Ain shams university
Faculty of engineering
Computer engineering and software systems programs

Course: EDA (CSE215)

Project 1

Submitted by: Shehab Ahmed Hassan Kotb

Id: 16p6014

Email: shehabktob@gmail.com

Submitted to: Dr. Mohamed Dessouky
Eng. Michael Hany

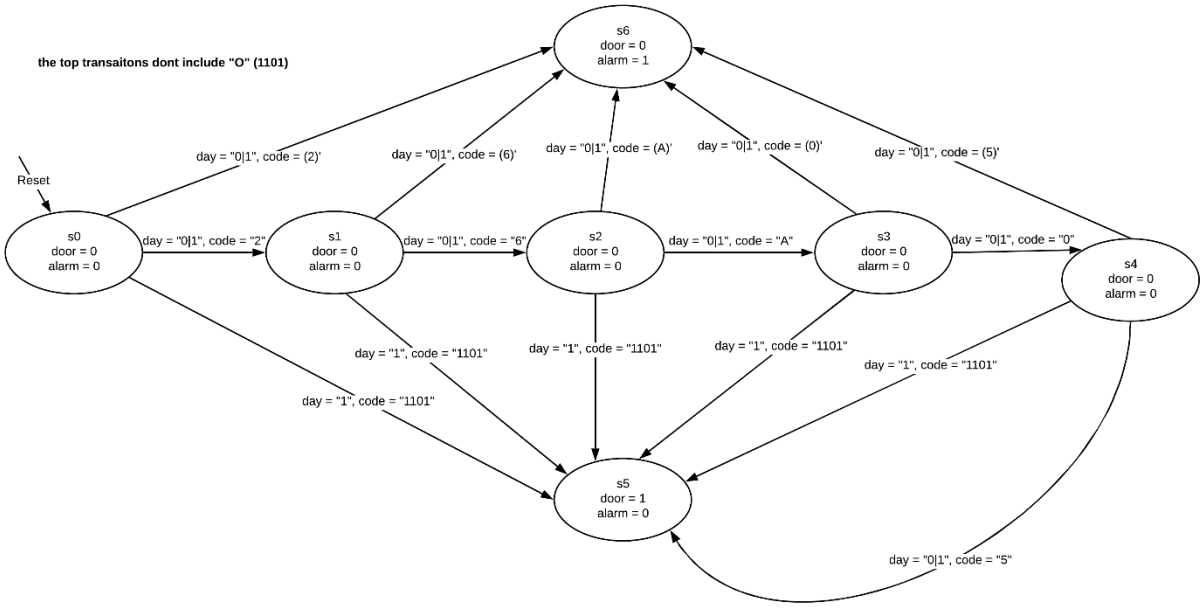
2.0 Introduction

This document details the design and implementation of a Digital access control system, this system is designed using MOORE fsm .

The first chapter of this document discusses the design of the system and it's vhdl code, also the testing strategy along with the testing bench code will be provided in this document.

The subsequent chapters will discuss the workflow with alliance tools to reach the final physical layout of the circuit.

3.1 state diagram



The top transitions don’t include code = “O” (1101) which opens the door when is equal to 1.

Moore was chosen over mealy for the following reasons:

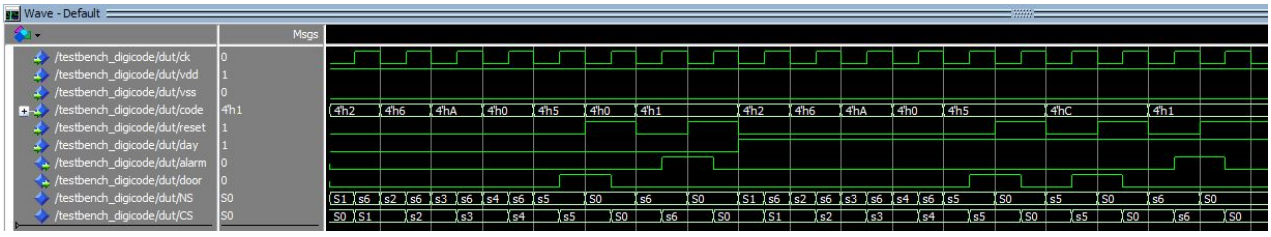
- Mealy has combinational paths between inputs and outputs which lead to long paths going through multiple components which slow down the system.
- Since this is an access control system Moore machine may be safer to use, because they change states on the clock edge.
- Moore model has simplified behavior and is easy to design.

3.2 testing bench strategy

tested feature	reset	day	code (hex)	delay	expected (door)	expected (alarm)
normal operation (night)	0	0	2	20 ns	0	0
			6			
			a			
			0			
			5		1	
reset (night)	1	0	0		0	0
wrong code (night)	0	0	1		0	1
normal operation (day)	0	1	2		0	0
			6			
			a			
			0			
			5		1	
reset (day)	1	1	0		0	0
O button (day)	0	1	1101 (binary)		1	0
wrong code (day)	0	1	1		0	1

We don’t cover everything here we just test for basic functionality of the system.

3.3 simulation results



Digicode

```vhdl
library IEEE;
USE ieee.std_logic_1164.ALL;

entity DigiCode is
port  (
    ck    : in  std_logic;
    vdd   : in  std_logic;
    vss   : in  std_logic;
    code  : in  std_logic_vector (3 downto 0);
    reset : in  std_logic;
    day   : in  std_logic;
    alarm : out std_logic;
    door  : out std_logic
      );
end DigiCode;

architecture MOORE of DigiCode is
  type STATE_TYPE is (S0, S1, s2, s3, s4, s5, s6);
  signal NS, CS : STATE_TYPE;

begin
  process (CS, reset, code, day)
  begin

    if (reset='1') then
      NS<=S0;
    else
      case CS is
        when S0 =>
          if (day='1' and code = "1101") then --code = o
            NS <= S5;
          elsif (code = x"2") then
            NS <= s1;
          else
            Ns <= s6;
          end if;

        when S1 =>
          if (day='1' and code = "1101") then --code = o
            NS <= S5;
          elsif (code = x"6") then
            NS <= s2;
          else
            Ns <= s6;
          end if;


        when S2 =>
          if (day='1' and code = "1101") then --code = o
            NS <= S5;
          elsif (code = "1010") then
            NS <= s3;
          else
            Ns <= s6;
          end if;


        when S3 =>
          if (day='1' and code = "1101") then --code = o
            NS <= S5;
          elsif (code = x"0") then
            NS <= s4;
          else
            Ns <= s6;
          end if;

```

```
67
68          when S4 =>
69            if (day='1' and code = "1101") then --code = o
70              NS <= S5;
71            elsif (code = x"5") then
72              NS <= s5;
73            else
74              Ns <= s6;
75            end if;
76
77
78          when S5 =>
79            --door <= '1';
80            null;
81            --alarm <= '0';
82            -- if (day='1' and code == '1101') then --code = o
83              -- NS <= S5;
84            -- elsif (code == '0')
85              -- NS <= s4;
86            -- else
87              -- Ns <= s6;
88            -- end if;
89
90
91          when S6 =>
92            --door <= '0';
93            --alarm <= '1';
94            null;
95            -- if (day='1' and code == '1101') then --code = o
96              -- NS <= S6;
97            -- elsif (code == '0')
98              -- NS <= s5;
99            -- else
100             -- Ns <= s6;
101           -- end if;
102
103         when others => assert(false)
104         report "illegal state" severity error;
105
106
107       end case;
108   end if;
109
110   case cs is
111       when s0 =>
112       door <= '0';
113       alarm <= '0';
114       when s1 =>
115       door <= '0';
116       alarm <= '0';
117       when s2 =>
118       door <= '0';
119       alarm <= '0';
120       when s3 =>
121       door <= '0';
122       alarm <= '0';
123       when s4 =>
124       door <= '0';
125       alarm <= '0';
126       when s5 =>
127       door <= '1';
128       alarm <= '0';
129       when s6 =>
130       door <= '0';
131       alarm <= '1';
132       end case;
```

```
    end process;

  -- Process (2): State update (sequential)
    process(ck)
    begin
      if(ck = '1' and not ck'stable)then
        CS <= NS;
      end if;
    end process;

  end MOORE;

```

Digicode_tb

```vhdl
library IEEE;
USE ieee.std_logic_1164.ALL;

-- Entity declaration for your testbench. Don't declare any ports here
ENTITY testbench_DigiCode IS
END ENTITY testbench_DigiCode;

ARCHITECTURE test_DigiCode OF testbench_DigiCode IS

-- Component Declaration for the Device Under Test (DUT)
COMPONENT DigiCode IS
port  (
    ck    : in  std_logic;
    vdd   : in  std_logic;
    vss   : in  std_logic;
    code  : in  std_logic_vector (3 downto 0);
    reset : in  std_logic;
    day   : in  std_logic;
    alarm : out std_logic;
    door  : out std_logic
      );
END COMPONENT DigiCode;

FOR dut: DigiCode USE ENTITY WORK.DigiCode (MOORE);

-- Declare input signals and initialize them
SIGNAL ck    : std_logic := '0';
SIGNAL vdd   : std_logic := '1';
SIGNAL vss   : std_logic := '0';
SIGNAL code  : std_logic_vector (3 downto 0) := x"0";
SIGNAL alarm : std_logic := '0';
SIGNAL reset : std_logic := '0';
SIGNAL day   : std_logic := '0';
SIGNAL door  : std_logic := '0';

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

-- Instantiate the Device Under Test (DUT)
dut: DigiCode PORT MAP (ck, vdd, vss, code, reset, day, alarm, door);

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

-- Stimulus process, refer to clock signal
stim_proc: PROCESS IS
BEGIN
    -- case 1 correct code entered at night "day = 0"
    code <= x"2";
    WAIT FOR clk_period;
    ASSERT alarm = '0' and door = '0'
    REPORT "output error"
    SEVERITY error;

    code <= x"6";
    WAIT FOR clk_period;
    ASSERT alarm = '0' and door = '0'
    REPORT "output error"
    SEVERITY error;
```

```
67
68          code <= x"a";
69          WAIT FOR clk_period;
70          ASSERT alarm = '0' and door = '0'
71          REPORT "output error"
72          SEVERITY error;
73
74          code <= x"0";
75          WAIT FOR clk_period;
76          ASSERT alarm = '0' and door = '0'
77          REPORT "output error"
78          SEVERITY error;
79
80          code <= x"5";
81          WAIT FOR clk_period;
82          ASSERT alarm = '0' and door = '1'
83          REPORT "output error"
84          SEVERITY error;
85
86          --case 2 reset test
87          code <= x"0";
88          reset <= '1';
89          WAIT FOR clk_period;
90          ASSERT alarm = '0' and door = '0'
91          REPORT "output error"
92          SEVERITY error;
93
94          --case 3 testing wrong code at night "day = 0"
95          code <= x"1";
96          reset <= '0';
97          WAIT FOR clk_period;
98          ASSERT alarm = '1' and door = '0'
99          REPORT "output error"
100         SEVERITY error;
101
102         --returning to s0
103         reset <= '1';
104         WAIT FOR clk_period;
105         ASSERT alarm = '0' and door = '0'
106         REPORT "output error"
107         SEVERITY error;
108
109         --case 4 testing correct code with day = 1
110         code <= x"2";
111         day <= '1';
112         reset <= '0';
113         WAIT FOR clk_period;
114         ASSERT alarm = '0' and door = '0'
115         REPORT "output error"
116         SEVERITY error;
117
118         code <= x"6";
119         day <= '1';
120         reset <= '0';
121         WAIT FOR clk_period;
122         ASSERT alarm = '0' and door = '0'
123         REPORT "output error"
124         SEVERITY error;
125
126         code <= x"a";
127         day <= '1';
128         reset <= '0';
129         WAIT FOR clk_period;
130         ASSERT alarm = '0' and door = '0'
131         REPORT "output error"
132         SEVERITY error;
```

```
133
134         code <= x"0";
135         day <= '1';
136         reset <= '0';
137         WAIT FOR clk_period;
138         ASSERT alarm = '0' and door = '0'
139         REPORT "output error"
140         SEVERITY error;
141
142         code <= x"5";
143         day <= '1';
144         reset <= '0';
145         WAIT FOR clk_period;
146         ASSERT alarm = '0' and door = '1'
147         REPORT "output error"
148         SEVERITY error;
149
150         --case 5 testing rest when day = 1
151         reset <= '1';
152         WAIT FOR clk_period;
153         ASSERT alarm = '0' and door = '0'
154         REPORT "output error"
155         SEVERITY error;
156
157         --case 6 testing "o" button when day = 1
158         code <= "1101";
159         day <= '1';
160         reset <= '0';
161         WAIT FOR clk_period;
162         ASSERT alarm = '0' and door = '1'
163         REPORT "output error"
164         SEVERITY error;
165
166         --returning to s0
167         reset <= '1';
168         WAIT FOR clk_period;
169         ASSERT alarm = '0' and door = '0'
170         REPORT "output error"
171         SEVERITY error;
172
173         --case 6 wrong code when day = 1
174         code <= x"1";
175         day <= '1';
176         reset <= '0';
177         WAIT FOR clk_period;
178         ASSERT alarm = '1' and door = '0'
179         REPORT "output error"
180         SEVERITY error;
181
182         --returning to s0
183         reset <= '1';
184         WAIT FOR clk_period;
185         ASSERT alarm = '0' and door = '0'
186         REPORT "output error"
187         SEVERITY error;
188
189
190
191     WAIT; -- stop process simulation run
192
193     END PROCESS stim_proc;
194     END ARCHITECTURE test_DigiCode;
195
196
197
```