Electronic security code lock system Project

Created by

Medhat Ashraf Abdo Alhaddad

Table OF Contents

Introduction:-

In this project, you'll go through the high level design of a FSM.

We will design a digital access control system. While this system could be designed as combinational circuit we used sequential circuit to review the FSM.

We used Moore FSM as the output is not a prerequisite for state transition. And it is easier to design.

FSM Diagram:-

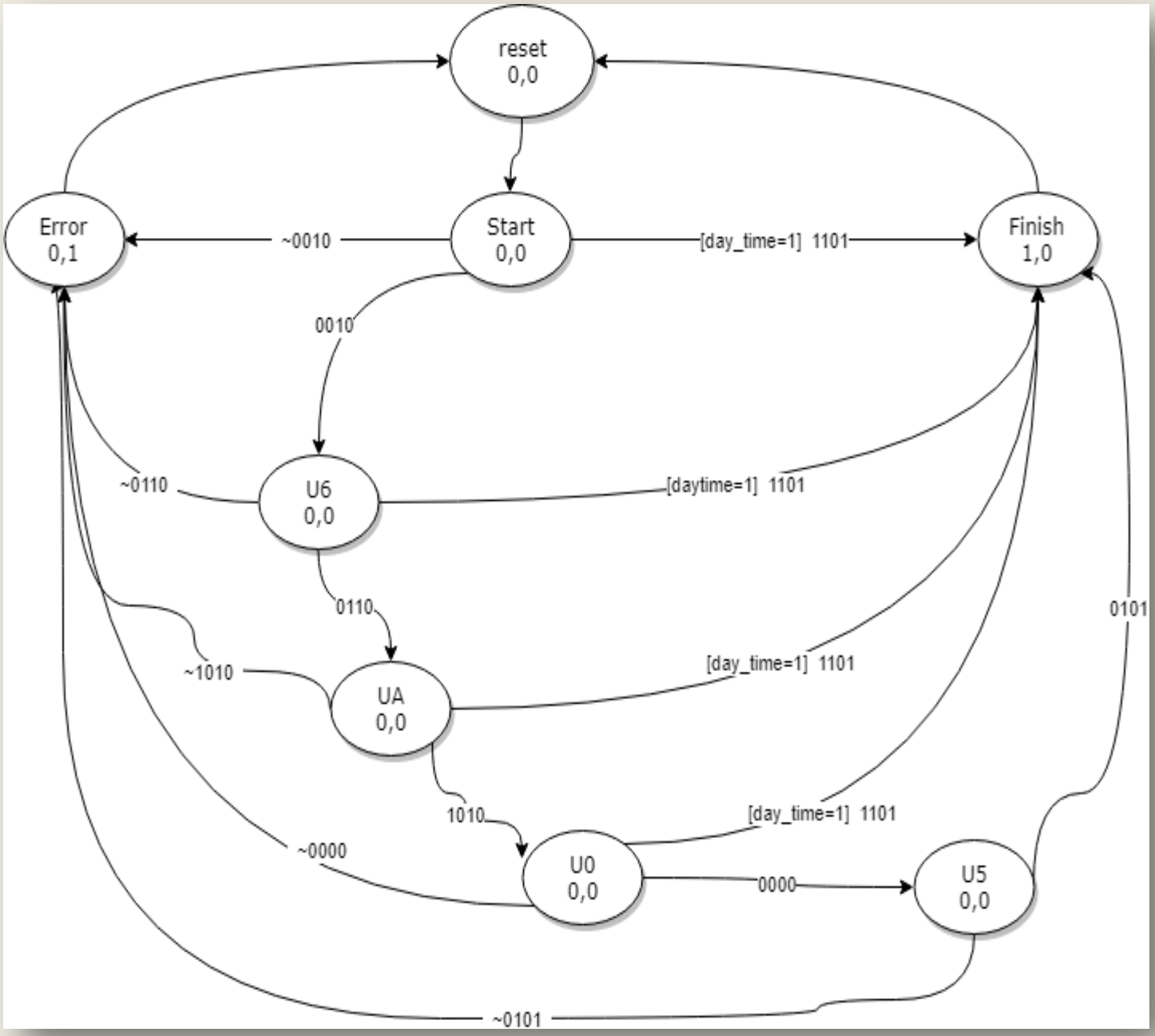

Figure 1: FSM Diagram

Test Bench test strategy table:-

Tested Feature	code	current state	day time	clock	delay	exp. Door	exp. Alarm	next state
successful scenario	0010	Start	0	0	10 ns	0	0	U6
	0010	U6	0	1		0	0	UA
	0110	U6	0	0		0	0	UA
	0110	UA	0	1		0	0	U0
	1010	UA	0	0		0	0	U0
	1010	U0	0	1		0	0	U5
	0000	U0	0	0		0	0	U5
	0000	U5	0	1		0	0	finish
	0101	U5	0	0		0	0	finish
	0101	Finish	0	1		1	0	reset
	0000	Finish	0	0		1	0	reset
	0000	Reset	0	1		0	0	start
	0000	Reset	0	0		0	0	start
	0000	Start	0	1		0	0	…….
daytime scenario	1101	Start	1	0	10 ns	0	0	finish
	1101	Finish	1	1		1	0	reset
	1101	Finish	1	0		1	0	reset
	0000	Reset	1	1		1	0	start
	0000	Reset	1	0		1	0	start
	0000	Start	1	1		0	0	…….
	0010	Start	1	0		0	0	U6
	0010	U6	1	1		0	0	finish
	1101	U6	1	0		0	0	finish
	1101	Finish	1	1		1	0	reset
	0000	Finish	1	0		1	0	reset
	0000	Reset	1	1		1	0	start
	0000	Reset	1	0		1	0	start
	0000	Start	1	1		0	0	…….
unsuccessful scenario	0010	Start	0	0	10 ns	0	0	U6
	0010	U6	0	1		0	0	error
	1111	U6	1	0		0	0	error
	1111	Error	1	1		0	1	reset
	0000	Error	1	0		0	1	reset
	0000	Reset	0	1		1	0	start
	0000	Reset	0	0		1	0	start
	0000	Start	0	1		0	0	…….

Figure 2: Test Strategy Table

Source Code:-

- Entity and used library

```
library ieee;
library STD;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use std.standard.all;

entity fsm is
	port (
		clk	: in  std_logic                    := '0';
		vdd	: in  std_logic                    := '1';
		vss	: in  std_logic                    := '0';
		code	: in  std_logic_vector(3 downto 0) := "0000";
		daytime : in  std_logic                    := '0';
		rst	: in  std_logic                    := '0';
		door	: out std_logic                    := '0';
		alarm	: out std_logic                    := '0'
	);

 end entity fsm;
```

Behavioral architecture:-

```
-- Architecture Declaration

ARCHITECTURE moore OF fsm IS
 type state_type is (reset, start, U6, UA, U0, U5, finish, error);
 SIGNAL current_state : state_type := start;
 SIGNAL next_state    : state_type := start;
BEGIN
 cs : PROCESS(clk)
 BEGIN
       IF (clk = '1' and not clk'stable) THEN
             current_state <= next_state;
       END IF;
 END PROCESS cs;

 ns : PROCESS(rst, current_state, code, daytime)
 BEGIN
       IF rst = '1' THEN
             next_state <= reset;
       end if;
       case current_state is
             when start =>
                   if code = "0010" then
                         next_state <= U6;
                   elsif (daytime = '1' and code = "1101") then
                         next_state <= finish;
                   else
                         next_state <= error;
                   end if;
             when U6 =>
                   if code = "0110" then
                         next_state <= UA;
                   elsif (daytime = '1' and code = "1101") then
                         next_state <= finish;
                   else
                         next_state <= error;
                   end if;
             when UA =>
                   if code = "1010" then
                         next_state <= U0;
                   elsif (daytime = '1' and code = "1101") then
                         next_state <= finish;
                   else
                         next_state <= error;
                   end if;
             when U0 =>
                   if code = "0000" then
                         next_state <= U5;
                   elsif (daytime = '1' and code = "1101") then
                         next_state <= finish;
                   else
                         next_state <= error;
                   end if;
             when U5 =>
                   if code = "0101" then
                         next_state <= finish;
                   elsif (daytime = '1' and code = "1101") then
```

```
                        next_state <= finish;
                  else
                        next_state <= error;
                  end if;
            when finish =>
                  door       <= '1';
                  next_state <= reset;
            when error =>
                  alarm      <= '1';
                  next_state <= reset;
            when reset =>
                  door       <= '0';
                  alarm      <= '0';
                  next_state <= start;
            when others =>
                  assert (false) report "illegal state" severity error;

      end case;
 END PROCESS ns;
END ARCHITECTURE moore;
```

Test Bench:-

- Behavioral Testing

```
ENTITY testbench IS
END ENTITY testbench;

ARCHITECTURE test OF testbench IS

	COMPONENT fsm IS
		port(
			clk	: in  std_logic;
			vdd	: in  std_logic;
			vss	: in  std_logic;
			code	: in  std_logic_vector(3 downto 0);
			daytime : in  std_logic;
			rst	: in  std_logic;
			door	: out std_logic;
			alarm	: out std_logic
		);
	END COMPONENT fsm;

	FOR dut : fsm USE ENTITY WORK.fsm(moore);

	SIGNAL clk	: std_logic                    := '0';
	SIGNAL vdd	: std_logic                    := '1';
	SIGNAL vss	: std_logic                    := '0';
	SIGNAL code	: std_logic_vector(3 downto 0) := "0000";
	signal daytime : std_logic                   := '0';
	SIGNAL rst	: std_logic                    := '0';
	SIGNAL door	: std_logic                    := '0';
	signal alarm	: std_logic                    := '0';

	-- Constants and Clock period definitions
	constant clk_period : time := 20 ns;
BEGIN

	-- Instantiate the Device Under Test (DUT)
	dut : fsm PORT MAP(clk, vdd, vss, code, daytime, rst, door, alarm);

	-- Clock process definitions( clock with 50% duty cycle )
	clk_process : process
	begin
		clk <= '0';
		wait for clk_period / 2;
		clk <= '1';
		wait for clk_period / 2;
	end process;

	-- Stimulus process, refer to clock signal
	stim_proc : PROCESS IS
	BEGIN
		code	<= "0010";
		daytime <= '0';
		rst	<= '0';
```

```
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "0110";
daytime <= '0';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "1010";
daytime <= '0';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "0000";
daytime <= '0';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "0101";
daytime <= '0';
rst      <= '0';
wait for 11 ns;
Assert (door = '1' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "0000";
daytime <= '0';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "0000";
daytime <= '0';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "1101";
daytime <= '1';
rst      <= '0';
wait for 11 ns;
Assert (door = '1' and alarm = '0') Report "wrong exp. output"
Severity Error;
```

```
wait for 9 ns;

code     <= "0000";
daytime <= '0';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "0010";
daytime <= '0';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "0010";
daytime <= '1';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "1101";
daytime <= '1';
rst      <= '0';
wait for 11 ns;
Assert (door = '1' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "0000";
daytime <= '0';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "0000";
daytime <= '0';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '0') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "1111";
daytime <= '1';
rst      <= '0';
wait for 11 ns;
Assert (door = '0' and alarm = '1') Report "wrong exp. output"
Severity Error;
wait for 9 ns;

code     <= "0000";
```

```
            daytime <= '0';
            rst      <= '0';
            wait for 11 ns;
            Assert (door = '0' and alarm = '0') Report "wrong exp. output"
            Severity Error;
            wait for 9 ns;

            code     <= "0000";
            daytime <= '0';
            rst      <= '0';
            wait for 11 ns;
            Assert (door = '0' and alarm = '0') Report "wrong exp. output"
            Severity Error;
            wait for 9 ns;

            WAIT;                          -- stop process simulation run

      END PROCESS;
END ARCHITECTURE test;
```

Modalism output:-

- Simulation

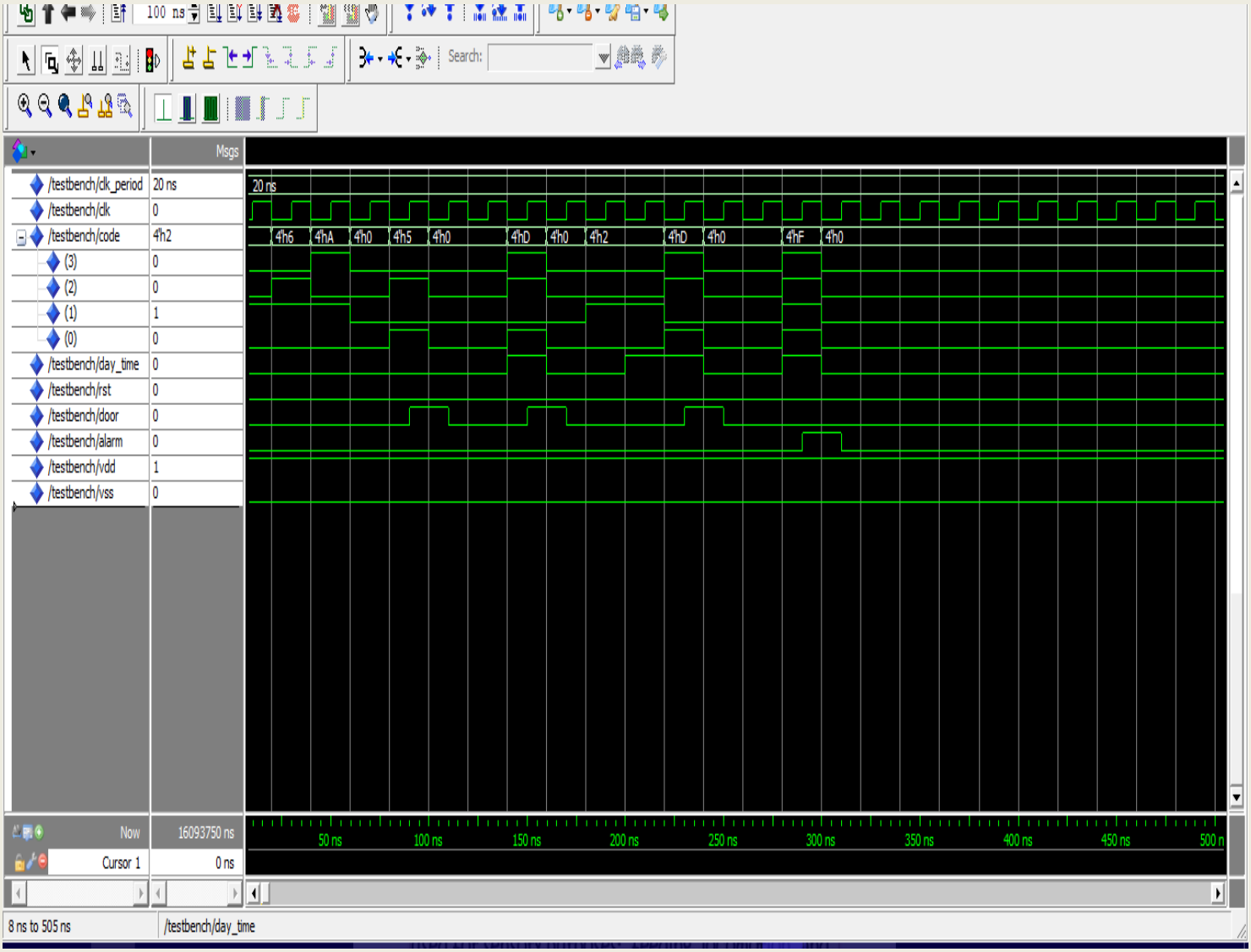

Figure 3: Modalism simulation

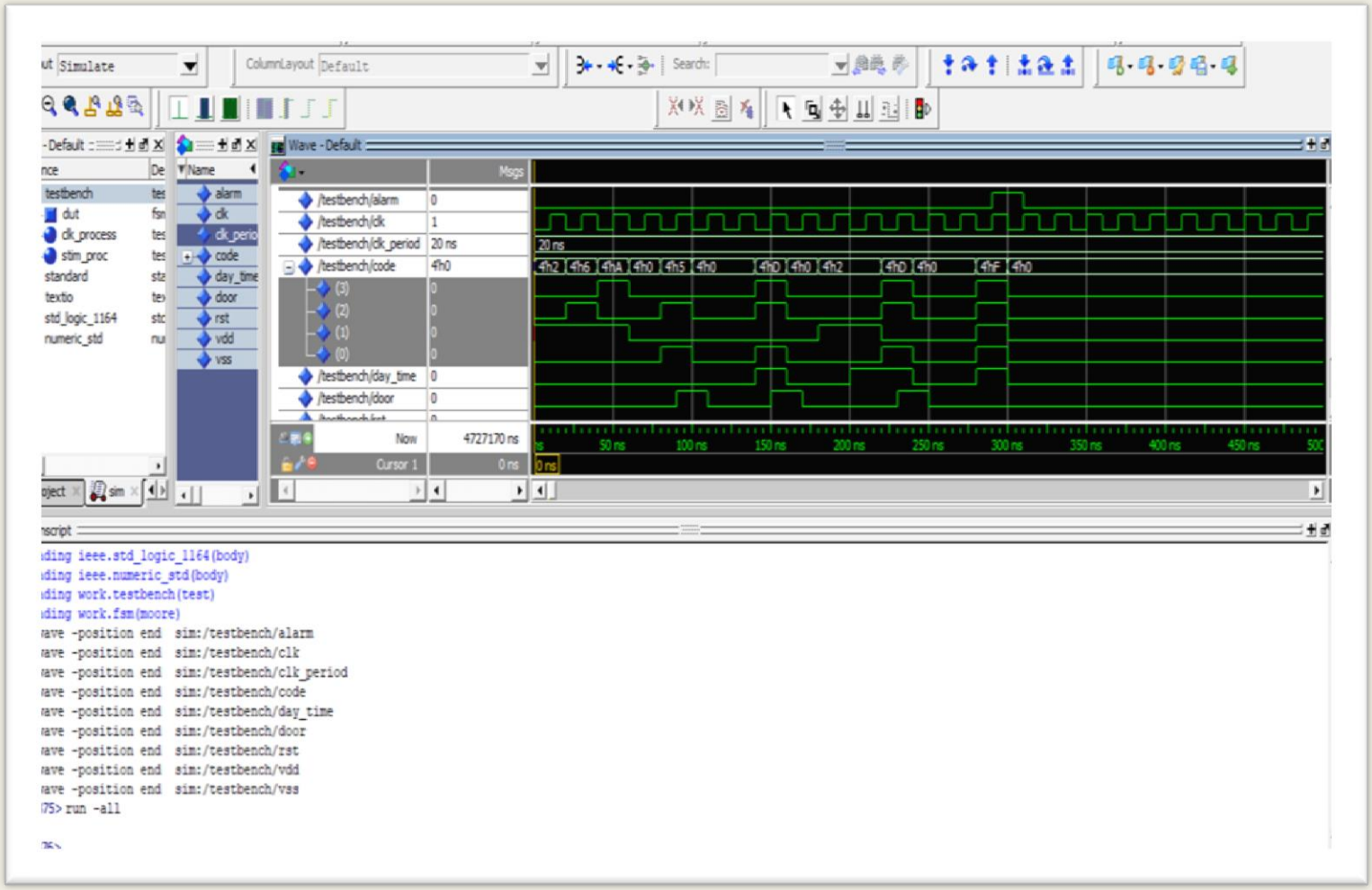

Figure 4: Modelsim output

References:-

- EDA: W. Wolf, “Modern VLSI Design”, PEARSON, Prentice Hall. 3rd Edition.
- EDA: M. Smith, "Application-Specific Integrated Circuits", Addison Wesley
- Design: Neil H.E. Westeand David Harris, "CMOS VLSI Design", Third Edition, Pearson, Addison Wesley
- Design: J. Rabaey, A. Chandrakasanand B. Nikolic. "Digital Integrated Circuits: A Design Perspective", Prentice-Hall. 2nd Edition.
- M216A: Design of VLSI Circuits and Systems. By C.K. Ken Yang. U. of California at Los Angeles
- EE271: Introduction to VLSI Systems. By Mark Horowitz. Stanford U.
- Franck Wajsbürtand Jean-Paul Chaput, “Outils de CAO pour VLSI -Flot de Conception VLSI Alliance”, University of Paris VI.
- Franck Wajsbürtand Jean-Paul Chaput, “Outils de CAO pour VLSI -Flot de Conception VLSI Alliance”, University of Paris VI.
- Paolo PRINETTO, “Lecture: The design cycle”, Politecnicodi Torino (Italy).
- The lecture notes and slides

Table of Figures:-