**Table of Contents**

[**INTRODUCTION 2**](#_Toc84279131)

[**METHOD 4**](#_Toc84279132)

[**RESULTS AND DISCUSSIONS 10**](#_Toc84279133)

[**FUTURE SCOPE 16**](#_Toc84279134)

[**CONCLUSION 17**](#_Toc84279135)

[**REFERENCES 17**](#_Toc84279136)

[**APPENDIX 18**](#_Toc84279137)

# 

# **INTRODUCTION**

Fires impact people, property and the environment in all countries around the world. In some cases, the resulting losses are extraordinary, causing hundreds of deaths, widespread damage to property and contents and significant impacts on the environment. More often, fires may cause a single casualty or affect a single home, though the effects are still highly significant to those affected and collectively are substantial.

Thirty-five Indians die in a fire incident daily, says the National Crime Records Bureau (NCRB) based on the 12,748 lives in 2018. “Our fire safety laws are archaic and our fire departments are not well equipped to handle the modern high-rise buildings. We have suboptimal fire infrastructure, fewer fire stations, poorly trained staff and low budgetary grants,” say experts.

Fire hazards can be greatly reduced if we can realize early warning of the fire. In order to prevent and reduce fire's crisis to people, humans require a fire alarm system. So, fire can be detected in the initial stage, which is helpful to realize the early warning and control of fire. A fire alarm system warns people when smoke, fire, carbon monoxide, or other fire-related emergencies are detected.

Fire alarms have many advantages. In the event of a fire, they provide detection and notification. They can also automatically dispatch the fire department to the location of the fire. The greatest benefits of an efficient fire alarm system also include:

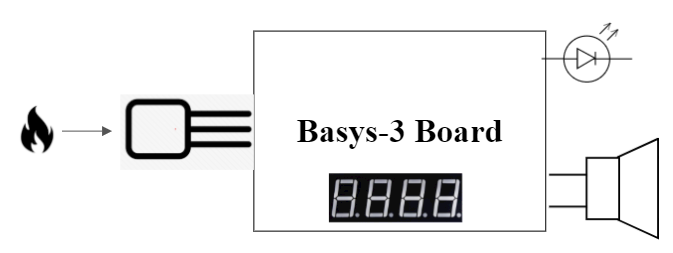
* Avoid Smoke Inhalation: Smoke inhalation is the leading cause of death related to fires; more so than heat or flames. A mixture of particles, chemicals, and gases, smoke causes everything from skin and mucous membrane irritation to swelling, respiratory distress, and airway collapse. Without immediate medical help, smoke inhalation can lead to suffocation and death. A fire alarm can protect you against exposure to harmful substances such as carbon monoxide, ammonia, hydrogen cyanide, and others often found in smoke.
* Early Fire Detection: An audible or visual signal enables you to seek safety soon after the fire starts. Once fire and/or smoke conditions trigger the system, one can use an available fire extinguisher to protect one’s home and belongings or rush their family to safety. Early detection can enable one to avoid serious damage or destruction, so it is of extreme importance. In addition to providing security in the kitchen, bathroom, bedrooms, and family rooms, a fire alarm can quickly alert firefighters so they can help minimize the damage.
* Discounts on Insurance: When you install fire alarms, you can save money on home. They help cut costs for insurance companies by reducing the risk of a fire destroying your home, appliances, and possessions. Instant notification of a potential fire and fast action can minimize the amount of an insurance claim.
* Decreased Risk of Fire Damage: Property damage can require large investments and a great deal of time to repair. One can also avoid damage to nearby properties when they install a residential fire alarm. This level of prevention is possible because a fire alarm system can enable firefighters to respond and put out the fire before it gets out of control. The less damage there is, the quicker one can get each affected room back in order.

The primary purpose of the fire alarm system is to provide an early warning of fire so that people can be evacuated & immediate action can be taken to stop or eliminate the fire as soon as possible. These alarms may be activated automatically from smoke detectors and temperature sensors. There can also be speaker strobes that sound an alarm and alert people for evacuation. The temperature sensor is utilized as a part of the Fire Alarm System to recognize a fire. The temperature sensor records the temperature of the room. If the room temperature reaches a certain threshold - fire threat is detected, the designed fire alarm system gets automatically activated and alerts the user by three mechanisms: a countdown timer, an alarm and the glow of a bright red LED.

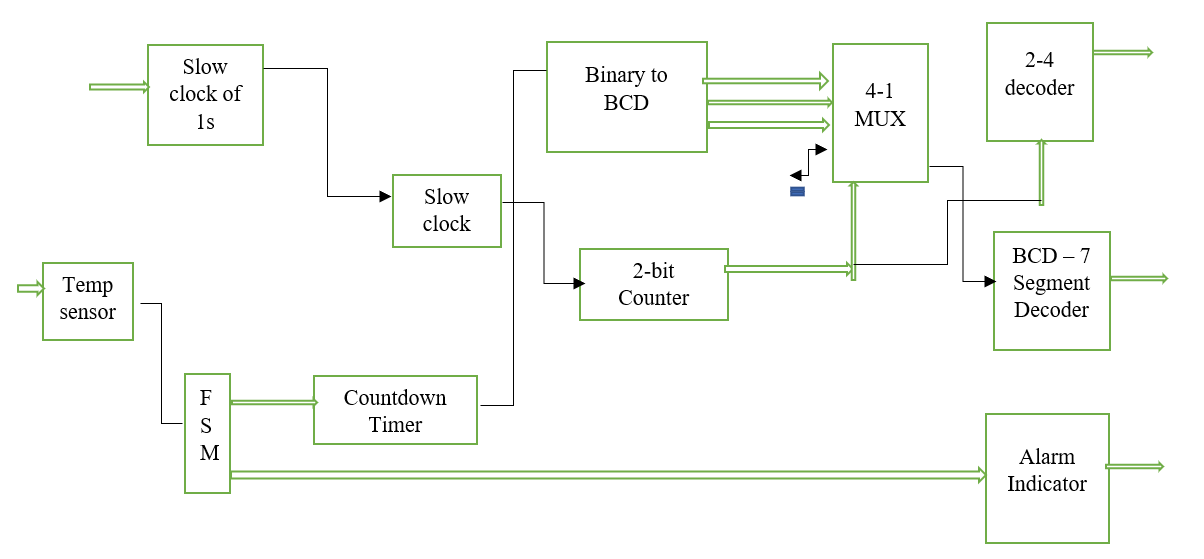
In this project, we develop an efficient design of a fire alarm system using Verilog and a possible solution where the user controls the device by employing a central Field Programmable Gate Array (FPGA) controller to which the devices and sensors are interfaced. The FPGA can meet the needs of high-speed real-time with its hardware features. SoPC technology can make the design flexible, software and hardware in the system programmable and updated. This project is a reflection of digital system design to achieve our goal. We simulate the design in Verilog using Xilinx Vivado.

# 

# **METHOD**

**BASIC BLOCK DIAGRAM**

**BASIC SCHEMATIC**

****

**LIST OF SUB MODULES REQUIRED TO IMPLEMENT FIRE ALARM SYSTEM IN VERILOG**

* Temperature Sensor to sense outside temperature and fire threats
* Finite State Machine design
* Countdown Timer to alert people for evacuation
* Slow clock to decrement 1 second
* Slow clock for blinking LED
* Two-bit counter for generating the select signal
* Binary to BCD Converter for converting the Count
* 4-1 MUX for selecting the digits position in BCD number
* BCD to 7 segment display to display output count
* Buzzer to alert people
* 2-4 Decoder to indicate the position of output in 7 segment display

**COMPONENTS USED**

1. Xilinx Artix-7 based FPGA Development Board (Basys 3)
2. Temperature Sensor
3. Alarm System - Buzzer
4. Breadboard
5. Connecting Cables

**COMPONENTS SPECIFICATIONS**

**XILINX ARTIX-7 BASED FPGA DEVELOPMENT BOARD (BASYS 3)**

The Basys 3 board is a complete, ready-to-use digital circuit development platform based on the latest Artix-7 Field Programmable Gate Array (FPGA) from Xilinx®. With its high-capacity FPGA (Xilinx part number XC7A35T1CPG236C), low overall cost, and collection of USB, VGA, and other ports, the Basys 3 can host designs ranging from introductory combinational circuits to complex sequential circuits like embedded processors and controllers. It includes enough switches, LEDs, and other I/O devices to allow a large number of designs to be completed without the need for any additional hardware, and enough uncommitted FPGA I/O pins to allow designs to be expanded using Digilent Pmods or other custom boards and circuits.

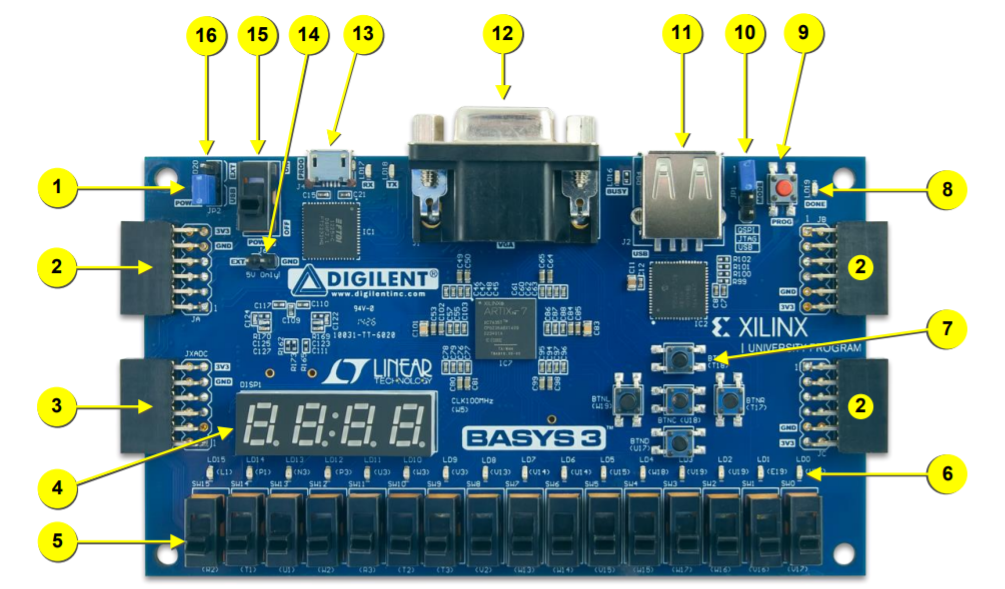
The Artix-7 FPGA is optimized for high performance logic, and offers more capacity, higher performance, and more resources than earlier designs. Artix-7 35T features include:

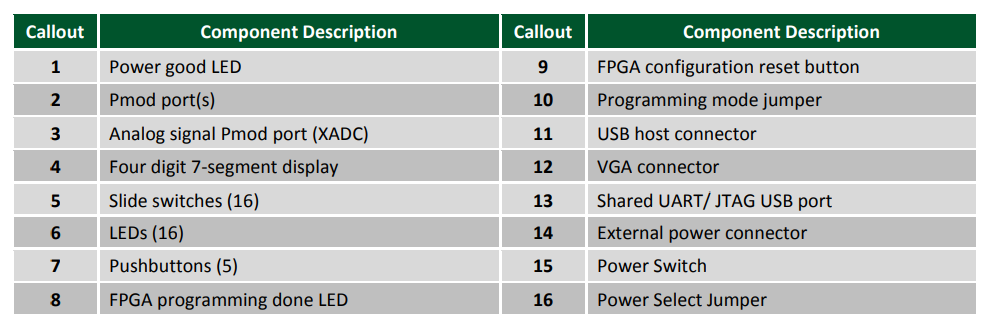
* 33,280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops)
* 1,800 Kbits of fast block RAM
* Five clock management tiles, each with a phase-locked loop (PLL)
* 90 DSP slices
* Internal clock speeds exceeding 450MHz
* On-chip analog-to-digital converter (XADC)

The Basys 3 also offers an improved collection of ports and peripherals, including:

* 16 user switches
* 16 user LEDs
* 5 user pushbuttons
* 4-digit 7-segment display
* Three Pmod ports
* Pmod for XADC signals
* 12-bit VGA output USB-UART Bridge
* Serial Flash
* Digilent USB-JTAG port for FPGA programming and communication
* USB HID Host for mice, keyboards and memory sticks

The Basys 3 works with Xilinx's new high-performance Vivado Design Suite. Vivado includes many new tools and design flows that facilitate and enhance the latest design methods. It runs faster, allows better use of FPGA resources, and allows designers to focus their time evaluating design alternatives. The System Edition includes an on-chip logic analyzer, high-level synthesis tool, other cutting-edge tools, and the free WebPACK version allows Basys 3 designs to be created at no additional cost.

****

****

**TEMPERATURE SENSOR**

A temperature sensor is an electronic device that measures the temperature of its environment and converts the input data into electronic data to record monitor or signal temperature changes. The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling.

The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range. Lower cost is assured by trimming and calibration at the wafer level. The low-output impedance, linear output, and precise inherent calibration of the LM35 device make interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 μA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy).

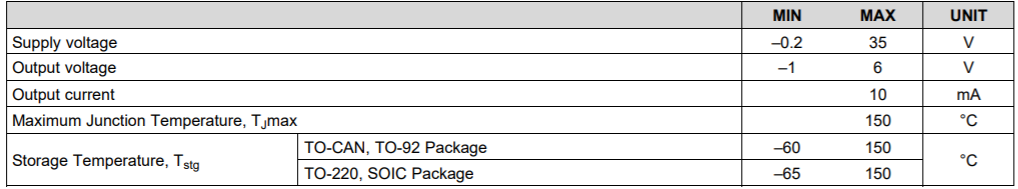
The accuracy specifications of the LM35 are given with respect to a simple linear transfer function:

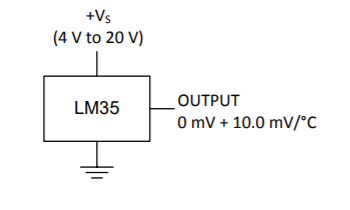
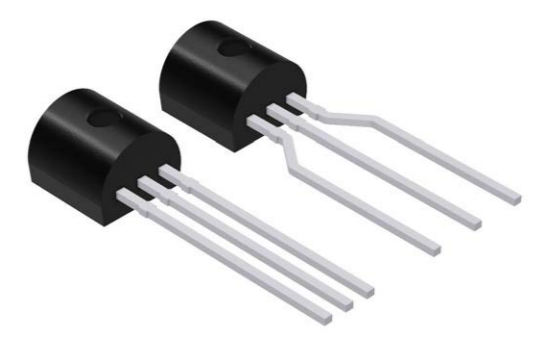
VOUT = 10 mv/°C × T

where

• VOUT is the LM35 output voltage

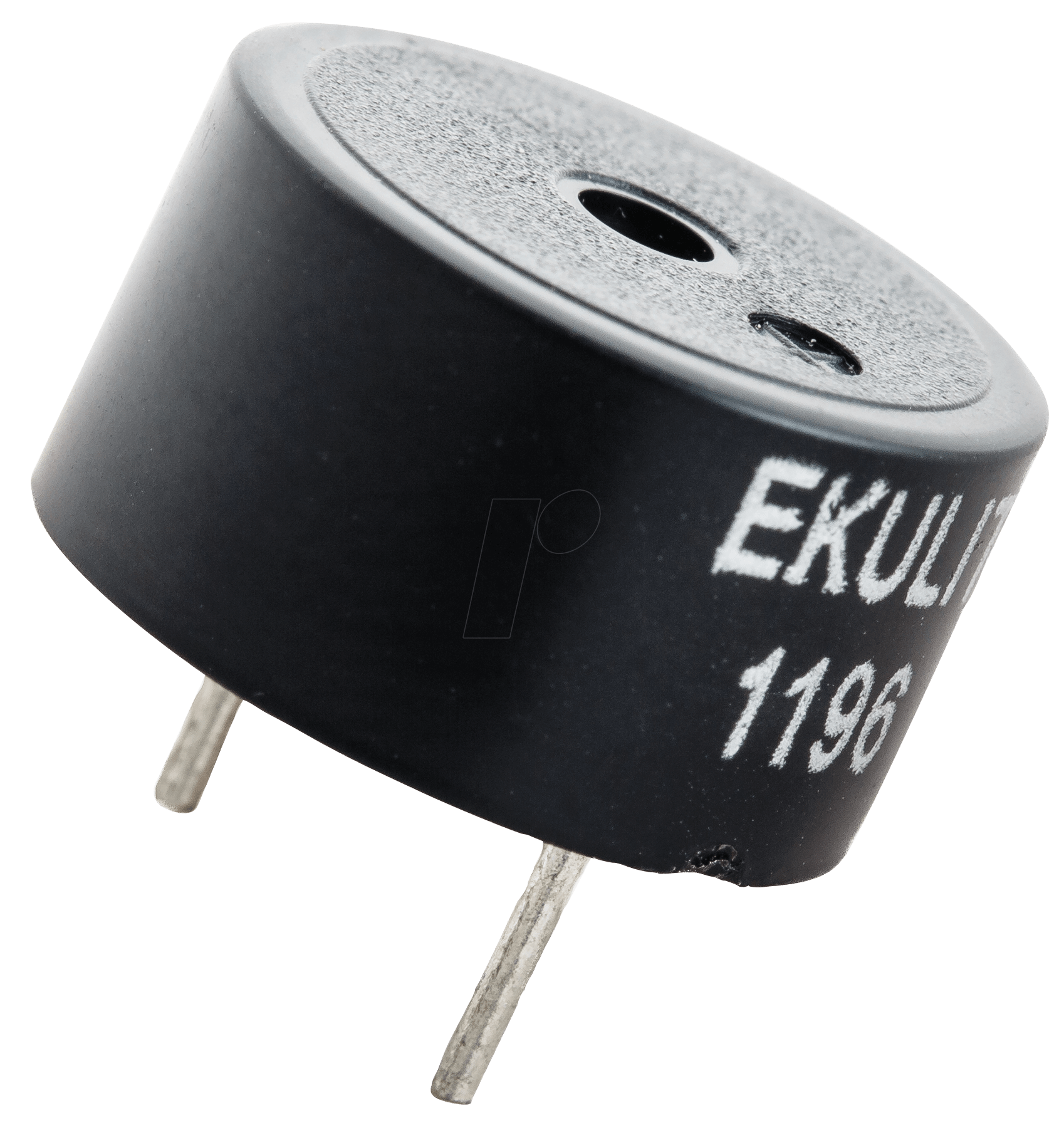
• T is the temperature in °C

**ABSOLUTE MAXIMUM RATINGS**

**PIEZO - ELECTRIC BUZZER**

In simplest terms, a piezo buzzer is a type of electronic device that’s used to produce a tone, alarm or sound. It’s lightweight with a simple construction, and it’s typically a low-cost product. Yet at the same time, depending on the piezo ceramic buzzer specifications, it’s also reliable and can be constructed in a wide range of sizes that work across varying frequencies to produce different sound outputs.

Thanks to both the reliability and flexibility of piezoelectric vibration plates to produce audible signals — ranging from monotone buzzes and alarms to multi-tones and melodies — their applications in small, high-density assemblies are wide-ranging. Their low power consumption makes them ideal for many battery-operated devices.

With such characteristics, piezo buzzers are regularly used in alarms, warning devices and automobile alerts. In addition, since they can produce a wide range of audible signals, they’re also used in pest deterrent devices.

The typical specifications of a Piezo - Electric Buzzer include:

* The frequency range is 3,300Hz
* Operating Temperature ranges from – 20° C to +60°C
* Operating voltage ranges from 3V to 24V DC
* The sound pressure level is 85dBA or 10cm
* The supply current is below 15mA

A basic piezo buzzer with these characteristics is used.

**SOFTWARE USED**

**VERILOG**

Verilog, standardized as IEEE 1364, is a [hardware description language](https://en.wikipedia.org/wiki/Hardware_description_language) (HDL) used to model [electronic systems](https://en.wikipedia.org/wiki/Electronic_system). It is most commonly used in the design and verification of [digital circuits](https://en.wikipedia.org/wiki/Digital_electronics) at the [register-transfer level](https://en.wikipedia.org/wiki/Register-transfer_level) of [abstraction](https://en.wikipedia.org/wiki/Abstraction_(computer_science)). It is also used in the verification of [analog circuits](https://en.wikipedia.org/wiki/Analogue_electronics) and [mixed-signal circuits](https://en.wikipedia.org/wiki/Mixed-signal_integrated_circuit), as well as in the design of [genetic circuits](https://en.wikipedia.org/wiki/Synthetic_biological_circuit).

Hardware description languages such as Verilog are similar to [software](https://en.wikipedia.org/wiki/Software) [programming languages](https://en.wikipedia.org/wiki/Programming_language) because they include ways of describing the propagation time and signal strengths (sensitivity). There are two types of [assignment operators](https://en.wikipedia.org/wiki/Assignment_operator); 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](https://en.wikipedia.org/wiki/Temporary_storage_variable). 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](https://en.wikipedia.org/wiki/Schematic_capture) software and specially written software programs to document and [simulate electronic circuits](https://en.wikipedia.org/wiki/Electronic_circuit_simulation).

A Verilog design consists of a [hierarchy of modules](https://en.wikipedia.org/wiki/Hierarchy_of_modules). Modules encapsulate *design hierarchy*, and communicate with other modules through a set of declared input, output, and [bidirectional ports](https://en.wikipedia.org/w/index.php?title=Bidirectional_port&action=edit&redlink=1). Internally, a module can contain any combination of the following: net/variable declarations (wire, reg, integer, etc.), [concurrent](https://en.wikipedia.org/wiki/Concurrency_(computer_science)) and sequential [statement blocks](https://en.wikipedia.org/wiki/Statement_block), 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](https://en.wikipedia.org/wiki/Dataflow_language).

During the development cycle the description has to become more and more precise until it is actually possible to manufacture the product. The (automatic) transformation of a less detailed description into a more elaborated one is called synthesis. Existing synthesis tools are capable of mapping specific constructs of hardware description languages directly to the standard components of integrated circuits. This way, a formal model of the hardware system can be used from the early design studies to the final net list. Software support is available for the necessary refinement steps.

**VIVADO**

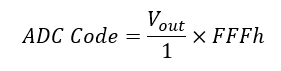
Vivado Design Suite is a software suite produced by [Xilinx](https://en.wikipedia.org/wiki/Xilinx) for synthesis and analysis of [HDL](https://en.wikipedia.org/wiki/Hardware_description_language) designs, superseding [Xilinx ISE](https://en.wikipedia.org/wiki/Xilinx_ISE) with additional features for [system on a chip](https://en.wikipedia.org/wiki/System_on_a_chip) development and [high-level synthesis](https://en.wikipedia.org/wiki/High-level_synthesis). Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE). Vivado was introduced in April 2012, and is an integrated design environment (IDE) with system-to-IC level tools built on a shared scalable data model and a common debug environment. Vivado includes electronic system level (ESL) design tools for synthesizing and verifying C-based algorithmic IP; standards-based packaging of both algorithmic and RTL IP for reuse; standards-based IP stitching and systems integration of all types of system building blocks; and the verification of blocks and systems.

# **RESULTS AND DISCUSSIONS**

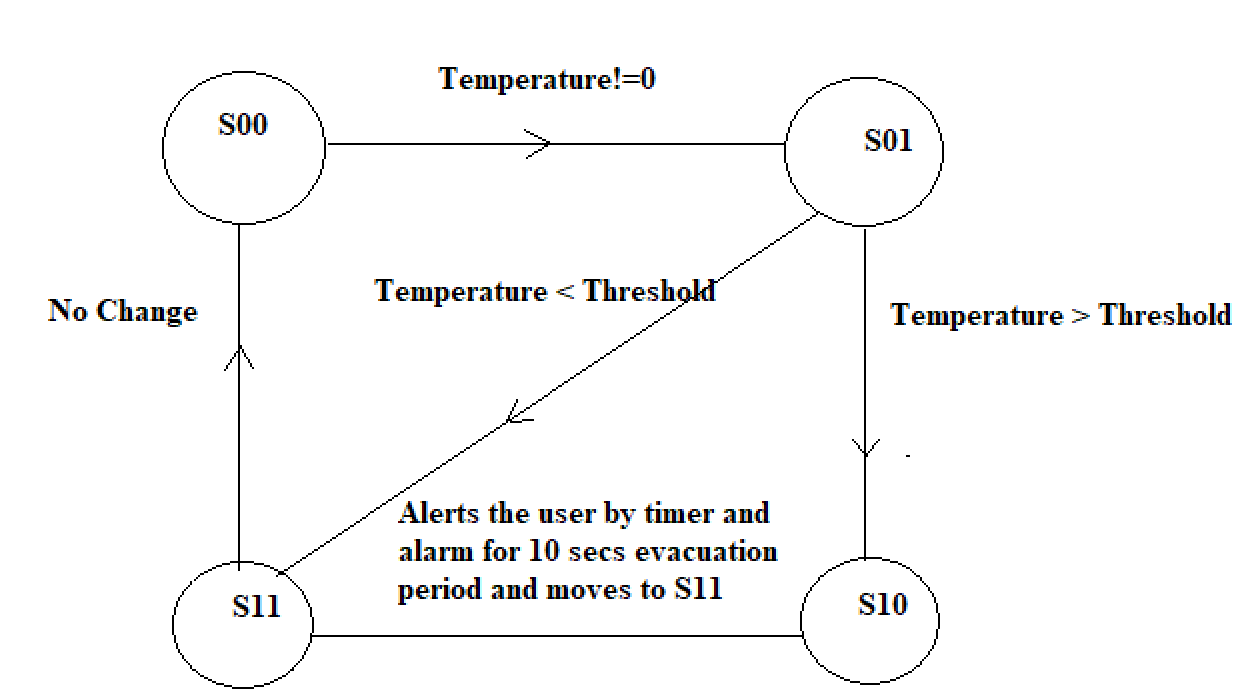
The coding part is implemented using Verilog in Xilinx Vivado and the behavioral simulation is carried out. Synthesis and software implementation is performed and verified. Then, the code is loaded onto the Basys 3 FPGA board and the fire alarm system working is realized.

We now take a closer look into each of the individual modules and their functionalities in detail.

* **Temperature Sensor:** The outside temperature is sensed via an analog temperature sensor LM35. It is interfaced with the Basys 3 Board via PMOD JXADC. The corresponding digital value is obtained and is stored in the tempV register.



* **FSM - Finite State Machine:** The Finite State Machine is designed as follows:



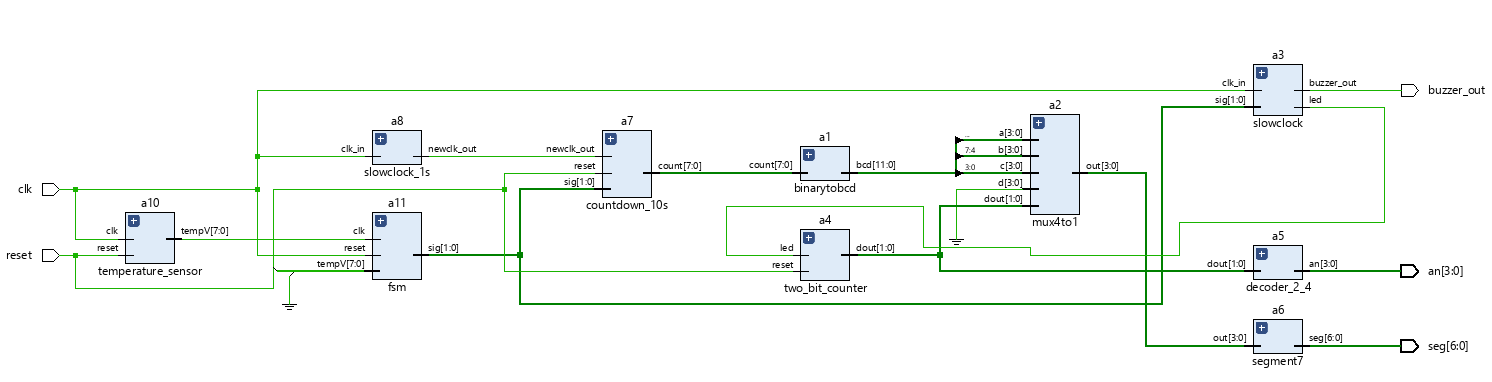
S00: This state checks the value of outside temperature. If the temperature is a non-negative value, it moves to the next state S01.

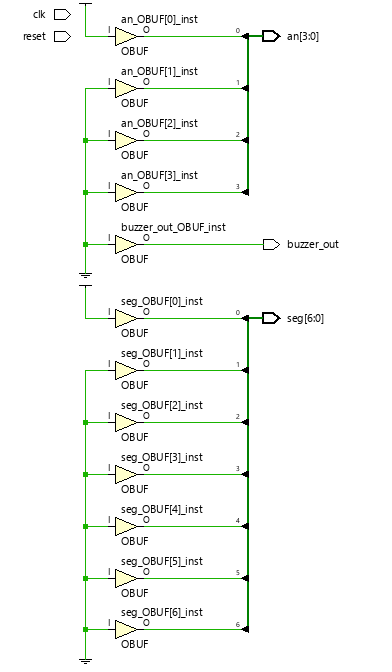
S01: This state checks whether the value of temperature is above the threshold value which is set as 50 degrees Centigrade. If it is above the threshold value, it moves to the next state S10. Otherwise, it moves to state S11.

S10: Once the temperature above the threshold has been detected, the fire alarm system triggers the buzzer and countdown timer with the LED flash. It alerts the user for a period of 10 seconds before moving to the state S11.

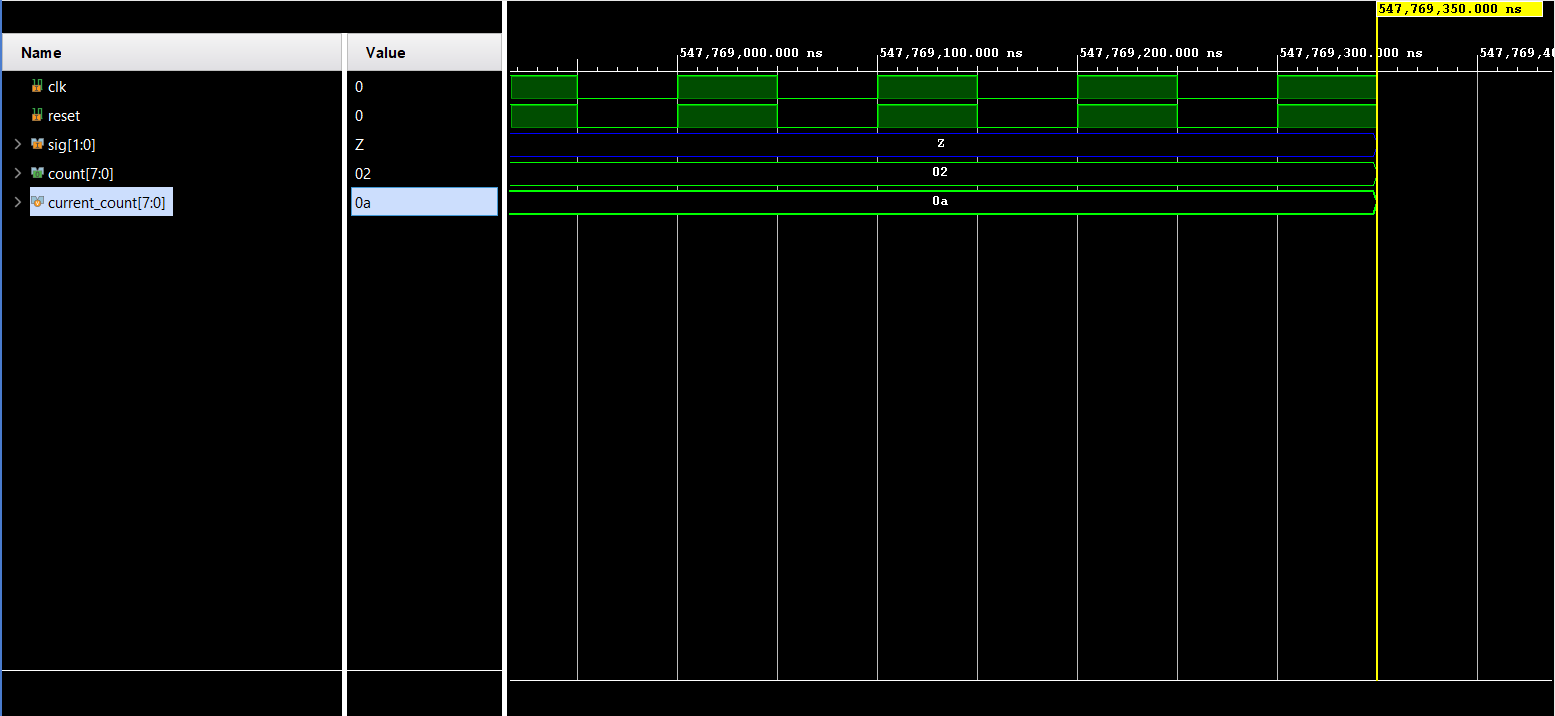
S11: This system is reset to S00 state and it becomes ready again to detect an impending danger.

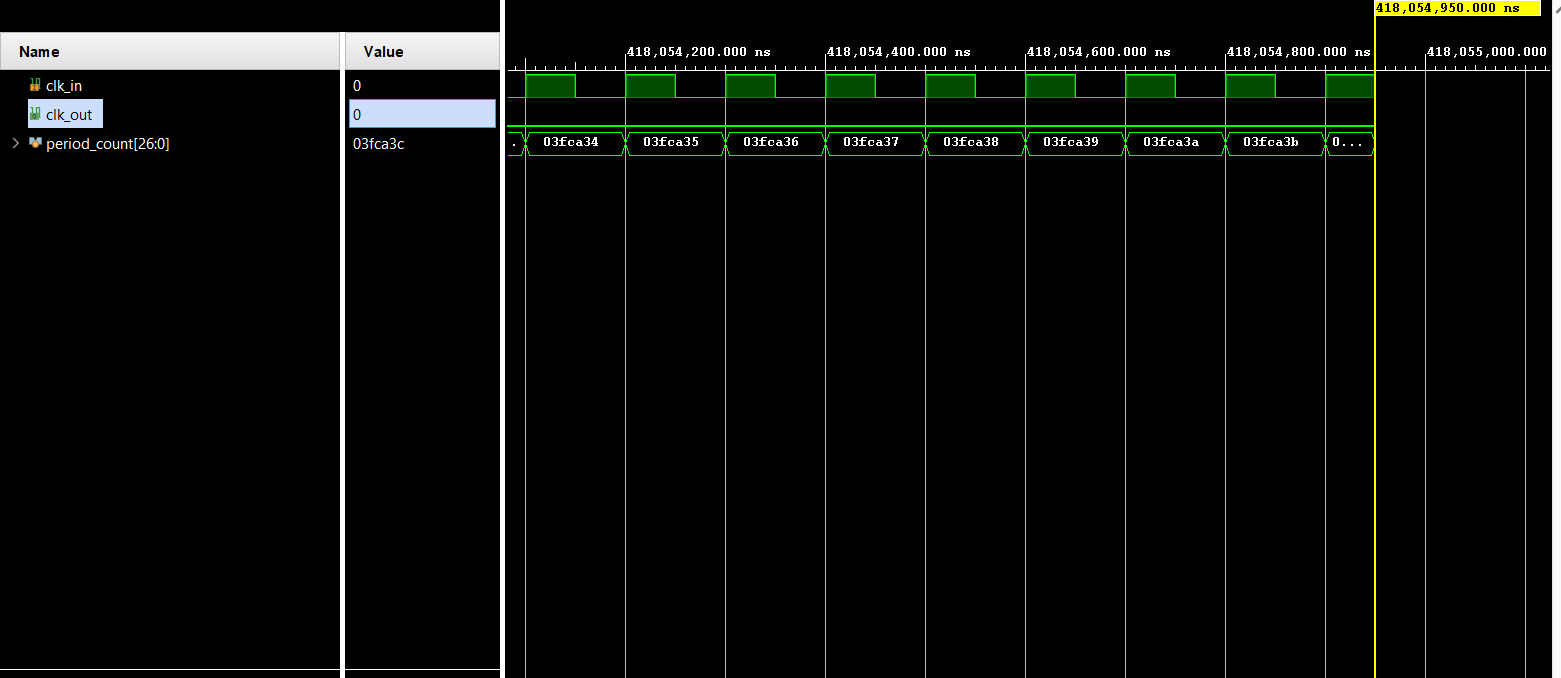
* **Slow Clock of 1 second:** Basys 3 has an inbuilt clock of frequency 100MHz. Only when the period count reaches 100\*106 clock cycles, it results in decrementing 1 second.
* **Countdown Timer:** After the FSM reaches the S10 state, the countdown timer decrements from the value of 10. This is shown using a seven-segment display which is available on the Basys 3 FPGA board.
* **Slow Clock:** This is a 100 Hz clock. We are stepping down the frequency from 100 MHz to 100 Hz in order to acquire sufficient time for blinking of the LED and buzzing of the alarm.
* **Binary to BCD:** The binary count value obtained from the Countdown Timer module is converted to its equivalent BCD value so that it can be fed into the BCD to seven segment display module.
* **Two bit Counter:** Two bit counter which counts from 0 to 3, acts as a select signal for the upcoming 4x1 multiplexer module.
* **4x1 MUX:** The multiplexer selects hundreds or tens or unit positions of the BCD number generated by the binary to BCD module using a select signal from the two-bit counter. The output from 4x1 MUX goes to BCD to seven segment display.
* **2-4 Decoder:** The decoder determines in which position the number needs to be displayed.
* **BCD to seven segment display:** The BCD number coming from 4x1 MUX is coupled onto the seven-segment display and hence the countdown can be observed.

****

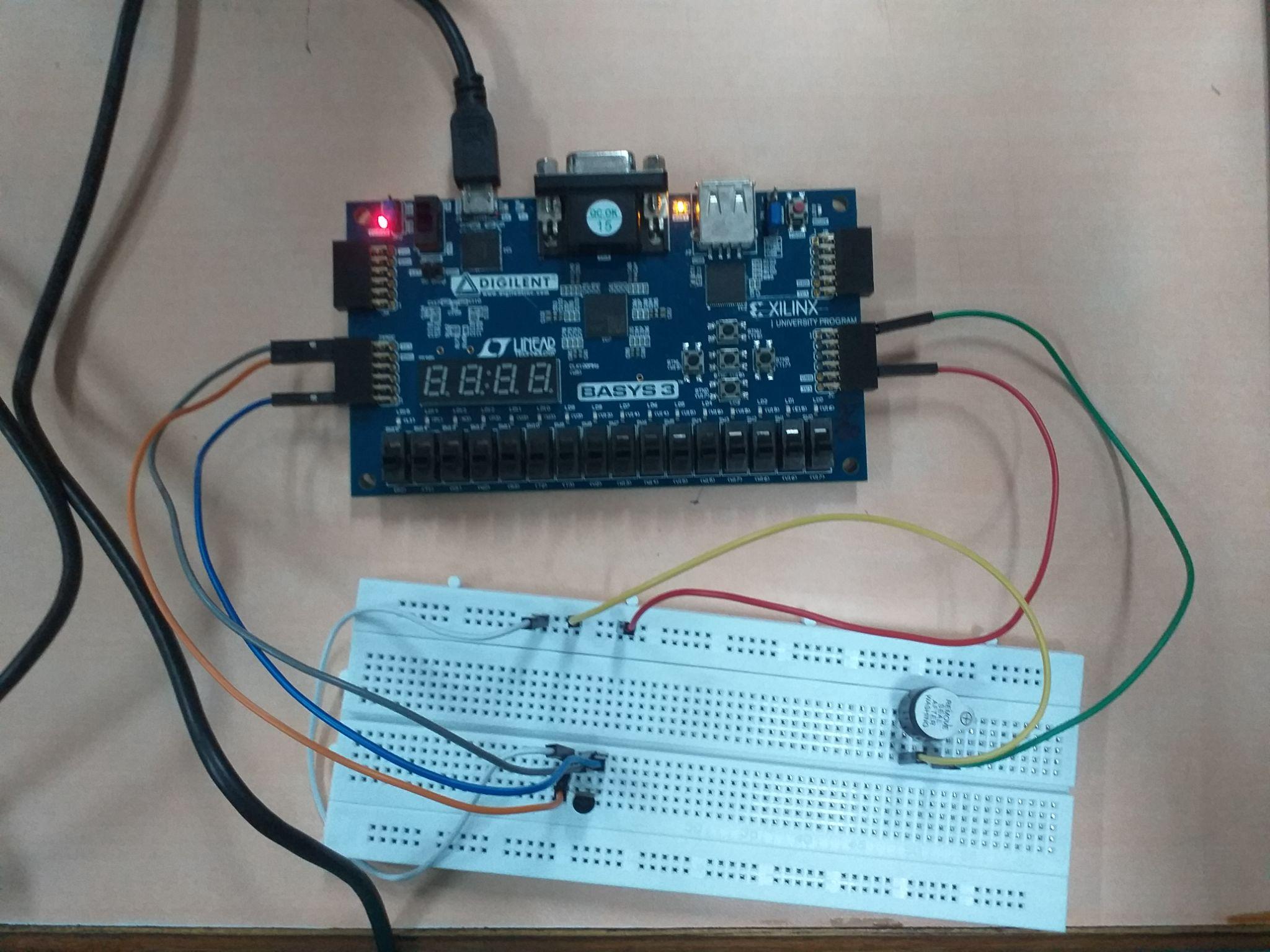
**SCHEMATIC POST SYNTHESIS**

**SIMULATION RESULTS**

**OUTPUT OF COUNTDOWN TIMER**

**OUTPUT OF SLOWCLOCK OF 1 SECOND**

**HARDWARE IMPLEMENTATION**



Hardware implementation is done post successful synthesis of the designed fire alarm system verilog code on Xilinx Vivado. This involves mapping the developed code to the Basys-3 board. We have additional hardware components apart from the Basys-3 board on the input and output sides, LM 35 temperature sensor on the input side and a piezo electric buzzer on the output side. These components need to be interfaced with the Basys-3 board. Thanks to the Pmod ports of Basys-3, this can be achieved efficiently. Basys-3 hosts different Pmod ports, namely Pmod JA, Pmod JB, Pmod JC, Pmod XDAC. The temperature sensor- LM 35 is connected to the JXADC port belonging to the Pmod XDAC group. This is done because the LM 35 converts the sensed temperature into an analog signal which is to be made digital in order to process on the FPGA. But the inbuilt JXADC port enables this. It converts the sensed analog signal from the sensor and converts it to digital for processing. To connect the buzzer, any of the ports belonging to the Pmod JA, Pmod JB, Pmod JC group can be made use of. The buzzer and temperature sensor are connected to a bread board and joined with the aforementioned Pmod ports with the help of connecting wires. Once this arrangement is made, and the code is successfully dumped onto the Basys-3 board, the fire alarm system can be realized.

**FUTURE SCOPE**

Owing to the increase in the number of fire threats these days, there is an exponential rise in the necessity of developments in the fire alarm system. Smoke detectors and alarms are migrating from just the detection of smoke, to combination detectors and multicriteria detectors.

**Detection of many gas components:**

The future will be with multicriteria detection in which the detector will be more of a sensor, with the detection more for the products of combustion, such as carbon monoxide, carbon dioxide, sulfur dioxide, nitrogen oxides in addition to heat and particulate matter.

**Integration of AI to reduce the evacuation time:**

Sensors will also have the ability to sense or track when a room is occupied or not and have the ability to be integrated with occupant notification and evacuation. The development of more advanced algorithms and artificial intelligence, both within the sensor itself and the front-end control unit will decrease the time from the beginning of an event to the notification of the event. It is not improbable that detection technology will be able to detect an incipient fire at that stage rather than at the flaming stage. This at the same time could reduce the likelihood of an unwanted activation from occurring.

**Easy detection of fire through image processing:**

Within the next decade, video image detection (VID) will become more mainstream in which, through analytics, the image of either smoke or flame will be able to be isolated and detected from within a room or space. The VID system would also be able to detect if an individual is within the space and through the integration with the notification appliances, provide a path of exit.

**Integration of evacuation guide:**

If the individual is familiar with the building, they generally know the location of exits. However, if one is not familiar with the occupancy or if there may be a blockage of one of the exits, then the notification system should be able to aid the occupants with a direct and safe exit. This may be through messaging via notification appliances but could also be through the interface of the detection and notification system to the smart device that the building occupants would have on them.

# **CONCLUSION**

Fire alarm systems are devices created and designed to alarm the user by sound systems and LEDs. They provide very high efficiency and can be used for security reasons. Furthermore, the proposed fire alarm system can be easily implemented and applied to large areas by installing these systems in every room thereby reducing the evacuation time and the risks of fire threats. The proposed system has low hardware costs. Many technologies can be integrated within the fire alarm system in the future to reduce the number of fire accidents.

# **REFERENCES**

[1] Peter J. Finley, Jr., Executive Analysis of Fire Service Operations in Emergency Management, Vineland Fire Department, Vineland, New Jersey

[2] Moore, C. (2015, 05 21). *Fire Detection – the Past, Present and Future*. Retrieved from International Fire Protection: <https://ifpmag.mdmpublishing.com/fire-detection-the-past-present-and-future/>

[3] *Zip CPU*. (2018, 04 17). Retrieved from Zip Timer: A simple countdown timer: <https://zipcpu.com/zipcpu/2018/04/17/ziptimer.html>

[4] A Home Automation System Design Using Hardware Descriptive Tools <https://www.ijert.org/research/a-home-automation-system-design-using-hardware-descriptive-tools-IJERTV2IS70169.pdf>

[5] Xilinx Artix-7 based FPGA Development Board (Basys 3) User Manual <https://digilent.com/reference/_media/basys3:basys3_rm.pdf>

[6] Verilog HDL: A Guide to Digital Design and Synthesis by Samir Palnitkar <https://www.google.co.in/books/edition/Verilog_HDL/fCSIpgsqkhkC?hl=en>

[7] *AllDataSheet*. Retrieved from LM35 Datasheet (PDF) - Texas Instruments: <https://pdf1.alldatasheet.com/datasheet-pdf/view/517588/TI1/LM35.html>

[8] *ElProCus*.Retrieved from What is a Buzzer : Working & Its Applications: <https://www.elprocus.com/buzzer-working-applications/>

# **APPENDIX**

`timescale 1ns / 1ps

//////////////////////////////////////////////////////////////////////////////////

// Company:

// Engineer:

//

// Create Date: 01.10.2021 13:12:40

// Design Name:

// Module Name: fire\_alarm\_system\_top\_module

// Project Name:

// Target Devices:

// Tool Versions:

// Description:

//

// Dependencies:

//

// Revision:

// Revision 0.01 - File Created

// Additional Comments:

//

//////////////////////////////////////////////////////////////////////////////////

module fire\_alarm\_system\_top\_module(

input clk,reset,

output [3:0] an,

output [6:0] seg,

output buzzer\_out

);

parameter zero = 4'b0000;

wire clk\_out;

wire newclk\_out;

wire [1:0] sig;

wire [3:0] out;

wire [7:0] count;

wire [11:0] bcd;

wire ioport;

wire [7:0] tempV;

wire [1:0] dout;

wire led;

**// Instantiating modules**

temperature\_sensor a10(clk,reset,ioport,tempV);

fsm a11(tempV,clk,reset,sig);

countdown\_10s a7 (newclk\_out,reset,sig,count);

slowclock\_1s a8(clk,newclk\_out);

binarytobcd a1 (count,bcd[11:0]);

//binarytobcd a1 (count,bcd[11:8],bcd[7:4],bcd[3:0]); old

slowclock a3 (clk,sig,led,buzzer\_out);

two\_bit\_counter a4(led,reset,dout);

mux4to1 a2(bcd[11:8],bcd[7:4],bcd[3:0],4'b0000,dout,out);

segment7 a6(out,seg);

decoder\_2\_4 a5(dout,an);

endmodule

**// Binary to BCD Conversion**

module binarytobcd(

count,

bcd

);

input [7:0] count;

output [11:0] bcd;

reg [11 : 0] bcd;

reg [3:0] i;

always @(count)

begin

bcd =0; //initialize bcd to zero.

for (i = 0; i < 8; i = i+1)

begin

bcd = {bcd[10:0],count[7-i]};

if(i < 7 && bcd[3:0] > 4)

bcd[3:0] = bcd[3:0] + 3;

if(i < 7 && bcd[7:4] > 4)

bcd[7:4] = bcd[7:4] + 3;

if(i < 7 && bcd[11:8] > 4)

bcd[11:8] = bcd[11:8] + 3;

end

end

endmodule

**// 4-1 MUX**

module mux4to1 ( input [3:0] a,

input [3:0] b,

input [3:0] c,

input [3:0] d,

input [1:0] dout,

output reg [3:0]out

);

always @ (a or b or c or d or dout[0] or dout[1])

begin

case (dout[0] | dout[1])

2'b00: out = a;

2'b01: out = b;

2'b10: out = c;

2'b11: out = 4'b0000;

endcase

end

endmodule

**// LED BLINKING AND BUZZER (ALARM)**

module slowclock(

input clk\_in, input [1:0]sig,

output led,

output reg buzzer\_out

);

reg [25:0]count=0;

reg [25:0] count1 = 0;

reg clk\_out;

assign led=0;

always@ (posedge clk\_in)

begin

if(sig == 2'b10)

begin

count<=count+1;

if (count==50\_000\_000)

begin

count<=0;

clk\_out=-clk\_out;

end

if (count1 <= 50\_000\_000)

begin

buzzer\_out <= 1;

end

else if (count1 >= 50\_000\_000 && count1<= 100\_000\_000)

begin

buzzer\_out <=0;

end

else

begin

buzzer\_out <=0;

end

end

end

assign led=clk\_out;

endmodule

**// 2-BIT COUNTER**

module two\_bit\_counter(led ,reset ,dout );

output [1:0] dout ;

reg [1:0] dout ;

input led ;

wire clk ;

input reset ;

wire reset ;

initial dout = 0;

always @ (posedge led)

begin

if (reset)

dout <= 0;

else

dout <= dout + 1;

end

endmodule

**// 2-4 DECODER**

module decoder\_2\_4(

input [1:0] dout,

output reg [3:0]an

);

always@(dout)

begin

case(dout)

2'b00: an = 4'b0001;

2'b01: an = 4'b0010;

2'b10: an = 4'b0100;

2'b11: an = 4'b1000;

endcase

end

endmodule

**// BCD - 7 segment Decoder**

module segment7(

out,

seg

);

//Declare inputs,outputs and internal variables.

input [3:0] out;

output [6:0] seg;

reg [6:0] seg;

//always block for converting bcd digit into 7 segment format

always @(out)

begin

case (out) //case statement

0 : seg = 7'b0000001;

1 : seg = 7'b1001111;

2 : seg = 7'b0010010;

3 : seg = 7'b0000110;

4 : seg = 7'b1001100;

5 : seg = 7'b0100100;

6 : seg = 7'b0100000;

7 : seg = 7'b0001111;

8 : seg = 7'b0000000;

9 : seg = 7'b0000100;

//switch off 7 segment character when the bcd digit is not a decimal number.

default : seg = 7'b1111111;

endcase

end

endmodule

**// COUNTDOWN 10S**

module countdown\_10s

(

input newclk\_out,

input reset,

input [1:0] sig,

output [7:0]count

);

reg [7:0]current\_count=0;

always @(posedge newclk\_out)

begin

if(reset)

current\_count<=10;

else if(sig==2'b00)

current\_count<=10;

else if ((sig==2'b10)&&(current\_count!=0))

//else if ((sig==2'b10)&(current\_count!=0)) old

current\_count<=current\_count-1;

else

current\_count<=current\_count;

end

assign count=current\_count;

endmodule

**// SLOW CLOCK 1S**

module slowclock\_1s(

input clk\_in,

output reg newclk\_out);

reg[26:0] period\_count=0;

always @(posedge clk\_in)

begin

if (period\_count != 100\_000\_000-1)

begin

period\_count<=period\_count+1;

newclk\_out<=0;

end

else

begin

period\_count<=0;

newclk\_out<=1;

end

end

endmodule

**// TEMPERATURE SENSOR**

module temperature\_sensor(

input wire clk,

input wire reset,

inout ioport,

output reg [7:0] tempV

);

always @(posedge clk)

begin

tempV <= ioport;

end

endmodule

**// FINITE STATE MACHINE**

module fsm (

input clk, reset,

input [7:0] tempV,

output [1:0] sig,

output reg flag\_in

);

localparam s00=0,s01=1,s10=2,s11=3;

reg [1:0] current\_state = 2'b00;

reg [1:0] next\_state = 2'b00;

reg setsig = 2'b00;

always @(posedge clk)

begin

if (reset)

current\_state <= s00;

else

current\_state <= next\_state;

end

**//Next State Combinational Logic**

always@(current\_state)

begin

case(current\_state)

s00:

begin

if(tempV != 8'b00000000)

next\_state = s01;

end

s01:

begin

if(tempV >= 8'b00110010)

next\_state =s10;

else

next\_state = s11;

end

s10:

begin

//countdown\_10s a7(newclk\_out,reset,sig,count);

//slowclock a9 (clk,newclk\_out);

end

s11:

begin

next\_state=s00;

end

endcase

end

**// Combinational Output Logic for each state**

always@(posedge clk)

begin

case(current\_state)

s00: begin

setsig = 2'b00;

end

s01: begin

setsig = 2'b01;

end

s10: begin

setsig = 2'b10;

end

s11: begin

setsig = 2'b11;

end

endcase

end

assign sig = setsig;

endmodule

**CONSTRAINTS FILE**

**#Temperature Sensor**

set\_property PACKAGE\_PIN J3 [get\_ports ioport]

set\_property IOSTANDARD LVCMOS33 [get\_ports ioport]

#Sch name = XA2\_P

set\_property PACKAGE\_PIN L3 [get\_ports {vauxp14}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {vauxp14}]

#Sch name = XA3\_P

set\_property PACKAGE\_PIN M2 [get\_ports {vauxp7}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {vauxp7}]

#Sch name = XA4\_P

set\_property PACKAGE\_PIN N2 [get\_ports {vauxp15}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {vauxp15}]

#Sch name = XA1\_N

set\_property PACKAGE\_PIN K3 [get\_ports {vauxn6}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {vauxn6}]

#Sch name = XA2\_N

set\_property PACKAGE\_PIN M3 [get\_ports {vauxn14}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {vauxn14}]

#Sch name = XA3\_N

set\_property PACKAGE\_PIN M1 [get\_ports {vauxn7}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {vauxn7}]

#Sch name = XA4\_N

set\_property PACKAGE\_PIN N1 [get\_ports {vauxn15}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {vauxn15}]

**#7 segment display**

set\_property PACKAGE\_PIN W7 [get\_ports {seg[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[6]}]

set\_property PACKAGE\_PIN W6 [get\_ports {seg[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[5]}]

set\_property PACKAGE\_PIN U8 [get\_ports {seg[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[4]}]

set\_property PACKAGE\_PIN V8 [get\_ports {seg[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[3]}]

set\_property PACKAGE\_PIN U5 [get\_ports {seg[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[2]}]

set\_property PACKAGE\_PIN V5 [get\_ports {seg[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[1]}]

set\_property PACKAGE\_PIN U7 [get\_ports {seg[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {seg[6]}]

**#buzzer**

set\_property PACKAGE\_PIN K17 [get\_ports buzzer\_out]

set\_property IOSTANDARD LVCMOS33 [get\_ports buzzer\_out]

##Sch name = JC2

set\_property PACKAGE\_PIN M18 [get\_ports {JC[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {JC[1]}]

##Sch name = JC3

set\_property PACKAGE\_PIN N17 [get\_ports {JC[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {JC[2]}]

##Sch name = JC4

set\_property PACKAGE\_PIN P18 [get\_ports {JC[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {JC[3]}]

##Sch name = JC7

set\_property PACKAGE\_PIN L17 [get\_ports {JC[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {JC[4]}]

##Sch name = JC8

set\_property PACKAGE\_PIN M19 [get\_ports {JC[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {JC[5]}]

##Sch name = JC9

set\_property PACKAGE\_PIN P17 [get\_ports {JC[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {JC[6]}]

##Sch name = JC10

set\_property PACKAGE\_PIN R18 [get\_ports {JC[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {JC[7]}]

**# LED**

set\_property PACKAGE\_PIN U16 [get\_ports led]

set\_property IOSTANDARD LVCMOS33 [get\_ports led]

set\_property PACKAGE\_PIN E19 [get\_ports {LED[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[1]}]

set\_property PACKAGE\_PIN U19 [get\_ports {LED[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[2]}]

set\_property PACKAGE\_PIN V19 [get\_ports {LED[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[3]}]

set\_property PACKAGE\_PIN W18 [get\_ports {LED[4]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[4]}]

set\_property PACKAGE\_PIN U15 [get\_ports {LED[5]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[5]}]

set\_property PACKAGE\_PIN U14 [get\_ports {LED[6]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[6]}]

set\_property PACKAGE\_PIN V14 [get\_ports {LED[7]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[7]}]

set\_property PACKAGE\_PIN V13 [get\_ports {LED[8]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[8]}]

set\_property PACKAGE\_PIN V3 [get\_ports {LED[9]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[9]}]

set\_property PACKAGE\_PIN W3 [get\_ports {LED[10]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[10]}]

set\_property PACKAGE\_PIN U3 [get\_ports {LED[11]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[11]}]

set\_property PACKAGE\_PIN P3 [get\_ports {LED[12]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[12]}]

set\_property PACKAGE\_PIN N3 [get\_ports {LED[13]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[13]}]

set\_property PACKAGE\_PIN P1 [get\_ports {LED[14]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[14]}]

set\_property PACKAGE\_PIN L1 [get\_ports {LED[15]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {LED[15]}]

set\_property PACKAGE\_PIN U2 [get\_ports {an[0]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[0]}]

set\_property PACKAGE\_PIN U4 [get\_ports {an[1]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[1]}]

set\_property PACKAGE\_PIN V4 [get\_ports {an[2]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[2]}]

set\_property PACKAGE\_PIN W4 [get\_ports {an[3]}]

set\_property IOSTANDARD LVCMOS33 [get\_ports {an[3]}]