# PROJECT REPORT

# Combinational Logic Circuit

**ELL201/ELP201**

Harsh chaudhary 2022MT11261

**Tuesday Batch**
**Table Number: 12**

2022EE1766 Niranjan Rajeev ,2022MT61961 Dev Elvis
2022PH11853 Aditya Limkar

# CONTENTS

**Page no.**

भारतीय प्रौद्योगिकी संस्थान दिल्ली
INDIAN INSTITUTE OF TECHNOLOGY

# PROJECT OVERVIEW

## 1.1 Objective

This project aims to develop and deploy a secure digital locker system utilizing the Verilog Hardware Description Language (VHDL). The primary goal is to construct a digital lock capable of accommodating a wide array of combinations, totaling 2^6 = 64 unique possibilities

## 1.2 Project Insights

Delve into the intricacies of our project, where users have the autonomy to establish their unique password combinations. Upon desiring access, users must input the predetermined password stored within the Verilog file. Upon successful authentication, the system illuminates all lights, signaling access approval.

The input-output format is described below:

## 1.3 Input-Output Format

Password Setting Input

Users have the capability to modify the password directly within the original Verilog code, which is subsequently implemented on the CPLD.

Output Indicators for Password Entry

In case of incorrect, incomplete, or correct password entries, the system provides intuitive feedback. If alternate bits are illuminated (e.g., 1010), it indicates an incorrect password entry or an incomplete input. Conversely, if all four bits are lit up, it signifies a correct password entry. Example:

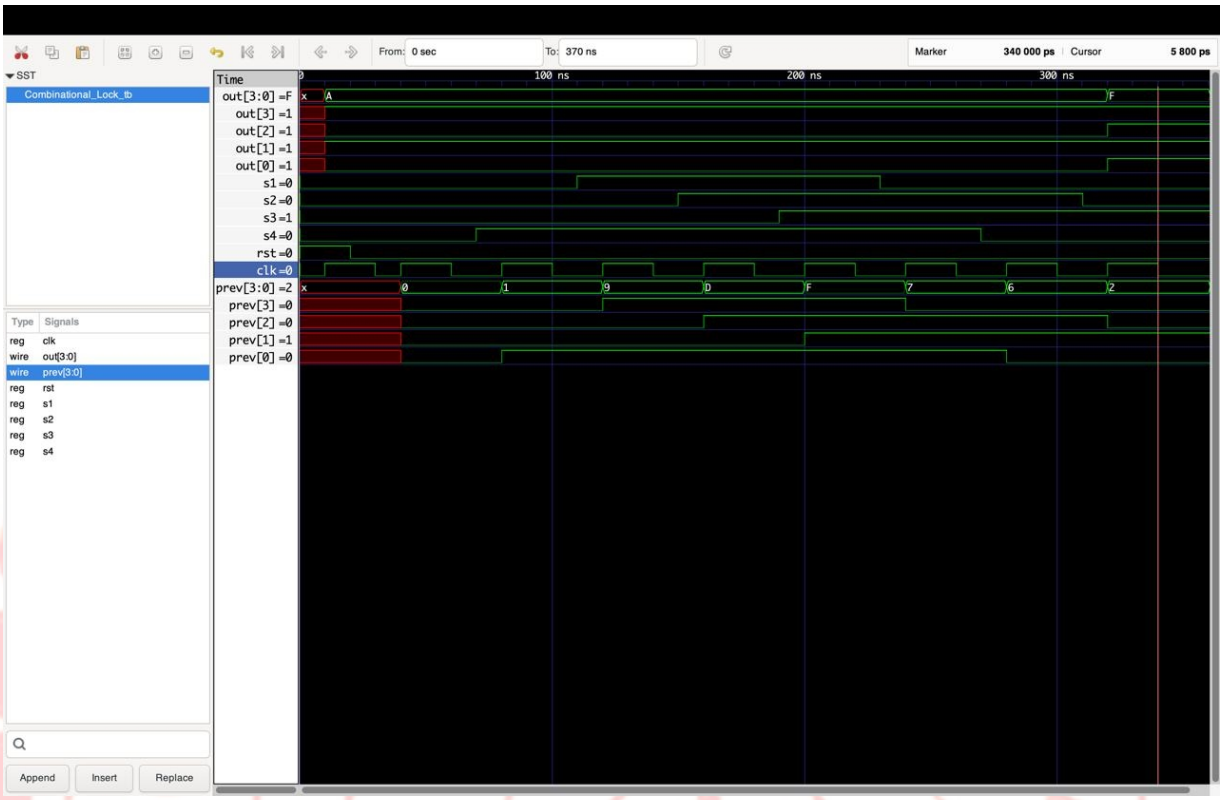
If the password is 456654:

Then the specific sequence for opening and closing the switches will be:

- First, open switch s4, then s5, then s6.

- Next, turn off these 3 switches i.e. turning off s6, then s5, then s4.- This process generates one of the 2^6 possible combinations for setting the password on the CPLD digital lock

## 2. Wave form



## 3. Verilog code

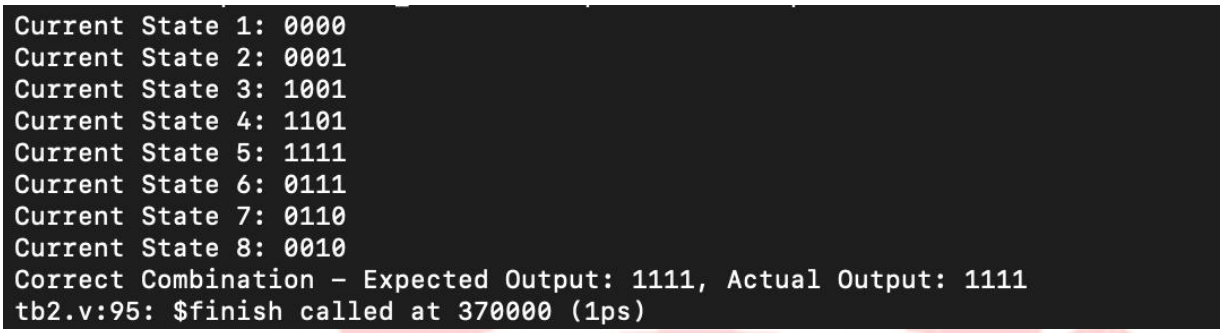
```
module Combinational_Lock(
    input clk, s1, s2, s3, s4, rst,
    output reg [3:0] out,
    output reg [3:0] prev
);

    reg [3:0] combination0 = 4'b0000;
    reg [3:0] combination1 = 4'b0001;
    reg [3:0] combination2 = 4'b1001;
    reg [3:0] combination3 = 4'b1101;
    reg [3:0] combination4 = 4'b1111;
    reg [3:0] combination5 = 4'b0111;
    reg [3:0] combination6 = 4'b0110;
    reg [3:0] combination7 = 4'b0010;
    reg [3:0] temp_arr [0:7];
    integer temp_index = 0;
    reg [3:0] curr_state;
    reg [3:0] is_ok;
    reg prev_s1=1'b1, prev_s2=1'b1, prev_s3=1'b1, prev_s4=1'b1;

    always @(posedge clk) begin
        if(rst) begin
            out=4'b1010;
            is_ok=4'b0001;
            curr_state=4'b0000;
            temp_index=0;
            temp_arr[0]=4'b0000;
            temp_arr[1]=4'b0000;
            temp_arr[2]=4'b0000;
            temp_arr[3]=4'b0000;
            temp_arr[4]=4'b0000;
            temp_arr[5]=4'b0000;
            temp_arr[6]=4'b0000;
            temp_arr[7]=4'b0000;
        end
        else begin
            if (s1 != prev_s1 || s2 != prev_s2 || s3 != prev_s3 || s4 != prev_s4) begin
                curr_state = {s1, s2, s3, s4};
                temp_arr[temp_index] = curr_state;
                prev=temp_arr[temp_index];
                temp_index = temp_index + 1;
                temp_index = temp_index%8;
                prev_s1=s1;
                prev_s2=s2;
                prev_s3=s3;
                prev_s4=s4;
                if( (temp_arr[0]==combination0) &&
                    (temp_arr[1]==combination1) &&
                    (temp_arr[2]==combination2) &&
                    (temp_arr[3]==combination3) &&
                    (temp_arr[4]==combination4) &&
                    (temp_arr[5]==combination5) &&
                    (temp_arr[6]==combination6) &&
                    (temp_arr[7]==combination7)
                ) out=4'b1111;
                else out=4'b1010;
            end
        end
    end
endmodule
```

## 4. Test cases

This image shows the condition when the switch is unlocked




else if the switch remains locked then it showa the output 0101.

## 5. PROJECT OVERVIEW

The project involves designing a digital lock system that unlocks when a specific sequence of inputs is entered via switches. It comprises several key components:

Inputs: The lock operates by receiving input signals from a clock signal and switches, essential for its functioning.

Reset Signal: This signal serves to reset the lock to its default state, pivotal for rectifying errors or initiating a fresh start.

Output: Indicating whether it's unlocked or locked, the lock's output signal is instrumental in managing auxiliary systems such as doors or security systems.

The operational flow encompasses:

Starting : The lock commences its operation in a secured state, with all internal registers and variables reset to their default values.

Detecting Input Changes: Vigilantly monitoring switch states, the system identifies alterations, signaling the initiation of a new phase within the input sequence.

Updating the State: Upon recognizing a switch alteration, the lock dynamically adjusts its state to mirror the updated combination of switch values. This revised state is then stored within an internal array.

Checking the Combination: The system cross-references the stored sequence with the predetermined correct combination to ascertain whether unlocking the lock is warranted.

Outputting the Result: Upon entry of the correct combination, the lock is promptly unlocked, as indicated by the output signal (in our case 1111). Conversely, it remains firmly secured if an incorrect combination is entered(which is 0101 ).

Additionally, a reset signal is provided, affording users the ability to clear the internal array and commence anew, facilitating repeated usage or rectification following an erroneous combination entry. Ultimately, the system's functionality hinges upon its adeptness at detecting input changes, preserving sequences, and validating them against the predefined combination to regulate lock status.