**A Mini Project Report On**

**HOME AUTOMATION USING VERILOG**

In partial fulfilment of the requirement for the Mini Project

by

**A. RAMYA SREE – B191978**

**MD. ZAHEER – B192091**

Under the guidance of

**MRS. TEJASVEY MADAM,**

**ASSISTANT PROFESSOR**

**Dept. of Electronics and Communication Engineering**



**DEPARTMENT OF**

**ELECTRONICS AND COMMUNICATION ENGINEERING**

**BASAR, TELANGANA, 504107**

**ABSTRACT**

In this modern world, everything is working under the technology. With the advancement of Technology, the devices are implementing smart. One of the smartest things is home automation. The home automation mainly deals with two parameters i.e., Safety and security. These include lighting and temperature control for Safety. The security system detects the fire and invaders through the door. The digital design is the platform to design this smart home. With the help of FSM concepts Verilog HDL code has been synthesized in Xilinx Vivado software, RTL schematic diagram is used to represent the pin diagram. The simulated waveforms are obtained and verified. This system is very flexible so that the new devices can be added and the core appliances can be controlled.

This project presents a comprehensive home automation system designed using Verilog, a hardware description language. The system integrates various sensors, such as temperature, luminosity, door, window, garage, and fire sensors, to enhance the security and comfort of a household. The system is divided into key modules: the Password Verification Module ensures only authorized access; the Security Module monitors and responds to potential security threats; and the Comfort Module adjusts environmental conditions for optimal living. The system’s design is represented through block diagrams, state diagrams, RTL schematics, and Verilog code. This project aims to offer an affordable and efficient solution for in-house automation, providing unique detection mechanisms for different scenarios. The project also includes a comparative analysis of various home automation approaches, highlighting the advantages and challenges of IoT, Bluetooth, microcontroller, gesture-based, and remote-control systems. The use of Verilog allows for precise control and easy modification of the system, ensuring reliability and flexibility in operation.

**TABLE OF CONTENT**

1. INTRODUCTION
2. HOME AUTOMATION
3. FLOW CHART
4. BLOCK DIAGRAM
5. STATE DIAGRAM
6. VERILOG MODULES
7. VERILOG CODE
8. RTL SCHEMATIC
9. STIMULATION
10. COMPARATIVE ANALYSIS
11. CONCLUSION

**INTRODUCTION**

Smart home technology refers to the integration of various devices and systems within a household to automate and enhance living experiences. This includes features like energy management, home security, entertainment systems, and remote access through connected devices and networks. Smart homes leverage advancements in Internet of Things (IoT), artificial intelligence, and cloud computing to provide homeowners with increased convenience, efficiency, and control over their living environment.

In the semiconductor and electronic design industry, Verilog is a hardware description language (HDL) used to model electronic systems. Verilog HDL, not to be confused with VHDL (a competing language), is most commonly used 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 analogy and mixed-signal circuits. Hardware description languages such as Verilog differ from software programming languages because they include ways of describing the propagation of time and signal dependencies (sensitivity). There are two assignment operators, a blocking assignment (=), and a non-blocking (<=) assignment. The non-blocking assignment allows designers to describe a state-machine update without needing to declare and use temporary storage variables. Since these concepts are part of Verilog's language semantics, designers could quickly write descriptions of large circuits in a relatively compact and concise form. At the time of Verilog's introduction (1984), Verilog represented a tremendous productivity improvement for circuit designers who were already using graphical schematic capture software and specially written software programs to document and simulate electronic circuits. A Verilog design consists of a hierarchy of modules. Modules encapsulate design hierarchy, and communicate with other modules through a set of declared input, output, and bidirectional ports. Internally, a module can contain any combination of the following: net/variable declarations (wire, reg, integer, etc.), concurrent and sequential statement blocks, and instances of other modules (sub-hierarchies). Sequential statements are placed inside a begin/end block and executed in sequential order within the block. However, the blocks themselves are executed concurrently, making Verilog a dataflow language.

Verilog is a powerful hardware description language (HDL) used to design and simulate digital systems. It allows engineers to model the behaviour of complex electronic systems at various levels of abstraction, from gate-level to system-level. Verilog’s concise syntax and powerful simulation capabilities make it a popular choice for designing and verifying digital circuits, ranging from simple logic gates to sophisticated microprocessors.

**HOME AUTOMATION**

Home automation, also known as domotics, refers to the use of technology to control and automate various household systems and devices. This can include everything from lighting, heating, and air conditioning to security systems, home entertainment systems, and kitchen appliances. Home automation systems allow homeowners to monitor and control these devices remotely through smartphones, tablets, or computers, often via a centralized hub or platform. The ultimate goal of home automation is to enhance the convenience, efficiency, and security of home living.

* **Increased Security**

Home automation systems enhance security by providing real-time monitoring of sensors, such as door and window contacts, motion detectors, and smoke detectors. In case of unauthorized entry or potential hazards, the system can trigger alarms, send notifications, and activate security measures, providing homeowners with peace of mind.

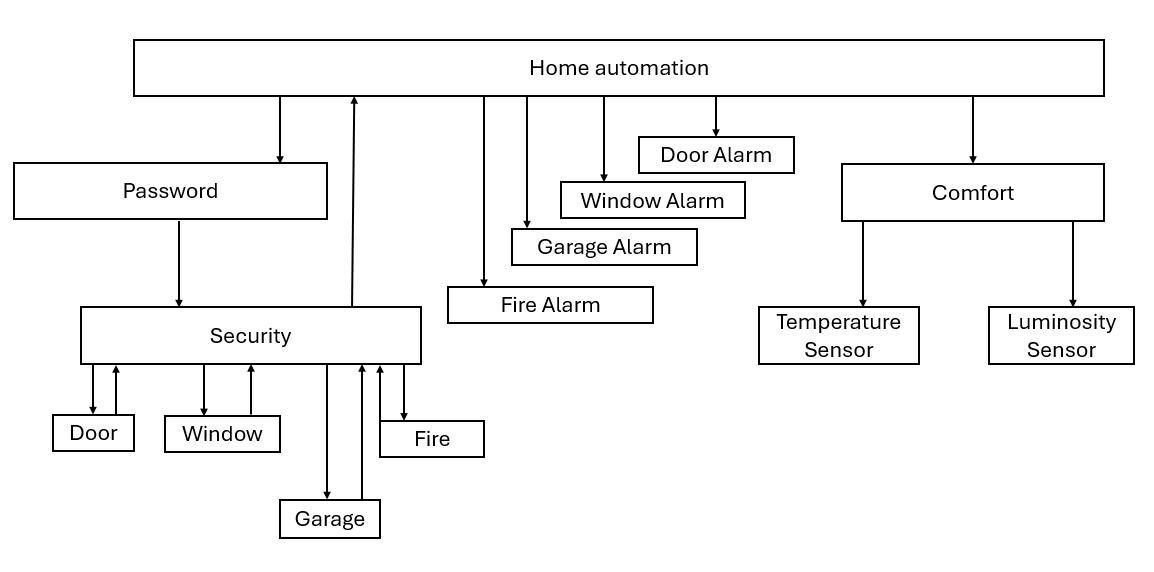
* **Energy Efficiency**

By automating lighting and temperature control based on occupancy and ambient conditions, home automation systems can optimize energy consumption and reduce utility bills. Smart thermostats, for instance, can adjust the heating and cooling systems based on schedules and preferences, minimizing energy waste.

* **Enhanced Comfort**

Home automation systems enhance comfort by providing personalized control over various aspects of the home environment. Users can adjust lighting levels, set desired temperatures, control appliances, and even remotely manage their homes from anywhere with an internet connection.

**FLOW CHART**

**BLOCK DIAGRAM**

HOME

AUTOMATION

Clk lock

Reset flag

[11:0]data\_in

Room\_door door\_alarm

Garage\_door garage\_alarm

Room\_window window\_alarm

Smoke fire\_alarm

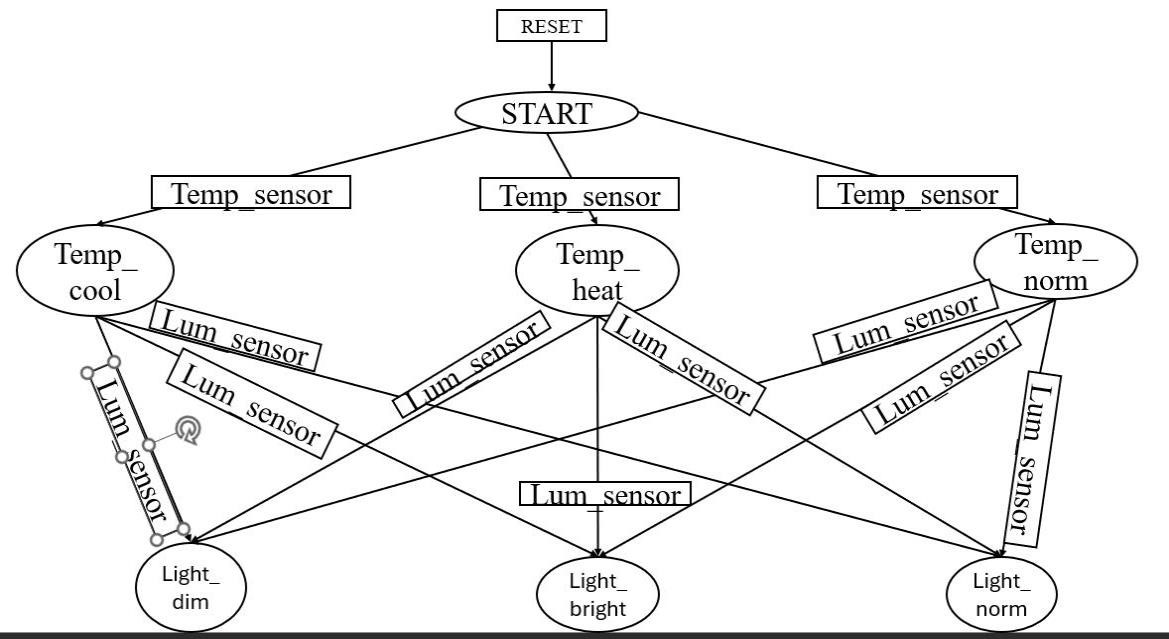
Motion\_sensor

Tank\_full water\_alarm

[7:0]temp\_sensor heater, ac

[7:0]lumen\_sensor light

**STATE DIAGRAM**



**VERILOG MODULES**

* **Password verification Module**

This module implements a password verification mechanism to ensure only authorized users can access and control the home automation system. It compares the input password with a stored reference value, granting access only if they match. This security feature prevents unauthorized individuals from manipulating the system’s functionalities.

* **Security Module**

The security module continuously monitors data from sensors related to security, such as door and window contacts, motion detectors, and fire alarms. It analyses this data in real-time, triggering appropriate actions based on predefined rules. This includes activating alarms, sending notifications, and controlling security measures like door locks and cameras.

* **Comfort Module**

The comfort module focuses on managing the home’s environment for optimal user comfort. It receives data from sensors related to temperature, light levels, and occupancy, and controls actuators like lights, heaters, and air conditioners based on predefined settings and user preferences. This module provides personalized control over the home environment for a comfortable and energy-efficient living experience.

**RTL SCHEMATIC**

* **Input Stage**

The input stage receives data from various sensors, such as door and window contacts, motion detectors, temperature sensors, light sensors, and smoke detectors. This data is digitized and processed before being sent to the respective modules.

* **Door Alarm**

The door alarm module monitors door contacts, triggering an alarm if an unauthorized door opening is detected. It can send notifications to the user, activate external security measures like sirens, and even control door locks to prevent entry.

* **Window Alarm**

Similar to the door alarm, the window alarm module monitors window contacts, detecting unauthorized openings. It triggers alarms, sends notifications, and can even activate window blinds or shutters to deter entry.

* **Garage Alarm**

The garage alarm module secures the garage by monitoring garage door sensors. It triggers alarms if unauthorized opening is detected and can even control garage door openers, ensuring the garage remains secure.

* **Fire Alarm**

The fire alarm module detects smoke and heat, triggering an alarm in case of a potential fire hazard. It can send notifications to the user, activate sirens, and even control sprinklers to mitigate the fire.

* **Light Control**

The light control module adjusts the lighting based on ambient light levels, occupancy, and user preferences. It can automatically dim lights in the evening, switch off lights in empty rooms, and even create personalized lighting scenes for different activities.

* **Temperature Control**

The temperature control module manages the heating and air conditioning system based on indoor temperature, occupancy, and user preferences. It can automatically adjust the temperature to maintain a comfortable indoor environment, reducing energy consumption and improving comfort.

* **Output Stage**

The output stage sends control signals to actuators, such as lights, heaters, air conditioners, alarms, and other appliances, based on the processed data and module decisions. This stage ensures the system’s commands are executed effectively, automating various functions in the home.

**VERILOG CODE**

module homeauto(

input clk,

input reset,

input [11:0] data\_in,

input room\_door,

input garage\_door,

input room\_window,

input smoke,

input motion\_sens,

input tank\_full,

input [7:0] temp\_sens,

input [7:0] lumen\_sens,

output door\_alarm,

output garage\_alarm,

output window\_alarm,

output fire\_alarm,

output water\_alarm,

output heater,

output ac,

output light,

output lock,

output flag

);

reg [11:0] password = 12'b101010101010; // Example 12-bit password

reg lock\_reg;

assign lock = lock\_reg;

// Password authentication

always @(posedge clk or posedge reset) begin

if (reset) begin

lock\_reg <= 1; // Lock the system on reset

end else if (data\_in == password) begin

lock\_reg <= 0; // Unlock if the password is correct

end else begin

lock\_reg <= 1; // Keep locked if the password is incorrect

end

end

door\_module door\_inst(

.clk(clk),

.reset(reset),

.room\_door(room\_door),

.lock(lock\_reg),

.door\_alarm(door\_alarm)

);

garage\_module garage\_inst(

.clk(clk),

.reset(reset),

.garage\_door(garage\_door),

.lock(lock\_reg),

.garage\_alarm(garage\_alarm)

);

window\_module window\_inst(

.clk(clk),

.reset(reset),

.room\_window(room\_window),

.lock(lock\_reg),

.window\_alarm(window\_alarm)

);

fire\_module fire\_inst(

.clk(clk),

.reset(reset),

.smoke(smoke),

.lock(lock\_reg),

.fire\_alarm(fire\_alarm)

);

flag\_module flag\_inst(

.clk(clk),

.reset(reset),

.motion\_sens(motion\_sens),

.lock(lock\_reg),

.flag(flag)

);

water\_pump\_module water\_pump\_inst(

.clk(clk),

.reset(reset),

.tank\_full(tank\_full),

.water\_alarm(water\_alarm)

);

// Comfort Module

reg heater\_reg, ac\_reg, light\_reg;

assign heater = heater\_reg;

assign ac = ac\_reg;

assign light = light\_reg;

always @(posedge clk or posedge reset) begin

if (reset) begin

heater\_reg <= 0;

ac\_reg <= 0;

light\_reg <= 0;

end else if (motion\_sens) begin

if (temp\_sens < 15) begin

heater\_reg <= 1;

ac\_reg <= 0;

end else if (temp\_sens > 28) begin

ac\_reg <= 1;

heater\_reg <= 0;

end else begin

heater\_reg <= 0;

ac\_reg <= 0;

end

if (lumen\_sens < 8'd200) begin

light\_reg <= 1; // Bright light

end else if (lumen\_sens > 8'd250) begin

light\_reg <= 0; // Dim light

end else begin

light\_reg <= 0; // Normal light

end

end else begin

heater\_reg <= 0;

ac\_reg <= 0;

light\_reg <= 0;

end

end

endmodule

module door\_module (

input clk,

input reset,

input room\_door,

input lock,

output reg door\_alarm

);

always @(posedge clk or posedge reset) begin

if (reset) begin

door\_alarm <= 0;

end else if (!lock) begin

door\_alarm <= room\_door;

end

end

endmodule

module garage\_module (

input clk,

input reset,

input garage\_door,

input lock,

output reg garage\_alarm

);

always @(posedge clk or posedge reset) begin

if (reset) begin

garage\_alarm <= 0;

end else if (!lock) begin

garage\_alarm <= garage\_door;

end

end

endmodule

module window\_module (

input clk,

input reset,

input room\_window,

input lock,

output reg window\_alarm

);

always @(posedge clk or posedge reset) begin

if (reset) begin

window\_alarm <= 0;

end else if (!lock) begin

window\_alarm <= room\_window;

end

end

endmodule

module fire\_module (

input clk,

input reset,

input smoke,

input lock,

output reg fire\_alarm

);

always @(posedge clk or posedge reset) begin

if (reset) begin

fire\_alarm <= 0;

end else if (!lock) begin

fire\_alarm <= smoke;

end

end

endmodule

module flag\_module (

input clk,

input reset,

input motion\_sens,

input lock,

output reg flag

);

always @(posedge clk or posedge reset) begin

if (reset) begin

flag <= 0;

end else if (!lock && motion\_sens) begin

flag <= 1; // Owner is at home

end else begin

flag <= 0; // Owner is not at home

end

end

endmodule

module water\_pump\_module (

input clk,

input reset,

input tank\_full,

output reg water\_alarm

);

always @(posedge clk or posedge reset) begin

if (reset) begin

water\_alarm <= 0;

end else begin

water\_alarm <= tank\_full;

end

end

endmodule

**TEST BENCH**

module homeautotb();

reg clk;

reg reset;

reg [11:0] data\_in;

reg room\_door;

reg garage\_door;

reg room\_window;

reg smoke;

reg motion\_sens;

reg tank\_full;

reg [7:0] temp\_sens;

reg [7:0] lumen\_sens;

wire door\_alarm;

wire garage\_alarm;

wire window\_alarm;

wire fire\_alarm;

wire water\_alarm;

wire heater;

wire ac;

wire light;

wire lock;

wire flag;

homeauto uut (

.clk(clk),

.reset(reset),

.data\_in(data\_in),

.room\_door(room\_door),

.garage\_door(garage\_door),

.room\_window(room\_window),

.smoke(smoke),

.motion\_sens(motion\_sens),

.tank\_full(tank\_full),

.temp\_sens(temp\_sens),

.lumen\_sens(lumen\_sens),

.door\_alarm(door\_alarm),

.garage\_alarm(garage\_alarm),

.window\_alarm(window\_alarm),

.fire\_alarm(fire\_alarm),

.water\_alarm(water\_alarm),

.heater(heater),

.ac(ac),

.light(light),

.lock(lock),

.flag(flag)

);

initial begin

// Initialize inputs

clk = 0;

reset = 1;

data\_in = 12'b0;

room\_door = 0;

garage\_door = 0;

room\_window = 0;

smoke = 0;

motion\_sens = 0;

tank\_full = 0;

temp\_sens = 8'd0;

lumen\_sens = 8'd0;

// Reset the system

#10 reset = 0;

// Test sequence

#10 data\_in = 12'b101010101010; // Correct password

#10 motion\_sens = 1; // Motion detected

temp\_sens = 8'd10; // Temperature below 15°C

lumen\_sens = 8'd180; // Luminosity less than 200 Lux

room\_door = 1; // Door open

garage\_door = 0;

room\_window = 0;

smoke = 0;

tank\_full = 0;

#20 temp\_sens = 8'd30; // Temperature above 28°C

lumen\_sens = 8'd260; // Luminosity more than 250 Lux

#20 temp\_sens = 8'd20; // Temperature normal

lumen\_sens = 8'd220; // Luminosity normal

#20 room\_window = 1; // Window open

smoke = 1; // Smoke detected

tank\_full = 1; // Tank is full

#20 data\_in = 12'b000000000000; // Wrong password

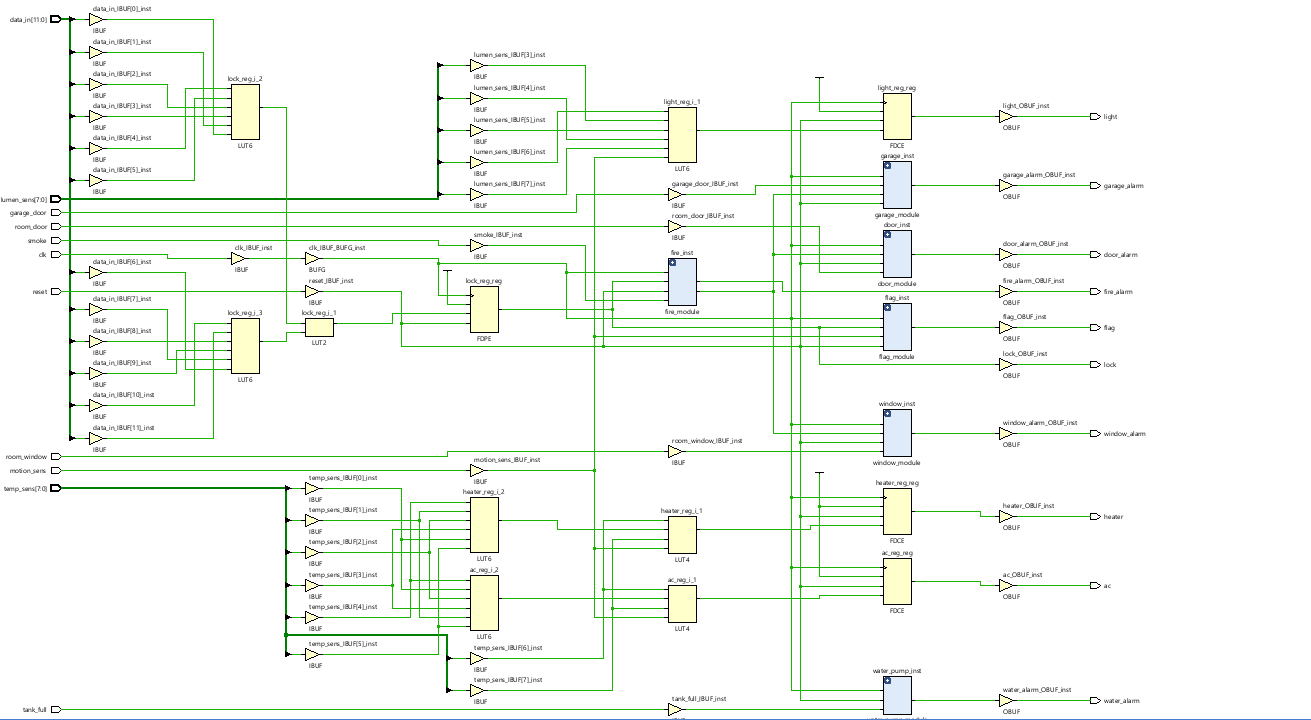
#20 $finish;

end

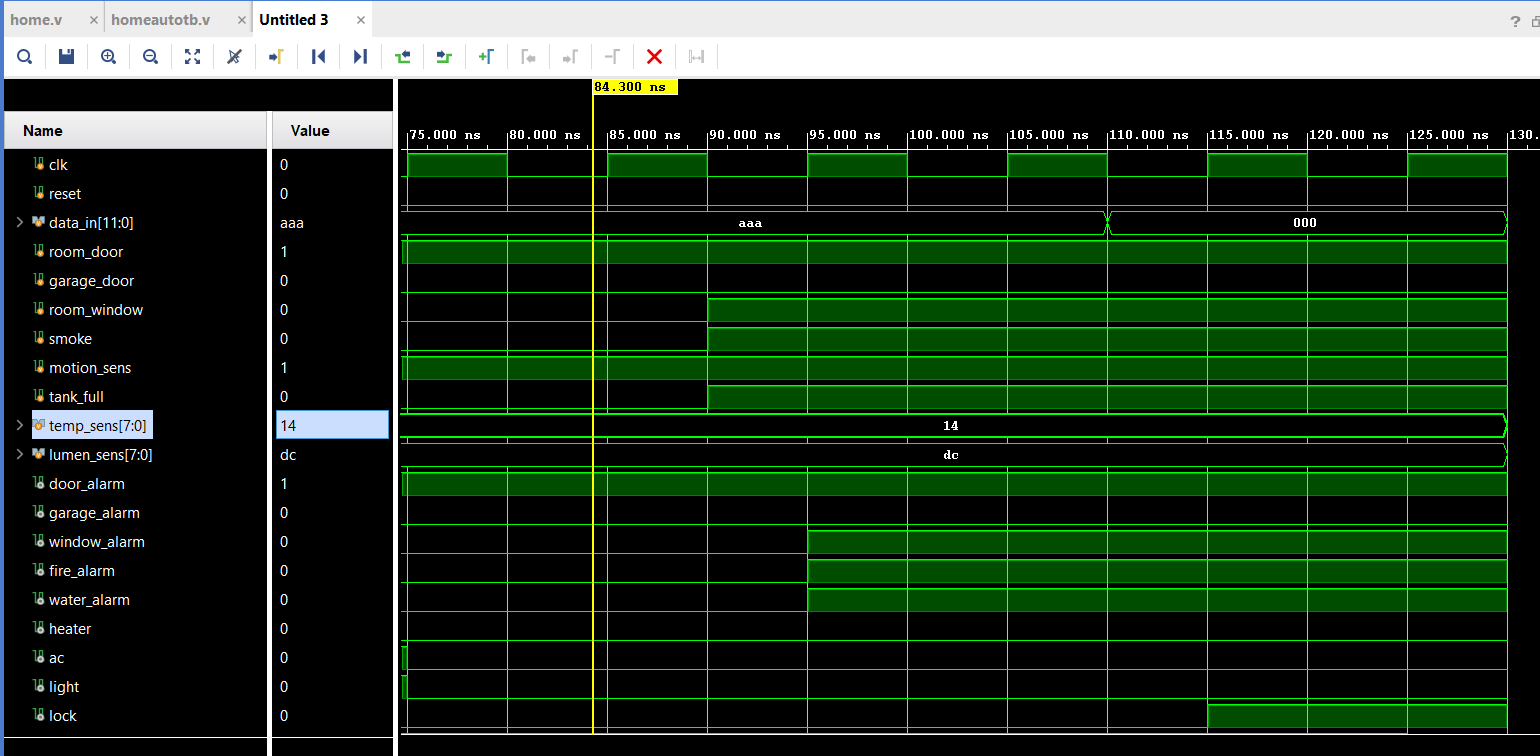
always #5 clk = ~clk; // Clock generation

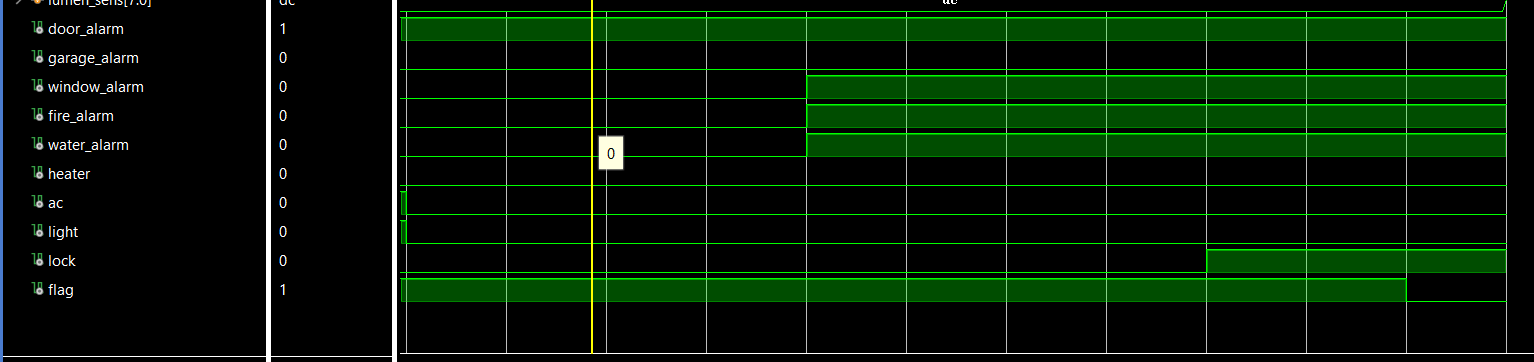
endmodule

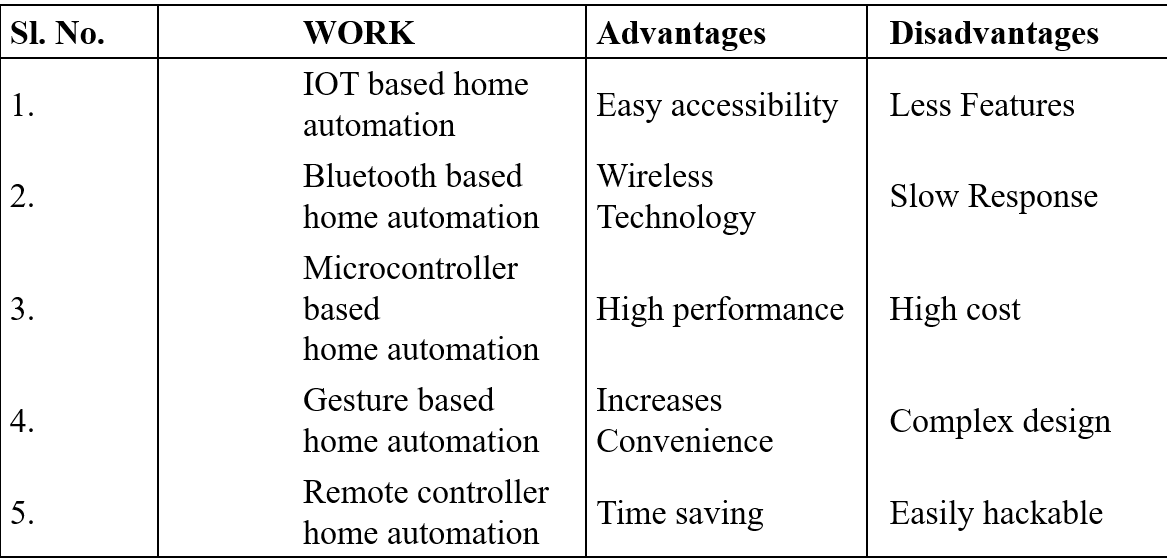
**RTL SCHEMATIC**



**STIMULATION**

****

****

**COMPARATIVE ANALYSIS**

**CONCLUSION**

* Verilog provides RTL description which is advantageous for designer for debugging. RTL schematic can be modified by the designer and implemented by editing Verilog code. The wave form is obtained as desired.
* Concluding the main aim of this project is to design a controller at an affordable price which contains an automated in-house security system. Every sensor in this controller has unique detection system for different incidents such as motion detection, fire, temperature and light etc.