TEIS

Assignment 4 VHDL Course

Designing the door lock system

Jasim Abbasi

2014-10-04

The basic purpose of this project is to design the two different types of automatic door lock system. A simple door and a door with more advance door lock system.

CONTENTS

1 INTRODUCTION

1.1 Background

A new customer has ordered prototypes of an automatic door control structure to be made in two different versions,

- Småland customers want a simpler (cheaper) than those living Stockholm.
- A simple door and a more advanced door with lock.

2 REQUIREMENTS SPECIFICATION

2.1 System requirements

The system requirement from the customer is that the design should be made with two different types of state machines and want it single door controller with both synchronous outputs (Mealy + Moore).

2.2 Functional requirements

The client has a requirement to design two different types of door lock system.

- Simple door lock system
- Advance door lock system

2.3 Construction code requirements

The client has requirements that code should be written in VHDL and result should be simulate on the modelsim.

2.4 Delivery requirements

The delivery of the product should be before 6 October, 2014.

3 TEST PROTOCOL

The table 1 and 2 shown the functionality of simple and advance door lock system.

KEY1	KEY0	LEDG1	LEDG0	Description
0	1	1	0	When door is closed
1	0	0	1	When door is open

Table 1: Simple door test protocol

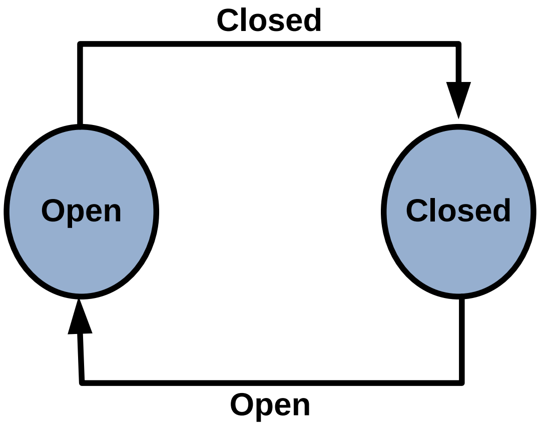


Figure 1: Simple door Stateflow diagram

KEY3	KEY2	KEY1	KEY0	LEDG2	LEDG1	LEDG0	LEDR1	Description
0	0	0	1	0	1	0	0	Door is close
0	0	1	0	0	0	1	0	Door is open
0	1	0	0	1	0	0	0	Door is lock
1	0	0	0	0	0	0	1	Error

Table 2: Advance door test protocol

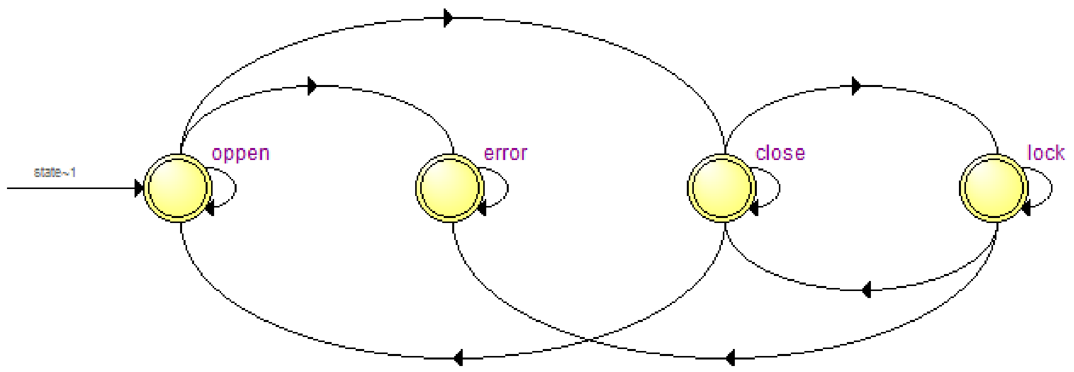


Figure 2: Advance door Stateflow diagram

4 VERIFICATION

4.1 Result from simulation

The tables 3 and 4 shows the test protocol of verifying simple and advance doors. The results can be verified on modelsim.

Case no	Description	OK	Verifying (Modelsim)	Validation (DE2-115)
1	Reset=0	LEDG=01 (door open)	OK	OK
2	Key=10	LEDG=10 (door close)	OK	OK
3	Key=01	LEDG=01 (door open)	OK	OK
4	Key=10	LEDG=10 (door close)	OK	OK
5	Reset=0	LEDG=01 (door open)	OK	OK

Table 3: Test protocol of verifying simple door

Case no	Description	OK	Verifying (Modelsim)	Validation (DE2-115)

1	Reset=0
	Door open
		LEDG=001
		LEDR=0
			OK	OK

2	Key=0001
	Close
		LEDG=010
		LEDR=0
			OK	OK

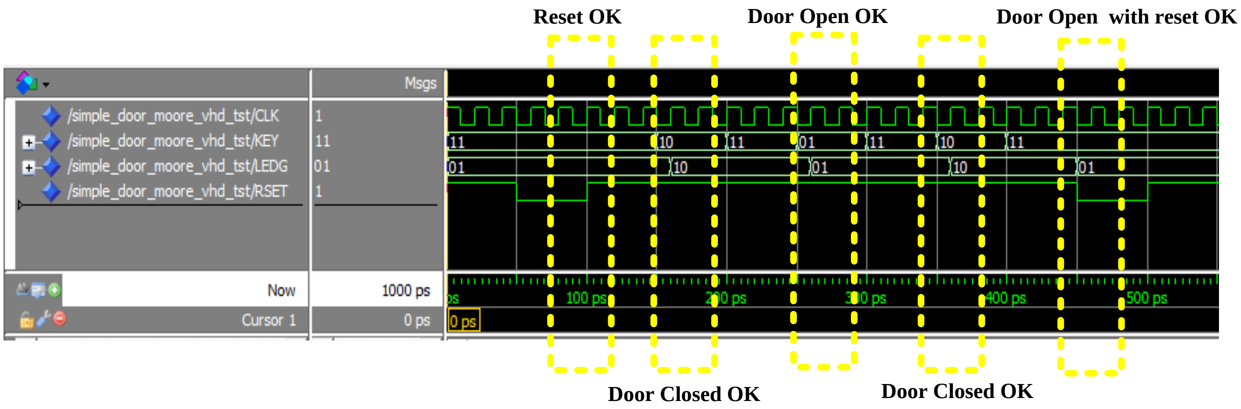
3	Key=0100
	Lock
		LEDG=100
		LEDR=0
			OK	OK

4	Key=1000
	Error
		LEDG=000
		LEDR=1
			OK	OK

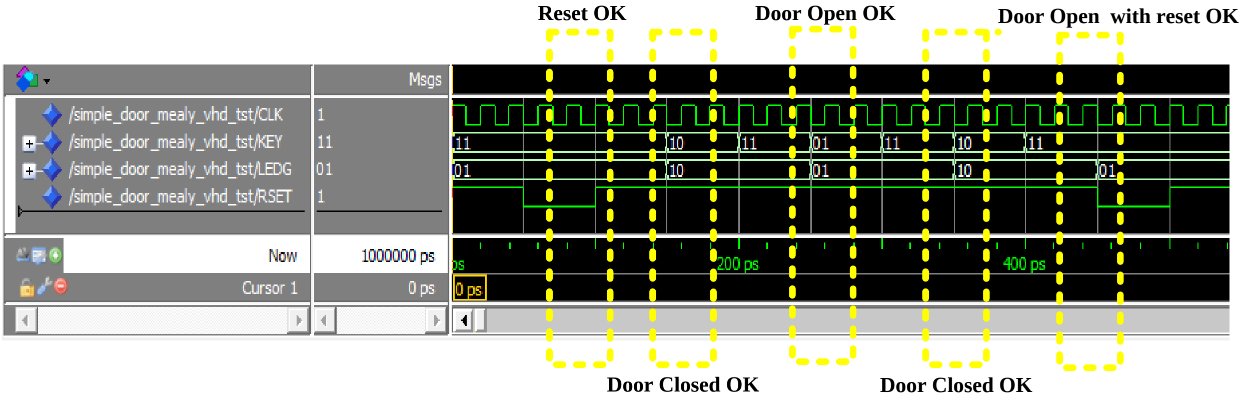
5	Key=1111
	Error
		LEDG=000
		LEDR=1
			OK	OK

Table 4: Test protocol of verifying advance door

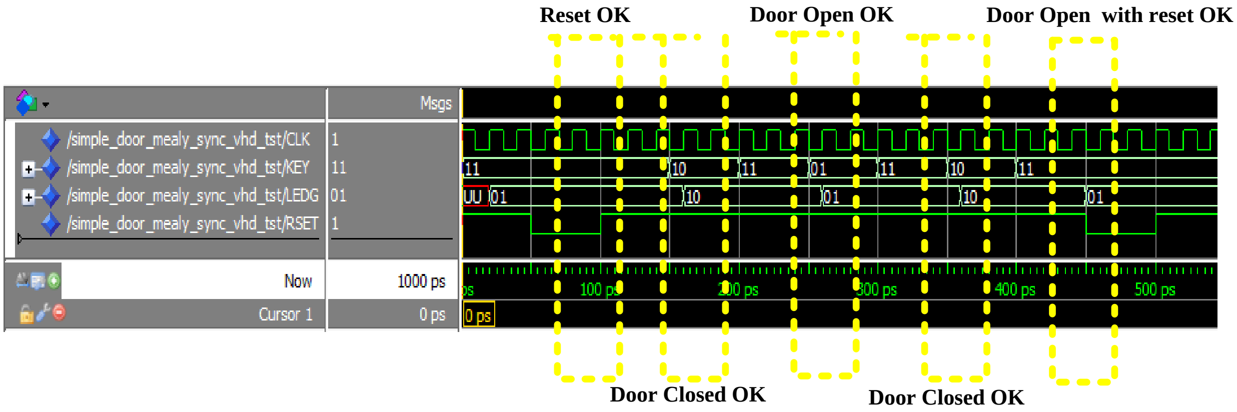
Simple door moore machine



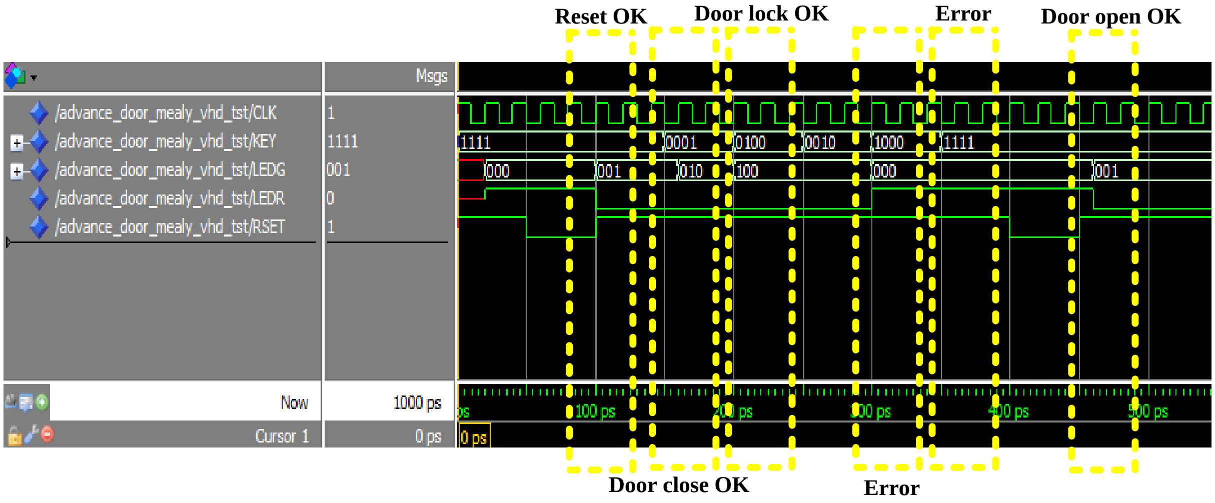
Simple door mealy machine


Reset OK
Door Open OK
Door Open with reset OK
Msgs
/simple_door_mealy_vhd_tst/CLK 1
/simple_door_mealy_vhd_tst/KEY 11
/simple_door_mealy_vhd_tst/LEDG 01
/simple_door_mealy_vhd_tst/RSET 1
Now 1000000 ps
Cursor 1 0 ps
200 ps
400 ps
0 ps
Door Closed OK
Door Closed OK

Simple door synchronous outputs


Reset OK
Door Open OK
Door Open with reset OK
Msgs
/simple_door_mealy_sync_vhd_tst/CLK 1
/simple_door_mealy_sync_vhd_tst/KEY 11
/simple_door_mealy_sync_vhd_tst/LEDG 01
/simple_door_mealy_sync_vhd_tst/RSET 1
Now 1000 ps
Cursor 1 0 ps
100 ps
200 ps
300 ps
400 ps
500 ps
0 ps
Door Closed OK
Door Closed OK

Advance door


Reset OK
Door lock OK
Error
Door open OK
Msgs
/advance_door_mealy_vhd_tst/CLK 1
/advance_door_mealy_vhd_tst/KEY 1111
/advance_door_mealy_vhd_tst/LEDG 001
/advance_door_mealy_vhd_tst/LEDR 0
/advance_door_mealy_vhd_tst/RSET 1
Now 1000 ps
Cursor 1 0 ps
100 ps
200 ps
300 ps
400 ps
500 ps
0 ps
Door close OK
Error

APPENDIX

Simple door moore VHDL Code

```
LIBRARY ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_arith.all;

use ieee.std_logic_unsigned.all;

entity simple_door_moore is

    port

       (

         CLK: in std_logic; -- Active on positiv flank

                              RSET: in std_logic;

         KEY: in std_logic_vector(1 downto 0); -- address in slide switches

         LEDG: out std_logic_vector(1 downto 0) -- data in slide switches

               );

end entity;

 architecture model of simple_door_moore is

type STATE_TYPE is (oppen, close);

signal state: STATE_TYPE;

begin

clk_reset: PROCESS (rset, clk)

   BEGIN

     IF (rset='0') THEN

          state<= oppen;

     ELSIF rising_edge(clk) THEN

       case state is

         when oppen =>
```

```
          if(KEY = "10") then
           state<= close;
       end if;
              when close =>
          if KEY = "01" then
            state<= oppen;
          end if;
                                when others=>
                                 state<= oppen;
      end case ;
                    end if;
          --end if;
 end process;
 process(state)
   begin
   case state is
    when oppen=>
     LEDG<="01";
      when close=>
      LEDG<="10";
      end case;
  end process;
 end model;
```

Simple door moore Do file

```
onerror {resume}
quietly WaveActivateNextPane {} 0
```

```
add wave -noupdate /simple_door_moore_vhd_tst/CLK
add wave -noupdate /simple_door_moore_vhd_tst/KEY
add wave -noupdate /simple_door_moore_vhd_tst/LEDG
add wave -noupdate /simple_door_moore_vhd_tst/RSET
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {0 ps} 0}
quietly wave cursor active 0
configure wave -namecolwidth 224
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0
configure wave -gridperiod 1
configure wave -griddelta 40
configure wave -timeline 0
configure wave -timelineunits ps
update
WaveRestoreZoom {0 ps} {931 ps}
```

Simple door test bench

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
```

```
ENTITY simple_door_moore_vhd_tst IS
END simple_door_moore_vhd_tst;
ARCHITECTURE simple_door_moore_arch OF simple_door_moore_vhd_tst IS
-- constants
-- signals
SIGNAL CLK : STD_LOGIC;
SIGNAL KEY : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL LEDG : STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL RSET : STD_LOGIC;
COMPONENT simple_door_moore
        PORT (
        CLK : IN STD_LOGIC;
        KEY : IN STD_LOGIC_VECTOR(1 DOWNTO 0);
        LEDG : OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
        RSET : IN STD_LOGIC
        );
END COMPONENT;
BEGIN
        i1 : simple_door_moore
        PORT MAP (
-- list connections between master ports and signals
        CLK => CLK,
        KEY => KEY,
        LEDG => LEDG,
        RSET => RSET
```

```
          );
 clk_signal: process
    begin
           --------clOCK SIGNAL---------
        clk<='1';
                      wait for 10 ps;
                    --wait on clk;
        clk<='0';
        wait for 10 ps;
                    --wait on clk;
 end process;
init : PROCESS
-- variable declarations
BEGIN
   -- code that executes only once
    KEY<= "11"; --
           rset<='1';
           wait for 50ps ;
           rset<='0';
           wait for 50ps ;
           rset<='1';
    ------CASE 2-------
           wait for 50ps ;
    KEY<= "10"; --
           wait for 50ps ;
           KEY<= "11";
```

```
            ------CASE 2-------
            wait for 50ps ;
    KEY<= "01"; --
            wait for 50ps ;
            KEY<= "11";
            ------CASE 3-------
    wait for 50ps ;
    KEY<= "10"; --
            wait for 50ps ;
            KEY<= "11";
            ------CASE 4-------
    wait for 50ps ;
            rset<='0';
            wait for 50ps ;
             rset<='1';
--  ---------------------
 --  KEY<= "00"; --
 --  wait for 50ps ;
WAIT;
END PROCESS init;
always : PROCESS
-- optional sensitivity list
-- (         )
-- variable declarations
BEGIN
        -- code executes for every event on sensitivity list
```

```
WAIT;

END PROCESS always;

END simple_door_moore_arch;
```

Simple door mealy VHDL Code

```
LIBRARY ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_arith.all;

use ieee.std_logic_unsigned.all;

entity simple_door_mealy is

    port

       (

         CLK: in std_logic; -- Clock signal

                               RSET: in std_logic; --reset signal

         KEY: in std_logic_vector(1 downto 0); -- Input keys

         LEDG: out std_logic_vector(1 downto 0) -- output key

               );

end entity;

 architecture model of simple_door_mealy is

type STATE_TYPE is (oppen, close);

signal current_state,next_state: STATE_TYPE;

signal state: STATE_TYPE;

begin

clk_reset: PROCESS (rset,clk)

   BEGIN

     IF (rset='0') THEN

          state<= oppen;
    ELSIF rising_edge(clk) THEN
      case state is
        when oppen =>
          if(KEY = "10") then
           state<= close;
      end if;
             when close =>
          if KEY = "01" then
            state<= oppen;
          end if;
                               when others=>
                                state<= oppen;
     end case ;
                   end if;
         --end if;
end process;
process(state,key)
  begin
  case state is
   when oppen=>
           if(KEY = "01" or KEY = "11")then
    LEDG<="01";
            else
                   LEDG<="10";
                   end if;
    when close=>
```

```
            if(KEY = "10" or KEY = "11") then
      LEDG<="10";
                  else
                  LEDG<="01";
                  end if;
      end case;
  end process;
 end model;
```

Simple door mealy do file

```
onerror {resume}
quietly WaveActivateNextPane {} 0
add wave -noupdate /simple_door_mealy_vhd_tst/CLK
add wave -noupdate /simple_door_mealy_vhd_tst/KEY
add wave -noupdate /simple_door_mealy_vhd_tst/LEDG
add wave -noupdate /simple_door_mealy_vhd_tst/RSET
TreeUpdate [SetDefaultTree]
WaveRestoreCursors {{Cursor 1} {0 ps} 0}
quietly wave cursor active 0
configure wave -namecolwidth 220
configure wave -valuecolwidth 100
configure wave -justifyvalue left
configure wave -signalnamewidth 0
configure wave -snapdistance 10
configure wave -datasetprefix 0
configure wave -rowmargin 4
configure wave -childrowmargin 2
configure wave -gridoffset 0

configure wave -gridperiod 1

configure wave -griddelta 40

configure wave -timeline 0

configure wave -timelineunits ps

update

WaveRestoreZoom {0 ps} {935 ps}
```

Synchronous mealy VHDL Code

```
LIBRARY ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_arith.all;

use ieee.std_logic_unsigned.all;

entity simple_door_mealy_sync is

    port

       (

         CLK: in std_logic; -- Clock signal

                              RSET: in std_logic; -- reset signal

         KEY: in std_logic_vector(1 downto 0); -- Input keys

         LEDG: out std_logic_vector(1 downto 0) --  outputs

               );

end entity;

 architecture model of simple_door_mealy_sync is

type STATE_TYPE is (oppen, close);

signal current_state,next_state: STATE_TYPE;

signal state: STATE_TYPE;

begin
```

```
clk_reset: PROCESS (rset,clk)
  BEGIN
    IF (rset='0') THEN
       state<= oppen;
                        LEDG<="01";
    ELSIF rising_edge(clk) THEN
     case state is
       when oppen =>
                        LEDG<="01";
        if(KEY = "10") then
                         LEDG<="10";
         state<= close;
     end if;
           when close =>
                  LEDG<="10";
        if KEY = "01" then
                           LEDG<="01";
          state<= oppen;
        end if;
    end case ;
               end if;
        --end if;
 end process;
 end model;
```

Synchronous do file

```
onerror {resume}
```

```
quietly WaveActivateNextPane {} 0

add wave -noupdate /simple_door_mealy_sync_vhd_tst/CLK

add wave -noupdate /simple_door_mealy_sync_vhd_tst/KEY

add wave -noupdate /simple_door_mealy_sync_vhd_tst/LEDG

add wave -noupdate /simple_door_mealy_sync_vhd_tst/RSET

TreeUpdate [SetDefaultTree]

WaveRestoreCursors {{Cursor 1} {0 ps} 0}

quietly wave cursor active 0

configure wave -namecolwidth 239

configure wave -valuecolwidth 100

configure wave -justifyvalue left

configure wave -signalnamewidth 0

configure wave -snapdistance 10

configure wave -datasetprefix 0

configure wave -rowmargin 4

configure wave -childrowmargin 2

configure wave -gridoffset 0

configure wave -gridperiod 1

configure wave -griddelta 40

configure wave -timeline 0

configure wave -timelineunits ps

update

WaveRestoreZoom {0 ps} {917 ps}
```

Advance door VHDL

```
LIBRARY ieee;

use ieee.std_logic_1164.all;

use ieee.std_logic_arith.all;

use ieee.std_logic_unsigned.all;

entity advance_door_mealy is

   port

      (

        CLK: in std_logic; -- Clk

                             RSET: in std_logic;-- Reset signal

        KEY: in std_logic_vector(3 downto 0); -- input keys

        LEDG: out std_logic_vector(2 downto 0);-- outputs

                      LEDR :out std_logic---- output

              );

end advance_door_mealy;

 architecture model of advance_door_mealy is

type STATE_TYPE is (error, oppen, close,lock);

signal state: STATE_TYPE;

--variable TMP : std_logic;

begin

clk_reset: PROCESS (rset,clk)

   BEGIN

     IF (rset='0') THEN

        state<= oppen;

   ELSIF rising_edge(clk) THEN

   case state is
```

```
when oppen =>
            if(KEY = "0001") then
   state<= close;
                         LEDG<="010";
                         LEDR <='0';
                 elsif(KEY = "1000") then
               state<= error;
                          LEDG<="000";
                          LEDR <='1';
 else
 state <= oppen;
                 LEDG<="001";
                 LEDR <='0';
 end if;
     when close =>
  if(KEY = "0100") then
   state<=lock;
                         LEDG<="100";
                         LEDR <='0';
                 elsif(KEY = "0010") then
               state<= oppen;
                          LEDG<="001";
                          LEDR <='0';
 else
 state <= close;
                 LEDG<="010";
```

```
                   LEDR <='0';
     end if;
             when lock =>
               if(KEY = "0001") then
       state<= close;
                          LEDG<="010";
                          LEDR <='0';
                   elsif(KEY = "1000") then
                 state<= error;
                          LEDG<="000";
                          LEDR <='1';
     else
     state <= lock;
                   LEDG<="100";
                   LEDR <='0';
     end if;
             when error =>
                 LEDG<="000";
             LEDR<='1';
                   when others =>
                        LEDG<="111";
   end case ;
      end if;
end process;
end model;
```

Advance door mealy do file

```
onerror {resume}

quietly WaveActivateNextPane {} 0

add wave -noupdate /advance_door_mealy_vhd_tst/CLK

add wave -noupdate /advance_door_mealy_vhd_tst/KEY

add wave -noupdate /advance_door_mealy_vhd_tst/LEDG

add wave -noupdate /advance_door_mealy_vhd_tst/LEDR

add wave -noupdate /advance_door_mealy_vhd_tst/RSET

TreeUpdate [SetDefaultTree]

WaveRestoreCursors {{Cursor 1} {0 ps} 0}

quietly wave cursor active 0

configure wave -namecolwidth 246

configure wave -valuecolwidth 100

configure wave -justifyvalue left

configure wave -signalnamewidth 0

configure wave -snapdistance 10

configure wave -datasetprefix 0

configure wave -rowmargin 4

configure wave -childrowmargin 2

configure wave -gridoffset 0

configure wave -gridperiod 1

configure wave -griddelta 40

configure wave -timeline 0

configure wave -timelineunits ps

update

WaveRestoreZoom {0 ps} {910 ps}
```

Advance door Test bench

```
LIBRARY ieee;

USE ieee.std_logic_1164.all;


ENTITY advance_door_mealy_vhd_tst IS

END advance_door_mealy_vhd_tst;

ARCHITECTURE advance_door_mealy_arch OF advance_door_mealy_vhd_tst IS

-- constants

-- signals

SIGNAL CLK : STD_LOGIC;

SIGNAL KEY : STD_LOGIC_VECTOR(3 DOWNTO 0);

SIGNAL LEDG : STD_LOGIC_VECTOR(2 DOWNTO 0);

SIGNAL LEDR : STD_LOGIC;

SIGNAL RSET : STD_LOGIC;

COMPONENT advance_door_mealy

        PORT (

        CLK : IN STD_LOGIC;

        KEY : IN STD_LOGIC_VECTOR(3 DOWNTO 0);

        LEDG : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);

        LEDR : OUT STD_LOGIC;

        RSET : IN STD_LOGIC

        );

END COMPONENT;

BEGIN

        i1 : advance_door_mealy

        PORT MAP (
-- list connections between master ports and signals
        CLK => CLK,
        KEY => KEY,
        LEDG => LEDG,
        LEDR => LEDR,
        RSET => RSET
        );
        clk_signal: process
   begin
         --------clOCK SIGNAL---------
      clk<='1';
                   wait for 10 ps;
                 --wait on clk, reset;
      clk<='0';
      wait for 10 ps;
                 --wait on clk, reset;
 end process;
init : PROCESS
-- variable declarations
BEGIN
-- code that executes only once
          ------CASE 1-------
   KEY<= "1111"; --
         rset<='1';
         wait for 50ps ;
         rset<='0';
         wait for 50ps ;
         rset<='1';
         wait for 50ps ;
   KEY<= "0001"; -- close door
         wait for 50ps ;
         ------CASE 1-------
          KEY<= "0100"; -- lock door
         wait for 50ps ;
   ----------------------
          KEY<= "0010"; -- open door
          wait for 50ps ;
   ----------------------
          KEY<= "1000"; -- error
          wait for 50ps ;
   ----------------------
          KEY<= "1111"; -- error
          wait for 50ps ;
          ----------------------
         rset<='0';
         wait for 50ps ;
         rset<='1';
WAIT;
END PROCESS init;
always : PROCESS
-- optional sensitivity list
-- (        )
-- variable declarations

BEGIN

        -- code executes for every event on sensitivity list

WAIT;

END PROCESS always;

END advance_door_mealy_arch;
```