Digital Assignment Report

EEE F313 : Analog & Digital VLSI Design

Problem Set 29

Birla Institute of Technology & Science, Pilani - Pilani Campus
First Semester - Academic Year 2023-24

Kedar Date	2021A3B52396P
Mausam Tripathi	2021A3PS0310P
Gopalchetty Tejaswi	2021A3PS2384P

October 16, 2023

Abstract

This report describes the problem statements, the design methodologies, and the optimization processes undertaken to achieve the goals described in the problem statement. The problem selected by this group for the project, out of the various problems provided, is Problem Set 29. The problem has two parts: (1)Design of an EXOR gate(Implemented in MICROWIND Software) (2)Design of a combination lock(Implemented in Verilog HDL). The report shows the results of the design process in the form of waveforms of inputs and outputs. All the team members would like to thank Prof. Anu Gupta for providing this unique opportunity. The team members would also like to thank those whose help was indirect, particularly for the MICROWIND Software tutorials.

1 CMOS Implementation of Two Input EXOR Gate

Problem Statement Design a 2-input Full CMOS implementation of XOR function circuit at 500MHz frequency with 500fF load capacitance.
NOTE: Use less than 15 transistors including both NMOS and PMOS in your design.

Theory & Calculations The primary design goal is to make the gate operational at 500MHz frequency. The parameters concerned with this are the τ_{pHL} which is the time taken for the high to low transition of the output upon input change, and the τ_{pLH} which is the time taken for the low to high transition of the output upon input change.
The definitions of τ_{pHL} & τ_{pLH} are given as follows:

$$\tau_{pHL} = \frac{C_{load}}{k_n(V_{OH} - |V_{T_0,n}|)} \left[\frac{2\,|V_{T_0,n}|}{V_{OH} - |V_{T_0,n}|} + ln\left(\frac{4(V_{OH} - |V_{T_0,n}|)}{V_{OH} + V_{OL}} - 1 \right) \right] \tag{1}$$

$$\tau_{pLH} = \frac{C_{load}}{k_p(V_{OH} - |V_{T_0,p}|)} \left[\frac{2\,|V_{T_0,p}|}{V_{OH} - |V_{T_0,p}|} + ln\left(\frac{4(V_{OH} - |V_{T_0,p}|)}{V_{OH} + V_{OL}} - 1 \right) \right] \tag{2}$$

The values of k_n and k_p are given by:

$$k_n = \mu_n C_{ox} \frac{W_n}{L_n} \tag{3}$$

$$k_p = \mu_p C_{ox} \frac{W_p}{L_p} \tag{4}$$

The value of C_{load} is the Load capacitance(provided in the Problem Statement as 500pF) in parallel with parasitic capacitances(C_{GD} & C_{DB}) of the output MOSFETs. Since the Load Capacitance is considerably higher than the typical values of C_{GD} & C_{DB}, hence value of C_{load} can be approximated to 500pF without incresing error percentage.
For a balanced output, design parameters k_n & k_p were assumed to be equal. Hence, the relation $\frac{W_n}{W_p} = \frac{\mu_p}{\mu_n}$ is obtained.

Optimisation / Innovation The 180nm technology node used in this project had a $\frac{\mu_n}{\mu_p}$ ratio of 1.9 .
The MICROWIND Software introduces a constraint of minimum length of any drawing to be of 200nm, thus making the gate size to be 200nm in all MOSFETs.
The $\frac{W}{L}$ ratios were optimised such as to minimise Drain Currents and thus power. However pure optimisation of drain currents leads to reduced levels of V_{OH} and increased levels of V_{OL}. Hence, a combined optimisation of the two parameters was undertaken.
Two MOSFETS at the output end are minimised by implementing the complement of EXOR(EXNOR). During the design, one redundant path was removed in the PMOS part of the implementation. Thus material required for interconnects was reduced. To reduce the Silicon area used, Euler Path was recognised and implemented.

Schematic The Schematic of the Two Input EXOR Gate implementation in CMOS technology is shown in Figure 1.

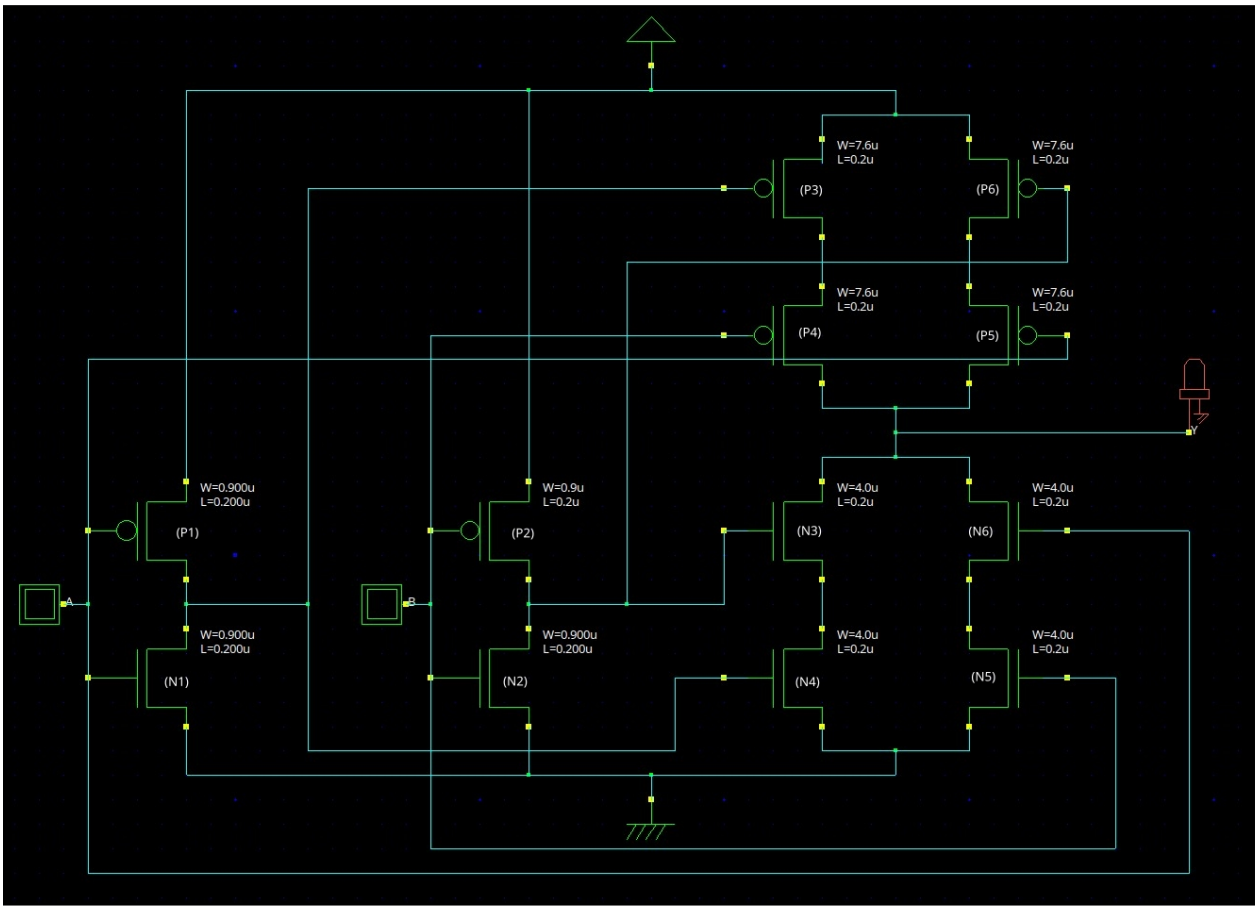


Figure 1: Schematic of CMOS Implemetation of Two Input EXOR.

$\frac{W}{L}$ of MOSFETs The Table 1 gives the values of $\frac{W}{L}$ ratios of all the PMOSs and NMOSs used in the design and implementation of CMOS two input EXOR gate.

Table 1: $\frac{W}{L}$ Ratios of all MOSFETs used in Schematic.

NMOS Label	$\frac{W_n}{L_n}$	**PMOS Label**	$\frac{W_p}{L_p}$
N1	5	P1	9.5
N2	5	P2	9.5
N3	10	P3	19
N4	10	P4	19
N5	10	P5	19
N6	10	P6	19

Layout of EXOR Gate The Silicon layout of the two input CMOS implemetation EXOR Gate is shown in Figure 2.

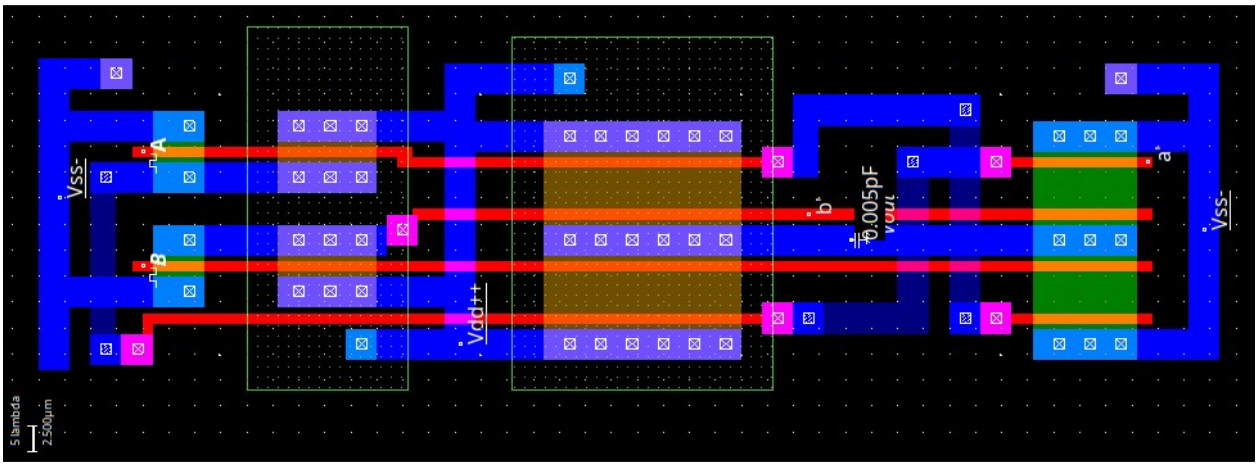


Figure 2: Layout of CMOS Implemetation of Two Input EXOR.

Gate Function Output Waveforms The EXOR Gate implemented in Full CMOS Implementation has the function $F = A \oplus B = \bar{A} \cdot B + A \cdot \bar{B}$. The Signal A has a time period of 4ns, while the B signal has a time period of 2ns which corresponds to 500MHz frequency of input signal. Both the input signals have $\tau_{rise} = \tau_{fall} = 0.050ps$. The output of the EXOR Gate can be verified in Figure 3.

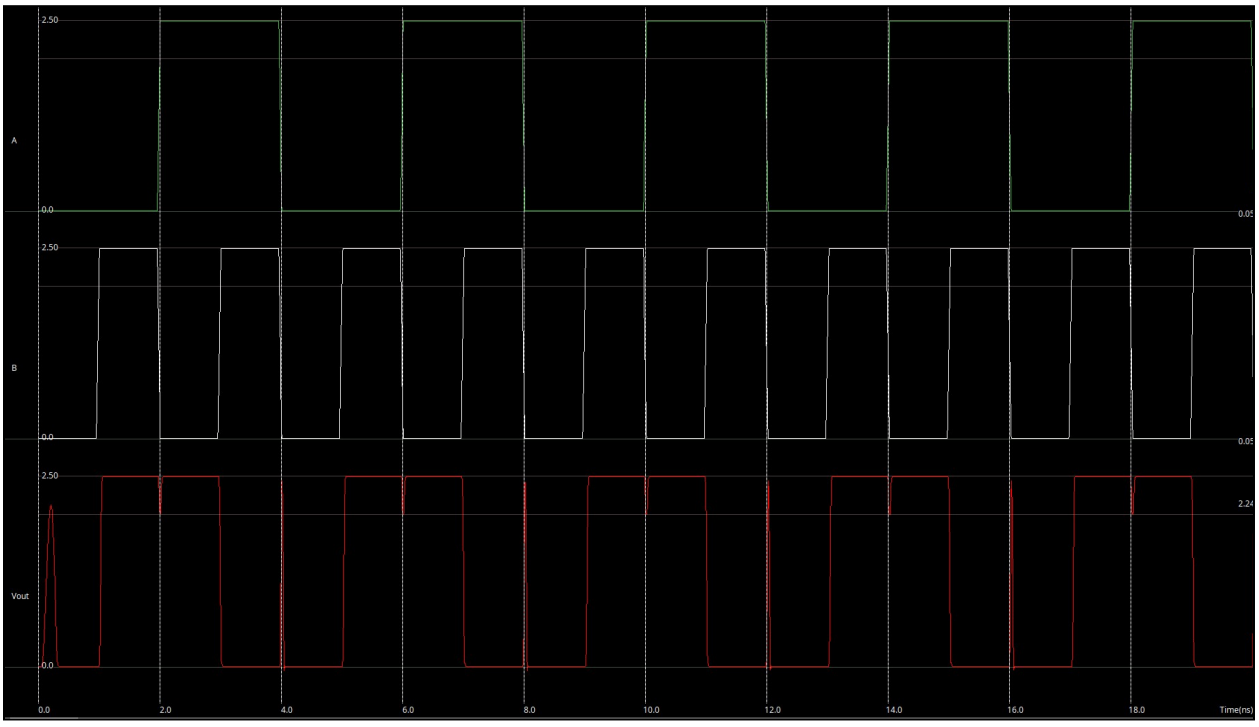
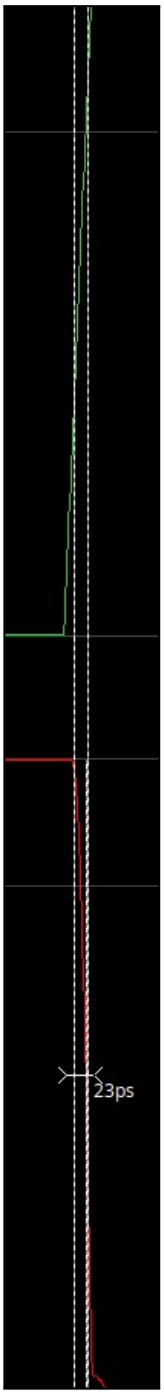
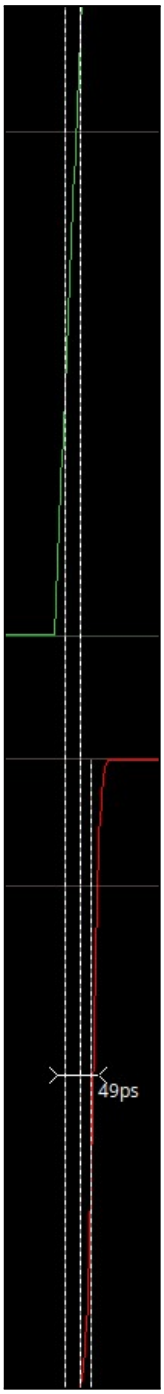


Figure 3: Time signal verification of Function Implementation of EXOR Gate.

Timing Diagrams The $\tau_{pHL} = 23ps$ & $\tau_{pLH} = 49ps$ for the designed gate can be seen in Figures 4a & 4b respectively.



(a) $\tau_{pHL} = 23ps$



(b) $\tau_{pLH} = 49ps$

Conclusion The Full CMOS implementation of the two-input EXOR Gate designed satisfies the requirements of the problem statement of providing functionality at 500MHz rate of input by providing $\tau_{pLH} = 49ps$ & $\tau_{pHL} = 23ps$. The power consumed by the gate is 0.145mW. This power is mainly due to short circuit power due to $\tau_{rise} = \tau_{fall} = 0.050ps$ of the input signals.

2 Combination Lock System

Problem Statement Build an electronic combination lock with two number buttons (0 and 1), a reset button and a unlock output. The combination should be 01011.
a) Design FSM.
b) Write verilog code for 8:1 MUX and D FF
c) Use 8:1 MUX and DFF to impelement FSM using Verilog.

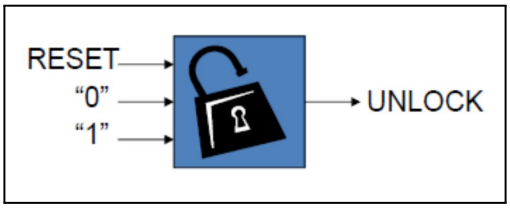


Figure 5: Diagram of Lock System given in Problem Statement.

Design of Finite State Machine(FSM) A Moore State Machine was designed, since the even after the final key press of the unlock sequence has been completed, the lock should still remain unlocked. Hence, as the "UNLOCK" output should not depend on the key pressed inputs in the final state, the Moore State Machine was preferred.
The State Diagram is presented in the Figure 6.

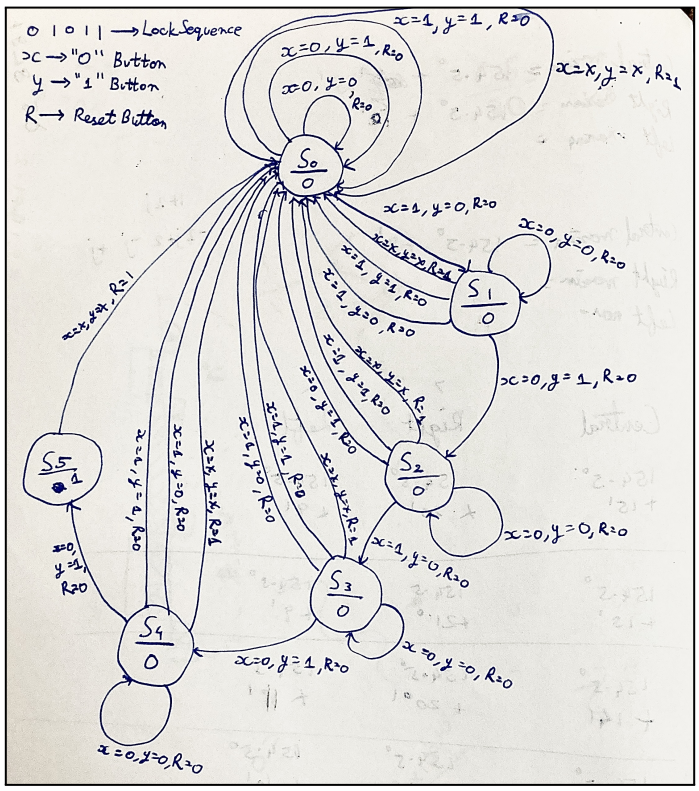


Figure 6: State Diagram for Moore State Machine for Combination Lock System.

The Table 2 gives the binary coded states used for synthesis. From the State Diagram in Figure 6 we see that the States S_6 & S_7 remain unused.

Table 2: State Reference Table

S	Q_2	Q_1	Q_0
S_0	0	0	0
S_1	0	0	1
S_2	0	1	0
S_3	0	1	1
S_4	1	0	0
S_5	1	0	1
S_6	1	1	0
S_7	1	1	1

Verilog Subsystem Design of 8:1 Multiplexer and D Flip-Flop The 8:1 MUX and the DFF was designed and implemented in Verilog HDL using the Gate Level Design Method.
The D Flip-Flop has been modelled on the functionality shown by Texas Instruments SN74LS74. The Verilog HDL code for the D Flip Flop Module can be found in Code 1.

Code 1: D Flip Flop Verilog HDL Code.

```
`timescale 1s/1ms
module dff( q , q_bar , d , clk , clr_bar , pre_bar );
wire [3:0] y ;
output q , q_bar ;
input  d , clk , clr_bar , pre_bar ;
nand( y[0] , pre_bar , y[3] , y[1] ) ;
nand( y[1] , clr_bar , y[0] , clk ) ;
nand( y[2] , clk , y[1] , y[3] ) ;
nand( y[3] , d , y[2] , clr_bar ) ;
nand( q , q_bar , y[1] , pre_bar ) ;
nand( q_bar , q , clr_bar , y[2] ) ;
endmodule
```

The 8:1 MUX has been modelled on the functionality shown by Texas Instruments SN74LS152. The Verilog HDL Code for the 8:1 MUX Module van be found in Code 2.

Code 2: 8:1 MUX Verilog HDL Code.

```
`timescale 1s/1ms
module mux8_to_1 ( in , out , s , en_bar );
output out ;
input [7:0] in ;
input [2:0] s ;
input en_bar ;
wire out_before_z ;
wire [7:0] y ;
wire [2:0] sn ;
not( sn[0] , s[0] ) ;
not( sn[1] , s[1] ) ;
not( sn[2] , s[2] ) ;
and( y[0] , in[0] , sn[2] , sn[1] , sn[0] ) ;
and( y[1] , in[1] , sn[2] , sn[1] , s[0] ) ;
and( y[2] , in[2] , sn[2] , s[1] , sn[0] ) ;
```

```
16 and( y[3] , in[3] , sn[2] , s[1] , s[0] ) ;
17 and( y[4] , in[4] , s[2] , sn[1] , sn[0] ) ;
18 and( y[5] , in[5] , s[2] , sn[1] , s[0] ) ;
19 and( y[6] , in[6] , s[2] , s[1] , sn[0] ) ;
20 and( y[7] , in[7] , s[2] , s[1] , s[0] ) ;
21 or( out_before_z , y[0] , y[1] , y[2] , y[3] , y[4] , y[5] , y[6] , y[7] ) ;
22 bufif0( out , out_before_z , en_bar ) ;
23 endmodule
```

Circuit Diagram of FSM Implemented using DFFs and 8:1 MUXs In the implementation of the Moore State Machine designed in the previous section for the Combination Lock System, a total of three DFFs and seven 8:1 MUXs were utilised.
The final circuit contains 3 User Inputs, namely the "RESET Button", "ZERO Button" & the "ONE Button". The output"UNLOCK" remains low in any wrong combination cases, or when the "RESET Button" is pressed.The output "UNLOCK" goes high when the final key of the correct sequence is pressed, and remains high until any one of the three keys is pressed.
The system designed is asynchronous, meaning, there can be any amount of delay between consecutive key presses, without the delay affecting the output of the State Machine.
The Circuit Diagram can be seen in Figure 7

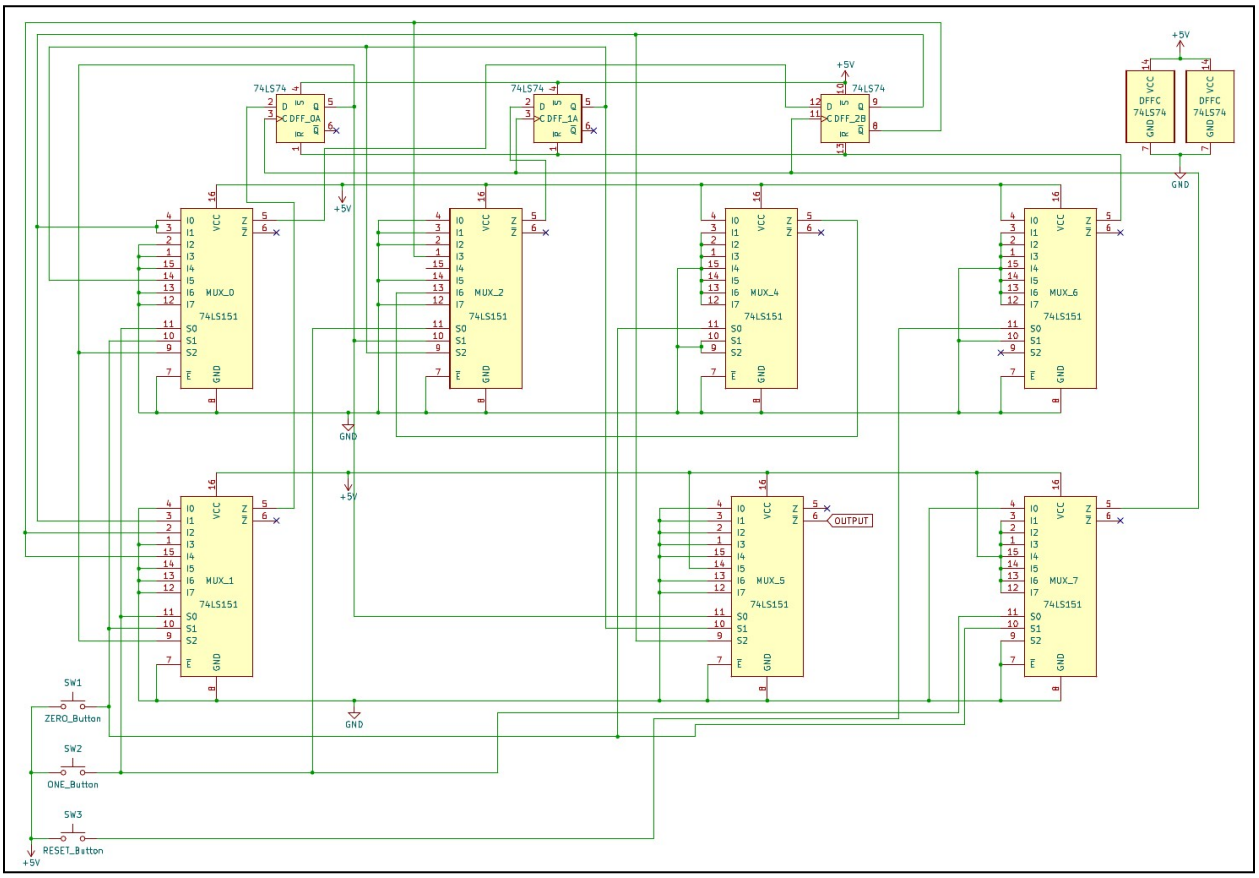


Figure 7: Circuit Diagram of Combination Lock System.

Combination Lock System Verilog HDL Code The System designed using only DFFs and 8:1 MUXs uses seven 8:1 MUXs and three DFFs. The system designed is asynchronous, a Moore Machine. Since the Verilog Code is written at Gate Level implementation, hence, to avoid undefined signals at a level transition of input of DFF and the input also depending upon output of the DFF, a small delay has been incorporated in the CLOCK input of all DFFs, which is not far from the real behaviour of these components.
The Verilog HDL code for the Combination Lock System can be found in Code 3.

Code 3: Combination Lock System Verilog HDL Code.

```
`include "MUX8_to_1.v"
`include "DFF.v"
`timescale 1s/1ms
module system( RESET_Button , ZERO_Button , ONE_Button , UNLOCK_Output );
input RESET_Button , ZERO_Button , ONE_Button ;
output UNLOCK_Output ;
wire clk , clr_bar ;
reg pre_bar ;
wire dff_0_Q , dff_0_Q_Bar , dff_0_D ;
wire dff_1_Q , dff_1_Q_Bar , dff_1_D ;
wire dff_2_Q , dff_2_Q_Bar , dff_2_D ;
assign #0.1 clk = mux_7_out ;
dff dff_0( dff_0_Q , dff_0_Q_Bar , dff_0_D , clk , clr_bar , pre_bar );
dff dff_1( dff_1_Q , dff_1_Q_Bar , dff_1_D , clk , clr_bar , pre_bar );
dff dff_2( dff_2_Q , dff_2_Q_Bar , dff_2_D , clk , clr_bar , pre_bar );
wire [7:0] mux_0_in ;
wire [2:0] mux_0_s ;
wire mux_0_out ;
wire mux_0_en_bar ;
wire [7:0] mux_1_in ;
wire [2:0] mux_1_s ;
wire mux_1_out ;
wire mux_1_en_bar ;
wire [7:0] mux_2_in ;
wire [2:0] mux_2_s ;
wire mux_2_out ;
wire mux_2_en_bar ;
wire [7:0] mux_4_in ;
wire [2:0] mux_4_s ;
wire mux_4_out ;
wire mux_4_en_bar ;
wire [7:0] mux_5_in ;
wire [2:0] mux_5_s ;
wire mux_5_out ;
wire mux_5_en_bar ;
wire [7:0] mux_6_in ;
wire [2:0] mux_6_s ;
wire mux_6_out ;
wire mux_6_en_bar ;
wire [7:0] mux_7_in ;
wire [2:0] mux_7_s ;
wire mux_7_out ;
wire mux_7_en_bar ;
reg mux_6_S_2 ;
mux8_to_1 mux_0( mux_0_in , mux_0_out , mux_0_s , mux_0_en_bar ) ;
mux8_to_1 mux_1( mux_1_in , mux_1_out , mux_1_s , mux_1_en_bar ) ;
mux8_to_1 mux_2( mux_2_in , mux_2_out , mux_2_s , mux_2_en_bar ) ;
mux8_to_1 mux_4( mux_4_in , mux_4_out , mux_4_s , mux_4_en_bar ) ;
mux8_to_1 mux_5( mux_5_in , mux_5_out , mux_5_s , mux_5_en_bar ) ;
```

```
mux8_to_1 mux_6( mux_6_in , mux_6_out , mux_6_s , mux_6_en_bar ) ;
mux8_to_1 mux_7( mux_7_in , mux_7_out , mux_7_s , mux_7_en_bar ) ;
assign mux_0_in = { 1'b0 , 1'b0 , dff_1_Q , 1'b0 , 1'b0 , 1'b0 , dff_2_Q , dff_2_Q } ;
assign mux_1_in = { 1'b0 , 1'b0 , 1'b0 , dff_2_Q_Bar , 1'b0 , dff_2_Q_Bar , dff_2_Q , 1'b0 } ;
assign mux_2_in = { 1'b0 , mux_4_out , 1'b0 , ZERO_Button , dff_2_Q_Bar , 1'b0 , 1'b0 , 1'b0 } ;
assign mux_4_in = { 1'b0 , 1'b0 , 1'b0 , 1'b0 , 1'b0 , 1'b0 , 1'b0 , 1'b1 } ;
assign mux_5_in = { 1'b0 , 1'b0 , 1'b1 , 1'b0 , 1'b0 , 1'b0 , 1'b0 , 1'b0 } ;
assign mux_6_in = { 1'b0 , 1'b0 , 1'b0 , 1'b0 , 1'b0 , 1'b0 , 1'b0 , 1'b1 } ;
assign mux_7_in = { 1'b1 , 1'b1 , 1'b1 , 1'b1 , 1'b1 , 1'b1 , 1'b1 , 1'b0 } ;
assign mux_0_s = { dff_0_Q , ZERO_Button , ONE_Button } ;
assign mux_1_s = { dff_0_Q , ZERO_Button , ONE_Button } ;
assign mux_2_s = { dff_1_Q , dff_0_Q , ONE_Button } ;
assign mux_4_s = { 1'b0 , 1'b0 , ZERO_Button } ;
assign mux_5_s = { dff_2_Q , dff_1_Q , dff_0_Q } ;
assign mux_6_s = { mux_6_S_2 , 1'b0 , RESET_Button } ;
assign mux_7_s = { 1'b0 , ZERO_Button , ONE_Button } ;
assign mux_0_en_bar = 0 ;
assign mux_1_en_bar = 0 ;
assign mux_2_en_bar = 0 ;
assign mux_4_en_bar = 0 ;
assign mux_5_en_bar = 0 ;
assign mux_6_en_bar = 0 ;
assign mux_7_en_bar = 0 ;
assign dff_2_D = mux_0_out ;
assign dff_0_D = mux_1_out ;
assign dff_1_D = mux_2_out ;
assign UNLOCK_Output = mux_5_out ;
assign clr_bar = mux_6_out ;
initial begin
    mux_6_S_2 = 1'b1 ;
    pre_bar = 1'b1 ;
    mux_6_S_2 = 1'b0 ;
end
endmodule
```

The Testbench which simulates the various Button Pressings and the output is given by Code 4.

Code 4: Combination Lock System Testbench Verilog HDL Code.

```
`include "System.v"
`timescale 1s/1ms
module System_Testbench ;
reg RESET_Button , ZERO_Button , ONE_Button ;
wire UNLOCK_Output ;
system lock_combination( RESET_Button , ZERO_Button , ONE_Button , UNLOCK_Output ) ;
initial
begin
    $dumpfile("System_Testbench.vcd");
    $dumpvars;
    ZERO_Button = 0 ;
    ONE_Button = 0 ;
    RESET_Button = 1 ;
    #2.5 RESET_Button = 0;
    #5 ZERO_Button = 1 ;
    #5 ZERO_Button = 0 ;
    #5 ONE_Button = 1 ;
    #5 ONE_Button = 0 ;
    #5 ZERO_Button = 1 ;
    #5 ZERO_Button = 0 ;
    #5 ONE_Button = 1 ;
```

```
22      #5 ONE_Button = 0 ;
23      #5 ONE_Button = 1 ;
24      #5 ONE_Button = 0 ;
25      #5 ZERO_Button = 1 ;
26      #5 ZERO_Button = 0 ;
27      $finish ;
28  end
29  endmodule
```

Results The waveforms of the Inputs and Output are shown in Figure 8

Figure 8: Simulated Waveforms of Combination Lock System.

Region I depicts the initial phase, when the system is initialised. This phase is not repeated everytime the Lock Unlocks, but only when the System is turned ON for the first time.
The region II depicts the input of the correct combination "01011".
The region III depicts the UNLOCKING of the Combination Lock System. We see that it unlocks immediately upon the press of the last correct button of the combination and remains UNLOCKED until the user presses any other key, as wee see at the onset of region IV. In this region we see that the UNLOCK ouptput has returned to the low state, meaning the the Combination Lock System has Locked adn will unlock only after the correct sequence is received.

Conclusion The Combination Lock System designed fulfils the requirements of the Problem Statement. The DFF & 8:1 MUX Modules designed and implemented in Verilog HDL are used in the implementation of the Combination Lock System, thus verifying the functionality of the Modules.