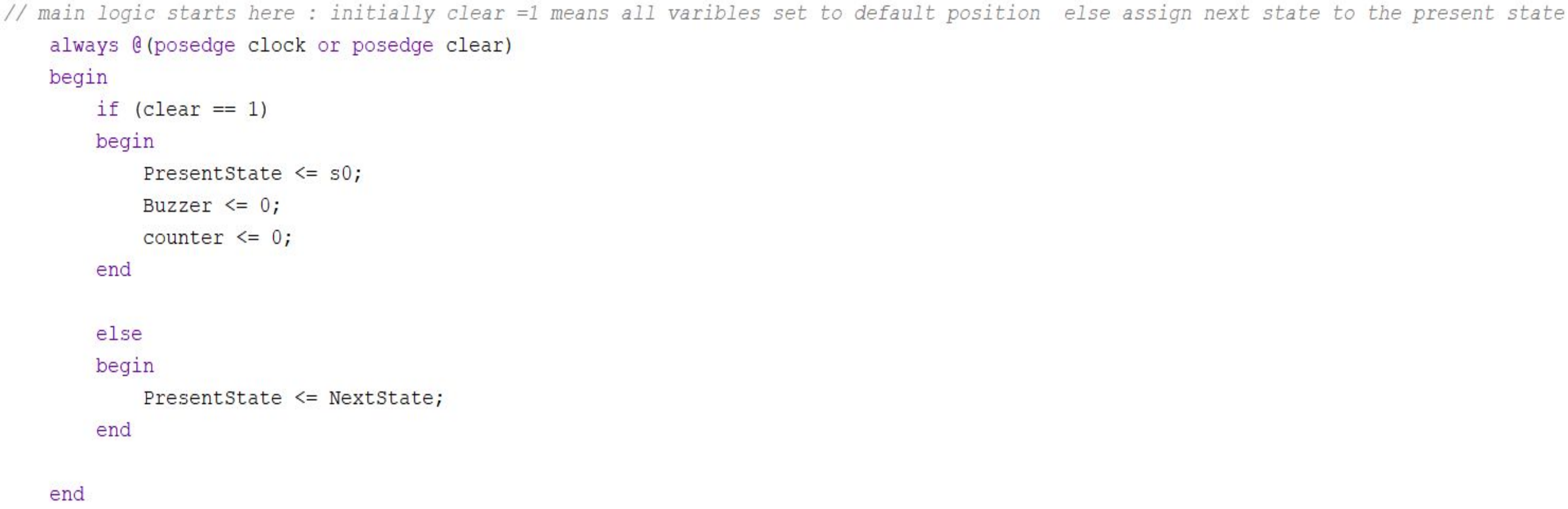
```
`timescale 1ns / 1ps

// decalring the inputs and outputs
module FSM_Door(
        // clk - system clock // clear - clear the input // b0 - 1 // b1 - 2 // b2 - 3 // b3 - 4
        input clk, clear, b0, b1, b2, b3,
        // frequency division
        output clock,
        // PasswordRight - LED for password match, PasswordWrong - LED for password unmatch, Buzzer - LED for 3 wrong inputs.
        output reg PasswordRight, PasswordWrong, Buzzer,
        // counter
        output [1:0] NoOfWrongAttempts
    );
```



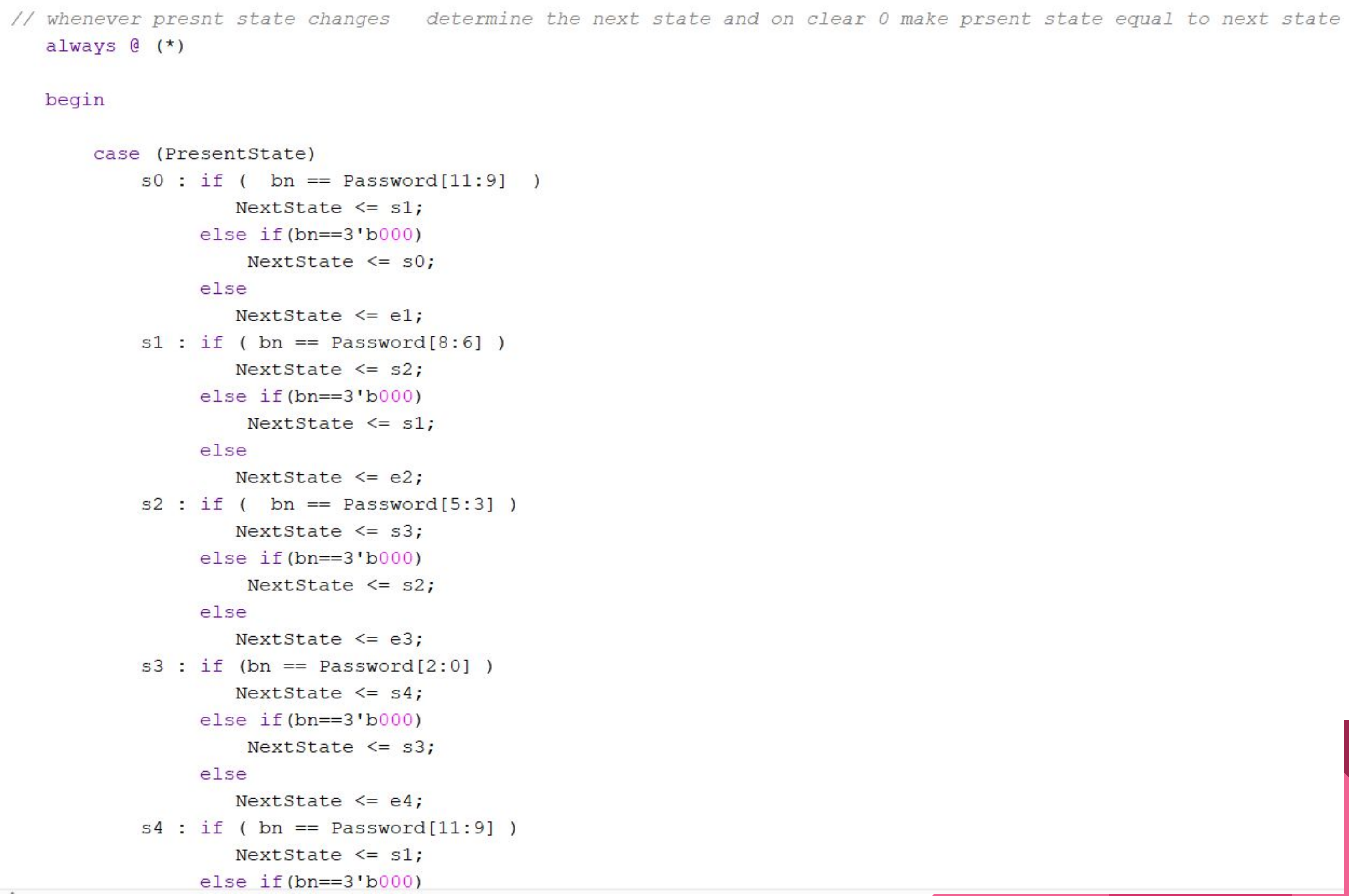
```
// declaring the password for the door lock
    wire [11:0] Password;
    assign Password = 12'b001010011100; // 1234

//encoding the button inputs
    reg [3:1] bn=3'b000;
    always @(b0 or b1 or b2 or b3)
        begin
        if(b0 == 1)
            begin
                bn = 3'b001;
            end
        else if(b1 == 1)
            begin
                bn = 3'b010;
            end
        else if(b2 == 1)
            begin
                bn = 3'b011;
            end
        else if(b3 == 1)
            begin
                bn = 3'b100;
            end
        end
```



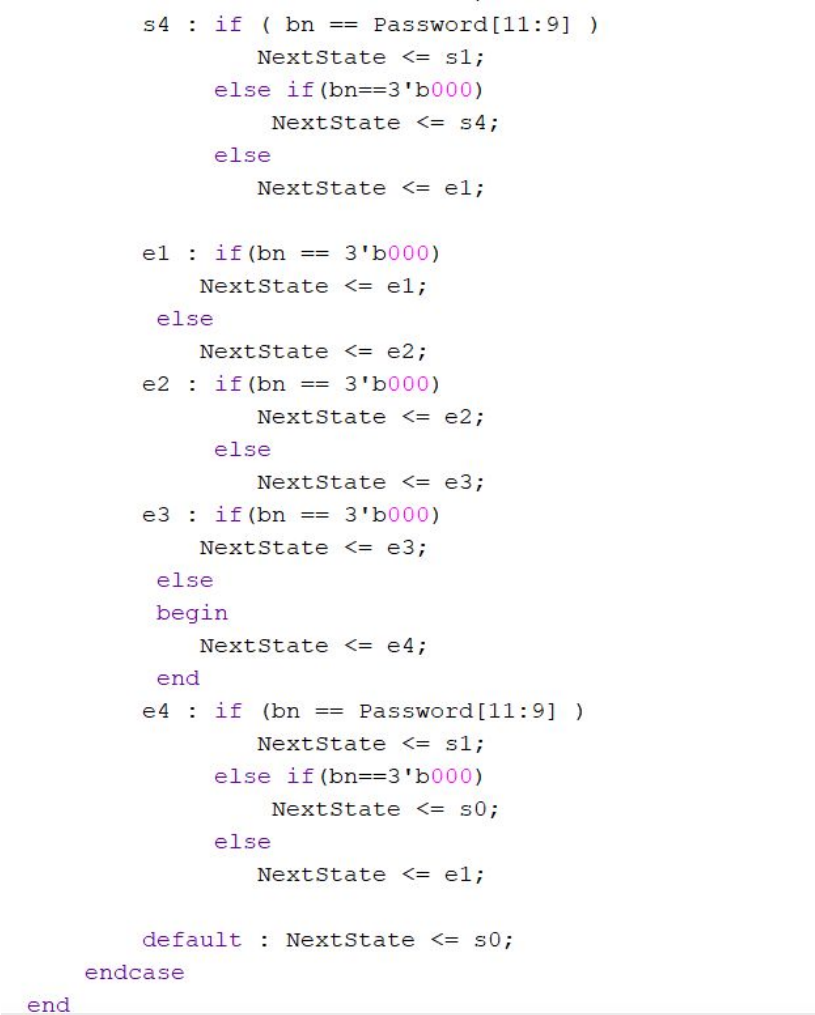
```verilog
// declaring states , counter , present and next state variables
    reg [2:0] counter=0;
    reg [3:0] PresentState, NextState;
    reg [27:0] count=0;
    parameter s0 = 4'b0000, s1 = 4'b0001, s2 = 4'b0010, s3 = 4'b0011, s4 = 4'b0100,
              e1 = 4'b0101, e2 = 4'b0110, e3 = 4'b0111, e4 = 4'b1000;

// clock divider
    always@(posedge clk)
        count = count + 1;
    assign clock = count[27];
```



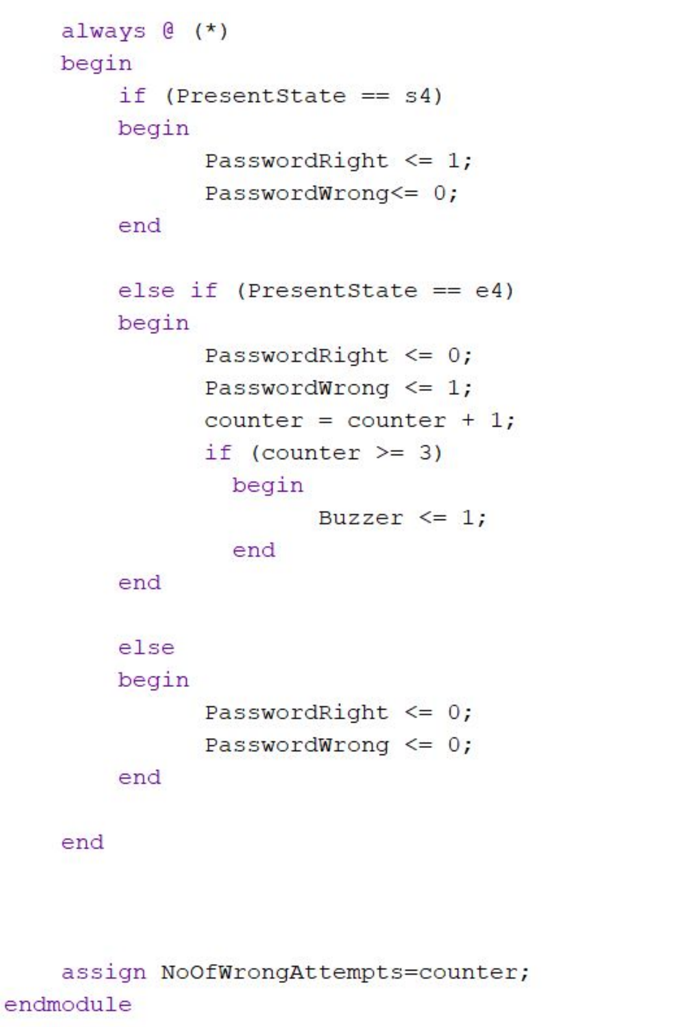
```
// main logic starts here : initially clear =1 means all varibles set to default position  else assign next state to the present state
    always @(posedge clock or posedge clear)
    begin
        if (clear == 1)
        begin
            PresentState <= s0;
            Buzzer <= 0;
            counter <= 0;
        end

        else
        begin
            PresentState <= NextState;
        end

    end
```



```
// whenever presnt state changes   determine the next state and on clear 0 make prsent state equal to next state
   always @ (*)

   begin

       case (PresentState)
           s0 : if (  bn == Password[11:9]  )
                   NextState <= s1;
                else if(bn==3'b000)
                    NextState <= s0;
                else
                   NextState <= e1;
           s1 : if ( bn == Password[8:6] )
                   NextState <= s2;
                else if(bn==3'b000)
                    NextState <= s1;
                else
                   NextState <= e2;
           s2 : if (  bn == Password[5:3] )
                   NextState <= s3;
                else if(bn==3'b000)
                    NextState <= s2;
                else
                   NextState <= e3;
           s3 : if (bn == Password[2:0] )
                   NextState <= s4;
                else if(bn==3'b000)
                    NextState <= s3;
                else
                   NextState <= e4;
           s4 : if ( bn == Password[11:9] )
                   NextState <= s1;
                else if(bn==3'b000)
```



```
        s4 : if ( bn == Password[11:9] )
                NextState <= s1;
             else if(bn==3'b000)
                 NextState <= s4;
             else
                NextState <= e1;

        e1 : if(bn == 3'b000)
            NextState <= e1;
         else
            NextState <= e2;
        e2 : if(bn == 3'b000)
                NextState <= e2;
             else
                NextState <= e3;
        e3 : if(bn == 3'b000)
            NextState <= e3;
         else
         begin
            NextState <= e4;
         end
        e4 : if (bn == Password[11:9] )
                NextState <= s1;
             else if(bn==3'b000)
                 NextState <= s0;
             else
                NextState <= e1;

        default : NextState <= s0;
    endcase
end
```



```
    always @ (*)
    begin
        if (PresentState == s4)
        begin
              PasswordRight <= 1;
              PasswordWrong<= 0;
        end

        else if (PresentState == e4)
        begin
              PasswordRight <= 0;
              PasswordWrong <= 1;
              counter = counter + 1;
              if (counter >= 3)
                begin
                      Buzzer <= 1;
                end
        end

        else
        begin
              PasswordRight <= 0;
              PasswordWrong <= 0;
        end

    end



    assign NoOfWrongAttempts=counter;
endmodule
```

Implementation on Board