digital lock system

Final Project

Digital Logic Laboratory

Table of Contents

[Table of Tables 2](#_Toc58258621)

[Table of Figures 2](#_Toc58258622)

[Abstract 3](#_Toc58258623)

[Introduction 3](#_Toc58258624)

[Design 3](#_Toc58258625)

[Equipment and Programs 3](#_Toc58258626)

[Block Diagram and Module Break Down 3](#_Toc58258627)

[Top Module 3](#_Toc58258628)

[Buttons 4](#_Toc58258629)

[Block Diagram 4](#_Toc58258630)

[Code 5](#_Toc58258631)

[Counter Module 6](#_Toc58258632)

[Code 6](#_Toc58258633)

[Display Module 6](#_Toc58258634)

[Code 7](#_Toc58258635)

[Finite State Machine (FSM) Module 9](#_Toc58258636)

[States 9](#_Toc58258637)

[Reset 9](#_Toc58258638)

[State Diagram 10](#_Toc58258639)

[Table Diagram 10](#_Toc58258640)

[Code 11](#_Toc58258641)

[Constraints 17](#_Toc58258642)

[Code 17](#_Toc58258643)

[Test Bench 19](#_Toc58258644)

[Scenarios 19](#_Toc58258645)

[Results 20](#_Toc58258646)

[Conclusion 22](#_Toc58258647)

# Table of Tables

[Table 1 Table Diagram of FSM 10](#_Toc58258648)

# Table of Figures

[Figure 1 button representation on Nexys 4 Board 4](#_Toc58258649)

[Figure 2 Top module schematic 4](#_Toc58258650)

[Figure 3 State Diagram of FSM 10](#_Toc58258651)

[Figure 4 Right from the first try 19](#_Toc58258652)

[Figure 5 Right from second try 19](#_Toc58258653)

[Figure 6 Wrong twice 20](#_Toc58258654)

[Figure 7 Output 1 20](#_Toc58258655)

[Figure 8 Output 2 21](#_Toc58258656)

[Figure 9 Output 3 22](#_Toc58258657)

# Abstract

Design a three digit combinational digital lock and implement it using the **Nexys4 board**. The lock system has five input buttons each representing an integer number, two led lights representing the two states of the system “locked” and “unlocked” and a seven segment display that displays the entered input and number of tries. To unlock the system, the user should enter the combinational lock key “241”. He will have two tries to enter the right key or else the system will stay locked.

# Introduction

The lock system is comprised of three main modules; Counter module, display module, and finite state machine (FSM) module. The system can be in two states “locked” and “unlocked”. The top module is responsible for the connections between the three main modules and the inputs and outputs of the system. The test-bench module tests different inputs/outputs of the finite state machine (FSM) module.

# Design

## Equipment and Programs

* Nexys 4 board
* Vivado Design Suite (VDS) software
* A computer

## Block Diagram and Module Break Down

## Top Module

This module pieces all of the other modules together to get the full lock system. It receives information directly from the push buttons (BTNU / BTNR / BTNC BTNL / BTND) and passes them to the FSM module. It connects the clock received from the system to the Counter Module. It manages the inputs and outputs between the components. It passes the output from the display module to the seven segment display.

### Buttons

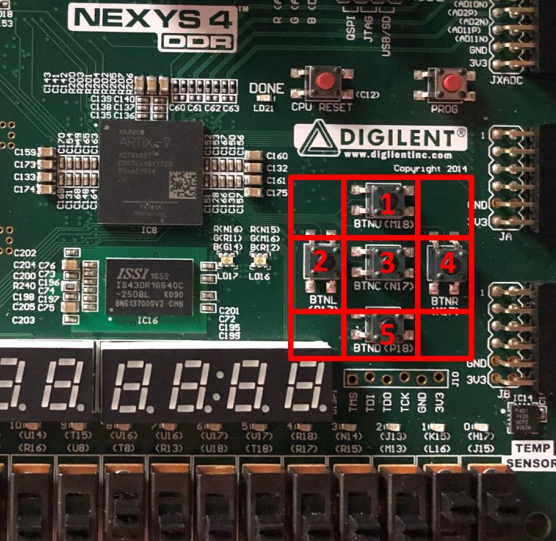


Figure button representation on Nexys 4 Board

Each button represents a digit

* BTNU = 1
* BTNL = 2
* BTNC = 3
* BTNR = 4
* BTND = 5

### Block Diagram

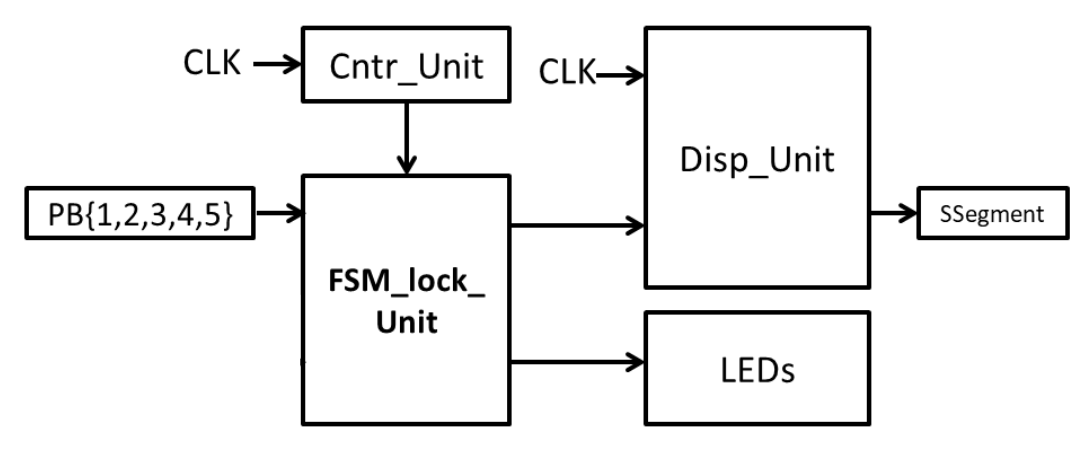


Figure Top module schematic

### Code

***module digital\_lock (***

***input wire clk ,***

***input wire reset ,***

***input wire d1 ,***

***input wire d2 ,***

***input wire d3 ,***

***input wire d4 ,***

***input wire d5 ,***

***output wire unlock,***

***output wire lock ,***

***output wire [3:0] an ,***

***output wire [7:0] sseg***

***);***

***wire [2:0] disp\_1\_wire ;***

***wire [2:0] disp\_2\_wire ;***

***wire [2:0] disp\_3\_wire ;***

***wire [2:0] disp\_N\_wire ;***

***wire new\_clk\_wire;***

***Cntr\_Unit Cntr (.clk(clk), .reset(reset), .new\_clk(new\_clk\_wire));***

***FSM\_lock\_Unit FSM\_lock (***

***.clk (new\_clk\_wire),***

***.lock (lock ),***

***.unlock(unlock ),***

***.disp\_1(disp\_1\_wire ),***

***.disp\_2(disp\_2\_wire ),***

***.disp\_3(disp\_3\_wire ),***

***.disp\_N(disp\_N\_wire ),***

***.reset (reset ),***

***.d1 (d1 ),***

***.d2 (d2 ),***

***.d3 (d3 ),***

***.d4 (d4 ),***

***.d5 (d5 )***

***);***

***Disp\_Unit Disp (***

***.clk (clk ),***

***.reset (reset ),***

***.disp\_1(disp\_1\_wire),***

***.disp\_2(disp\_2\_wire),***

***.disp\_3(disp\_3\_wire),***

***.disp\_N(disp\_N\_wire),***

***.sseg (sseg ),***

***.an (an )***

***);***

***endmodule***

## Counter Module

This module slows down the FPGA internal clock from 100MHz to 10Hz. It takes the system’s clock as input and according to a counter register it either outputs “HIGH” when it reaches a value of 10,000,000 or “LOW” otherwise.

### Code

***module Cntr\_Unit (***

***input clk ,***

***input reset ,***

***output reg new\_clk***

***);***

***reg [31:0] r\_reg;***

***always @ (posedge clk, posedge reset)begin***

***if (reset) begin***

***r\_reg <= 0;***

***new\_clk <= 0;***

***end else begin***

***if (r\_reg == 50000000) begin //10000000***

***new\_clk <= ~new\_clk;***

***r\_reg <= 0;***

***end else begin***

***new\_clk <= new\_clk;***

***r\_reg <= r\_reg + 1;***

***end***

***end***

***end***

***endmodule***

## Display Module

The display module will use the Seven Segment Display (SSD) to display the three values of the pressed buttons and the number of the users tries. The SSD is low level sensitive, meaning they light up when they are “LOW”. The module accepts the binary data for seven segment displays and decodes it using time division multiplexing.

### Code

***module Disp\_Unit (***

***input [2:0] disp\_1,***

***input [2:0] disp\_2,***

***input [2:0] disp\_3,***

***input [2:0] disp\_N,***

***input reset ,***

***input clk ,***

***output reg [7:0] sseg ,***

***output reg [3:0] an***

***);***

***localparam N = 18;***

***reg [N-1:0] q\_reg ;***

***wire [N-1:0] q\_next ;***

***reg [ 2:0] disp\_in ;***

***always @ ( posedge clk, posedge reset)***

***if (reset)***

***q\_reg <= 0;***

***else***

***q\_reg <= q\_next;***

***assign q\_next = q\_reg + 1;***

***always @\****

***case (q\_reg [N-1:N-2])***

***2'b00 :***

***begin***

***an = 4'b1110;***

***disp\_in = disp\_1;***

***end***

***2'b01 :***

***begin***

***an = 4'b1101;***

***disp\_in = disp\_2;***

***end***

***2'b10 :***

***begin***

***an = 4'b1011;***

***disp\_in = disp\_3;***

***end***

***default :***

***begin***

***an = 4'b0111;***

***disp\_in = disp\_N;***

***end***

***endcase***

***always @\****

***begin***

***case (disp\_in)***

***3'b000 : sseg[6:0] = 7'b1000000 ;***

***3'b001 : sseg[6:0] = 7'b1111001 ;***

***3'b010 : sseg[6:0] = 7'b0100100 ;***

***3'b011 : sseg[6:0] = 7'b0110000 ;***

***3'b100 : sseg[6:0] = 7'b0011001 ;***

***3'b101 : sseg[6:0] = 7'b0010010 ;***

***default : sseg[6:0] = 7'b1000000 ;***

***endcase***

***sseg[7] = 1'b1;***

***end***

***endmodule***

## Finite State Machine (FSM) Module

### States

The FSM has eight states:

### Reset

Initializes the system and sets the state register to the initial state “A”.

#### State A

* Resets everything but the number of tries to the following initial conditions:

1. Unlock led turned off
2. Lock led turned on
3. First display to zero
4. Second display to zero
5. Third display to zero

* It transitions into state “B” if the first input was equal to “2” or state “E” otherwise

#### State B

* sets the first display to the first input value
* It transitions into state “C” if the second input was equal to “4” or state “F” otherwise

#### State C

* sets the second display to the second input value
* It transitions into state “D” if the third input was equal to “1” or state “G” otherwise

#### State D

* sets the third display to the third input value
* Sets the unlock led to “HIGH”
* The system stays at this state

#### State E

* sets the first display to the first input value
* It transitions into state “F” once the second input is entered

#### State F

* sets the second display to the second input value
* It transitions into state “G” once the third input is entered

#### State G

* sets the third display to the third input value
* increments the number of tries “N”
* It transitions into state “H” if the number of tries “N” is less than two or state “A” otherwise.

#### State H

* sets lock led to “HIGH”
* The system stays at this state

### State Diagram

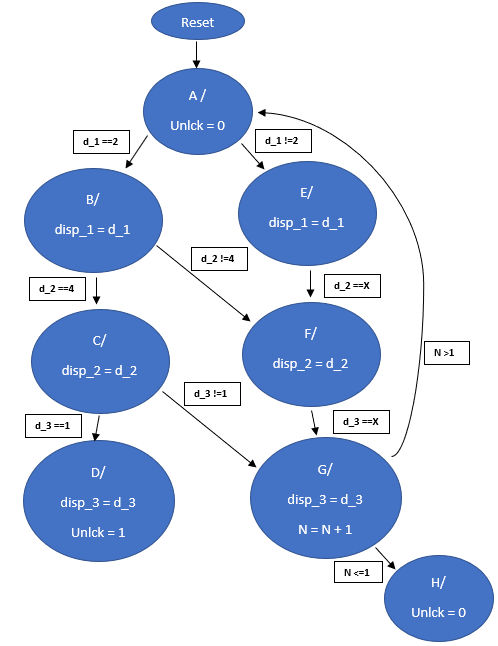


Figure State Diagram of FSM

### Table Diagram

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Current**  **state** | **Next State** | | | | | | | | | | **Output** | | | | | |
|  | **d\_1** | | **d\_2** | | | **d\_3** | | | **N** | | **disp\_1** | **disp\_2** | **disp\_3** | **N** | unlock | lock |
| **2** | **!2** | **4** | **!4** | **X** | **1** | **!1** | **X** | **<=1** | **>1** |
| **A** | **B** | **E** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **0** | **0** | **0** | **N** | **0** | **0** |
| **B** | **-** | **-** | **C** | **F** | **-** | **-** | **-** | **-** | **-** | **-** | **d\_1** | **0** | **0** | **N** | **0** | **0** |
| **C** | **-** | **-** | **-** | **-** | **-** | **D** | **G** | **-** | **-** | **-** | **d\_1** | **d\_2** | **0** | **N** | **0** | **0** |
| **D** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **d\_1** | **d\_2** | **d\_3** | **N** | **1** | **0** |
| **E** | **-** | **-** | **-** | **-** | **F** | **-** | **-** | **-** | **-** | **-** | **d\_1** | **0** | **0** | **N** | **0** | **0** |
| **F** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **G** | **-** | **-** | **d\_1** | **d\_2** | **0** | **N** | **0** | **0** |
| **G** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **H** | **A** | **d\_1** | **d\_2** | **d\_3** | **N - 1** | **0** | **0** |
| **H** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **-** | **d\_1** | **d\_2** | **d\_3** | **N** | **0** | **1** |

Table Table Diagram of FSM

### Code

***module FSM\_lock\_Unit (***

***input clk ,***

***input d1 ,***

***input d2 ,***

***input d3 ,***

***input d4 ,***

***input d5 ,***

***output reg lock ,***

***output reg unlock,***

***output reg [2:0] disp\_1,***

***output reg [2:0] disp\_2,***

***output reg [2:0] disp\_3,***

***output reg [2:0] disp\_N,***

***input reset***

***);***

***parameter [2:0] A = 3'b000, B = 3'b001, C = 3'b010, D = 3'b011,E = 3'b100, F = 3'b101, G = 3'b110, H = 3'b111;***

***reg [2:0] N ;***

***reg [2:0] N\_r ;***

***reg [2:0] state\_next;***

***reg [2:0] state\_reg ;***

***reg [2:0] disp\_1\_r;***

***reg [2:0] disp\_2\_r;***

***reg [2:0] disp\_3\_r;***

***reg [2:0] disp\_N\_r;***

***reg lock\_r;***

***reg unlock\_r;***

***reg d1\_r;***

***reg d2\_r;***

***reg d3\_r;***

***reg d4\_r;***

***reg d5\_r;***

***wire d1\_flag;***

***wire d2\_flag;***

***wire d3\_flag;***

***wire d4\_flag;***

***wire d5\_flag;***

***always @(posedge clk or posedge reset) begin***

***if(reset) begin***

***d1\_r <= 0;***

***d2\_r <= 0;***

***d3\_r <= 0;***

***d4\_r <= 0;***

***d5\_r <= 0;***

***end else begin***

***d1\_r <= d1;***

***d2\_r <= d2;***

***d3\_r <= d3;***

***d4\_r <= d4;***

***d5\_r <= d5;***

***end***

***end***

***assign d1\_flag = ~d1\_r & d1;***

***assign d2\_flag = ~d2\_r & d2;***

***assign d3\_flag = ~d3\_r & d3;***

***assign d4\_flag = ~d4\_r & d4;***

***assign d5\_flag = ~d5\_r & d5;***

***always @ ( posedge clk or posedge reset)***

***if (reset)***

***state\_reg <= A;***

***else***

***state\_reg <= state\_next;***

***always @ ( posedge clk or posedge reset)***

***if (reset) begin***

***N\_r <= 2;***

***disp\_1\_r <= 0;***

***disp\_2\_r <= 0;***

***disp\_3\_r <= 0;***

***disp\_N\_r <= 2;***

***unlock\_r <= 0;***

***lock\_r <= 0;***

***end else begin***

***N\_r <= N;***

***disp\_1\_r <= disp\_1;***

***disp\_2\_r <= disp\_2;***

***disp\_3\_r <= disp\_3;***

***disp\_N\_r <= disp\_N;***

***unlock\_r <= unlock;***

***lock\_r <= lock;***

***end***

***always @(state\_reg, d1\_flag, d2\_flag, d3\_flag, d4\_flag, d5\_flag) begin***

***disp\_1 = disp\_1\_r;***

***disp\_2 = disp\_2\_r;***

***disp\_3 = disp\_3\_r;***

***disp\_N = disp\_N\_r;***

***unlock = unlock\_r;***

***lock = lock\_r;***

***N = N\_r;***

***state\_next = state\_reg;***

***case (state\_reg)***

***A :***

***begin***

***disp\_1 = 3'b000;***

***disp\_2 = 3'b000;***

***disp\_3 = 3'b000;***

***if (d2\_flag)***

***begin***

***disp\_1 = 3'b010;***

***state\_next = B;***

***end***

***else***

***begin***

***if (d1\_flag) begin***

***disp\_1 = 3'b001;***

***state\_next = E;***

***end else if (d3\_flag) begin***

***disp\_1 = 3'b011;***

***state\_next = E;***

***end else if (d4\_flag) begin***

***disp\_1 = 3'b100;***

***state\_next = E;***

***end else if (d5\_flag) begin***

***disp\_1 = 3'b101;***

***state\_next = E;***

***end***

***end***

***end***

***B :***

***begin***

***if (d4\_flag)***

***begin***

***disp\_2 = 3'b100;***

***state\_next = C;***

***end***

***else***

***begin***

***if (d1\_flag) begin***

***disp\_2 = 3'b001;***

***state\_next = F;***

***end else if (d2\_flag) begin***

***disp\_2 = 3'b010;***

***state\_next = F;***

***end else if (d3\_flag) begin***

***disp\_2 = 3'b011;***

***state\_next = F;***

***end else if (d5\_flag) begin***

***disp\_2 = 3'b101;***

***state\_next = F;***

***end***

***end***

***end***

***C :***

***begin***

***if (d1\_flag)***

***begin***

***disp\_3 = 3'b001;***

***state\_next = D;***

***end***

***else***

***begin***

***if (d4\_flag) begin***

***disp\_3 = 3'b100;***

***state\_next = G;***

***end else if (d2\_flag) begin***

***disp\_3 = 3'b010;***

***state\_next = G;***

***end else if (d3\_flag) begin***

***disp\_3 = 3'b011;***

***state\_next = G;***

***end else if (d5\_flag) begin***

***disp\_3 = 3'b101;***

***state\_next = G;***

***end***

***end***

***end***

***D :***

***begin***

***unlock = 1;***

***lock = 0;***

***end***

***E :***

***begin***

***if (d1\_flag) begin***

***disp\_2 = 3'b001;***

***state\_next = F;***

***end else if (d4\_flag) begin***

***disp\_2 = 3'b100;***

***state\_next = F;***

***end else if (d2\_flag) begin***

***disp\_2 = 3'b010;***

***state\_next = F;***

***end else if (d3\_flag) begin***

***disp\_2 = 3'b011;***

***state\_next = F;***

***end else if (d5\_flag) begin***

***disp\_2 = 3'b101;***

***state\_next = F;***

***end***

***end***

***F :***

***begin***

***if (d1\_flag) begin***

***disp\_3 = 3'b001;***

***state\_next = G;***

***end else if (d4\_flag) begin***

***disp\_3 = 3'b100;***

***state\_next = G;***

***end else if (d2\_flag) begin***

***disp\_3 = 3'b010;***

***state\_next = G;***

***end else if (d3\_flag) begin***

***disp\_3 = 3'b011;***

***state\_next = G;***

***end else if (d5\_flag) begin***

***disp\_3 = 3'b101;***

***state\_next = G;***

***end***

***end***

***G :***

***begin***

***N = N\_r - 1;***

***disp\_N = N\_r - 1;***

***if (N\_r > 1)***

***state\_next = A;***

***else***

***begin***

***state\_next = H;***

***unlock = 0;***

***lock = 1;***

***end***

***end***

***H :***

***begin***

***unlock = 0;***

***lock = 1;***

***end***

***endcase***

***end***

***endmodule***

## Constraints

This is used for Hardware interfacing. It specifies the mapping of the FPGAs pins to other parts on the board and gives them names and some context.

### Code

***## Clock signal***

***set\_property -dict { PACKAGE\_PIN E3 IOSTANDARD LVCMOS33 } [get\_ports { clk }]; #IO\_L12P\_T1\_MRCC\_35 Sch=clk100mhz***

***create\_clock -add -name sys\_clk\_pin -period 10.00 -waveform {0 5} [get\_ports {clk}];***

***##Switches***

***set\_property -dict { PACKAGE\_PIN J15 IOSTANDARD LVCMOS33 } [get\_ports { reset }]; #IO\_L24N\_T3\_RS0\_15 Sch=sw[0]***

***## LEDs***

***set\_property -dict { PACKAGE\_PIN H17 IOSTANDARD LVCMOS33 } [get\_ports { unlock }]; #IO\_L18P\_T2\_A24\_15 Sch=led[0]***

***set\_property -dict { PACKAGE\_PIN K15 IOSTANDARD LVCMOS33 } [get\_ports { lock }]; #IO\_L24P\_T3\_RS1\_15 Sch=led[1]***

***##7 segment display***

***set\_property -dict { PACKAGE\_PIN T10 IOSTANDARD LVCMOS33 } [get\_ports { sseg[0] }]; #IO\_L24N\_T3\_A00\_D16\_14 Sch=ca***

***set\_property -dict { PACKAGE\_PIN R10 IOSTANDARD LVCMOS33 } [get\_ports { sseg[1] }]; #IO\_25\_14 Sch=cb***

***set\_property -dict { PACKAGE\_PIN K16 IOSTANDARD LVCMOS33 } [get\_ports { sseg[2] }]; #IO\_25\_15 Sch=cc***

***set\_property -dict { PACKAGE\_PIN K13 IOSTANDARD LVCMOS33 } [get\_ports { sseg[3] }]; #IO\_L17P\_T2\_A26\_15 Sch=cd***

***set\_property -dict { PACKAGE\_PIN P15 IOSTANDARD LVCMOS33 } [get\_ports { sseg[4] }]; #IO\_L13P\_T2\_MRCC\_14 Sch=ce***

***set\_property -dict { PACKAGE\_PIN T11 IOSTANDARD LVCMOS33 } [get\_ports { sseg[5] }]; #IO\_L19P\_T3\_A10\_D26\_14 Sch=cf***

***set\_property -dict { PACKAGE\_PIN L18 IOSTANDARD LVCMOS33 } [get\_ports { sseg[6] }]; #IO\_L4P\_T0\_D04\_14 Sch=cg***

***set\_property -dict { PACKAGE\_PIN H15 IOSTANDARD LVCMOS33 } [get\_ports { sseg[7] }]; #IO\_L19N\_T3\_A21\_VREF\_15 Sch=dp***

***set\_property -dict { PACKAGE\_PIN J17 IOSTANDARD LVCMOS33 } [get\_ports { an[0] }]; #IO\_L23P\_T3\_FOE\_B\_15 Sch=an[0]***

***set\_property -dict { PACKAGE\_PIN J18 IOSTANDARD LVCMOS33 } [get\_ports { an[1] }]; #IO\_L23N\_T3\_FWE\_B\_15 Sch=an[1]***

***set\_property -dict { PACKAGE\_PIN T9 IOSTANDARD LVCMOS33 } [get\_ports { an[2] }]; #IO\_L24P\_T3\_A01\_D17\_14 Sch=an[2]***

***set\_property -dict { PACKAGE\_PIN J14 IOSTANDARD LVCMOS33 } [get\_ports { an[3] }]; #IO\_L19P\_T3\_A22\_15 Sch=an[3]***

***##Buttons***

***set\_property -dict { PACKAGE\_PIN N17 IOSTANDARD LVCMOS33 } [get\_ports { d3 }]; #IO\_L9P\_T1\_DQS\_14 Sch=btnc***

***set\_property -dict { PACKAGE\_PIN M18 IOSTANDARD LVCMOS33 } [get\_ports { d1 }]; #IO\_L4N\_T0\_D05\_14 Sch=btnu***

***set\_property -dict { PACKAGE\_PIN P17 IOSTANDARD LVCMOS33 } [get\_ports { d2 }]; #IO\_L12P\_T1\_MRCC\_14 Sch=btnl***

***set\_property -dict { PACKAGE\_PIN M17 IOSTANDARD LVCMOS33 } [get\_ports { d4 }]; #IO\_L10N\_T1\_D15\_14 Sch=btnr***

***set\_property -dict { PACKAGE\_PIN P18 IOSTANDARD LVCMOS33 } [get\_ports { d5 }]; #IO\_L9N\_T1\_DQS\_D13\_14 Sch=btnd***

## Test Bench

The test-bench module tests the finite state machine (FSM) module. It can be used to assign inputs to the FSM and examine its outputs accordingly. Enables us to create different scenarios and stimulate the results.

### Scenarios

There are three possible scenarios:

1. The user enters the right combinational key

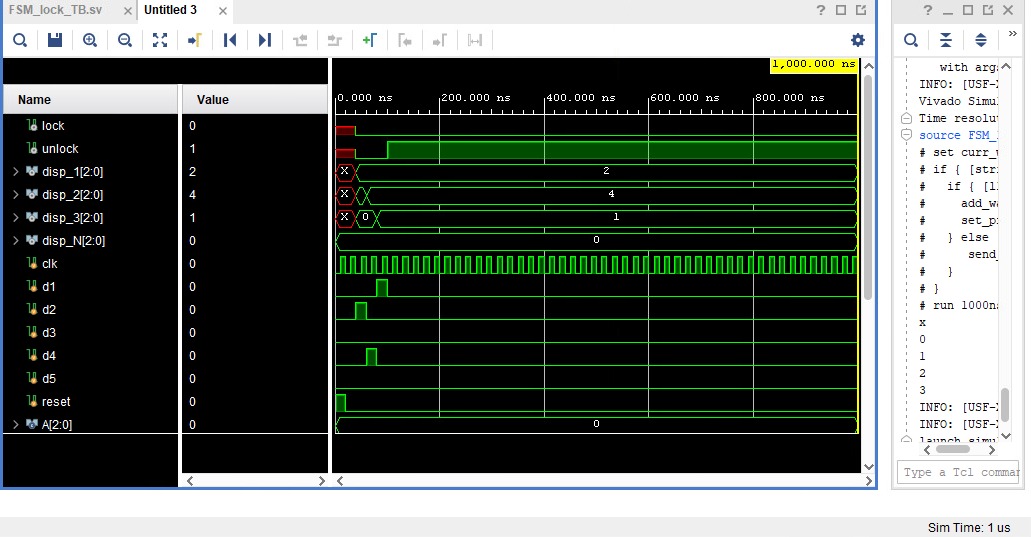


Figure Right from the first try

1. The user enters the wrong followed by the right combinational key

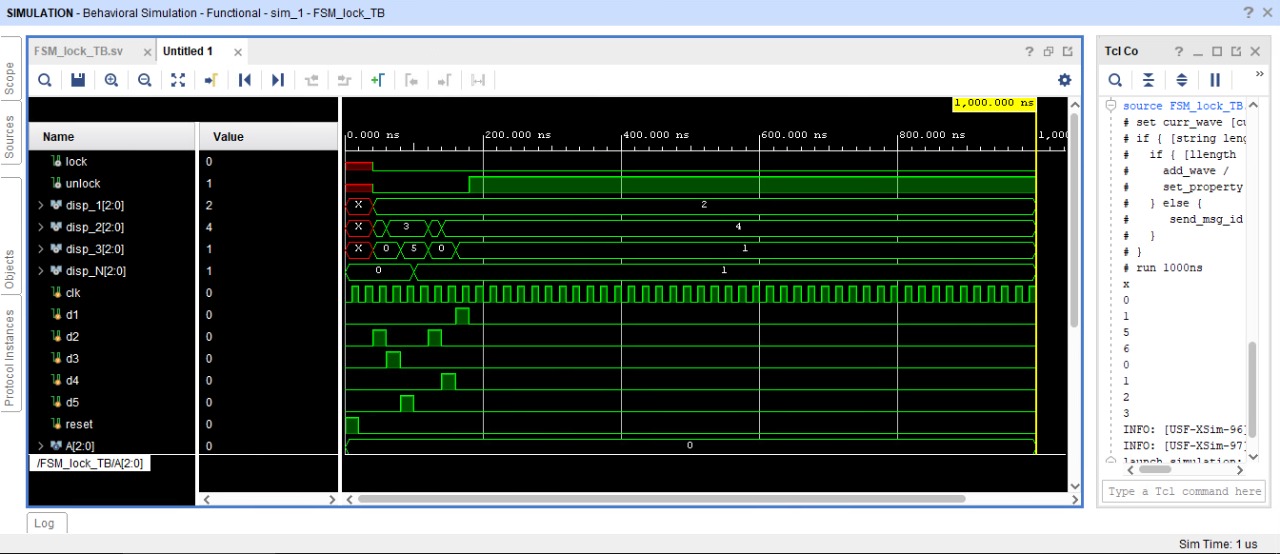


Figure Right from second try

1. The user enters the wrong combinational key twice in a row

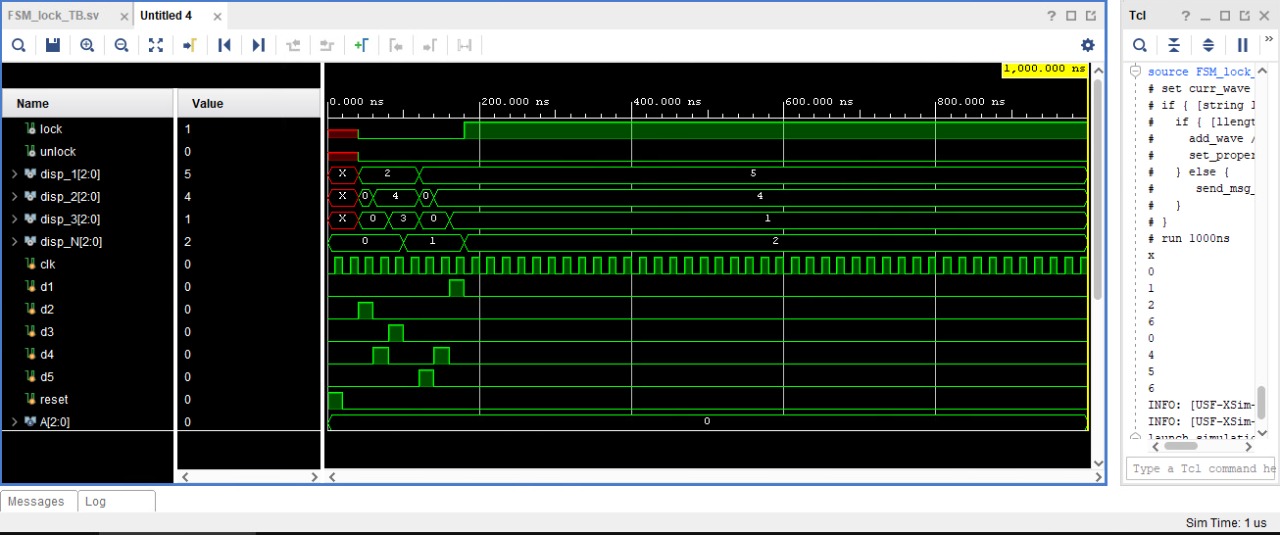


Figure Wrong twice

# Results



Figure Output 1



Figure Output 2



Figure Output 3

# Conclusion

The experimental results, theoretical results, and simulation results are identical as expected. Initially the digital lock displays zeros for the first, second and third digits and it diplays 1 for the number of remaining tries. Then The digital lock displays the decimal digits of the buttons as they are pressed. The number of tries decrements once the user enters the wrong lock key. The unlock led turns on when the user enters the right lock key and stays at that state no matter the buttons pressed. The lock led turns on when the user enters the wrong lock key twice in a row and stays at that state no matter the buttons pressed.