Mini Project on

MULTI-USER LOCK SYSTEM

SUBMITTED BY A11

204141-MANASI.
204142-AKSHITH.
204143-M.PRIYANKA.
204144-M.HARSHA VARDHAN.

Under the guidance of

Dr. HANUMANTH RAO T V K

&

Dr. Prithvi Pothupogu

ABSTRACT

The project deals with the simulation of a multi user login based security system using verilog VHDL with xilinx.

Multi user login based security system is secure and safe as it allows only specific users in the team to allow access to its contents and it is easy to design and works efficiently and easily in private cabins or in server rooms where only few are allowed.

INDEX

INTRODUCTION

The main objective of this project is to design a lock with a minimum cost having reliable security. It's easy to design, cost effective and it's easy to use, less power consuming and so much dependable on accuracy. This lock system can be used by multiple users. We have implemented this lock for four users to increase the security . We used a three digit hexadecimal number as our password in this design. This password gives us a total of 4096 possible combinations for each user without making the user memorize a long password. If a user gives the wrong input for three times continuously then the alarm rings and it locks the door until the admin presses the reset button for comparing the set password. We used a simple bit by bit comparator which compares every bit of the output which comes from the user input in Hexadecimal Format. If the Input Password matches the set password for that particular user then the user is allowed. For any wrong input password , the comparator will trigger the counter and no of will be increased. We also Mentioned the usage and applications of this lock system along with some more features that could be added to make our system more secure and user reliable.

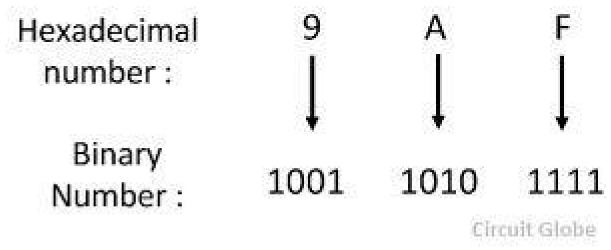
THEORY

HEXADECIMAL TO BINARY CONVERSION

To convert a hexadecial number to a binary number, convert each hexadecimal digit to its

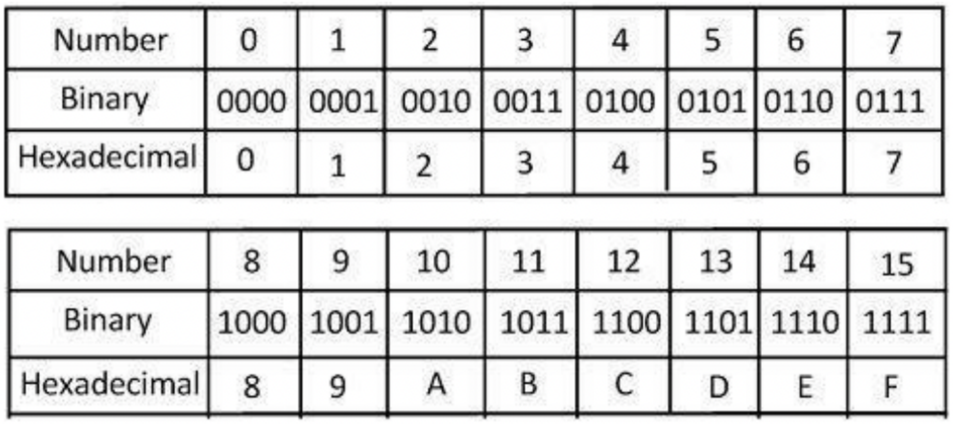
four-digit equivalent binary number. For example, consider the hexadecimal number 9AF

which is converted into a binary digit.The conversions are explained below.



Therefore, the equivalent binary number is 1001 1010 1111.

The number conversion table below.



Number	0	1	2	3	4	5	6	7
Binary	0000	0001	0010	0011	0100	0101	0110	0111
Hexadecimal	0	1	2	3	4	5	6	7

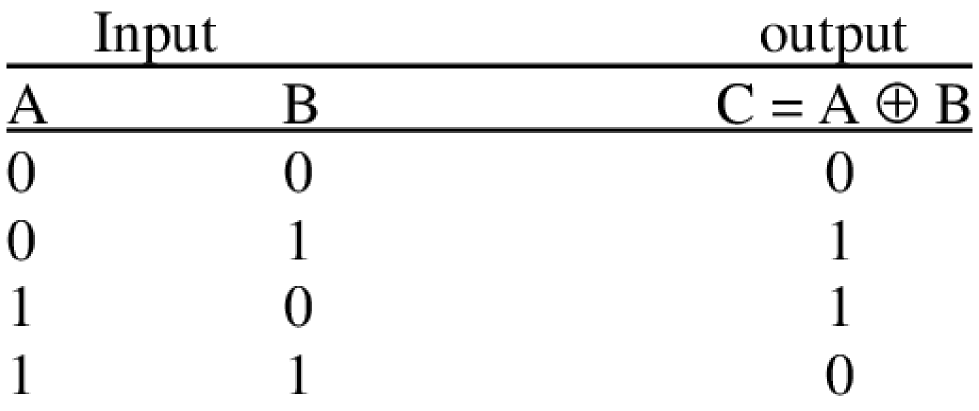
Number	8	9	10	11	12	13	14	15
Binary	1000	1001	1010	1011	1100	1101	1110	1111
Hexadecimal	8	9	A	B	C	D	E	F

Figure 2: number system conversion table

BITWISE XOR OPERATION

If two numbers are given A and B, We can perform the XOR operation between them to find whether they are equal or not.

Example: Let A=111 and B=110 are two 3-bit binary numbers. If we perform A XOR B then each bit in A and B are XORED. And Output will be 001.



Input		output
A	B	$C = A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table of XOR operation

If the preset password is the same as the input password then output will be zero. Else It will be other than Zero . We used This concept to find whether the input password matches the preset password.

Analyzing the XOR operation

If the Preset Password and Input password are matched, the Check variable will be Zero and It is passed to the Counter module which makes the Access possible for the User and If the user makes less than 3 wrong attempts ,It will not make the alarm Ring.

If the Preset Password and Input password are not matched, the Check variable Will a Non-zero and It is passed to the Counter module which makes the Access Denied and Increases the Count of wrong attempts. If the user makes less than 3 wrong attempts Then the Count will be increased (He/She will be given a chance) . If the User did more than 3 Wrong attempts) then the password match or mismatch doesn't matter, It always turns on the Alarm and Intruder will be caught and Alarm can be reset by admin with a reset button.

Counter with Reset

We used a Counter module which will count upto 3 and it stops counting and holds the value 3 till we reset the counter. when the wrong attempt is taken the counter will increment the previous count by 1. And we used an asynchronous reset (the counter will reset for whatever the clock signal is) . It is a forced reset option. The forced reset option is there to reset the counter to working well again after 3 wrong attempts by the admin and the alarm will be off if after less than 3 wrong attempts if 1 right attempt is performed then the counter should be reset and start to form the beginning again. After 3 wrong attempts, when the count is 3, The Alarm will turn on.

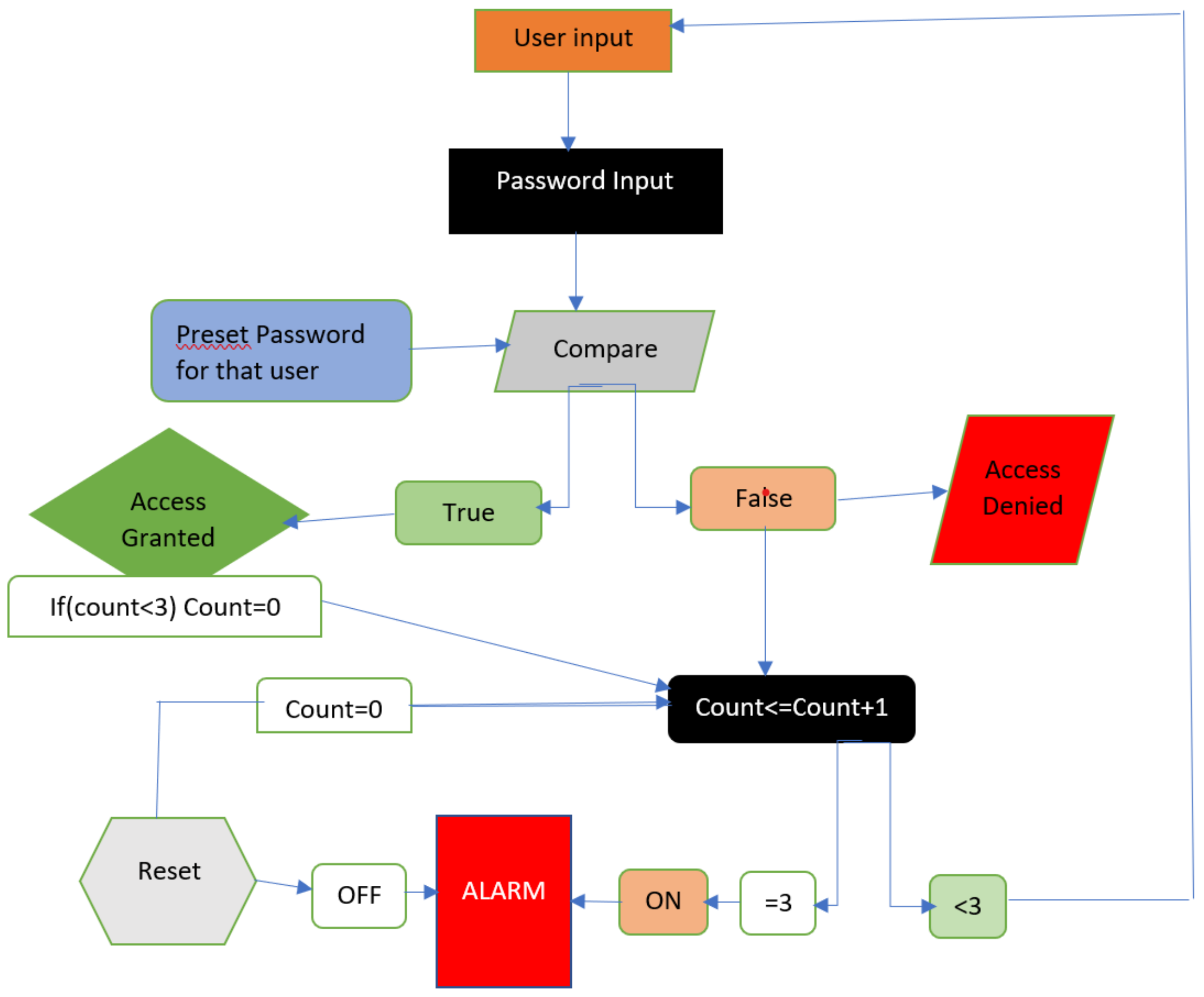
Verilog Coding

Verilog is used here to cross check our logic for our lock. We took the present password ,input password,the enter function and Reset as inputs in the main module.Access state,alarm and count as outputs.firstly, we used XOR to check whether the input password matches with the present password.if they match perfectly,0 is stored in Check .Otherwise a the XORED result is stored.

Here we send the entire function has clock as we need to check passwords whenever a new password is typed in and enter button is pressed.As we set our lock to buzz the alarm when count reaches 3,we need a mod 4 counter And the max value to be counted is 3 in binary 11 ,Only two digits length is enough.Now we have to check the outputs only when a password is entered or whenever reset is entered .

Now, independent of anything whenever a reset is pressed, we set everything to default i.e alarm is turned off if necessary, count is equal to zero. Else incase of a new wrong password meaning check is equal to 0, enable is equal to 1, while the count from previous cycle was 2, alarm is buzzed, access is off, count is increased by 1(count is now equal to 3). Now if count is equal to 3, in case of no reset, access is always off, alarm stays on. But, while count is not equal to 2 or 3, a wrong password is entered meaning enable is 1, count is increased by 1, access is 0, but alarm isn't turned on.Finally, if count is not equal to 3, and a correct password is given as an input, then check is equal to 1 and enable is equal to 0, access is granted, count is set to zero. Alarm default.

Flowchart



Explaination

1. Firstly , a Password is set for a user (Preset Password).
2. Then a password is taken as an input from outside (input Password).
3. Then compare them
4. If they are matched, then access is granted.
5. If they aren’t matched ,then access is denied, count is increased by 1.
6. If count < 3, then repeat from (2). [a password match sets count = 0]
7. If count = 3, it sets off an alarm and holds “Access Denied” state.
8. Now, for any input (even if correct Password), alarm stays on, access is not granted.
9. If and only if Reset is pressed (by Owner), only then the alarm turns off, sets count = 0.
10. Then repeat from (2).

Verilog Code

```
module project(reset,Enter,InputPass,user,Access,Setpass,Count,Alarm);
    input reset,Enter;
    input [11:0] InputPass;
    input [1:0] user;
    output Access,Alarm;
    output [1:0] Count;
    output reg [11:0]Setpass;
    wire [11:0] Check;


always@*
    begin
        if(user==2'b00)
        begin
            Setpass=12'hAB3;
        end
        else if(user==2'b01)
        begin
            Setpass=12'hF2A;
        end
        else if(user==2'b10)
        begin
            Setpass=12'hE93;
        end
        else if(user==2'b11)
        begin
            Setpass=12'h111;
        end
    end


    assign Check=(Setpass^InputPass);

    counter c1(Enter,Check,reset,Count,Access,Alarm);
endmodule

module counter(clk,E,rstn,cnt ,Access,Alarm);
    parameter N=4;
    input clk,rstn;
    input [11:0] E;
    output reg Access,Alarm;
    output reg  [1:0] cnt;
    initial begin
    cnt=0;
    Alarm=0;
    Access=0;
    end
    always @(posedge clk or posedge rstn)
        begin
```

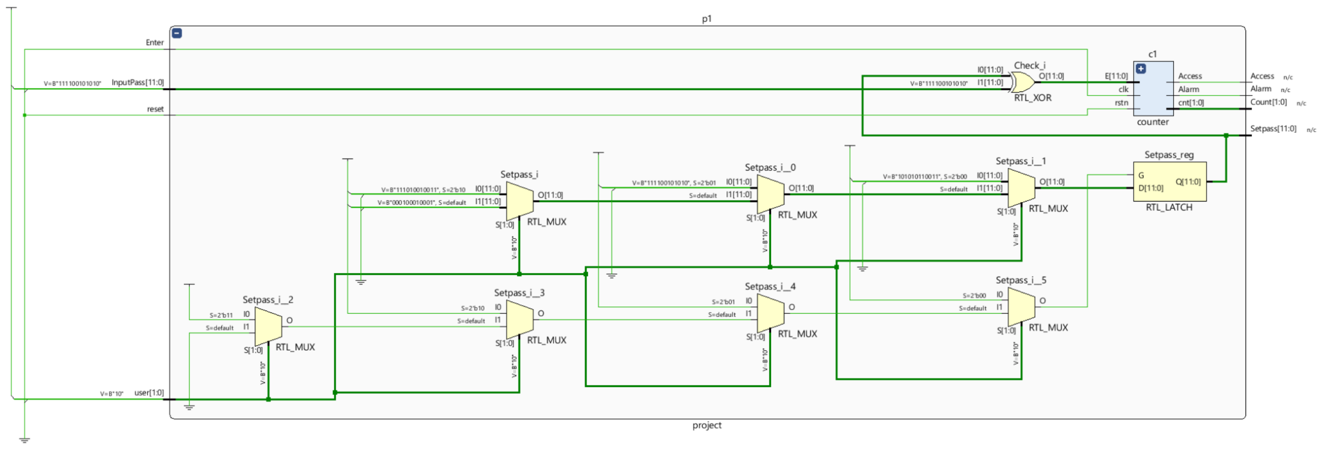
```
if(E==12'h0) begin
  Access<=1;
  Alarm=0;
  cnt<=0;
end
else if(E!=12'h0 & cnt==0)
  begin
  cnt<=cnt+1;
  Access<=0;
  Alarm<=0;
  end
else if(E!=12'h0)
  begin
  if(cnt==1)
      begin
      Alarm<=0;
      Access<=0;
      cnt<=cnt+1;
      end
  if(cnt==2)
      begin
      Alarm<=1;
      Access<=0;
      cnt<=cnt+1;
      end
  if(cnt==3)
      begin
      Alarm<=1;
      Access<=0;
```

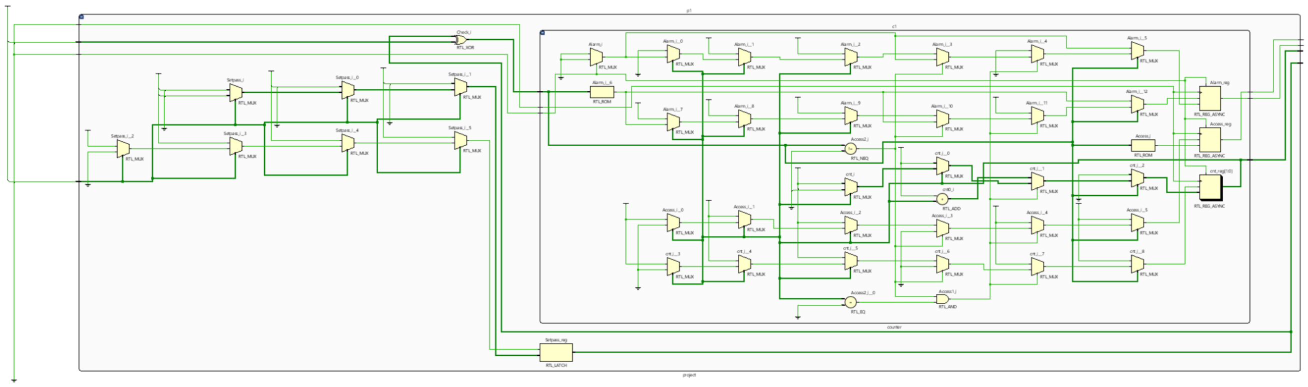
```
                cnt<=3;
                end
            end
          if(rstn==1)
            begin
            Alarm<=0;
            cnt<=0;
            Access<=0;
            end
        end
endmodule

module tb();
    reg [11:0] PassIn;
    reg reset,Enter;
    reg [1:0] user;
    wire [11:0] Setpass;
    wire Access,Alarm;
    wire [1:0] Count;
    project p1(reset,Enter,PassIn,user,Access,Setpass,Count,Alarm);
    initial
    begin
      user=2'b11;
        PassIn=12'h111;
        Enter=1;
        reset=0;
        #1;
        Enter=0;
        #99;
```

```
PassIn=12'h0AA;
Enter=1;
#1;
Enter=0;
#99;
PassIn=12'hF1A;
Enter=1;
#1;
Enter=0;
#99;
PassIn=12'h999;
Enter=1;
#1;
Enter=0;
#99;
PassIn=12'h001;
Enter=1;
#1;
Enter=0;
#99;
PassIn=12'h188;
reset=1;
Enter=1;
#10;
reset=0;
#1;
Enter=0;
#99;
PassIn=12'h111;
```

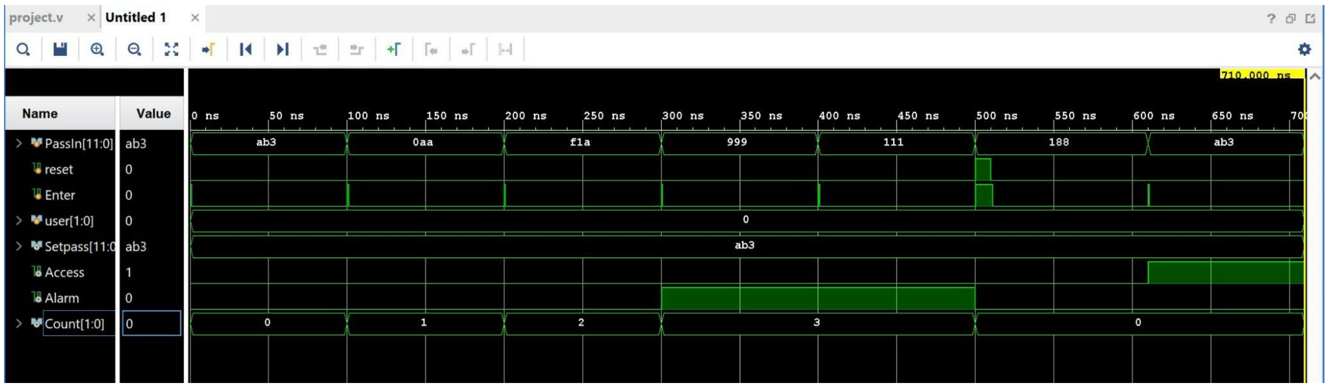
```
        Enter=1;
        #1;
        Enter=0;
        #99;
        $finish;

    end
Endmodule
```

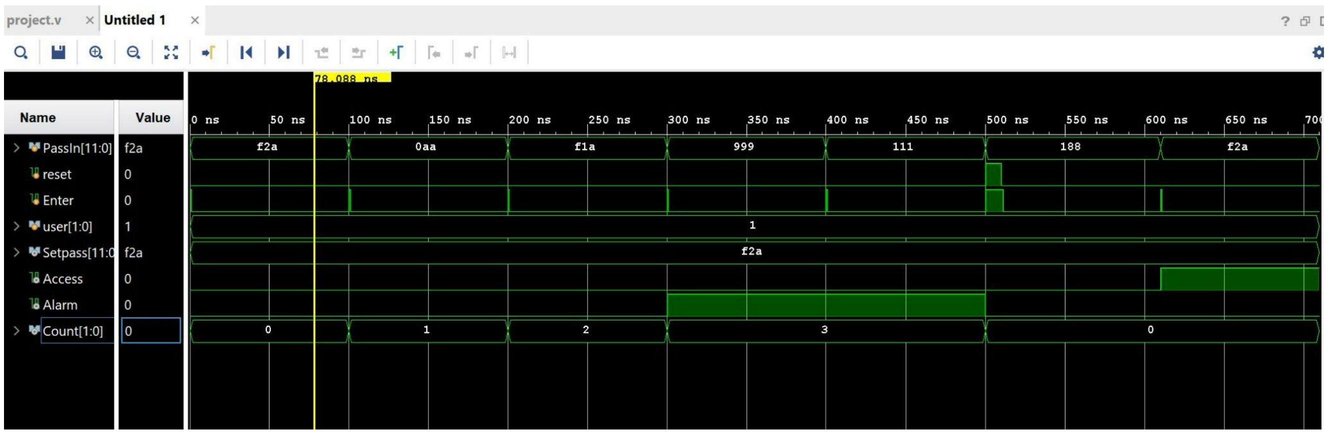
Schematic Diagram



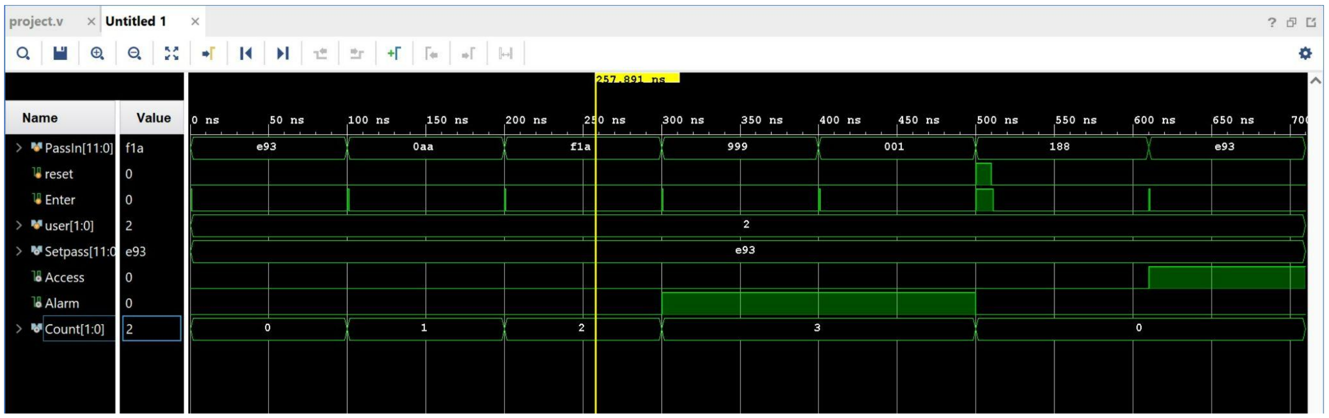


Outputs:

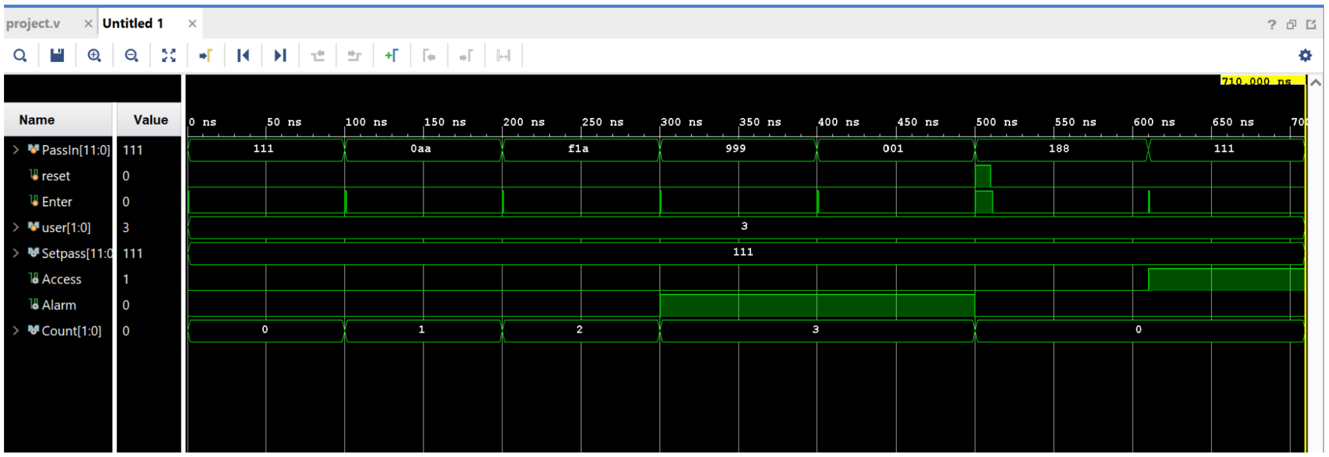

project.v
Untitled 1
710.000 ns
Name
Value
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns
300 ns
350 ns
400 ns
450 ns
500 ns
550 ns
600 ns
650 ns
PassIn[11:0] ab3
reset 0
Enter 0
user[1:0] 0
Setpass[11:0] ab3
Access 1
Alarm 0
Count[1:0] 0
ab3
0aa
f1a
999
111
188
ab3
0
ab3
0
1
2
3
0


project.v
Untitled 1
78.088 ns
0 ns 50 ns 100 ns 150 ns 200 ns 250 ns 300 ns 350 ns 400 ns 450 ns 500 ns 550 ns 600 ns 650 ns 700
f2a 0aa f1a 999 111 188 f2a
1
f2a
0 1 2 3 0

Name	Value
PassIn[11:0]	f2a
reset	0
Enter	0
user[1:0]	1
Setpass[11:0	f2a
Access	0
Alarm	0
Count[1:0]	0


project.v
Untitled 1
257.891 ns
0 ns 50 ns 100 ns 150 ns 200 ns 250 ns 300 ns 350 ns 400 ns 450 ns 500 ns 550 ns 600 ns 650 ns 700
e93 0aa f1a 999 001 188 e93
2
e93
0 1 2 3 0

Name	Value
PassIn[11:0]	f1a
reset	0
Enter	0
user[1:0]	2
Setpass[11:0	e93
Access	0
Alarm	0
Count[1:0]	2


project.v
Untitled 1
710.000 ns
0 ns 50 ns 100 ns 150 ns 200 ns 250 ns 300 ns 350 ns 400 ns 450 ns 500 ns 550 ns 600 ns 650 ns 70
111 0aa f1a 999 001 188 111
3
111
0 1 2 3 0

Name	Value
PassIn[11:0]	111
reset	0
Enter	0
user[1:0]	3
Setpass[11:0	111
Access	1
Alarm	0
Count[1:0]	0

Applications

.Residential Area

• Top class Organizations and Office buildings

• Schools and other Institutional buildings

• Safes and Lockers.

• Garage

• Storage

• Other places where security is required

Scope of development :

- Implementing Remote controlling
- Adding more flexibility for Secret button for resetting preset password
- Adding more flexibility for long passwords
- Adding camera or fingerprint recognition in parallel with the alarm capturing photo or biometric.
- Adding the capability to contact the owner/head in case of breaking-in.
- Adding safety against electric discharge by grounding the lock body

Reference

Verilog HDL A guide to Digital Design and Synthesis by Samir Palnitkar - for Verilog Language basics and syntax