

Home Automation System Design Using Verilog Hardware Descriptive Language

Komol Arafat. Gani, Farzana. Yasmin, A B M Najmul. Karim, Iqbalur. Rahman

Abstract—Home which is often referred as “sweet home” is indeed sweet if we introduce a home automation system. With such thought in mind a design of a home automation system is formed that will check security and comfort of a home. Security system includes detection of fire, intruders through doors, windows and garage protection. The comfort system is designed to control the temperature and luminosity only. In this paper we introduced an efficient design of home automation system using Verilog HDL and a possible solution where the user controls device by employing a central Field Programmable Gate Array (FPGA) controller to which the devices and sensors are interfaced. This project is a reflection of digital system design to achieve our goal. We simulated the design in Verilog HDL using Xilinx and ModelSim.

The solution of this project is in agreement with our expected output which is readily visible through our wave.

Keywords— Home Automation System, Simulation, Synthesis and Verilog HDL.

I. INTRODUCTION

IN the present time, everybody is always in great need of security. Mostly during recession, crime rate like burglary and thief reaches its peak and so home needs to be more secured. Such need can be fulfilled using a home automation system which will provide secure and comfortable environment for living. The security system engages itself to provide the security against any unwanted happening. The comfort system is responsible for providing a comfortable environment for the host in the house.

Security had always been a particular word which is of great importance in business and home. Whereas comfort is the prime word when a house is taken into consideration, when these two words meet a house becomes a perfect place for

living.

The basic home automation system will try to meet host's demand and make the home as safe and comfortable as possible [1]. Various products are available in the market which is often very costly. Sometimes you need to preplan about a home automation before making a home. Also there are some product those are discrete. This problem can be solved through a single program that checks security than comfort. Therefore home automation system can control all the desired things in one go. This will let the owner relax perfectly and will not have to panic around checking doors and windows often.

Our home automation system works in two phases. When the owner comes home from outside he disarm the system using password then he enter the home and arm the system again so that the entire home automation is activated. When the users are not present at home only the security system is activated. The entire system is dependent upon multiple sensors which act as input to the program.

A. Problem

The target of the project is to provide the maximum security and comfort to home users at the cheapest price.

B. Objective

The prime objective of the project is designing a home automation system that is capable of providing high security and always maintaining desired temperature, luminosity and controlling according to human presence.

The academic goal of this project is to develop specific skills in designing, programming, testing and debugging.

C. Project Scope

Our system was designed to control the door, window, garage door, fire alarm, luminosity, and temperature. It is not designed to control any other device.

D. Assumption

Due to the limitation of the time constraints and also due to the hardware inaccessibility and expense, it was assumed that readily available sensors are used on all the devices. The scope of the project is only controlling the device internally.

II. APPROACH

Initially the priority is set, and then the design is taken into consideration. By keeping a virtual house as role model the

Komol Arafat. Gani, is the final year student studying Electrical and Electronics Engineering, North South University, Block-B, Bashundhara, Baridhara, Dhaka 1229, Bangladesh (e-mail: komolfaruq@gmail.com).

Farzana. Yasmin is the final year student studying Electrical and Electronics Engineering, Department of Electrical, Electronics and Computer Science, North South University, Block-B, Bashundhara, Baridhara, Dhaka 1229, Bangladesh (e-mail: farzana.northsouth@gmail.com).

A B M Najmul Karim is a graduate student from Department of Electrical Engineering and Computer Science, North South University, Dhaka, Bangladesh (e-mail: tanvir.nsuer@gmail.com).

Iqbalur Rahman Rokan is a Faculty member in North South University, Dhaka, Bangladesh. Former Sr. Engineer, VLSI Chip Research and Development (R&D), Emulex Corporation, California, USA. (Phone: +88-01726246189 ; e-mail: irahman@northsouth.edu).

design set is drawn, then the placement of the sensors are decided. Later all the sensors are attached under one network, the devices are door, window, garage door, fire alarm and the temperature controller [2]. The RTL schematic is shown in Fig.1 by synthesizing under Xilinx showing all the inputs and outputs.

A. The block diagrams:

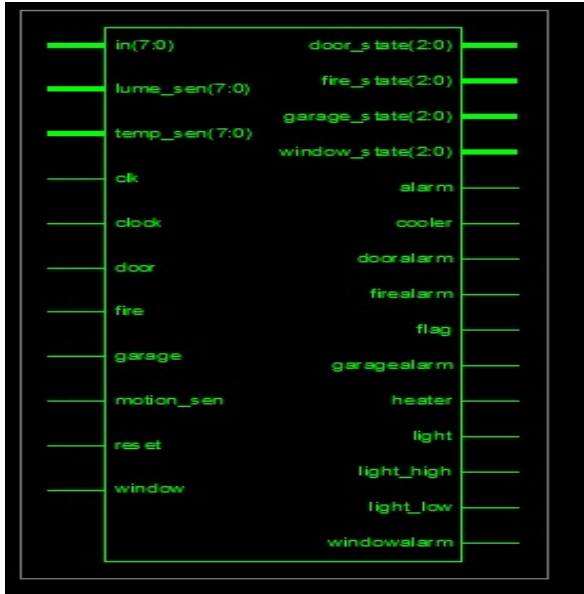


Fig. 1 Home Automation system [3]

The input signals are

In [7:0] = input password to arm and disarm security
 lume_sen [7:0] = signal from the optical sensor,
 temp_sen[7:0] = signal from the thermostat,
 clk = clock,
 door = signal from magnetic sensor,
 fire = signal from the smoke detector.
 garage = signal from laser sensor,
 motion_sen= detects presence of human.
 reset = reset,
 window = signal from magnetic sensor.

The output signals are:

door_state[2:0] = door open or close,
 fire_state[2:0] = detect smoke or not,
 window_state[2:0] = window open or close,
 garage_state[2:0] = garage open or close,
 alarm = alarm if password wrong otherwise zero,
 cooler = High when temp_sen > 30°C
 dooralarm = High when door is open, otherwise low
 firealarm = high when smoke is detected
 flag = high, then security module works.
 garagealarm = high when garage door is open, otherwise low.
 heater = High when temp_sen < 15 °C,
 light = when luminosity is between (10 – 15)lumen
 light_high = High when lume_sen < 10 lumen,
 light_low = High when lume_sen > 15 lumen.
 windowalarm = high when window is open, otherwise low.

The next state after the block diagram is to create the hierarchy and state diagram which explains diagram showing the different states and condition. The design is for every module, which includes the states, the devices, the conditions and the required action.

B. Hierarchy

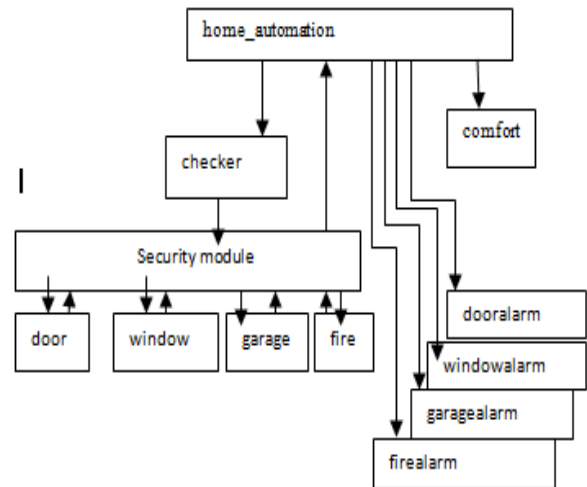


Fig. 2 Home Automation System

Home automation: As shown in Fig.2. This module is the one that takes all the inputs and provides output to the outer world. The user password which is input from the user is passed to the checker module. It also sets the security alarms high or low depending on the values from the other module outputs.

Checker module: is used to arm and disarm the security system. An 8 bit password is set, in[7:0], by the owner of the home. This password check module is responsible to turn on and off the security of the home.

Security module: It is responsible for transferring the data from the sensors to the individual door, garage, window, fire modules. It also plays role in passing the outputs from the door, garage, window and fire module to the home_automation module.

Door module: in this state the door is monitored continuously. If the door is open and the magnetic contact is broken. This is when the door alarm goes high. Both the window and garage modules work similarly.

Fire module: in this state the smoke detector is always set low (0). Upon coming in contact with smoke the smoke detector automatically sends high signal which turns on the fire alarm.

C. State Diagram

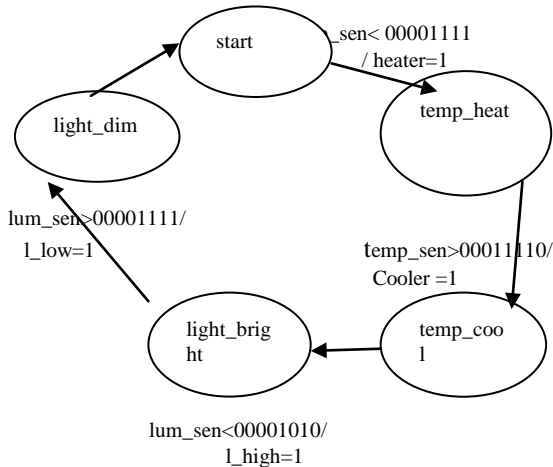


Fig.2 (a) Comfort Module

The comfort module is designed using Finite State Machine [4].

start state: as shown in Fig.2(a) The first state is the start state which can also be called the reset where the sensor of the comfort module is zero.

temp_heat state: When the temperature is below (00001111) i.e. 15°C, it sends out a high signal specifically to turn on the heater.

temp_cool state: When temperature is greater than (00011110) i.e. 30°C, cooler =1.

light_bright state: When the sensor detects luminosity less than 10 lumen then l_high is turned on to brighten the room.

light_dim state: Similarly if light is more than the specific lumen i.e. 15 then the light_low turn on to decrease the light

II. THEORY

A. Verilog HDL

In the semiconductor and electronic device industry, Verilog is a hardware descriptive language used to model electronic systems. Along with VHDL, Verilog HDL is another most commonly used language in the design, verification and implementation of digital logic chips at the register transfer level of abstraction. It is also used in the verification of analog and mixed signal circuit. A Verilog design contains of a hierarchy of modules which are later on joined and works as a complete system [5], [6].

III. IMPLEMENTATION

A. Codes

Home_Automation_system:

```

Top Module: ///////////////////////////////////////////////////////////////////
// University: North South University
// Students: Farzana Yasmin, Komol Arafat
// Create Date: 19:00:00 11/08/2013
// Module Name: Home Security
// Project Name: Home Automation
  
```

```

/////////////////////////////////////////////////////////////////
  
```

```

module Home_automation(
clock,reset,door,garage>window,fire, in,
clk,motion_sen,temp_sen,
lume_sen,light,heater,cooler,light_high,light_low,dooralarm,g
aragealarm>windowalarm,firealarm>window_state,garage_state
,door_state,fire_state,flag,alarm);
input [7:0]in;wire [7:0]in;reg [7:0]arr [0:3];reg [7:0]arr1
[0:3];
input clock,reset,door,garage, window, fire,clk;
output windowalarm,dooralarm,
garagealarm,firealarm,flag,alarm,light,heater,cooler,light_high,
light_low;
input motion_sen;input [7:0] temp_sen, lume_sen;wire [7:0]
temp_sen, lume_sen;
wire light,heater,cooler,light_high,light_low,clk;
output wire [2:0] garage_state,door_state,
window_state,fire_state;
wire
dooralarm,garagealarm,firealarm>windowalarm,flag,alarm,cloc
k,reset;
top_security T1 ( clock, reset,door,garage>window,fire, in,
clk,dooralarm,garagealarm>windowalarm,firealarm>window_st
ate,garage_state,door_state,fire_state,flag,alarm);
comfort C1
(clk,reset,motion_sen,temp_sen,lume_sen,light,heater,cooler,li
ght_high,light_low);
endmodule
  
```

• Top_security:

```

module top_security( clock, reset,door,garage>window,fire, in,
clk,dooralarm,garagealarm>windowalarm,firealarm>window_st
ate,garage_state,door_state,fire_state,flag,alarm);
input [7:0]in;wire [7:0]in;input clock, reset,clk,door,garage,
>window, fire;
output windowalarm,dooralarm,
garagealarm,firealarm,flag,alarm;output wire [2:0]
garage_state,door_state>window_state,fire_state;
wire
dooralarm,garagealarm,firealarm>windowalarm,flag,alarm,clk;
password ppa
(.clock(clock),.reset(reset),.in(in),.flag(flag),.alarm(alarm));
security S1
(.flag(flag),.clock(clock),.reset(reset),.door(door),.window(win
dow),.garage(garage),.fire(fire),.window_state(window_state),.
>windowalarm(windowalarm),.garage_state(garage_state),.gara
gealarm(garagealarm),.door_state(door_state),.dooralarm(door
alarm),.fire_state(fire_state),.firealarm(firealarm));
endmodule
  
```

• Password_check:

```

module password(clock, reset, in,flag,alarm);
input [7:0]in;input clock, reset;output reg flag,alarm;reg
[7:0] arr[3:0];reg [7:0] arr1[3:0];
integer i;
initial begin flag=1'b0;end
initial begin
for(i=0; i<8; i=i+8)
begin arr[i]=i+8;end
  
```

```

for(i=0; i<3; i=i+3)
begin arr1[i]=i+3;end end
always @ ( in )
begin flag=1'b0;
begin: block1
for(i=0; i<1; i=i+1)
if(in==arr[i])
begin flag=1'b1;alarm=1'b0;end
else if(in==arr1[i])
begin flag=1'b0;alarm=1'b0;end
else if(in==8'd0)odule password(clock, reset,
in,flag,alarm);
input [7:0]in; input clock, reset;output reg flag,alarm;reg
[7:0] arr[3:0];reg [7:0] arr1[3:0];
integer i;
initial begin flag=1'b0;end
initial begin
for(i=0; i<8; i=i+8)
begin arr[i]=i+8;end
for(i=0; i<3; i=i+3)
begin arr1[i]=i+3;end end
always @ ( in )
begin flag=1'b0;
begin: block1
for(i=0; i<1; i=i+1)
if(in==arr[i])
begin flag=1'b1;alarm=1'b0;end
else if(in==arr1[i])
begin flag=1'b0;alarm=1'b0;end
else if(in==8'd0)
alarm=1'b0;
else alarm=1'b1;
disable block1; end end
endmodule
alarm=1'b0;
else alarm=1'b1;
disable block1; end end
endmodule
• Security:
module security(flag,clock, reset, door>window,
garage,fire>window_state, windowalarm, garage_state,
garagealarm, door_state, dooralarm,fire_state, firealarm);
input clock, reset,flag,door, garage, fire, window; output [2:0]
window_state,garage_state,door_state,fire_state;
output garagealarm,dooralarm,firealarm>windowalarm; wire
garagealarm>windowalarm,dooralarm,firealarm;
fire f0
(.flag(flag),.clock(clock),.reset(reset),.fire(fire),.fire_state(fire_
state),.firealarm(firealarm));
door f1
(.flag(flag),.clock(clock),.reset(reset),.door(door),.door_state(d
oor_state),.dooralarm(dooralarm));
window f2
(.flag(flag),.clock(clock),.reset(reset),.window(window),.wind
ow_state(window_state),.windowalarm>windowalarm));
garage f3
(.flag(flag),.clock(clock),.reset(reset),.garage(garage),.garage_
state(garage_state),.garagealarm(garagealarm));

```

```

endmodule

```

- Fire

```

module fire(flag,clock, reset, fire, fire_state,
firealarm);
input clock, reset,flag, fire; output [2:0] fire_state;
output firealarm; wire firealarm;reg [2:0] fire_state;
assign firealarm = (fire_state == 1) ?(flag?0:1): 0; // if
burglary state, signal a burglary
always @(posedge clock)
fire_state <= fire ? 1 : 0; // go to burglary state if fire
is on
endmodule

```
- Door

```

module door(flag,clock, reset,
door,door_state,dooralarm);
input flag,clock, reset, door; output [2:0] door_state;
output dooralarm;wire dooralarm;
reg [2:0] door_state;
assign dooralarm = (door_state == 1) ? (flag?0:1) : 0;
// if burglary state, signal a burglary
always @(posedge clock)
door_state <= door ? 1 : 0; // go to burglary state if
fire is on
endmodule

```
- Window

```

module window(flag,clock, reset, window,
window_state, windowalarm);
input clock, reset,flag>window; output [2:0]
window_state;
output windowalarm; wire windowalarm;reg [2:0]
window_state;
assign windowalarm = (window_state == 1) ?
(flag?0:1): 0; // if burglary state, signal a burglary
always @(posedge clock)
window_state <= window ? 1 : 0; // go to burglary
state if window is on
endmodule

```
- Garage

```

module garage(flag,clock, reset, garage, garage_state,
garagealarm);
input clock, reset,flag, garage; output [2:0]
garage_state;
output garagealarm; wire garagealarm;
reg [2:0] garage_state;
assign garagealarm = (garage_state == 1) ?
(flag?0:1): 0; // if burglary state, signal a burglary
always @(posedge clock)
garage_state <= garage ? 1 : 0; // go to burglary state
if garage is on
endmodule

```
- Comfort

```

`define start 4'd0
`define temp_heat 4'd1
`define temp_cool 4'd2
`define light_bright 4'd3
`define light_dim 4'd4
module

```

```

comfort(clk,reset,motion_sen,temp_sen,lume_sen,light
t,heater,cooler,light_high,light_low);
input clk,reset,motion_sen;input [7:0]
temp_sen,lume_sen;
output reg heater,cooler,light_high,light_low,light;
reg [3:0] current_state;reg [3:0] next_state;wire clk;
initial begin
current_state=`start; next_state=`start;
heater='b0; cooler='b0;
light_high='b0; light_low='b0; light='b0;
end
always @(posedge clk)
current_state=next_state;
always @(current_state)
begin
case(current_state)
`start: begin
heater='b0; cooler='b0; light_high='b0;
light_low='b0;light='b0;
end
`temp_heat: begin if(motion_sen==1)
begin heater='b1;
cooler='b0;light='b1;end
else
heater='b0;end
`temp_cool:begin if(motion_sen==1)
begin cooler='b1;heater
='b0;light='b1;end
else cooler='b0;end
`light_bright:begin if(motion_sen==1)
begin light_high='b1;
light_low='b0;light='b1;end
else light_high='b0;end
`light_dim:begin if(motion_sen==1)
begin light_low='b1; light_high='b0;end
else light_low='b0; end
endcase
end
always
@(current_state,temp_sen,lume_sen,reset)
begin
if(reset=='b1)
next_state=`start;
else
case(current_state)
`start: begin
if(temp_sen> 'b00011110)
next_state=`temp_cool;
else if(temp_sen< 'b00001111)
next_state=`temp_heat;
else if(lume_sen > 'b00001111)
next_state=`light_dim;
else if (lume_sen < 'b00001010)
next_state=`light_bright;
end
`temp_cool: begin
if(temp_sen< 'b00001111)
next_state=`temp_heat;

```

```

else if (lume_sen > 'b00001111)
next_state=`light_dim;
else if(lume_sen < 'b00001010)
next_state=`light_bright;
end
`temp_heat: begin
if(temp_sen> 'b00011110)
next_state=`temp_cool;
else if (lume_sen > 'b00001111)
next_state=`light_dim;
else if(lume_sen < 'b00001010)
next_state=`light_bright;
end
`light_dim: begin
if(lume_sen < 'b00001111)
next_state=`light_bright;
end
`light_bright: begin
next_state=`start;end
endcase
end
endmodule

```

IV. RESULT

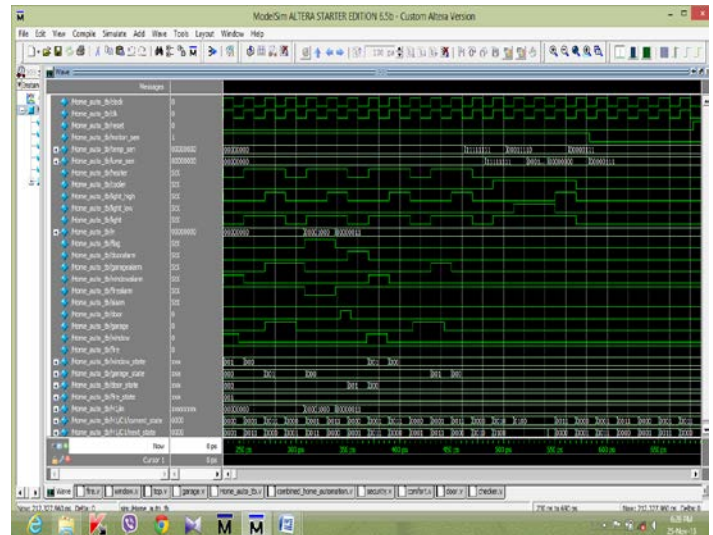


Fig. 3 ModelSim waveform after simulation

If the motion_sen(motion sensor) is high the comfort system works or else all appliances are off. At every positive edge of the clock values from the motion_sen, temp_sen, lume_sen are checked. the motionDuring the period when motion_sen is high, when the temp_sen is set to 8'b0 then heater starts which is 0001 state, then the lume_sen is checked which is set to 8'b0 so the next state shifts to light_bright state(0011) where the light_high is set high to increase the light intensity after that next state is the start state(0000) where the heater and light_high is set to zero. When the temp_sen is set to 8'b11111111 then at the positive edge of the clock the next state is cooler state(0010) where the cooler starts, then at the next positive edge of the clock the lume_sen is checked, which is set to 8'b11111111 and so the next state is changed to light_dim(0100) where the light_low is high, which indicates

to lower the light intensity. The next state is then `light_bright(0011)` where the `1_high` is set high to increase light intensity as `lume_sen` is set to `8'b0`. When the `motion_sen` is on only then light, cooler and heater is working or else they are off.

The input (`in`) acts as input from the user, when the `in` is `8'd8` then the alarms are deactivated, when the `in` is `8'd3` then the alarms are activated again and if wrong password then the alarm is on. The flag is set high if the alarm is deactivated or else flag is low, when flag low all the alarms are active. When the flag low(0), if any sensors (door, garage, window, fire) is high(1) then the corresponding alarm is set high(1). If the flag is high(1) which appears when `in=8'd8` then all the alarms are off.

The RTL schematics after synthesizing using Xilinx ISE 9.2i is shown below [7], [8] -

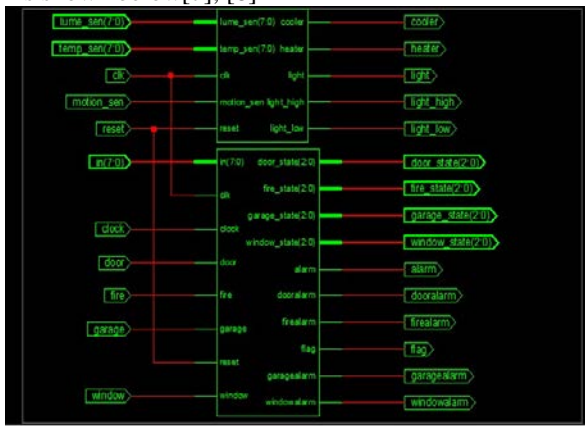


Fig.4 RTL diagram of complete Home Automation System

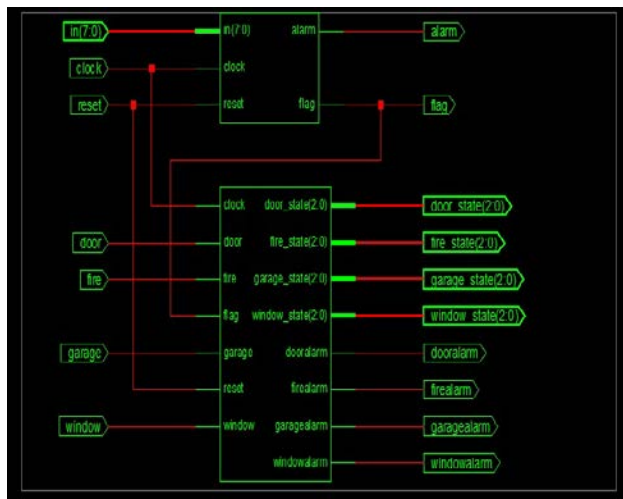


Fig.5 RTL schematic of security system

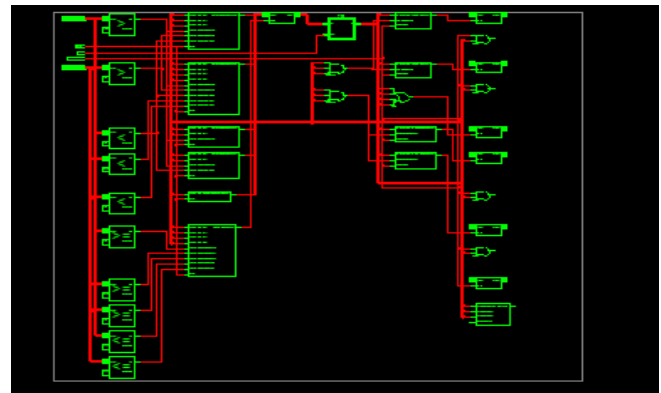


Fig. 7 RTL schematic of comfort system

V.CONCLUSION

Verilog allows use of RTL description that provides designer advantages while debugging, as the RTL description can be readily edited by the designer and implemented again with small cost of time [9].

The wave form is obtained as desired. Therefore we can conclude that the project was successfully done and is ready for implementation.

ACKNOWLEDGMENT

The work offered in this paper was carried out at the Department of EECS at North South University, Dhaka, Bangladesh. The thesis is regarded as a part of obtaining B.S. Degree in Electrical and Electronic Engineering at the University.

First of all, we acknowledge the praise to the Almighty Allah, Who gave strength and permitted us to accomplish the work. We would like to thank our respective faculty advisor Mr. IqbalurRahmanRokon for his essential guidance, precious advice. We would also like to express our gratitude to Syed Iqbal Javed for his endless support during the growth of this thesis work. Finally, we would like to show our appreciation to our family and beloved parents for their inestimable support during these years of our study.

REFERENCES

- [1] K. Madhuri, B. L. Sai and B. S. Sirisha, "A home automation system using hardware descriptive tool," *Ijert*, vol.2, July 2013.
- [2] C. Links. (2013). *Sensors Insights*.
- [3] Xilinx, "Using Block RAM in Spartan-3 FPGAs", XAPP463 (v1.1.2), July 23, 2003
- [4] Finite State Machine, Wikipedia the free encyclopedia.
- [5] Palnitkar, S. (2008). Verilog HDL (2nded.). Pearson Education.
- [6] Verilog, Wikipedia, the free encyclopedia. Retrieved March 23, 2012.
- [7] Xilinx, "ISE In-Depth Tutorial", UG695 (v1.2) June 24, 2012
- [8] Xilinx, "Spartan-3 Generation Configuration User Guide", UG332 (v1.6), October 26, 2009
- [9] Thomas L. Floyd, "Digital Fundamentals: Eighth edition", Pearson Education Limited, 2003