K L Deemed to be University

Department of Electronics & Communication Engineering

B. Tech ECE Third Year Semester-I

A.Y.2020-21, ODD Semester

Course Title: Technical Proficiency & Training -1

ID Number	Name
180040256	N Eswar
180040257	K Yogananda Sagar
180040263	P Sri Vidya

SYNCHRONOUS COMBINATIONAL LOCK USING VERILOG

CONTENTS

ABSTRACT

Security is a prime concern in our day-today life. Everyone wants to be as much secure as possible.

A classic approach to access control is the rotary combination lock , which can be found on safes, doorways, post office boxes, etc.

This typically mechanical device requires that a person wishing to gain access to a particular entryway rotate a knob left and right in a given sequence. If the sequence is correct, the entryway will

If the sequence is correct, the entryway will unlock; otherwise, the entryway will remain locked, while conveying no information about the required sequence.To accomplish such a feat, we will introduce the Moore machine, which is a type of Finite State Machine (FSM), and the state diagram, which is a convenient way to represent an FSM.

INTRODUCTION

- In this project we implemented a system security lock because at present time security system to be one of the most important technologies in this global world .
- Security refers to technique for ensuring that data stored in a computer cannot be read or used by anyone without authorization.
- A system cannot have high assurance if it has poor security and requirements. For high assurance, systems will logically include security requirement as well as availability, reliability and robustness requirements .
- When the information and material /object are seem to be much more important than system has to be mightily strong to protect it against hackers/thief or others.
- Due to the proper security controls in the system we are able to either prevent or minimize the negative impact of the hackers attack and due to this it increases the reliability, gaining trust and satisfaction of the people .
- Security has become an increasingly growing concern at the internet age and Nowadays, mobile phones are used for multiple purposes like internet-access, e-mails, calendar, mobile banking and so on.

We know that synchronous sequential circuits change affect their states for every positive or negative transition of the clock signal based on the input.
So, this behavior of synchronous sequential circuits can be represented in the graphical form and it is known as **state diagram**.
A synchronous sequential circuit is also called as **Finite State Machine** FSM, if it has finite number of states. There are two types of FSMs.

- **Mealy State Machine**
- **Moore State Machine**

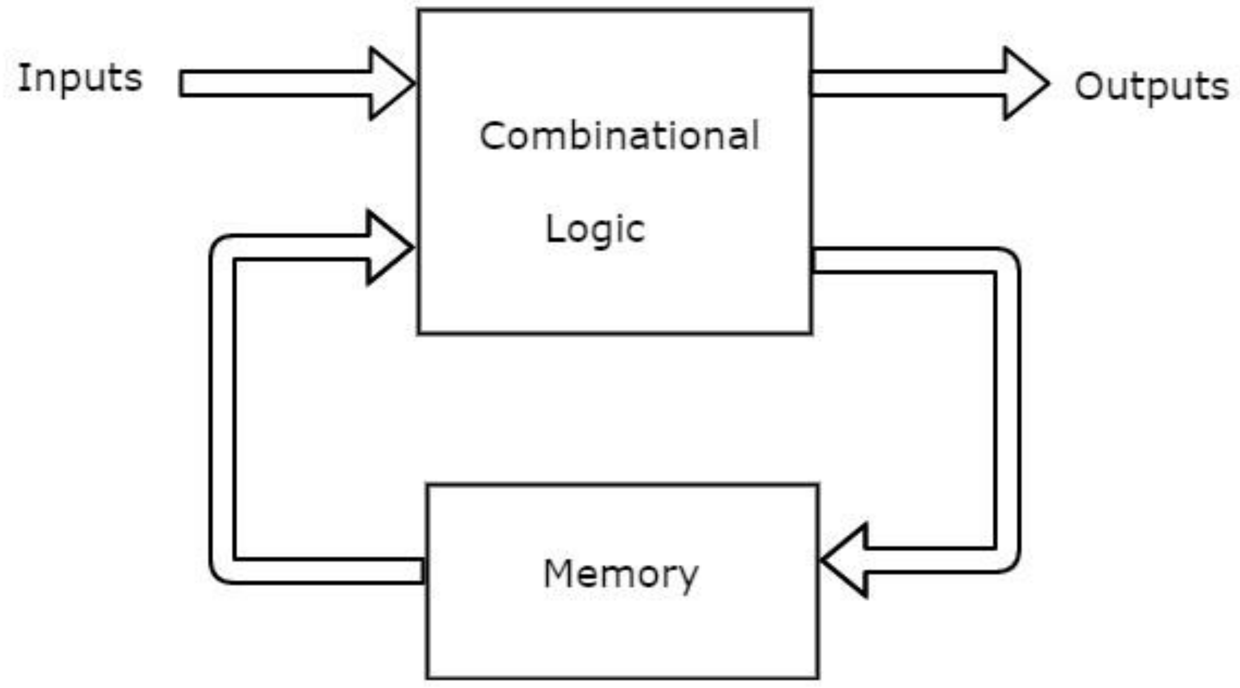
Now, let us discuss about these two state machines one by one.

MELAY STATE MACHINE

A Finite State Machine is said to be Mealy state machine, if outputs depend on both present inputs & present states. The block diagram of Mealy state machine is shown in the following figure.

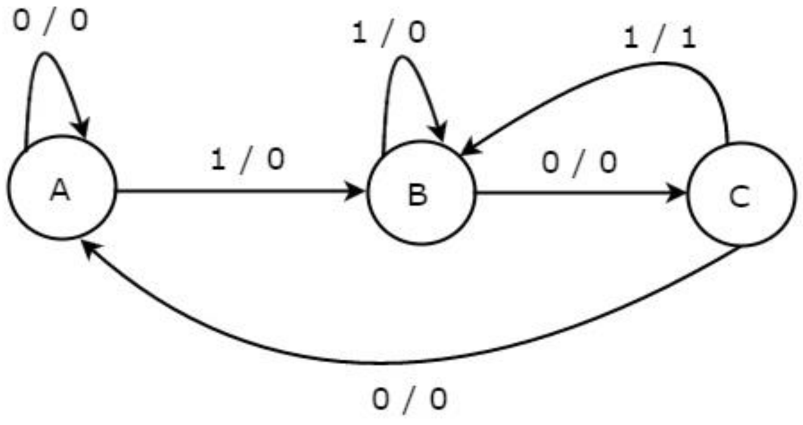
As shown in figure, there are two parts present in Mealy state machine. Those are combinational logic and memory. Memory is useful to provide some or part of previous outputs presentstates as inputs of combinational logic.

So, based on the present inputs and present states, the Mealy state machine produces outputs. Therefore, the outputs will be valid only at positive ornegativeornegative transition of the clock signal.



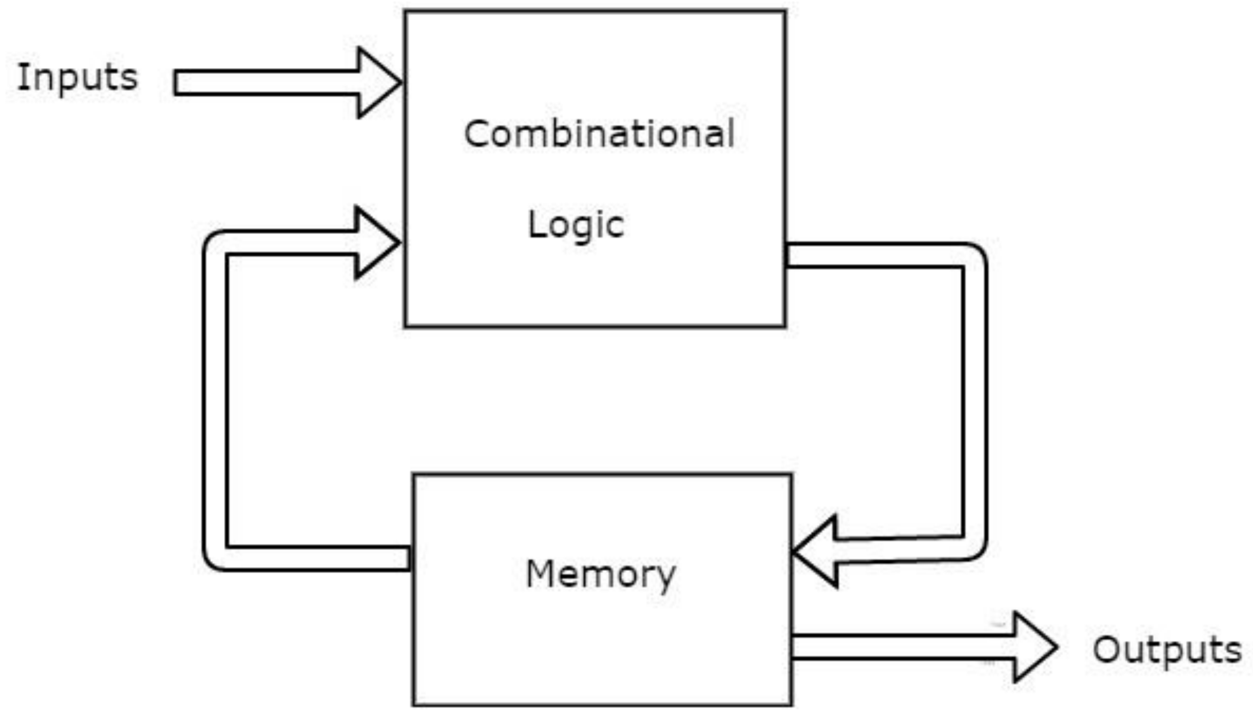
The state diagram of Mealy state machine is shown in the following figure.

In the above figure, there are three states, namely A, B & C. These states are labelled inside the circles & each circle corresponds to one state.
Transitions between these states are represented with directed lines. Here, 0 / 0, 1 / 0 & 1 / 1 denotes input / output. In the above figure, there are two transitions from each state based on the value of input, x.
In general, the number of states required in Mealy state machine is less than or equal to the number of states required in Moore state machine. There is an equivalent Moore state machine for each Mealy state machine.



MOORE STATE MACHINE

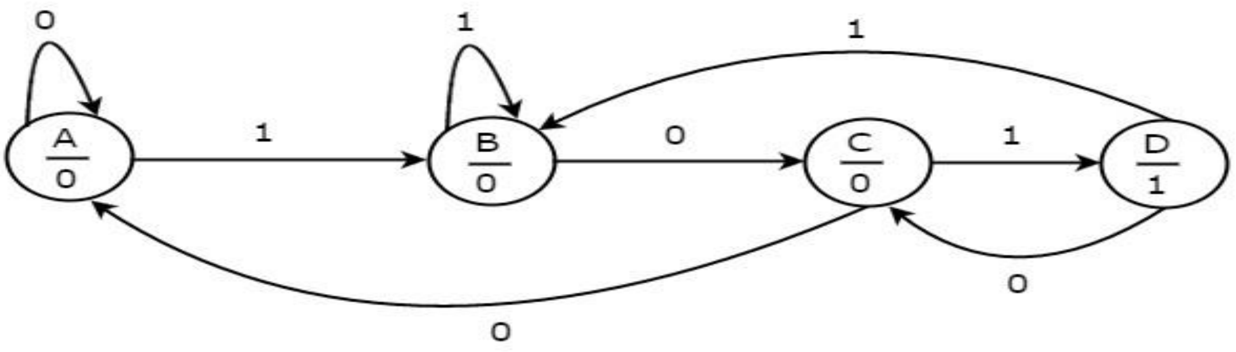
A Finite State Machine is said to be Moore state machine, if outputs depend only on present states. The block diagram of Moore state machine is shown in the following figure.
As shown in figure, there are two parts present in Moore state machine. Those are combinational logic and memory. In this case, the present inputs and present states determine the next states. So, based on next states, Moore state machine produces the outputs. Therefore, the outputs will be valid only after transition of the state.



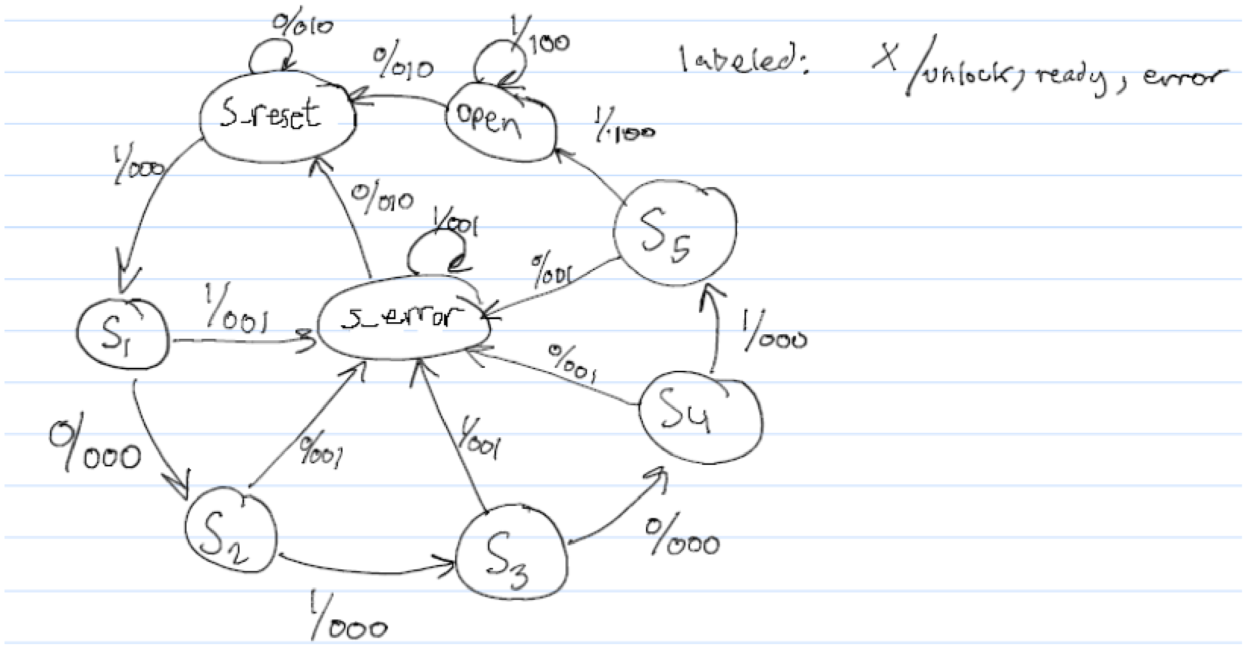
The **state diagram** of Moore state machine is shown in the following figure.

- In the above figure, there are four states, namely A, B, C & D. These states and the respective outputs are labelled inside the circles.
- Here, only the input value is labeled on each transition. In the above figure, there are two transitions from each state based on the value of input, x.
- In general, the number of states required in Moore state machine is more than or equal to the number of states required in Mealy state machine. There is an equivalent Mealy state machine for each Moore state machine. So, based on the requirement we can use one of them.

State diagram



METHODLGY



We are trying to implement a synchronous combination lock that will unlock once it receives "101011" using Verilog. It has one input: x, and three outputs: unlock, ready, and error. Following these rules:

- **Conditions**

- In initial state ready=1. Remains in initial while x=0
- Upon receiving 101011 on x, unlock=1. R
- If in unlock state and x=0, will go to initial state
- In all states other than initial and unlock, if input on x doesn't advance the sequence (101011), error=1.
- Remain in error state while x=1. If x=0 in error state, go to initial state From these rules I created the state diagram is 000000):

IMPLEMENTATION/CODE

SYNCHRONOUS COMBNATIONAL LOCK

```
module lock
(
    input  wire      clock,
    input  wire      reset,
    input  wire      x,
    output reg      ready,
    output reg      unlock,
    output reg      error
);
  reg [2:0] state;
  reg [2:0] next_state;
  localparam s_reset=3'b000,
           s1=3'b001,
           s2=3'b010,
           s3=3'b011,
           s4=3'b100,
           s5=3'b101,
           open=3'b110,
           s_error=3'b111;

  always @ (posedge clock, posedge reset) begin
     if(reset)
      state <= s_reset;
    else
      state <= next_state;
  end
  always @ * begin
    case(state)
      s_reset : if(x==1'b1) begin
                 next_state = s1;
                 unlock = 1'b0;
                 ready = 1'b0;
                 error = 1'b0;
```

```
          end
          else begin
            next_state = s_reset;
            unlock = 1'b0;
            ready = 1'b1;
            error = 1'b0;
          end

s1      : if(x==1'b0) begin
            next_state = s2;
            unlock = 1'b0;
            ready = 1'b0;
            error = 1'b0;
          end
          else begin
            next_state = s_error;
            unlock = 1'b0;
            ready = 1'b0;
            error = 1'b1;
          end

s2      : if(x==1'b1) begin
            next_state = s3;
            unlock = 1'b0;
            ready = 1'b0;
            error = 1'b0;
          end
          else begin
            next_state = s_error;
            unlock = 1'b0;
            ready = 1'b0;
            error = 1'b1;
          end

s3      : if(x==1'b0) begin
            next_state = s4;
            unlock = 1'b0;
            ready = 1'b0;
```

```
              error = 1'b0;
            end
            else begin
              next_state = s_error;
              unlock = 1'b0;
              ready = 1'b0;
              error = 1'b1;
            end

s4          : if(x==1'b1) begin
              next_state = s5;
              unlock = 1'b0;
              ready = 1'b0;
              error = 1'b0;
            end
            else begin
              next_state = s_error;
              unlock = 1'b0;
              ready = 1'b0;
              error = 1'b1;
            end

s5          : if(x==1'b1) begin
              next_state = open;
              unlock = 1'b0;
              ready = 1'b0;
              error = 1'b0;
            end
            else begin
              next_state = s_error;
              unlock = 1'b0;
              ready = 1'b0;
              error = 1'b1;
            end

open          : if(x==1'b0) begin
              next_state = s_reset;
              unlock = 1'b1;
```

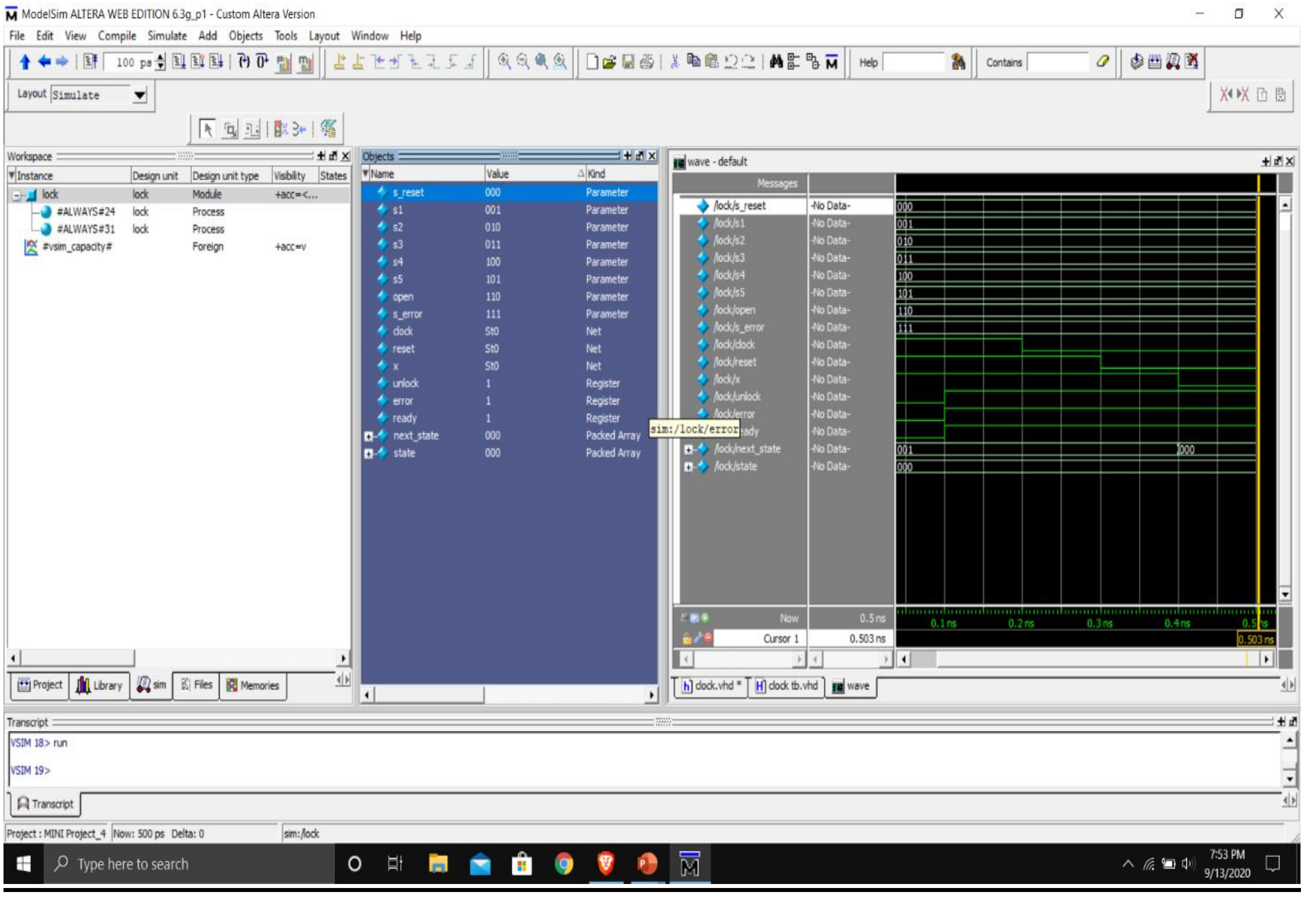
```
                  ready = 1'b0;
                  error = 1'b0;
                end
                else begin
                  next_state = open;
                  unlock = 1'b1;
                  ready = 1'b0;
                  error = 1'b0;
                end

      s_error  : if(x==1'b0) begin
                  next_state = s_reset;
                  unlock = 1'b0;
                  ready = 1'b1;
                  error = 1'b0;
                end
                else begin
                  next_state = s_error;
                  unlock = 1'b0;
                  ready = 1'b0;
                  error = 1'b1;
                end

      default : begin
                  next_state = s_reset;
                  unlock = 1'b0;
                  ready = 1'b1;
                  error = 1'b0;
                end
    endcase
  end

endmodule
```

OUTPUT:



ModelSim ALTERA WEB EDITION 6.3g_p1 - Custom Altera Version

Workspace

Instance	Design unit	Design unit type	Visibility	States
lock	lock	Module	+acc=<...	
#ALWAYS#24	lock	Process		
#ALWAYS#31	lock	Process		
#vsim_capacity#		Foreign	+acc=v	

Objects

Name	Value	Kind
s_reset	000	Parameter
s1	001	Parameter
s2	010	Parameter
s3	011	Parameter
s4	100	Parameter
s5	101	Parameter
open	110	Parameter
s_error	111	Parameter
clock	St0	Net
reset	St0	Net
x	St0	Net
unlock	1	Register
error	1	Register
ready	1	Register
next_state	000	Packed Array
state	000	Packed Array

wave - default

Messages		
/lock/s_reset	-No Data-	000
/lock/s1	-No Data-	001
/lock/s2	-No Data-	010
/lock/s3	-No Data-	011
/lock/s4	-No Data-	100
/lock/s5	-No Data-	101
/lock/open	-No Data-	110
/lock/s_error	-No Data-	111
/lock/clock	-No Data-	
/lock/reset	-No Data-	
/lock/x	-No Data-	
/lock/unlock	-No Data-	
/lock/error	-No Data-	
/lock/ready	-No Data-	
/lock/next_state	-No Data-	001 000
/lock/state	-No Data-	000
Now	0.5 ns	
Cursor 1	0.503 ns	0.503 ns

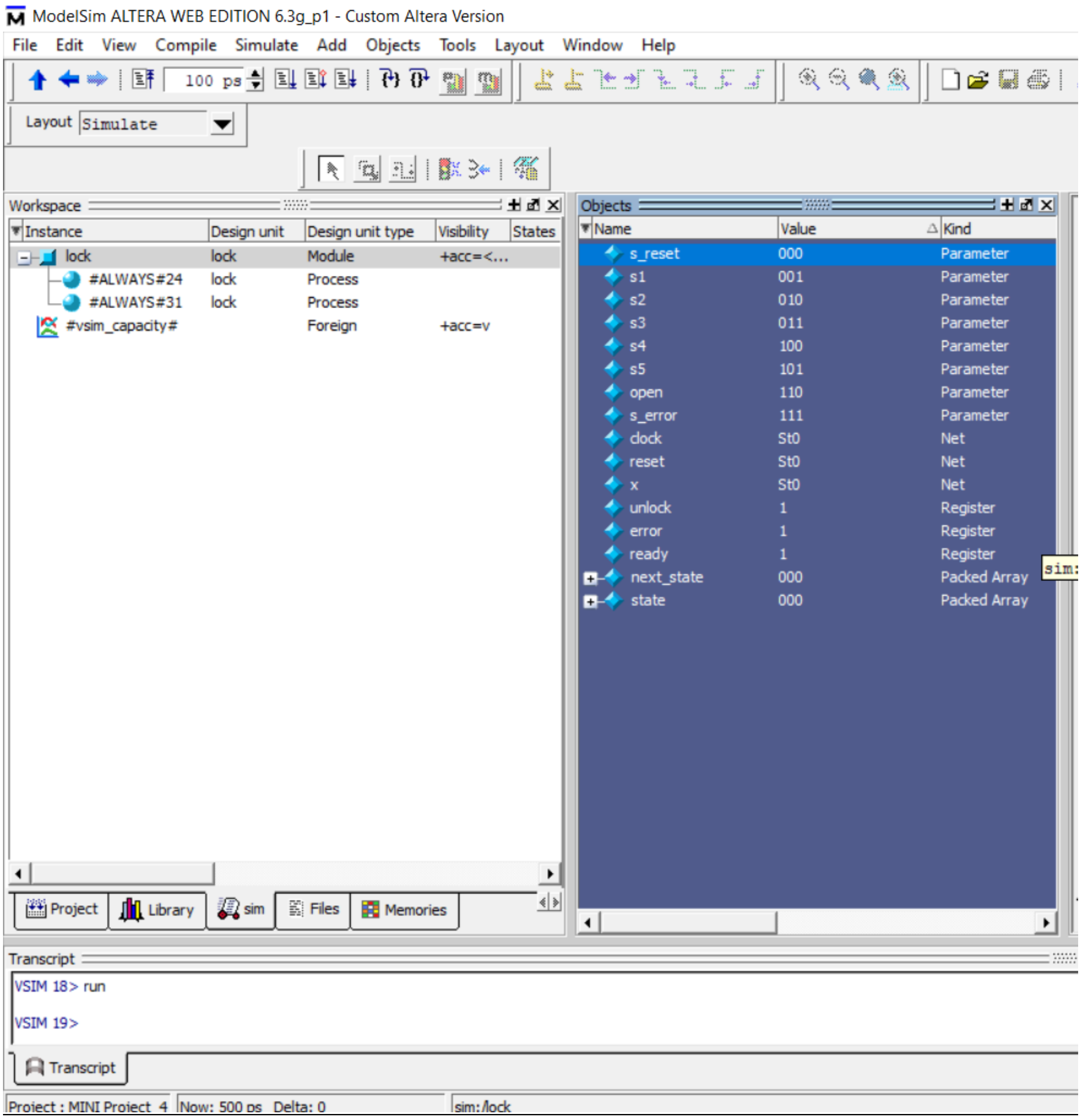
sim:/lock/error

Transcript

```
VSIM 18> run

VSIM 19>
```

Project : MINI Project_4 Now: 500 ps Delta: 0 sim:/lock


ModelSim ALTERA WEB EDITION 6.3g_p1 - Custom Altera Version
File Edit View Compile Simulate Add Objects Tools Layout Window Help
100 ps
Layout Simulate
Workspace
Project Library sim Files Memories
Objects
Transcript

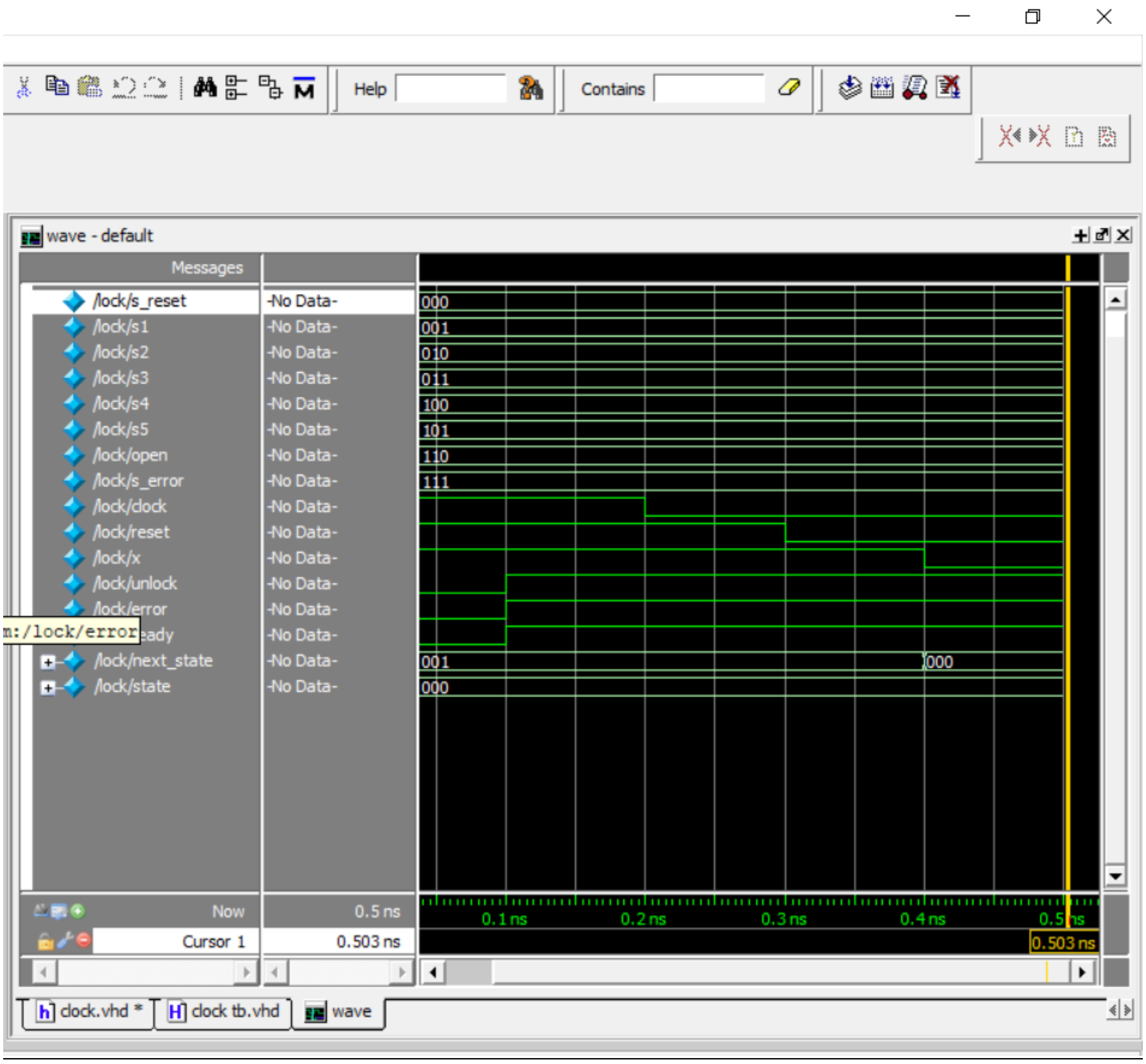
Instance	Design unit	Design unit type	Visibility	States
lock	lock	Module	+acc=<...	
#ALWAYS#24	lock	Process		
#ALWAYS#31	lock	Process		
#vsim_capacity#		Foreign	+acc=v	

Name	Value	Kind
s_reset	000	Parameter
s1	001	Parameter
s2	010	Parameter
s3	011	Parameter
s4	100	Parameter
s5	101	Parameter
open	110	Parameter
s_error	111	Parameter
clock	St0	Net
reset	St0	Net
x	St0	Net
unlock	1	Register
error	1	Register
ready	1	Register
next_state	000	Packed Array
state	000	Packed Array

```
VSIM 18> run

VSIM 19>
```

Project : MINI Project 4 Now: 500 ps Delta: 0 sim:/lock



Messages		
/lock/s_reset	-No Data-	000
/lock/s1	-No Data-	001
/lock/s2	-No Data-	010
/lock/s3	-No Data-	011
/lock/s4	-No Data-	100
/lock/s5	-No Data-	101
/lock/open	-No Data-	110
/lock/s_error	-No Data-	111
/lock/clock	-No Data-	
/lock/reset	-No Data-	
/lock/x	-No Data-	
/lock/unlock	-No Data-	
/lock/error	-No Data-	
m:/lock/error ready	-No Data-	
/lock/next_state	-No Data-	001 000
/lock/state	-No Data-	000

```
Now       0.5 ns     0.1 ns   0.2 ns   0.3 ns   0.4 ns   0.5 ns
Cursor 1  0.503 ns                                       0.503 ns
```

Conclusion: The increasing rate of crime, attacks by thieves, intruders and vandals, despite all forms of security gadgets and locks still need the attention of researchers to find a permanent solution to the well being of lives and properties of individuals. As such, we design a cheap and effective security system for buildings, cars, safes, doors and gates, so as to prevent unauthorized person from having access to ones properties through the use of codes, we therefore experiment the application of electronic devices as locks. The system works by combination lock which was divided into units and each unit designed separately before being coupled to form a whole functional system. Twenty tests were conducted to ascertain the reliability of the design with the first eight combinations being four in number, the next seven tests being five and the last five combinations being six. This was done because of the incorporation of 2 dummy switches in the combinations. From the result obtained, combinations 8, 11, 13 gave the correct output combination. However, 8 as the actual combination gave the required output. The general operation of the system and performance is dependent on the key combinations. The overall system was constructed and tested and it works perfectlyApplications: -

- **Common uses for digital combination locks include:**
- Enclosures
- Metal office furniture
- Wooden office furniture.
- Lockers
- Postal applications.

REFERENCE: -

- https://electronics.stackexchange.com/questions/225933/synchronuous-combination-lock
- https://vlsicoding.blogspot.com/2015/05/verilog-code-for-electronic-combination-lock-fsm.html
- https://www.scribd.com/doc/143125507/Design-and-Implementation-of-Digital-Code-LockUsing-Vhdl