EEE421

Introduction to VHDL

Automated House Security System

Project Report

Wasim Alfarram

210401063

Supervisor: Assist.Prof.Dr. Serpil Yılmaz

TABLE OF CONTENTS

1. Problem

This project is to design a controller, which provides an automated house security system at an affordable price. The parameter sensors connected to the controller will provide the required signals that will activate controller processes to and take the specified action.

2. Scope

A system is designed to control the doors, windows, fire alarm and the temperature. It is not intended or programmed to control anything else. Due to the short scope and to be realistically bound, certain assumptions are made in this applied project and certain concepts are overlooked at. This project is being implemented to control only these devices but can be expanded later to control more devices or processes and also can be web enabled.

3. Specifications

There are sensors for all four of the processes being automated. The sensors used are magnetic sensor (for doors and windows), smoke detector (for fire) and a thermostat (for temperature).

Every device will have a sensor that indicates how secure a system is by sending out a signal in case there is a change in the state. HIGH value (**1**) indicates the system will need to take action and LOW value (**0**) indicates the device is safe, so the next device can be checked.

The doors and windows operate on a magnetic sensor, if the signal on either is HIGH (**1**) then the operation is to close the door or buzz for the window to be closed. If the signal is a LOW (**0**), then the controller goes forward to check for the signal on the next device. There are separate magnetic sensors on both the front door and the rear door. The sensor on the fire alarm is a smoke detector, if it detects smoke then the signal goes to a HIGH (1) and it triggers off the buzzer, else it goes to the next step.

The temperature controller has a thermostat on it that responds to with the required action depending on the temperature and then goes back to the beginning. The no-action temperature range is from 50^{o} F – 70^{o} F. If the temperature goes below the range, then the heater turns itself on, if it goes above the range then the air conditioner turns itself on.

4. Assumptions

4.1. Operating the Front and the Rear Doors

Sensor on the door, which is a magnetic sensor, is always LOW (**0**) indicating the door is always closed. There are 2 doors and each has a different magnetic sensor attached to it. When the sensor detects a HIGH (**1**), the magnetic sensor automatically closes the door. This is achieved since a magnetic sensor is used.

4.2. Operating the Windows

The system covers all the windows. A +5V power is supplied to the system. Only the status of the window is checked for and if the window is open (**1**) then the system buzzes until it is manually turned off. The default state of the window is LOW (**0**). The sensor on the window, which is same as the sensor on the door, a magnetic sensor, is always on LOW (**0**) indicating the window is closed always.

4.3. Operating the Fire Alarm

A smoke detector is present to detect the smoke. There is a constant source of electricity of +5V to the sensor and devices. If there is any smoke detects a HIGH (**1**) on the sensor, the fire alarm starts buzzing and continues to do so until it has been turned off manually, it sends out the signal. The state of the fire alarm then returns to LOW (**0**), and this is the default state.

4.4. Operating the Temperature

The range from 50^{o} F to 70^{o} F is considered to be the comfortable range in the home. The sensor used here is the thermostat, which measures the changes in room temperature and turns on the heater or the cooler based on the temperature inside the room. Assuming the sensor records temperature below 50^{o} F, then the heater is turned on, and the air conditioner turns itself on if the temperature is above 70^{o} F. For both the conditions the sensor goes HIGH (**1**), else it remains on the LOW (**1**) state. The power is from a +5V power source.

5. Approach and Flow Diagram

Following the hardware design concepts, the first step taken in designing this system is to figure out what equipment needs to be secure. This is the initial stage wherein the devices that need to be connected for security are short-listed and then they are put in a network and so the corresponding work is done. The devices are doors (doors include front and rear doors), windows, fire alarm and the temperature controller. The block diagram showing all the inputs and outputs is drawn thus making it clear what inputs are required to drive which outputs.

As explained earlier, the signals on the attached sensors are captured and hence the required action is taken either on the same device or proceeds to the next device.

The Block diagram of the controller is as shown Figure 1:

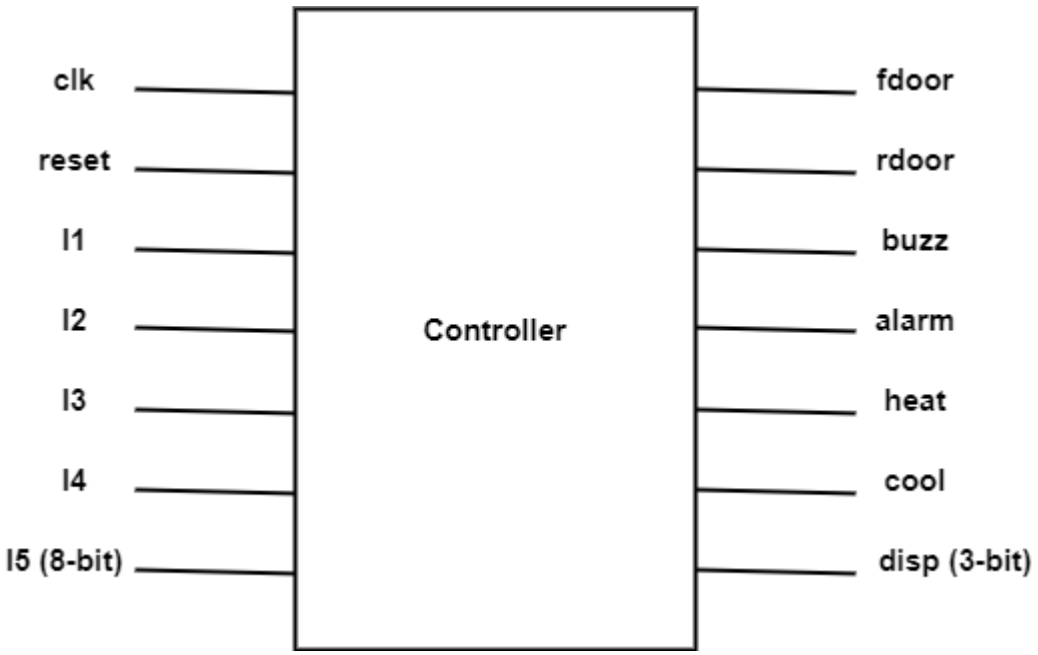

Figure 1: Controller Block Diagram of the System

The input signals are:
clk = clocks the device.
reset = resets the state of the system back to default.
I1 = signal from the sensor on the Front Door.
I2 = signal from the sensor on the Rear Door.
I3 = signal from the sensor on the Window.
I4 = signal from the sensor on the Fire Alarm.
I5 = signal from the sensor on the Temperature Controller.

The output signals are:

fdoor = HIGH (1) -> close the front door, LOW (0) -> reset and wait for the next command.
rdoor = HIGH (1) -> close the rear door, LOW (0) -> reset and wait for the next command.
buzz = buzz if window is open, i.e., HIGH (1).
alarm = buzz if fire alarm is on, i.e., HIGH (1).

heat = if temperature is less than 50^{o} F turn heater on.
cool = if temperature is more than 70^{o} F turn cooler on.
disp = 3-digit display that shows which state the system is in.

The next step after the block diagram is to obtain the flow diagram of the same showing the different states and conditions. Initially, a simple flowchart is drawn in which the conditions and the required action to be taken are mentioned and then the detailed flowchart follows, in which the states and the devices will be shown.

The state diagram step is the next to follow in which all the states are shown and also see if the states can be reduced. A state diagram is developed based on a FSM Moore Design that shows all the states and their behaviors. The states are examined for redundancy and reduced if any are found. This analysis and reduction help to obtain a minimum and race free system.

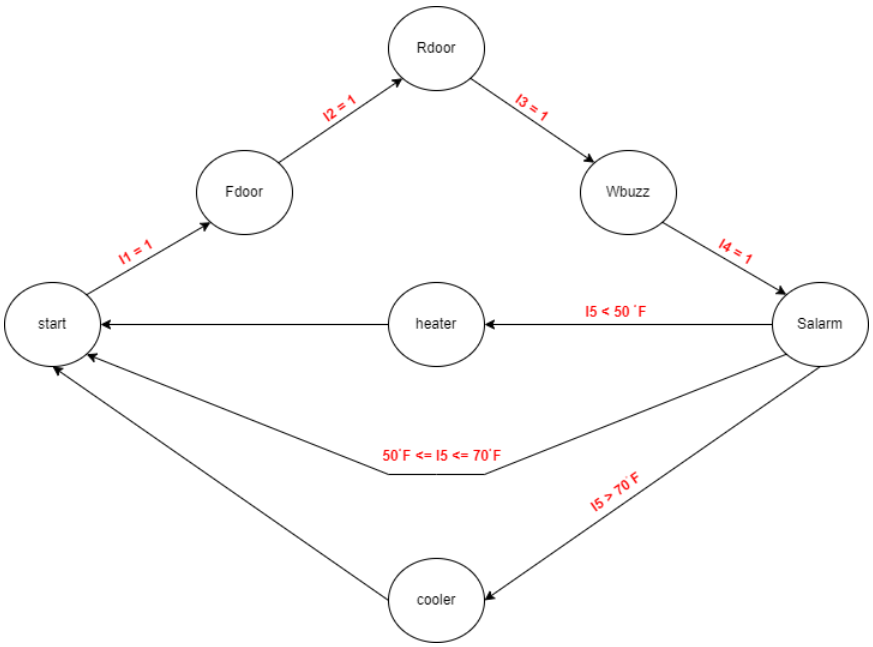

Figure 2: The State Diagram of the System

cur_st	**nxt_st**	**disp** (of the cur_st)
start	closeFdoor	000
closeFdoor	closeRdoor	001
closeRdoor	Wbuzz	010
Wbuzz	Salarm	011
Salarm	heater	100
heater	cooler	101
cooler	start	110

Table 1: The Current and Next State

6. States of the System

6.1. Front Door State

In this state the status of the front door is checked for and if the input from the magnetic sensor on the front door is LOW (**0**) that indicates the door is closed, while if it is HIGH (**1**) then the door is open. If I1 = 0, then the system goes to the next state else if I1 = 1, then the door is closed and the system goes to the next state which is the rear door state.

6.2. Rear Door State

Just as in the front door state, the status of the rear door state is checked for first and if the input it receives from the magnetic sensor on the rear door is LOW (**0**) that means the door is closed, else the door is open. If I2 = 0, then system goes to the next state, else if I2 = 1, then the rear door is closed and the system goes to the next state which is the window state.

6.3. Window State

Here also, the status of the window is checked, LOW (**0**) indicates the window is closed, and HIGH (**1**) means the window is open. The sensor on the window is a magnetic sensor. If I3 = 0, the system goes to the next state, else if I3 = 1, then the window triggers off a buzzer and it keeps buzzing until it is manually turned off. The system in the meanwhile goes to the next state, which is the fire alarm state.

6.4. Fire Alarm State

The sensor here is the smoke detector and if it detects smoke then the input it sends is a **1** else it remains at **0**. If I4 = 0, the system goes to the next state meaning there is no fire, but if I4 = 1, means there is smoke and hence it triggers of a buzzer and goes to the next state which is the temperature state. The buzzer is on till it is manually turned off.

6.5. Temperature Controller State

This state has 3 conditions to fulfill based on the input it receives. The thermostat on the temperature regulator keeps checking the temperature of the room. If it goes below 50^{o} F, then the system turns on the heater and if temperature goes beyond 70^{o} F, then the air conditioner is turned on. Else, the system goes back to the start state. The specifications for this state are as follows:

- If **I5 > 70^{o} F**, then turn on the air conditioner.
- If **I5 < 50^{o} F**, then the heater is turned on.
- If the temperature is **equal to 50^{o}F or 70^{o} F or anywhere between that span** then no action is taken and the system goes back to the start state.

7. Basic Code Implementation

```
library IEEE;
use IEEE.std_logic_1164.all;


entity project_entity is  -- should include IEEE.std_logic_1164.all
port(
clk, reset, I1, I2, I3, I4: in std_logic;
I5: in std_logic_vector(7 downto 0);
fdoor, rdoor, buzz, alarm, heat, cool: out std_logic;
disp: out std_logic_vector(2 downto 0)
);
end project_entity;


architecture project_architecture of project_entity is
type states is (start, closeFdoor, closeRdoor, Wbuzz, Salarm, heater, cooler);
signal cur_st, nxt_st: states;
begin

cur_st_pr: process(clk, reset)
begin
if (reset = '1') then
        cur_st <= start;
elsif (rising_edge(clk)) then
        cur_st <= nxt_st;
end if;
end process cur_st_pr;

nxt_st_pr: process(cur_st, I1, I2, I3, I4, I5)
begin
case cur_st is
when start =>
        if (I1 = '1') then
                nxt_st <= closeFdoor;
        elsif (I2 = '1') then
                nxt_st <= closeRdoor;
        elsif (I3 = '1') then
                nxt_st <= Wbuzz;
        elsif (I4 = '1') then
                nxt_st <= Salarm;
        elsif (I5 < "00110010") then
                nxt_st <= heater;
        elsif (I5 > "01000110") then
                nxt_st <= cooler;
        else
                nxt_st <= start;
        end if;
when closeFdoor =>
```

```
		if (I2 = '1') then
			nxt_st <= closeRdoor;
		elsif (I3 = '1') then
			nxt_st <= Wbuzz;
		elsif (I4 = '1') then
			nxt_st <= Salarm;
		elsif (I5 < "00110010") then
			nxt_st <= heater;
		elsif (I5 > "01000110") then
			nxt_st <= cooler;
		else
			nxt_st <= start;
		end if;
	when closeRdoor =>
		if (I3 = '1') then
			nxt_st <= Wbuzz;
		elsif (I4 = '1') then
			nxt_st <= Salarm;
		elsif (I5 < "00110010") then
			nxt_st <= heater;
		elsif (I5 > "01000110") then
			nxt_st <= cooler;
		else
			nxt_st <= start;
		end if;
	when Wbuzz =>
		if (I4 = '1') then
			nxt_st <= Salarm;
		elsif (I5 < "00110010") then
			nxt_st <= heater;
		elsif (I5 > "01000110") then
			nxt_st <= cooler;
		else
			nxt_st <= start;
		end if;
	when Salarm =>
		if (I5 < "00110010") then
			nxt_st <= heater;
		elsif (I5 > "01000110") then
			nxt_st <= cooler;
		else
			nxt_st <= start;
		end if;
	when heater =>
		if (I5 > "01000110") then
			nxt_st <= cooler;
		else
			nxt_st <= start;
		end if;
	when cooler =>
```

```
            nxt_st <= start;
    end case;
    end process nxt_st_pr;

    out_reg_pr: process(clk, reset)
    begin
    case cur_st is
    when start =>
            fdoor <= '0';
            rdoor <= '0';
            buzz <= '0';
            alarm <= '0';
            cool <= '0';
            heat <= '0';
            disp <= "000";
    when closeFdoor =>
            fdoor <= '1';
            rdoor <= '0';
            buzz <= '0';
            alarm <= '0';
            cool <= '0';
            heat <= '0';
            disp <= "001";
    when closeRdoor =>
            fdoor <= '0';
            rdoor <= '1';
            buzz <= '0';
            alarm <= '0';
            cool <= '0';
            heat <= '0';
            disp <= "010";
    when Wbuzz =>
            fdoor <= '0';
            rdoor <= '0';
            buzz <= '1';
            alarm <= '0';
            cool <= '0';
            heat <= '0';
            disp <= "011";
    when Salarm =>
            fdoor <= '0';
            rdoor <= '0';
            buzz <= '0';
            alarm <= '1';
            cool <= '0';
            heat <= '0';
            disp <= "100";
    when heater =>
            fdoor <= '0';
            rdoor <= '0';
```

```
            buzz <= '0';
            alarm <= '0';
            heat <= '1';
            cool <= '0';
            disp <= "101";
    when cooler =>
            fdoor <= '0';
            rdoor <= '0';
            buzz <= '0';
            alarm <= '0';
            heat <= '0';
            cool <= '1';
            disp <= "110";
    end case;
    end process out_reg_pr;

    end project_architecture;
```

8. Testbench Code Implementation

```
library IEEE;
use IEEE.std_logic_1164.all;


entity project_entity_TB is
end project_entity_TB;


architecture project_architecture_TB of project_entity_TB is  -- should include
IEEE.std_logic_1164.all
---------------------- Component Declaration ----------------------
component project_entity
port(
clk, reset, I1, I2, I3, I4: in std_logic;
I5: in std_logic_vector(7 downto 0);
fdoor, rdoor, buzz, alarm, heat, cool: out std_logic;
disp: out std_logic_vector(2 downto 0)
);
end component;
-----------------------------------------------------------------
---------------- Signals and Constants Declarations ---------------
signal clks, I1s, I2s, I3s, I4s: std_logic;
signal I5s: std_logic_vector(7 downto 0);
signal resets: std_logic := '0';
signal fdoors, rdoors, buzzs, alarms, heats, cools: std_logic;
signal disps: std_logic_vector(2 downto 0);
constant clock_period : time := 10 ns;
-----------------------------------------------------------------
begin
---------------------------- Port Map ----------------------------
project_entity_pm: project_entity port map(
clk => clks,
reset => resets,
I1 => I1s,
I2 => I2s,
I3 => I3s,
I4 => I4s,
I5 => I5s,
fdoor => fdoors,
rdoor => rdoors,
buzz => buzzs,
alarm => alarms,
heat => heats,
cool => cools,
disp => disps
);
-----------------------------------------------------------------
----------------------- (1) clk_PR Process -----------------------
```

```
clk_PR: process  -- the system waits until the process generates on the clock
begin
clks <= '0';
wait for clock_period/2;
clks <= '1';
wait for clock_period/2;
end process clk_PR;
-------------------------------------------------------------------
--------------------- (2) stimulus_PR Process ---------------------
stimulus_PR: process  -- generates on the outputs and the simulation
begin
---------------------------------- Case(1): Rear Door is closed, Heater is turned on.
I1s <= '0';
I2s <= '1';  -- (close the Rear Door!)
I3s <= '0';
I4s <= '0';
I5s <= "00111001";  -- 57° F
wait for clock_period;
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "00110000";  -- 48° F (turn on Heater!)
wait for clock_period;
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "00110100";  -- 52° F
wait for clock_period;
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "00111011";  -- 59° F
wait for clock_period;
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "01000000";  -- 64° F
wait for clock_period;
---------------------------------- Case(2): Window is closed.
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "00110100";  -- 52° F
wait for clock_period;
I1s <= '0';
```

```
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "00111001";  -- 57° F
wait for clock_period;
I1s <= '0';
I2s <= '0';
I3s <= '1';  -- (close the Window!)
I4s <= '0';
I5s <= "00111011";  -- 59° F
wait for clock_period;
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "01000000";  -- 64° F
wait for clock_period;
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "00111000";  -- 56° F
wait for clock_period;
---------------------------------- Case(3): Smoke Alarm is turned on, Cooler is turned on.
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "00110100";  -- 52° F
wait for clock_period;
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "00111011";  -- 59° F
wait for clock_period;
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '0';
I5s <= "01000001";  -- 65° F
wait for clock_period;
I1s <= '0';
I2s <= '0';
I3s <= '0';
I4s <= '1';  -- (turn on Smoke Alarm!)
I5s <= "01000100";  -- 68° F
wait for clock_period;
I1s <= '0';
I2s <= '0';
```

```
I3s <= '0';
I4s <= '0';
I5s <= "01001011";  -- 75° F (turn on Cooler!)
wait for clock_period;
end process stimulus_PR;
-------------------------------------------------------------
end project_architecture_TB;
```

9. Simulation Results

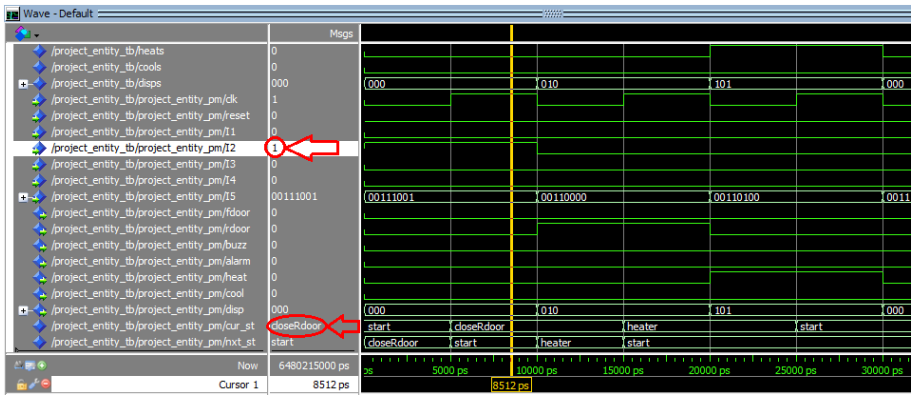

Figure 3: When Input Data of The Rear Door Sensor is 1

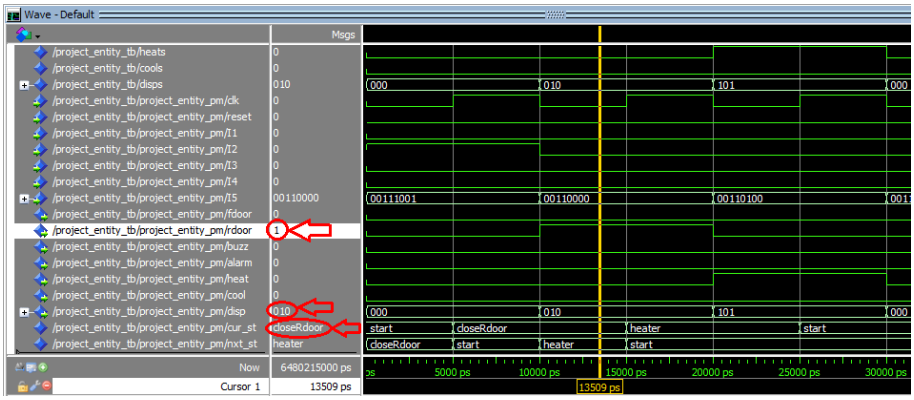

Figure 4: Closed Rear Door

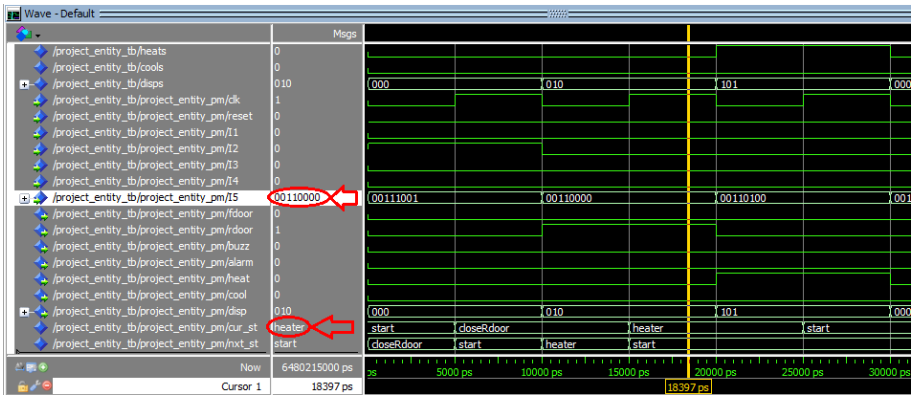

Figure 5: When Input Data of The Temperature Sensor is less than 50 F

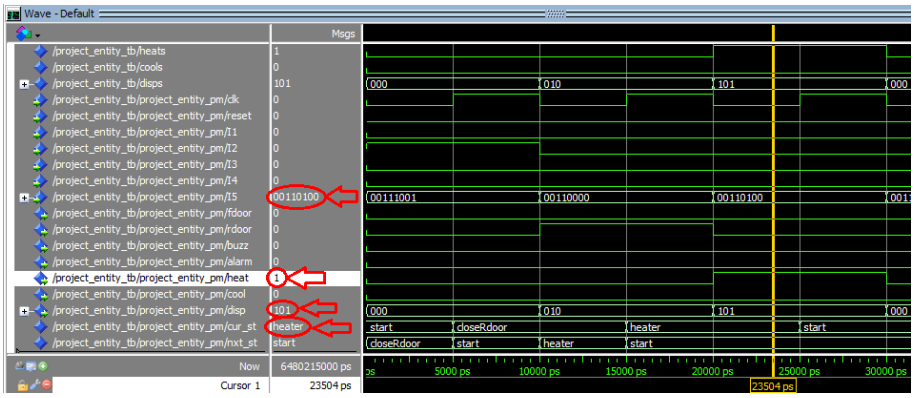

Figure 6: Activation of Heater

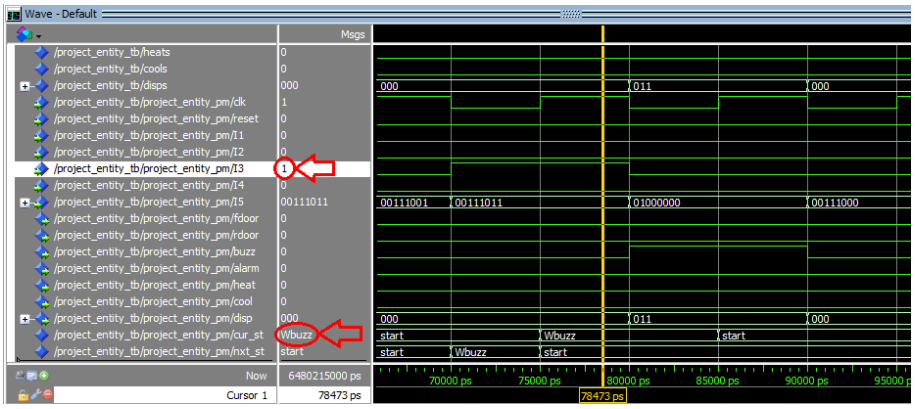

Figure 7: When Input Data of Window Sensor is 1

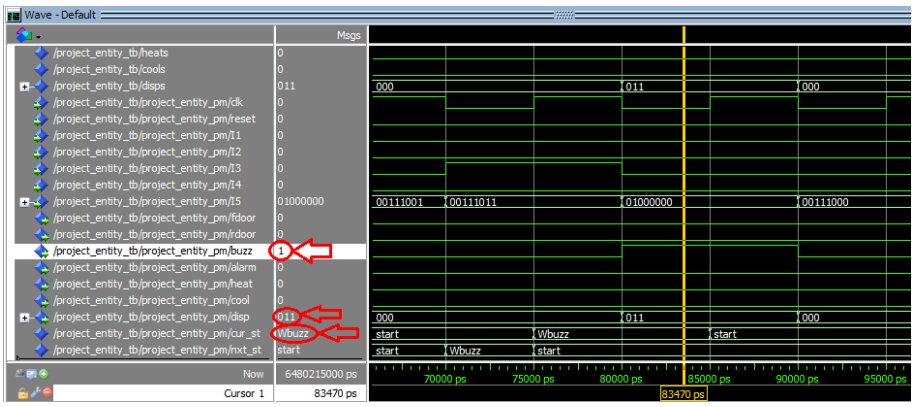

Figure 8: Closed Window

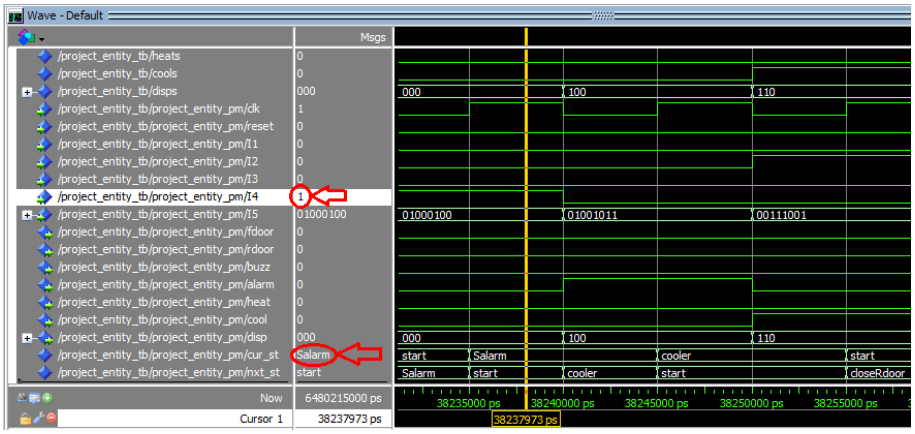

Figure 9: When Input Data of The Smoke Alarm Sensor is 1

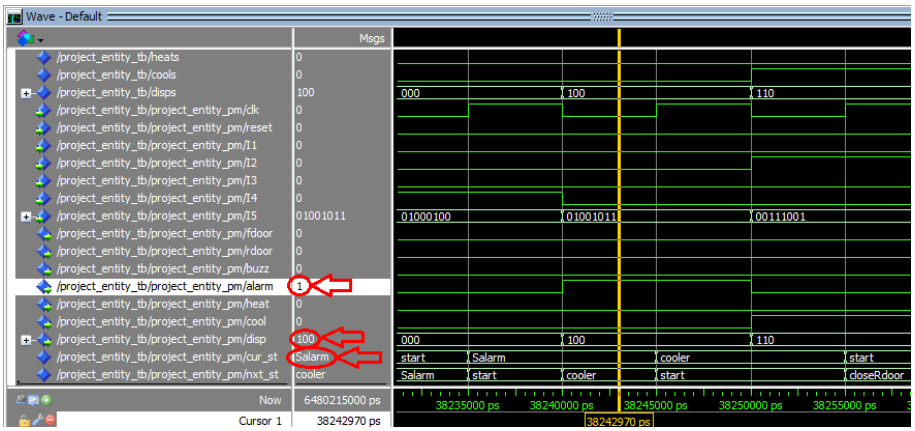

Figure 10: Activation of Smoke Alarm

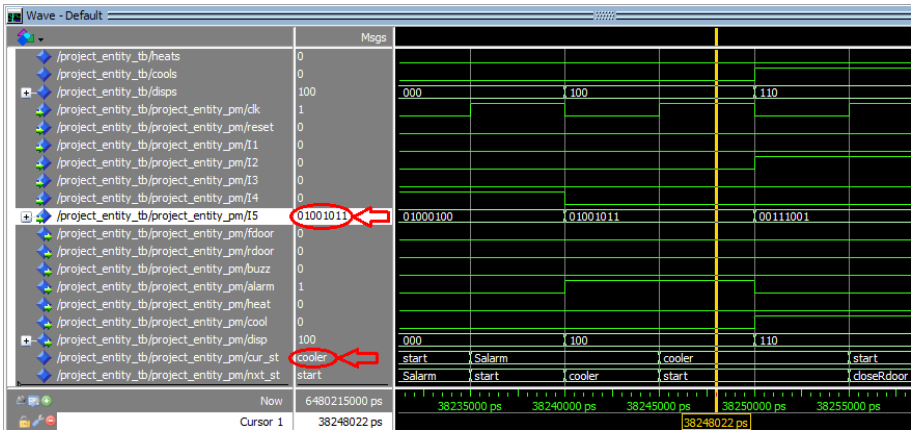

Figure 11: When Input Data of The Temperature Sensor is higher than 70 F

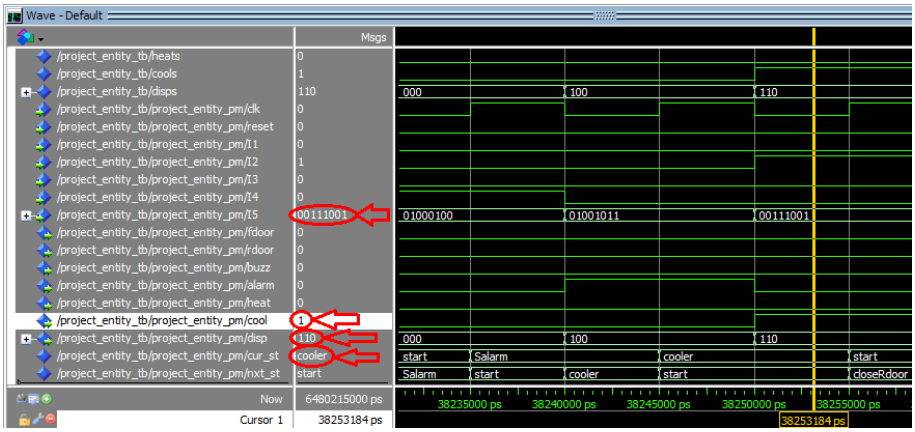

Figure 12: Activation of Cooler