Ain Shams University
Faculty of Engineering
CHEP- CESS

Mohammed Ehab Elsaeed
16P8160
m.elsaeed1998@gmail.com

Electronic Design Automation
CSE 215

Digital Access Control Finite State Machine

Project 1

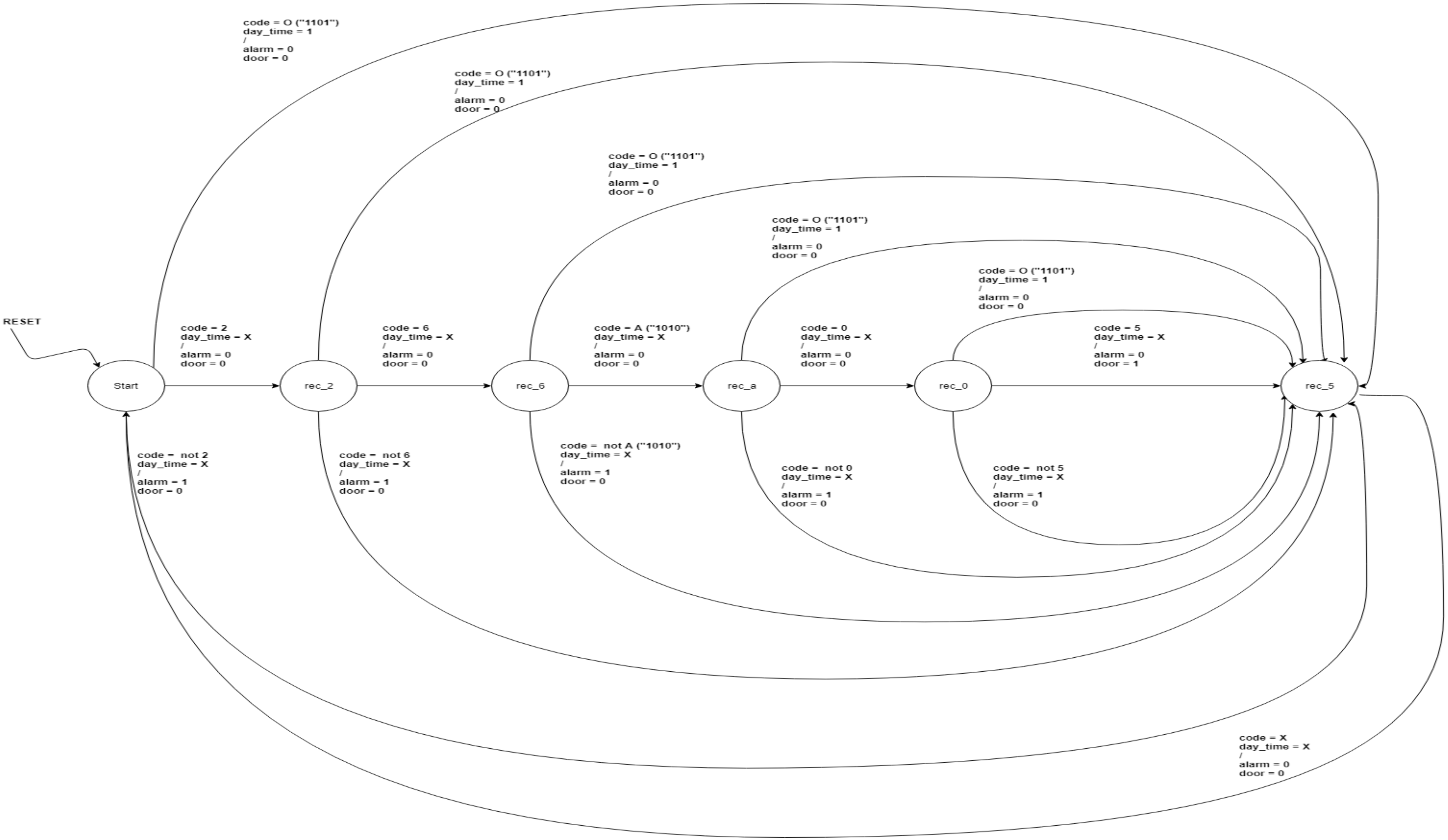
Introduction

The purpose of this document is to illustrate the work done to create a digital access control finite state machine in accordance to the specifications provided.

The finite state machine was implemented as a **Mealy** machine in an attempt to reduce the number of states due to the fact that outputs depend on transitions instead of states. As a sacrifice, the finite state machine is supposedly less stable than a Moore machine.

Table of Contents

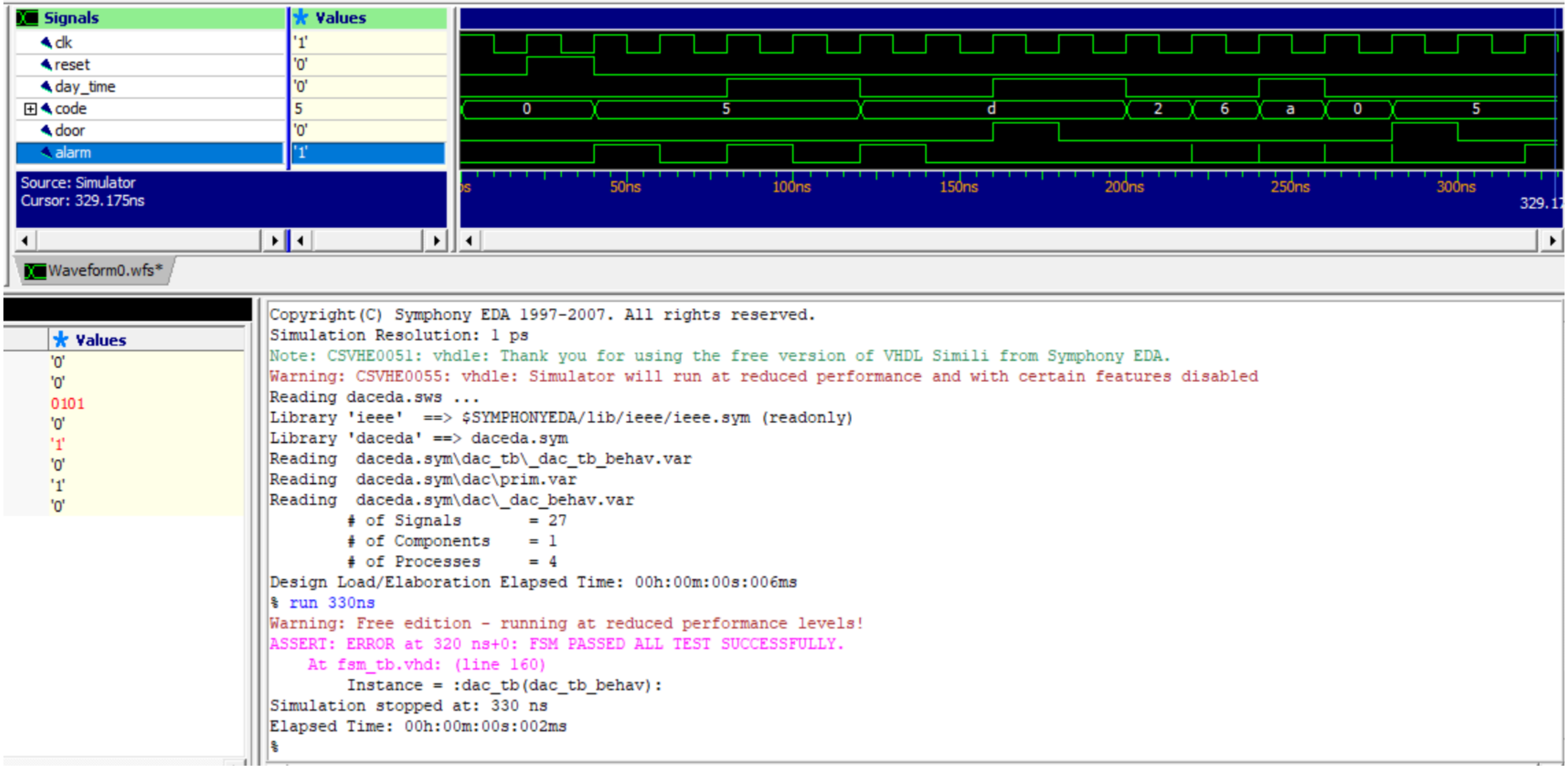
1- Finite State Machine State Diagram



2- Testbench Strategy

reset	day_time	code	door	alarm	Justification
1	X	X	0	0	Setting reset = 1 should reset the circuit to its initial state
0	0	5	0	1	Entering a wrong code, should trigger the alarm
0	**X**	**X**	**0**	**0**	**Waiting for a clock cycle, should reset both door and alarm, to zero**
0	1	5	0	1	Entering a wrong code, even during daytime should trigger the alarm
0	**X**	**X**	**0**	**0**	**Waiting for a clock cycle, should reset both door and alarm, to zero**
0	0	"O"	0	1	Entering "O" during night time should trigger the alarm
0	**X**	**X**	**0**	**0**	**Waiting for a clock cycle, should reset both door and alarm, to zero**
0	1	"O"	1	0	Entering "O" during daytime should open the door
0	**X**	**X**	**0**	**0**	**Waiting for a clock cycle, should reset both door and alarm, to zero**
0	0	2	0	0	Entering 2 neither opens the door, nor triggers the alarm
0	0	6	0	0	Entering 2, 6 neither opens the door, nor triggers the alarm
0	1	"A"	0	0	Entering 2, 6, A neither opens the door, nor triggers the alarm, switching daytime to 1 and not pressing "O", will not open the door nor trigger the alarm
0	0	0	0	0	Entering 2, 6, A, 0 neither opens the door, nor triggers the alarm
0	0	5	1	0	Entering 2, 6, A, 0, 5 opens the door, but doesn't trigger the alarm
0	**X**	**X**	**0**	**0**	**Waiting for a clock cycle, should reset both door and alarm, to zero**

3- Simulation



```
Copyright(C) Symphony EDA 1997-2007. All rights reserved.
Simulation Resolution: 1 ps
Note: CSVHE0051: vhdle: Thank you for using the free version of VHDL Simili from Symphony EDA.
Warning: CSVHE0055: vhdle: Simulator will run at reduced performance and with certain features disabled
Reading daceda.sws ...
Library 'ieee'   ==> $SYMPHONYEDA/lib/ieee/ieee.sym (readonly)
Library 'daceda' ==> daceda.sym
Reading  daceda.sym\dac_tb\_dac_tb_behav.var
Reading  daceda.sym\dac\prim.var
Reading  daceda.sym\dac\_dac_behav.var
        # of Signals       = 27
        # of Components    = 1
        # of Processes     = 4
Design Load/Elaboration Elapsed Time: 00h:00m:00s:006ms
% run 330ns
Warning: Free edition - running at reduced performance levels!
ASSERT: ERROR at 320 ns+0: FSM PASSED ALL TEST SUCCESSFULLY.
    At fsm_tb.vhd: (line 160)
        Instance = :dac_tb(dac_tb_behav):
Simulation stopped at: 330 ns
Elapsed Time: 00h:00m:00s:002ms
%
```

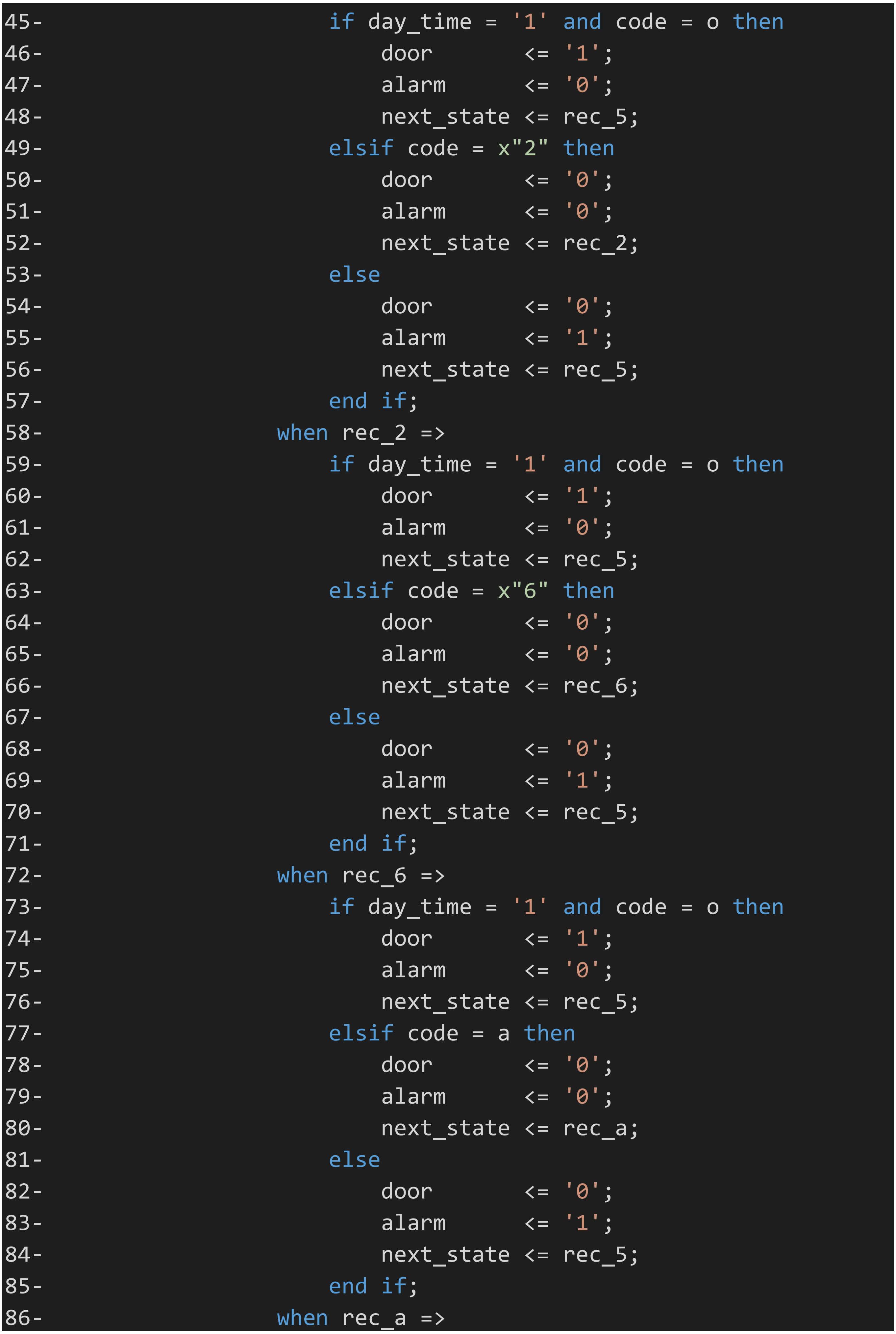
Note: “d” is the hexadecimal code for “1101” (Which is the **code** of input **O**)

Glitches in the alarm signal are due to the fact of using a **Mealy machine.**

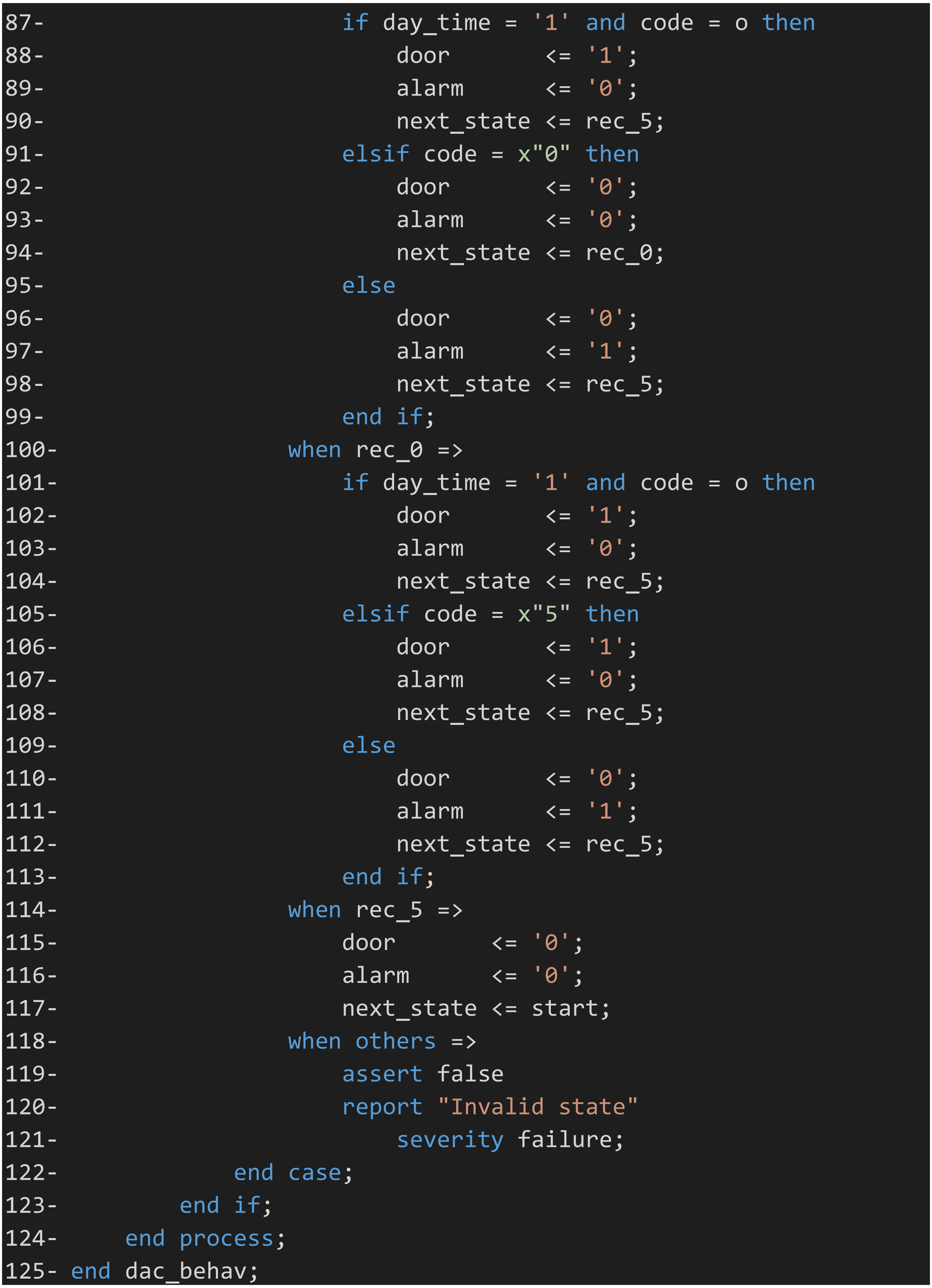
The system **passes** all the assertions.

4- Finite State Machine VHDL Code

```
-- 0 to 9 binary encoded
-- A => 1010
-- B => 1011
-- O => 1101
entity dac is
    port (
        reset    : in bit;
        day_time : in bit;
        code     : in bit_vector(3 downto 0);
        door     : out bit;
        alarm    : out bit;
        clk      : in bit;
        vdd      : in bit;
        vss      : in bit
    );
end dac;
architecture dac_behav of dac is
    type state is (start, rec_2, rec_6, rec_a, rec_0, rec_5);
    signal current_state : state;
    signal next_state    : state;
    constant a           : bit_vector(3 downto 0) := "1010";
    constant b           : bit_vector(3 downto 0) := "1011";
    constant o           : bit_vector(3 downto 0) := "1101";
    --pragma current_state current_state
    --pragma next_state next_state
    --pragma clock clk
begin
    process (clk)
    begin
        if clk = '1' and not clk'stable then
            current_state <= next_state;
        end if;
    end process;
    process (current_state, reset, day_time, code)
    begin
        if reset = '1' then
            next_state <= start;
        else
            case current_state is
                when start =>
```



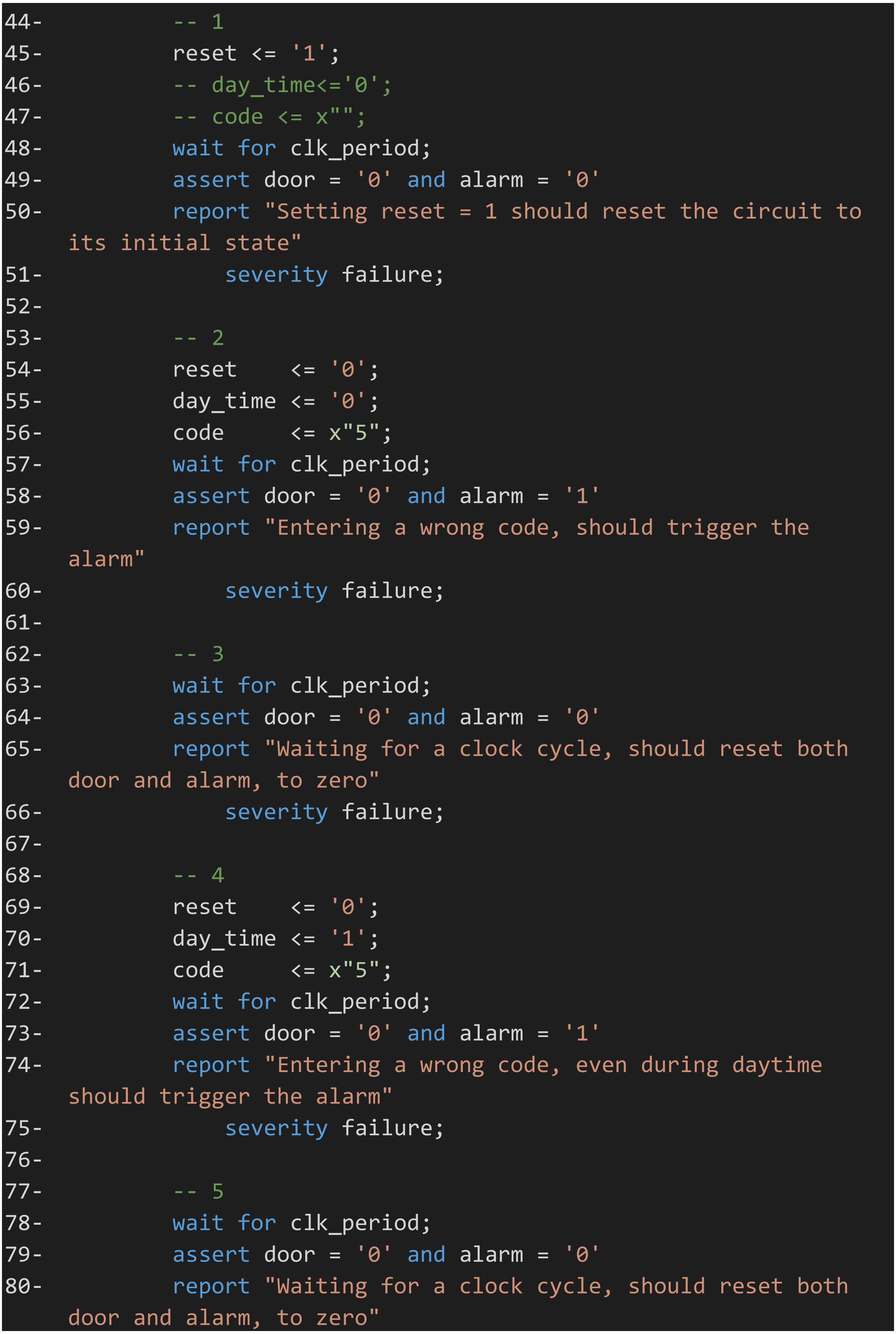
```vhdl
                    if day_time = '1' and code = o then
                        door       <= '1';
                        alarm      <= '0';
                        next_state <= rec_5;
                    elsif code = x"2" then
                        door       <= '0';
                        alarm      <= '0';
                        next_state <= rec_2;
                    else
                        door       <= '0';
                        alarm      <= '1';
                        next_state <= rec_5;
                    end if;
                when rec_2 =>
                    if day_time = '1' and code = o then
                        door       <= '1';
                        alarm      <= '0';
                        next_state <= rec_5;
                    elsif code = x"6" then
                        door       <= '0';
                        alarm      <= '0';
                        next_state <= rec_6;
                    else
                        door       <= '0';
                        alarm      <= '1';
                        next_state <= rec_5;
                    end if;
                when rec_6 =>
                    if day_time = '1' and code = o then
                        door       <= '1';
                        alarm      <= '0';
                        next_state <= rec_5;
                    elsif code = a then
                        door       <= '0';
                        alarm      <= '0';
                        next_state <= rec_a;
                    else
                        door       <= '0';
                        alarm      <= '1';
                        next_state <= rec_5;
                    end if;
                when rec_a =>
```



```
                    if day_time = '1' and code = o then
                        door       <= '1';
                        alarm      <= '0';
                        next_state <= rec_5;
                    elsif code = x"0" then
                        door       <= '0';
                        alarm      <= '0';
                        next_state <= rec_0;
                    else
                        door       <= '0';
                        alarm      <= '1';
                        next_state <= rec_5;
                    end if;
                when rec_0 =>
                    if day_time = '1' and code = o then
                        door       <= '1';
                        alarm      <= '0';
                        next_state <= rec_5;
                    elsif code = x"5" then
                        door       <= '1';
                        alarm      <= '0';
                        next_state <= rec_5;
                    else
                        door       <= '0';
                        alarm      <= '1';
                        next_state <= rec_5;
                    end if;
                when rec_5 =>
                    door       <= '0';
                    alarm      <= '0';
                    next_state <= start;
                when others =>
                    assert false
                    report "Invalid state"
                        severity failure;
            end case;
        end if;
    end process;
end dac_behav;
```

5- Testbench VHDL Code

```vhdl
entity dac_tb is
end dac_tb;

architecture dac_tb_behav of dac_tb is
    component dac is
        port (
            reset    : in bit;
            day_time : in bit;
            code     : in bit_vector(3 downto 0);
            door     : out bit;
            alarm    : out bit;
            clk      : in bit;
            vdd      : in bit;
            vss      : in bit
        );
    end component dac;
    signal reset        : bit;
    signal day_time     : bit;
    signal code         : bit_vector(3 downto 0);
    signal door         : bit;
    signal alarm        : bit;
    signal clk          : bit;
    signal vdd          : bit := '1';
    signal vss          : bit := '0';

    for all             : dac use entity work.dac(dac_behav);

    constant clk_period : time := 20 ns;
    constant a          : bit_vector(3 downto 0) := "1010";
    constant b          : bit_vector(3 downto 0) := "1011";
    constant o          : bit_vector(3 downto 0) := "1101";
begin

    dut : dac port map(reset, day_time, code, door, alarm, clk, vdd, vss);
    process begin

        wait for clk_period;

```



```vhdl
        -- 1
        reset <= '1';
        -- day_time<='0';
        -- code <= x"";
        wait for clk_period;
        assert door = '0' and alarm = '0'
        report "Setting reset = 1 should reset the circuit to its initial state"
            severity failure;

        -- 2
        reset    <= '0';
        day_time <= '0';
        code     <= x"5";
        wait for clk_period;
        assert door = '0' and alarm = '1'
        report "Entering a wrong code, should trigger the alarm"
            severity failure;

        -- 3
        wait for clk_period;
        assert door = '0' and alarm = '0'
        report "Waiting for a clock cycle, should reset both door and alarm, to zero"
            severity failure;

        -- 4
        reset    <= '0';
        day_time <= '1';
        code     <= x"5";
        wait for clk_period;
        assert door = '0' and alarm = '1'
        report "Entering a wrong code, even during daytime should trigger the alarm"
            severity failure;

        -- 5
        wait for clk_period;
        assert door = '0' and alarm = '0'
        report "Waiting for a clock cycle, should reset both door and alarm, to zero"
```

```
            severity failure;

        -- 6
        reset    <= '0';
        day_time <= '0';
        code     <= o;
        wait for clk_period;
        assert door = '0' and alarm = '1'
        report "Entering 'O' during night time should trigger the alarm"
            severity failure;

        -- 7
        wait for clk_period;
        assert door = '0' and alarm = '0'
        report "Waiting for a clock cycle, should reset both door and alarm, to zero"
            severity failure;

        -- 8
        reset    <= '0';
        day_time <= '1';
        code     <= o;
        wait for clk_period;
        assert door = '1' and alarm = '0'
        report "Entering 'O' during daytime should open the door"
            severity failure;

        -- 9
        wait for clk_period;
        assert door = '0' and alarm = '0'
        report "Waiting for a clock cycle, should reset both door and alarm, to zero"
            severity failure;

        -- 10
        reset    <= '0';
        day_time <= '0';
        code     <= x"2";
        wait for clk_period;
        assert door = '0' and alarm = '0'
        report "Entering 2 neither opens the door, nor triggers the alarm"
            severity failure;

        -- 11
        reset    <= '0';
        day_time <= '0';
        code     <= x"6";
        wait for clk_period;
        assert door = '0' and alarm = '0'
        report "Entering 2, 6 neither opens the door, nor triggers the alarm"
            severity failure;

        -- 12
        reset    <= '0';
        day_time <= '1';
        code     <= a;
        wait for clk_period;
        assert door = '0' and alarm = '0'
        report "Entering 2, 6, A neither opens the door, nor triggers the alarm, switching daytime to 1 and not pressing 'O', will not open the door nor trigger the alarm"
            severity failure;

        -- 13
        reset    <= '0';
        day_time <= '0';
        code     <= x"0";
        wait for clk_period;
        assert door = '0' and alarm = '0'
        report "Entering 2, 6, A, 0 neither opens the door, nor triggers the alarm"
            severity failure;

        -- 14
        reset    <= '0';
        day_time <= '0';
        code     <= x"5";
        wait for clk_period;
        assert door = '1' and alarm = '0'
        report "Entering 2, 6, A, 0, 5 opens the door, but doesn't trigger the alarm"
            severity failure;

        -- 15
        wait for clk_period;
        assert door = '0' and alarm = '0'
        report "Waiting for a clock cycle, should reset both door and alarm, to zero"
            severity failure;

        assert false
        report "FSM PASSED ALL TEST SUCCESSFULLY."
            severity error;
        wait;
    end process;

    process begin
        clk <= '1', '0' after (clk_period/2);
        wait for clk_period;
    end process;

end architecture dac_tb_behav;
```