

AIN SHAMS UNIVERSITY FACULTY OF ENGINEERING

Project 1 Report

Name: Hazem Hamada Abdellatif Mohamed

ID: 16p3100

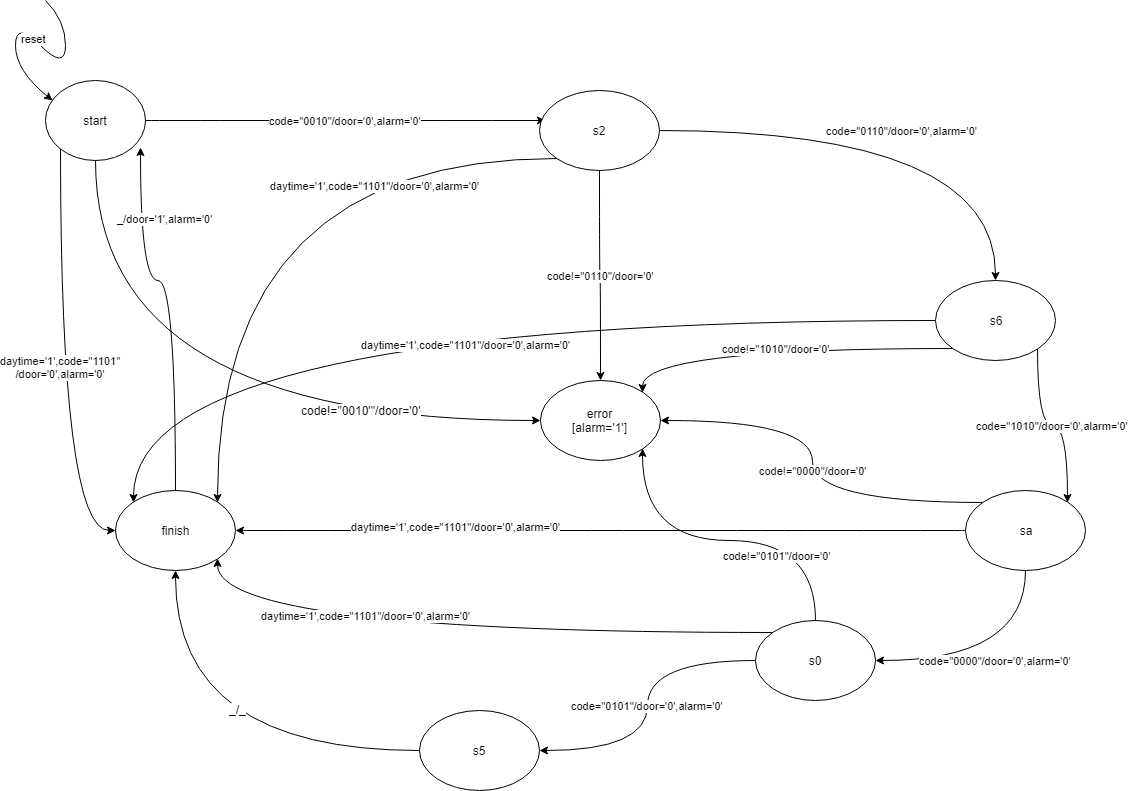
Group1, Section1.

Email: [zomahamada.hh@gmail.com](mailto:zomahamada.hh@gmail.com).

Program: Computer Engineering and Software Systems.

***Description:***

The project is about Digital Access Control System that is implemented as a finite state machine as shown below.



•The correct code is **26A05.**

•Operates in two modes:

–Daytime: in the morning, door opens when:

•Pressing “O”,

•Entering the complete correct code

•pressing “O” in the middle between any digit of the correct code.

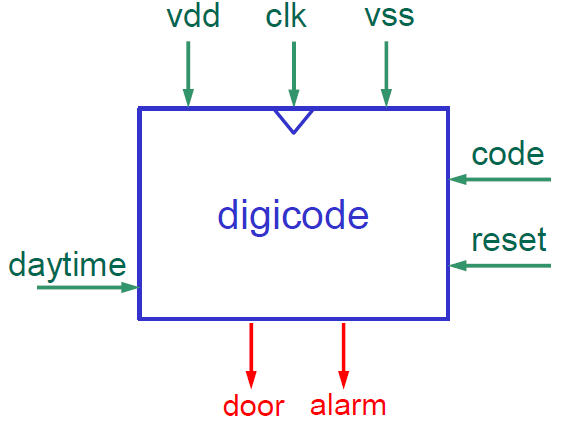
–Night: opens only if the code is correct.

•An alarm is triggered in case of:

–An incorrect entry, as soon as a wrong number is pressed.

–If “O” is pressed at night at any instance, even after any number of the correct code before it is complete.

• The FSM is implemented in mealy to minimize the number of states “no state for each output” and quickly get the output ,and the alarm output in the final state is moor as it is required in the error state the alarm should be activated all the time until the reset is set to ‘1’.



Inputs and Outputs:

•**code:** input code, *applied externally*, one number at a time, binary coded.

•**daytime:** =1 *applied externally* during the morning.

•**door:** =1 when correct code.

•**alarm**: =1, see previous slide.

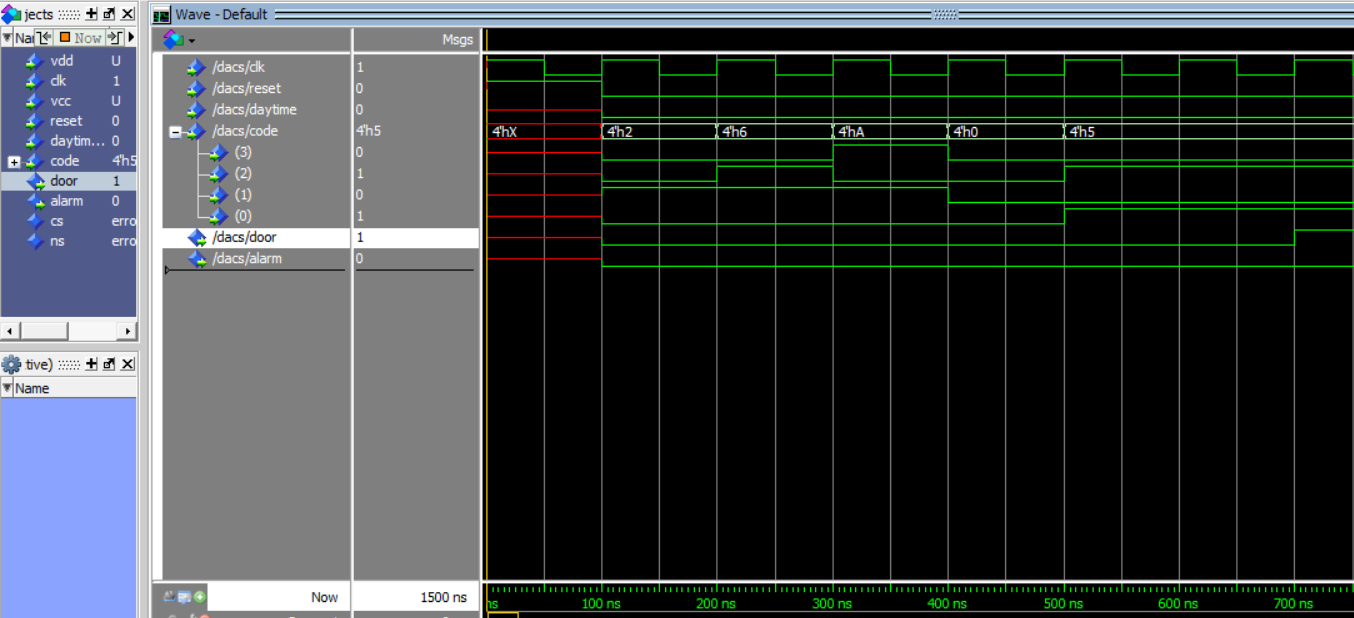
•**reset:** =1 *applied externally* after **alarm** is triggered (alarm=1)

Can be applied at any instance ,*i.e.* after any digit is pressed.

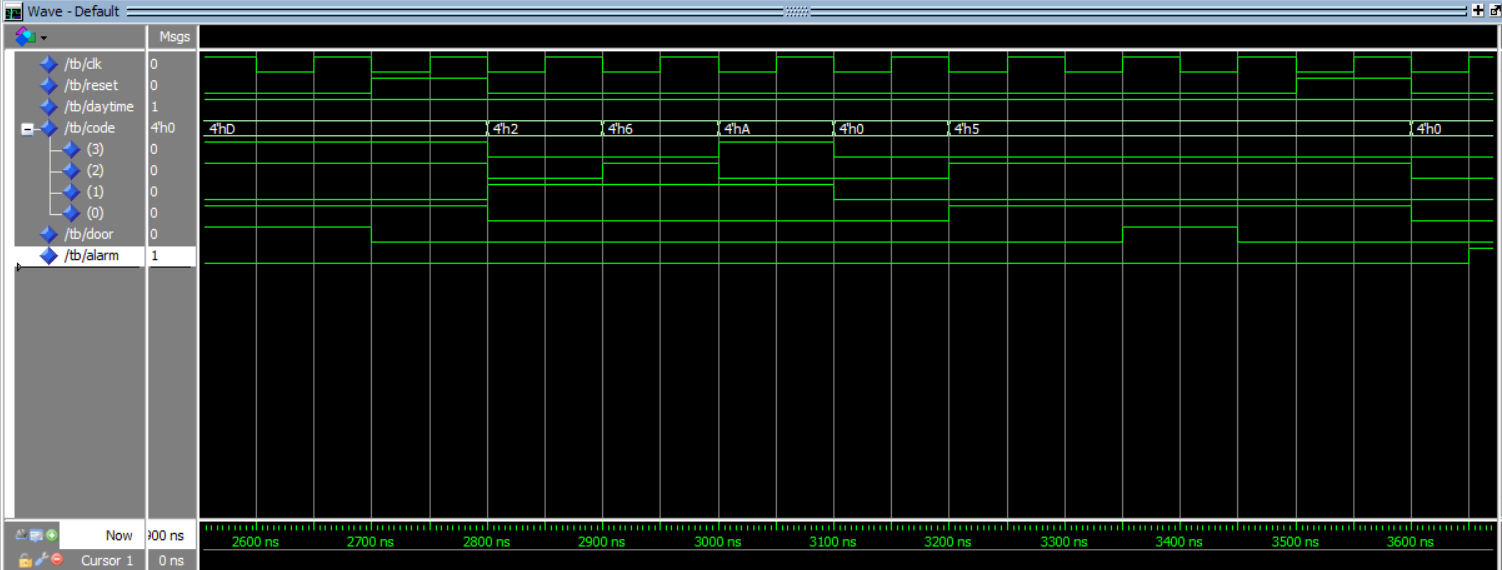
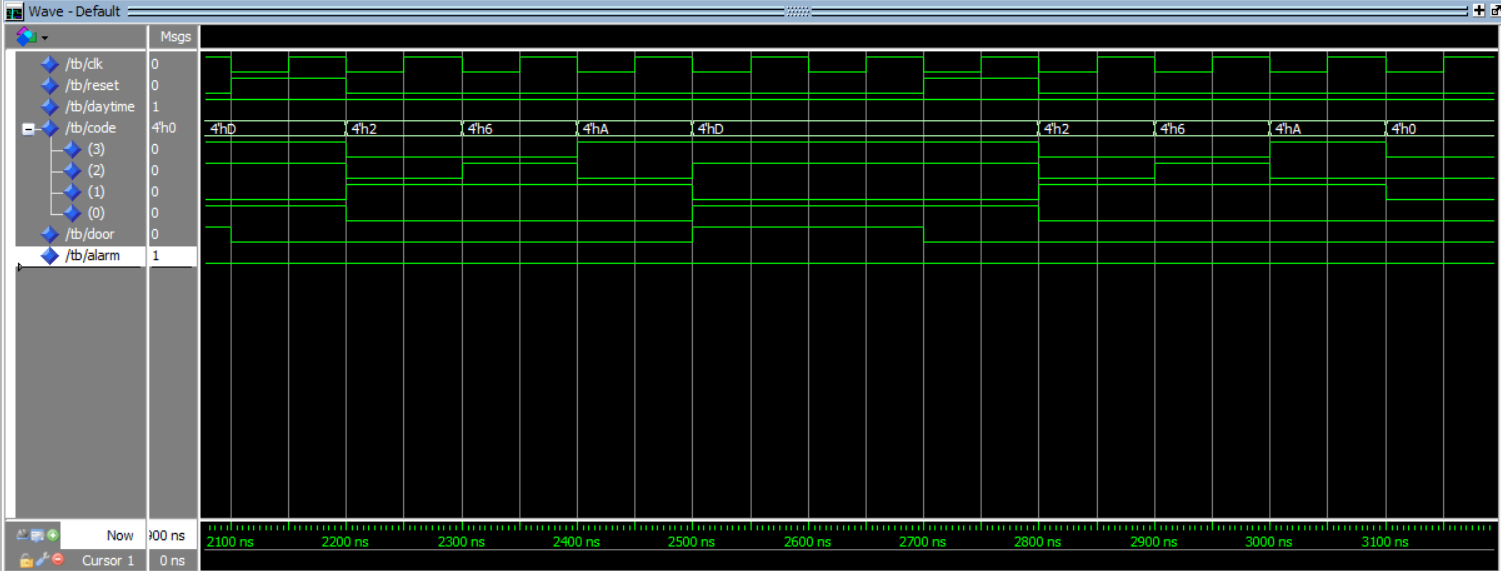
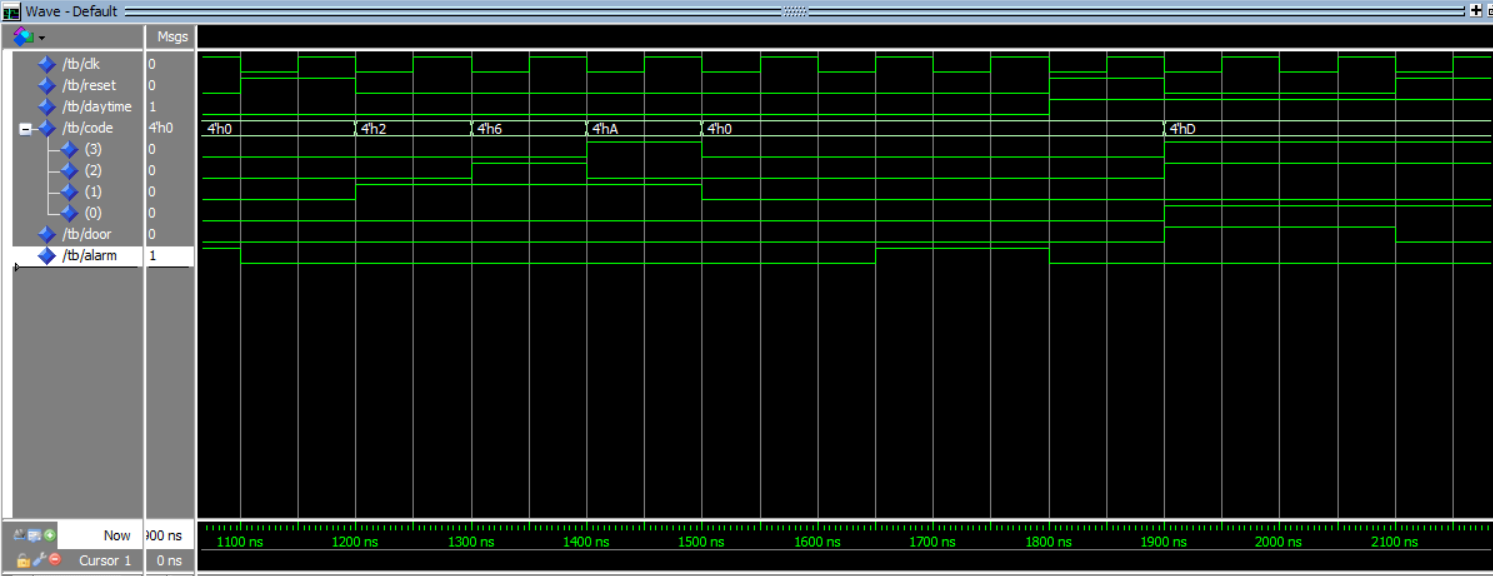
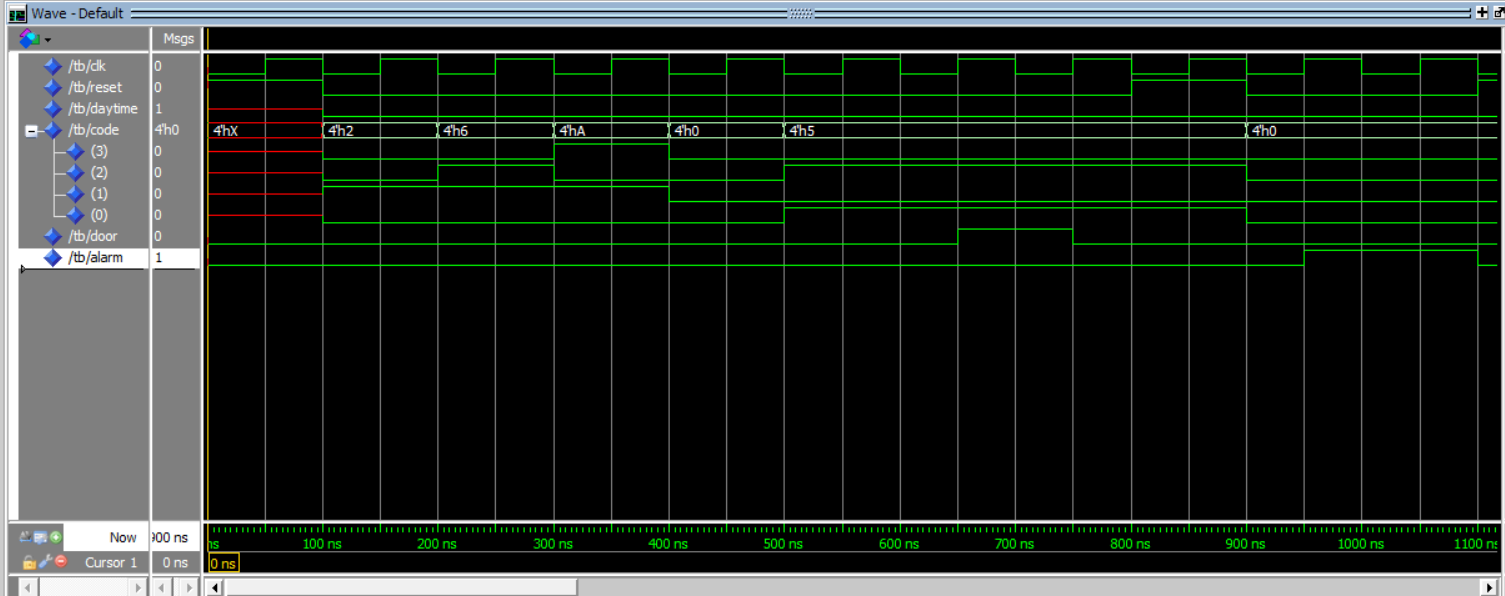
**Action** :deletes entered code and restarts from the beginning waiting for the first digit. Both door and alarm are set to 0.

**The reset is synchronous**.

***Wave forms of the FSM:***



***Test bench of wave form:***



***Test strategies:***

--happy scenario test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| reset | code | daytime | door | alarm |
| 1 | 0000 | 0 | 0 | 0 |
| 0 | 0010 | 0 | 0 | 0 |
| 0 | 0110 | 0 | 0 | 0 |
| 0 | 1010 | 0 | 0 | 0 |
| 0 | 0000 | 0 | 0 | 0 |
| 0 | 0101 | 0 | 1 | 0 |

--alarm scenario 1 test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| reset | code | daytime | door | alarm |
| 1 | 0000 | 0 | 0 | 0 |
| 0 | 0000 | 0 | 0 | 1 |

--alarm scenario 2 test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| reset | code | daytime | door | Alarm |
| 1 | 0000 | 0 | 0 | 0 |
| 0 | 0010 | 0 | 0 | 0 |
| 0 | 0110 | 0 | 0 | 0 |
| 0 | 1010 | 0 | 0 | 0 |
| 0 | 0000 | 0 | 0 | 0 |
| 0 | 0000 | 0 | 0 | 1 |

--daytime happy scenario 1 test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| reset | code | daytime | door | Alarm |
| 1 | 0000 | 1 | 0 | 0 |
| 0 | 1101 | 1 | 1 | 0 |

--daytime happy scenario 2 test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| reset | Code | daytime | door | Alarm |
| 1 | 0000 | 1 | 0 | 0 |
| 0 | 0010 | 1 | 0 | 0 |
| 0 | 0110 | 1 | 0 | 0 |
| 0 | 1010 | 1 | 0 | 0 |
| 0 | 1101 | 1 | 1 | 0 |

--daytime happy scenario 3 test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| reset | code | daytime | door | alarm |
| 1 | 0000 | 1 | 0 | 0 |
| 0 | 0010 | 1 | 0 | 0 |
| 0 | 0110 | 1 | 0 | 0 |
| 0 | 1010 | 1 | 0 | 0 |
| 0 | 0000 | 1 | 0 | 0 |
| 0 | 0101 | 1 | 1 | 0 |

--alarm scenario 3 test.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| reset | code | daytime | door | Alarm |
| 1 | 0000 | 1 | 0 | 0 |
| 0 | 0000 | 1 | 0 | 1 |

***FSM VHDL code:***

library ieee;

use ieee.std\_logic\_1164.ALL;

use ieee.std\_logic\_unsigned.ALL;

use ieee.numeric\_std.all;

entity dacs is

port( vdd,clk,vcc,reset,daytime: in std\_logic;

code: in std\_logic\_vector(3 downto 0);

door,alarm: out std\_logic );

end entity dacs;

architecture behav of dacs is

type state is (start, s2, s6, sa, s0, s5, finish, error);

signal cs,ns: state;

begin

p1:process(cs,reset,daytime,code)is

begin

if reset='1' then

alarm<='0';

door<='0';

ns<=start;

else

if daytime='1' then

if code ="1101" then

door<='1';

ns<=finish;

else

case cs is

when start=>

door<='0';

alarm<='0';

if code="0010" then

ns<=s2;

else

ns<=error;

end if;

when s2=>

if code="0110" then

ns<=s6;

else

ns<=error;

end if;

when s6=>

if code="1010" then

ns<=sa;

else

ns<=error;

end if;

when sa=>

if code="0000" then

ns<=s0;

else

ns<=error;

end if;

when s0=>

if code="0101" then

ns<=s5;

else

ns<=error;

end if;

when s5=>

ns<=finish;

when finish=>

door<='1';

ns<=start;

when error=>

door<='0';

alarm<='1';

end case;

end if;

else

case cs is

when start=>

door<='0';

alarm<='0';

if code="0010" then

ns<=s2;

else

ns<=error;

end if;

when s2=>

if code="0110" then

ns<=s6;

else

ns<=error;

end if;

when s6=>

if code="1010" then

ns<=sa;

else

ns<=error;

end if;

when sa=>

if code="0000" then

ns<=s0;

else

ns<=error;

end if;

when s0=>

if code="0101" then

ns<=s5;

else

ns<=error;

end if;

when s5=>

ns<=finish;

when finish=>

door<='1';

ns<=start;

when error=>

door<='0';

alarm<='1';

end case;

end if;

end if;

end process p1;

p2:process(clk)is

begin

if clk'event and clk='1' then

cs<=ns;

end if;

end process p2;

end architecture behav;

***FSM TEST BENCH VHDL code:***

library ieee;

use ieee.std\_logic\_1164.ALL;

use ieee.std\_logic\_unsigned.ALL;

use ieee.numeric\_std.all;

entity tb is

end entity tb;

architecture test of tb is

component dacs is

port( vdd,clk,vcc,reset,daytime: in std\_logic;

code: in std\_logic\_vector(3 downto 0);

door,alarm: out std\_logic );

end component dacs;

signal vdd,vcc,clk,reset,daytime,door,alarm:std\_logic;

signal code:std\_logic\_vector(3 downto 0);

constant clk\_period : time := 100ns;

for dut:dacs use entity work.dacs(behav);

begin

dut:dacs port map(vdd,clk,vcc,reset,daytime,code,door,alarm);

clk\_process :process

begin

clk <= '0';

wait for clk\_period/2;

clk <= '1';

wait for clk\_period/2;

end process;

p:process

begin

--happy scenario test.

reset<='1';

wait for clk\_period;

reset<='0';

daytime<='0';

code<="0010";

wait for clk\_period;

code<="0110";

wait for clk\_period;

code<="1010";

wait for clk\_period;

code<="0000";

wait for clk\_period;

code<="0101";

wait for 2\*clk\_period;

assert door='1' and alarm='0'

report "happy scenario error"

severity error;

wait for clk\_period;

--alarm scenario 1 test.

reset<='1';

wait for clk\_period;

reset<='0';

code<="0000";

wait for clk\_period;

assert door='0' and alarm='1'

report "alarm scenario 1 error"

severity error;

wait for clk\_period;

--alarm scenario 2 test.

reset<='1';

wait for clk\_period;

reset<='0';

code<="0010";

wait for clk\_period;

code<="0110";

wait for clk\_period;

code<="1010";

wait for clk\_period;

code<="0000";

wait for clk\_period;

code<="0000";

wait for clk\_period;

assert door='0' and alarm='1'

report "alarm scenario 2 error"

severity error;

wait for clk\_period;

--daytime happy scenario 1 test.

reset<='1';

daytime<='1';

wait for clk\_period;

reset<='0';

code<="1101";

wait for clk\_period;

assert door='1' and alarm='0'

report "daytime happy scenario 1 error"

severity error;

wait for clk\_period;

--daytime happy scenario 2 test.

reset<='1';

wait for clk\_period;

reset<='0';

code<="0010";

wait for clk\_period;

code<="0110";

wait for clk\_period;

code<="1010";

wait for clk\_period;

code<="1101";

wait for clk\_period;

assert door='1' and alarm='0'

report "daytime happy scenario 2 error"

severity error;

wait for clk\_period;

--daytime happy scenario 3 test.

reset<='1';

wait for clk\_period;

reset<='0';

code<="0010";

wait for clk\_period;

code<="0110";

wait for clk\_period;

code<="1010";

wait for clk\_period;

code<="0000";

wait for clk\_period;

code<="0101";

wait for 2\*clk\_period;

assert door='1' and alarm='0'

report "daytime happy scenario 3 error"

severity error;

wait for clk\_period;

--alarm scenario 3 test.

reset<='1';

daytime<='1';

wait for clk\_period;

reset<='0';

code<="0000";

wait for clk\_period;

assert door='0' and alarm='1'

report "alarm scenario 3 error"

severity error;

wait for clk\_period;

wait;

end process p;

end architecture test;