

Student Name: Omar Magdy Shaaban

<CESS>

<Project Phase 1>

UEL Module Title: Engineering Systems

Module Code: EG7423

ASU Course Title: Electronic Design Automation

Course Code: CSE215

Date of Submission: 15/4/2019

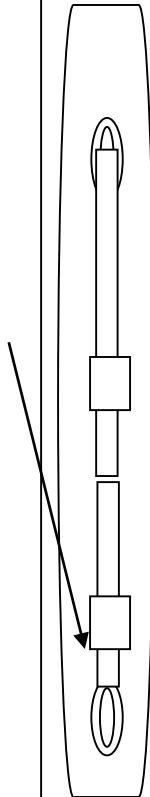
Lecturer: Dr. Mohamed Dessouky

UELID: u1927296

ASUID: 16P6034

Email: 16P6034@eng.asu.edu.eg

Using file
fastener



Abstract

This document demonstrates the finite state machine for a door keypad. The keypad is supposed to open under certain conditions and in between it transfers through some states to reach the final output (either the door will open or the alarm will be triggered). The following topics are covered: The suggested finite state machines models, the most suitable one chosen, suitable test cases, finite state machine code and testbench, and simulation outputs.

Contents

Figures.....	3
1. Proposed FSM models	4
2. Test Strategy	6
3. Implemented FSM.....	8
4. Testbench.....	11
5. Simulation Outputs	16
6. References	17

Figures

Figure 1: Moore fsm.....	4
Figure 2: Mealy fsm.....	5
Figure 3: Waveform (1).....	16
Figure 4: Waveform (2).....	16
Figure 5: Transcript window	17

1. Proposed FSM models

a) Moore: (Note: outputs are door and alarm respectively, daytime is entered first in state 0, then we just repeat mentioning it in transitions to keep a standard notation for showing transitions. Inputs are daytime and code respectively).

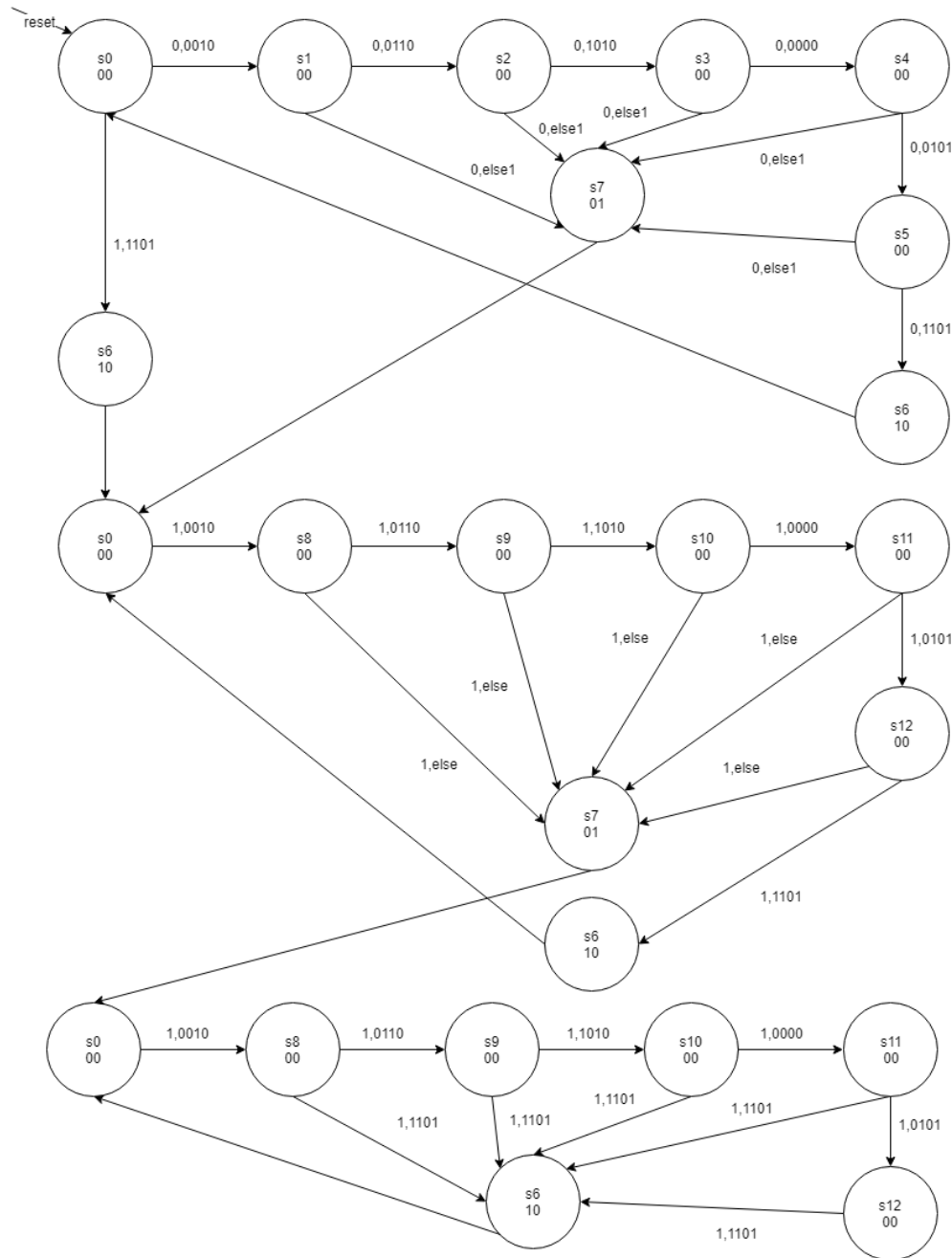


Figure 1: Moore fsm

b) Mealy:

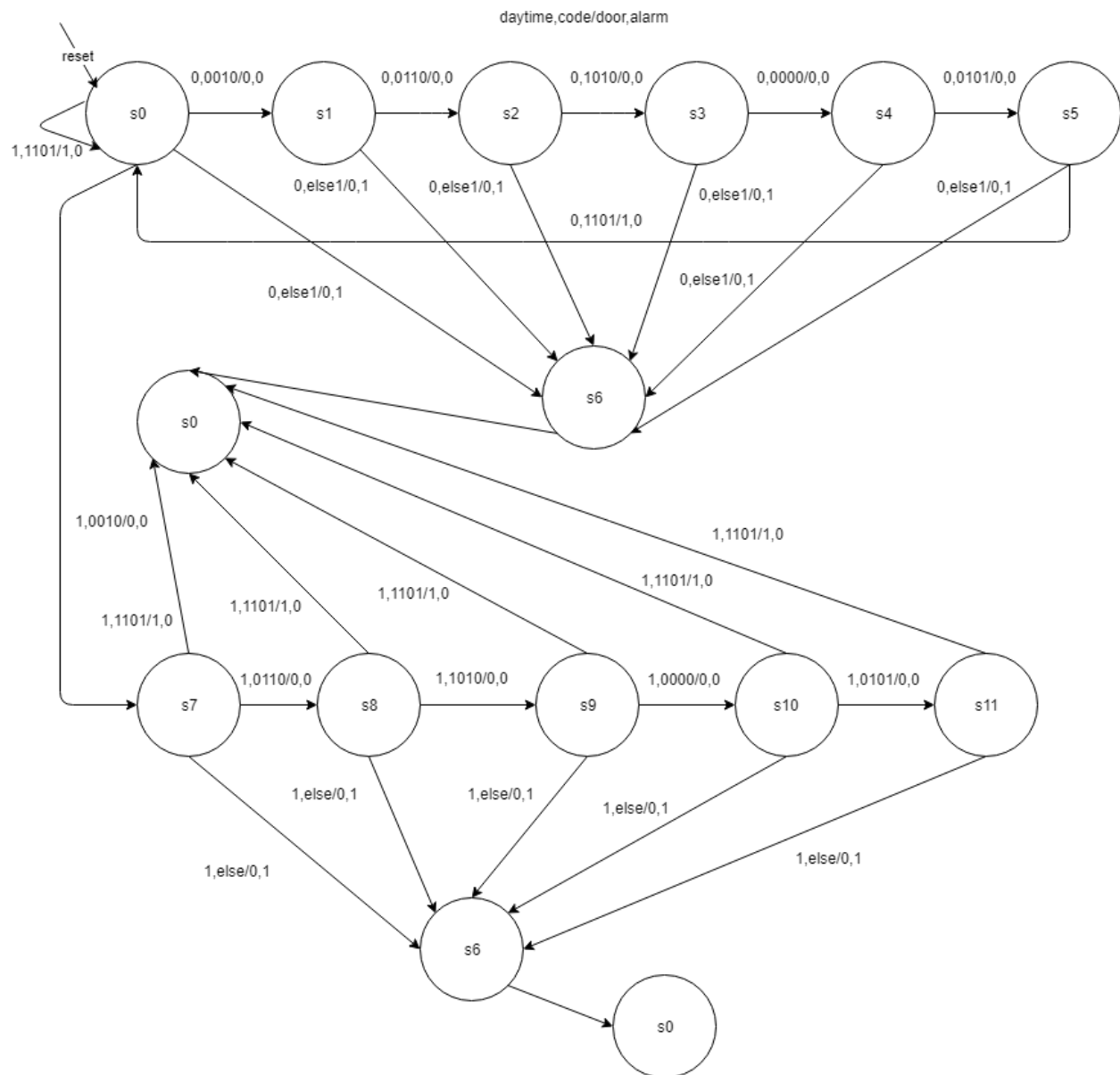


Figure 2: Mealy fsm

We decided that the most suitable finite state machine to work with is the Mealy model, as it has one less states and this should be better in terms of optimization when reducing logic gates. Also, the Mealy fsm is known for having earlier outputs which is also a good reason to choose Mealy over Moore.

2. Test Strategy

[3]

Tested Feature	VDD	VSS	Daytime	Reset	Clk	Delay	Code	Door	Alarm
Keypad	1	0	0	1	1	10 ns	0000	0	0
					0		0000	0	0
				0	1		0010	0	0
					0		0010	0	0
					1		0110	0	0
					0		0110	0	0
					1		1010	0	0
					0		1010	0	0
					1		0000	0	0
					0		0000	0	0
					1		0101	0	0
					0		0101	0	0
					1		1101	1	0
					0		1101	1	0
				1	1		1101	0	0
					0		1101	0	0
				0	1		0010	0	0
					0		0010	0	0
					1		0110	0	0
					0		0110	0	0
					1		1010	0	0
					0		1010	0	0
					1		1101	0	1
					0		1101	0	1
				1	1		1101	0	0
					0		1101	0	0
					1		1101	1	0
					0		1101	1	0
					1		1101	0	0
					0		1101	0	0
					1		0010	0	0
					0		0010	0	0
				0	1		0110	0	0
					0		0110	0	0
					1		0110	0	0
					0		0110	0	0

					1		1010	0	0
					0		1010	0	0
					1		0000	0	0
					0		0000	0	0
					1		0101	0	0
					0		0101	0	0
					1		1101	1	0
					0		1101	1	0
				1	1		1101	0	0
					0		1101	0	0
				0	1		0010	0	0
					0		0010	0	0
					1		0110	0	0
					0		0110	0	0
					1		1010	0	0
					0		1010	0	0
					1		0000	0	0
					0		0000	0	0
					1		1101	1	0
					0		1101	1	0
				1	1		1101	0	0
					0		1101	0	0
				0	1		0010	0	0
					0		0010	0	0
					1		0110	0	0
					0		0110	0	0
				1	1		0110	0	0
					0		0110	0	0
					1		0110	0	0
					0		0110	0	0
				0	1		0010	0	0
					0		0010	0	0
					1		0110	0	0
					0		0110	0	0
					1		1011	0	1
					0		1011	0	1

3. Implemented FSM

[1]

```
entity doorpass is
    port(
        daytime: in bit;
        reset: in bit;
        code: in bit_vector(3 downto 0);
        vdd: in bit;
        vss: in bit;
        clk: in bit;
        door: out bit;
        alarm: out bit
    );
end entity;

architecture fsm of doorpass is
    type state is
        (s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11);
    signal cs : state;
    signal ns : state;
-- Synthesis directives :
-- pragma current_state cs
-- pragma next_state ns
-- pragma clock clk
begin

process(clk)
begin
    if(clk = '1' and clk'event)then
        cs <= ns;
    end if;
end process;

process (cs, code, reset)
begin
    if (reset='1') then
        door <= '0'; alarm <= '0'; ns <= s0;
    else
        case cs is
            when s0 => if(daytime = '0') then
                if(code = "0010") then
                    door <= '0'; alarm <= '0'; ns <= s1;
                else
                    door <= '0'; alarm <= '1'; ns <= s6;
                end if;
            end if;
        end case;
    end if;
end process;
```

```

        end if;
    else
        if(code = "1101") then
            door <= '1'; alarm <= '0'; ns <= s0;
        else
            if(code = "0010") then
                door <= '0'; alarm <= '0'; ns <= s7;
            else
                door <= '0'; alarm <= '1'; ns <= s6;
            end if;
        end if;
    end if;
when s1 =>
    if(code = "0110") then
        door <= '0'; alarm <= '0'; ns <= s2;
    else
        door <= '0'; alarm <= '1'; ns <= s6;
    end if;
when s2 =>
    if(code = "1010") then
        door <= '0'; alarm <= '0'; ns <= s3;
    else
        door <= '0'; alarm <= '1'; ns <= s6;
    end if;
when s3 =>
    if(code = "0000") then
        door <= '0'; alarm <= '0'; ns <= s4;
    else
        door <= '0'; alarm <= '1'; ns <= s6;
    end if;
when s4 =>
    if(code = "0101") then
        door <= '0'; alarm <= '0'; ns <= s5;
    else
        door <= '0'; alarm <= '1'; ns <= s6;
    end if;
when s5 =>
    if(code = "1101") then
        door <= '1'; alarm <= '0'; ns <= s0;
    else
        door <= '0'; alarm <= '1'; ns <= s6;
    end if;
when s6 =>
    door <= '0'; alarm <= '0'; ns <= s0;
when s7 => if(code = "1101") then

```

```

        door <= '1'; alarm <= '0'; ns <= s0;
    else if(code = "0110") then
        door <= '0'; alarm <= '0'; ns <= s8;
    else
        door <= '0'; alarm <= '1'; ns <= s6;
    end if;
end if;
when s8 => if(code = "1101") then
    door <= '1'; alarm <= '0'; ns <= s0;
else if(code = "1010") then
    door <= '0'; alarm <= '0'; ns <= s9;
else
    door <= '0'; alarm <= '1'; ns <= s6;
end if;
end if;
when s9 => if(code = "1101") then
    door <= '1'; alarm <= '0'; ns <= s0;
else if(code = "0000") then
    door <= '0'; alarm <= '0'; ns <= s10;
else
    door <= '0'; alarm <= '1'; ns <= s6;
end if;
end if;
when s10 => if(code = "1101") then
    door <= '1'; alarm <= '0'; ns <= s0;
else if(code = "0101") then
    door <= '0'; alarm <= '0'; ns <= s11;
else
    door <= '0'; alarm <= '1'; ns <= s6;
end if;
end if;
when s11 => if(code = "1101") then
    door <= '1'; alarm <= '0'; ns <= s0;
else
    door <= '0'; alarm <= '1'; ns <= s6;
end if;

end case ;
end if;
end process;
end architecture;

```

4. Testbench

[2]

```
entity testbench2 is
end entity;

architecture mealy2 of testbench is
  component doorpass is
    port(
      daytime: in bit;    reset: in bit;
      code: in bit_vector(3 downto 0);    vdd: in bit;
      vss: in bit;    clk: in bit;
      door: out bit;    alarm: out bit
    );
  end component;

  for dp : doorpass use entity work.doorpass(fsm);

  signal clk : bit := '0';
  signal reset : bit := '1';
  signal daytime : bit := '0';
  signal code : bit_vector(3 downto 0) := "0000";
  signal vdd : bit := '1';
  signal vss : bit := '0';
  signal door : bit := '0';
  signal alarm : bit := '0';
  constant delay_time : time := 10 ns;

begin
  dp : doorpass port map(daytime, reset, code, vdd, vss, clk, door, alarm);

  process is
  begin
    clk <= '1';
    wait for delay_time/2;
    clk <= '0';
    wait for delay_time/2;
  end process;

  process is
  -- This procedure is used when current outputs are different from the next
  outputs, ex: current
  -- outputs, door = 1, alarm = 0, next outputs will be door = 0 and alarm = 0, to
  avoid assert errors
  -- at zero delay time
```

```

procedure reset1 is
begin
    reset <= '1';
    wait for 1 ns;
    assert door = '0' and alarm = '0'
    report "Next state s0, correct outputs: door = 0 and alarm = 0"
    severity error;
    wait for delay_time - 1 ns;

end procedure;
-- Resetting without change in delay, as current and next outputs don't
differ
procedure reset2 is
begin
    reset <= '1';
    assert door = '0' and alarm = '0'
    report "Next state s0, correct outputs: door = 0 and alarm = 0"
    severity error;
    wait for delay_time;

end procedure;

procedure success is
begin
    code <= "1101";
    wait for 1 ns;
    assert door = '1' and alarm = '0'
    report "Next state s0, correct outputs: door = 1 and alarm = 0"
    severity error;
    wait for delay_time - 1 ns;

end procedure;

procedure failure is
begin
    wait for 1 ns;
    assert door = '0' and alarm = '1'
    report "Next state s6, correct outputs: door = 0 and alarm = 1"
    severity error;
    wait for delay_time - 1 ns;

end procedure;
-- Up to 2 characters, 26
procedure partialcodedaylight is
begin

```

```

reset <= '0'; code <= "0010";
assert door = '0' and alarm = '0'
report "Next state s7, correct outputs: door = 0 and alarm = 0"
severity error;
wait for delay_time;

code <= "0110";
assert door = '0' and alarm = '0'
report "Next state s8, correct outputs: door = 0 and alarm = 0"
severity error;
wait for delay_time;

end procedure;
--Up to 2 characters, A0
procedure partialcodedaylight2 is
begin
code <= "1010";
assert door = '0' and alarm = '0'
report "Next state s9, correct outputs: door = 0 and alarm = 0"
severity error;
wait for delay_time;

code <= "0000";
assert door = '0' and alarm = '0'
report "Next state s10, correct outputs: door = 0 and alarm = 0"
severity error;
wait for delay_time;

end procedure;
--Up to 3 characters, 26A
procedure partialcodenight is
begin
reset <= '0'; code <= "0010";
assert door = '0' and alarm = '0'
report "Next state s1, correct outputs: door = 0 and alarm = 0"
severity error;
wait for delay_time;

code <= "0110";
assert door = '0' and alarm = '0'
report "Next state s2, correct outputs: door = 0 and alarm = 0"
severity error;
wait for delay_time;

code <= "1010";

```

```

    assert door = '0' and alarm = '0'
    report "Next state s3, correct outputs: door = 0 and alarm = 0"
    severity error;
    wait for delay_time;

end procedure;

begin
-- Test case 1: Enter full correct code at night, then press "0"
    reset2;
    partialcodenight;
    code <= "0000";
    assert door = '0' and alarm = '0'
    report "Next state s4, correct outputs: door = 0 and alarm = 0"
    severity error;
    wait for delay_time;

    code <= "0101";
    assert door = '0' and alarm = '0'
    report "Next state s5, correct outputs: door = 0 and alarm = 0"
    severity error;
    wait for delay_time;
    success;

--Test case 2: Enter partial correct code at night, then press "0"
    reset1;
    partialcodenight;
    code <= "1101";
    failure;

-- Test case 3: Press "0" directly in daylight
    daytime <= '1'; reset1;
    reset <= '0';
    success;

-- Test case 4: Enter full correct code at daylight, then press "0"
    reset1;
    reset <= '0';
    partialcodedaylight;
    partialcodedaylight2;

    code <= "0101";
    assert door = '0' and alarm = '0'
    report "Next state s11, correct outputs: door = 0 and alarm = 0"
    severity error;

```

```

wait for delay_time;
success;

-- Test case 5: Enter partial correct code at daylight, then press "0"
reset1;
partialcodedaylight;
partialcodedaylight2;
success;

-- Test case 6: Enter partial correct code at daylight, then reset
reset1;
partialcodedaylight;
reset2;

-- Test case 7: Enter partial wrong code at daylight
reset2;
partialcodedaylight;
code <= "1011";
failure;

wait;
end process;
end architecture;

```

Note: This is a modified testbench (hence named testbench2). We've included the first testbench in which no procedures are used, in case there is difficulty tracing the procedures in this testbench.

5. Simulation Outputs

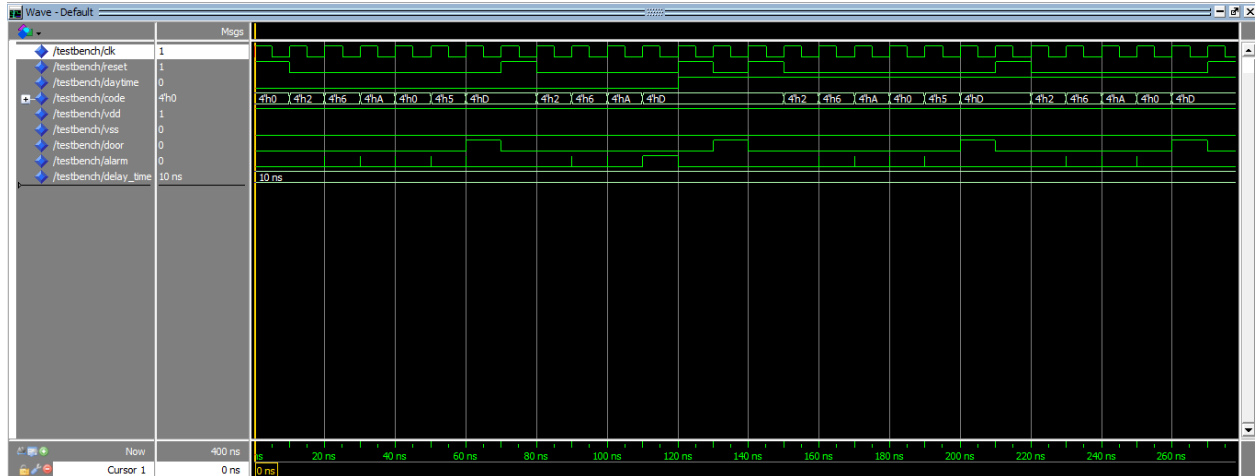
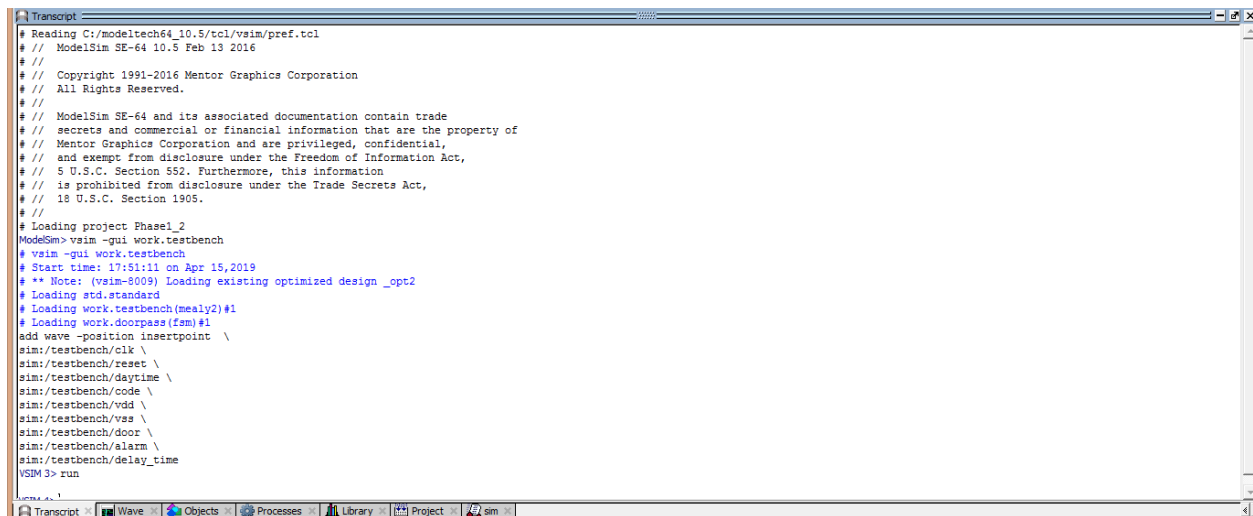


Figure 3: Waveform (1)



Figure 4: Waveform (2)



```
# Reading C:/modeltech64_10.5/tcl/vsim/pref.tcl
# // ModelSim SE-64 10.5 Feb 13 2016
# //
# // Copyright 1991-2016 Mentor Graphics Corporation
# // All Rights Reserved.
# //
# // ModelSim SE-64 and its associated documentation contain trade
# // secrets and commercial or financial information that are the property of
# // Mentor Graphics Corporation and are privileged, confidential,
# // and exempt from disclosure under the Freedom of Information Act,
# // 5 U.S.C. Section 552. Furthermore, this information
# // is prohibited from disclosure under the Trade Secrets Act,
# // 18 U.S.C. Section 1905.
# //
# Loading project Phase1_2
ModelSim> vsim -gui work.testbench
# vsim -gui work.testbench
# Start time: 17:51:11 on Apr 15, 2019
# ** Note: (vsim-8009) Loading existing optimized design _opt2
# Loading std.standard
# Loading work.testbench(mealy2)#1
# Loading work.doorpass(fsm)#1
add wave -position insertpoint \
sim:/testbench/clock \
sim:/testbench/reset \
sim:/testbench/daytime \
sim:/testbench/code \
sim:/testbench/vdd \
sim:/testbench/vss \
sim:/testbench/door \
sim:/testbench/alarm \
sim:/testbench/delay_time
VSIOM 3> run
```

Figure 5: Transcript window

6. References

- 1) Phase 1
- 2) Phase 1_tb2
- 3) ModelSim Testbenches Lecture 9
- 4) Phase 1_tb* (additional)