FPGA MINI PROJECT | 26–04-2023

IMPLEMENTING A TEMPERATURE MONITORING SYSTEM USING VERILOG HDL

MINI PROJECT OF FPGA LABORATORY AS PARTIAL FULFILLMENT OF B.TECH COURSE

SUBMITTED BY:

POOJA GUPTA – 21ECB0A44
POTNURI SRI ANJALI PRAVALLIKA – 21ECB0A45
SRINIDHI P- 21ECB0A46
SRI ABHIRAAMI K-21ECB0F01

SUBMITTED TO:
DR.HANUMANTH RAO

(DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING)
National Institute of Technology Warangal
2021-2025

Index

Abstract

Problem statement

Articles which describes few accidents involving temperature fluctuations brought the basic idea to the implementation Temperature monitoring system on Verilog HDL.

Methodology

This project deals with the simulation of TEMPERATURE MONITORING SYSTEM using Verilog HDL with Xilinx tool NEXYS 4 DDR tool, XC7a100tCG325-1 board. The Architecture basically consists of basic gates, ROMs, comparators to compare present data with higher and lower set limits and a 7 Segement display to display the temperature detected. The device is an auto running system with input data obtained from LM35, and an ADC CONVERTER MODULE which uses a simple 8bit 392 KSPS ADC with parallel output. Lower limit and higher limit of system can be updated through 2 more inputs data attached with the design to serve the purpose.

Outcome

We will see the result which is temperature on seven segment display connected to fpga and control it by adjusting the inputs.

Theory

Introduction

The device starts with power supply connected to 3 basic components that are used i.e LM35(temperature sensor), ADC(analog digital convertor) convertor and the XILINX NEXYS 4DDR.

Primarily LM35 ,a thermocoupleor resistance temperature detector that provides temperature measurementit a readable form through an electrical signal i.e it detects temp and outputs the detected temp value equivalent to voltage, now this Analog signal is passed to an ADC converter whose prime goal is to convert Analog to digital in parallel bits. Now that this is binary input to the system, there are 2 more inputs set which are used to set least and highest acceptable temperature limits.

Now that the system VERILOG is designed in such a way that whenever input binary value changes, 7 segment will DISPLAY the input binary value. comparators check and compare the range of input to the BINARY INPUT SET LIMITS . A BUZZER (precisely a light / led) that lits up as an alarming sign, if the comparator conditions show failed results there by there on a 7 segement display, comparators and an alarm buzz (can be a light or a sound).

The same operation continues so that there is an analysis on TEMPERATURE fluctuations and there can be interpreted for where ever there are fire and temperature incidents.

Procedure

- Conversion of analog signal to digital using analog to digital convertor.
- Reading binary data and converting it to seven segment display form and inputing the data in seven segment display device.
- Checking if binary data breaks the user set data limits which has lower value and a higher value.
- Activating buzzer or light if the binary input data breaks the user set data limits .
- If it does not break the limits, buzzer or light is in off state and waits till the process shows negative results.

Modules used

There are two modules in this implementation process

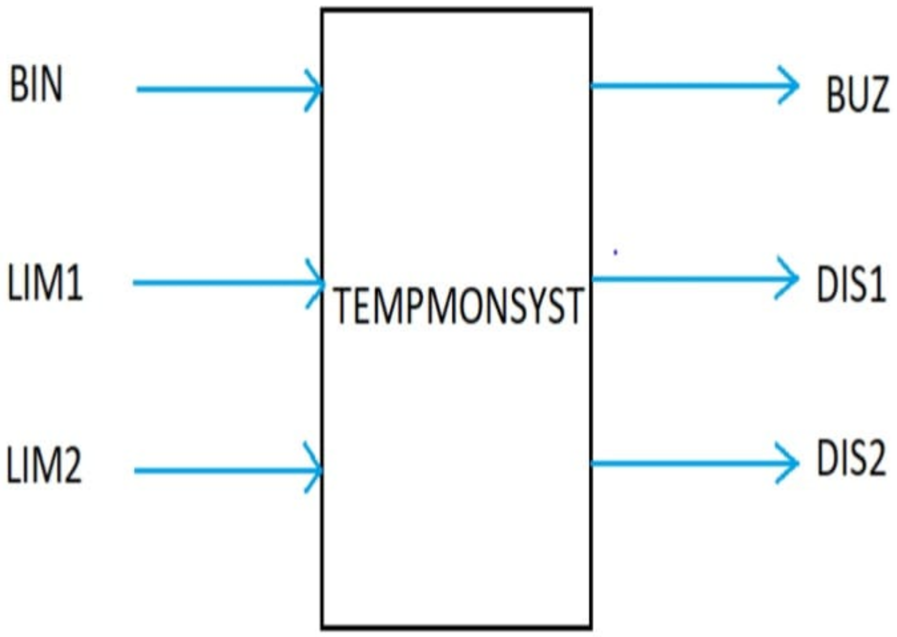
- 7segment display module
- Comparator module

These modules are defined using behavioral modelling of Verilog HDL.

Components used

- Temperature sensor LM35
- 8 bit 392 KSPS ADC with parallel output
- Xilinx Nexys 4 DDR
- 7 segment display block
- comparators, buzzer
- basic gates for the activation for buzzer or led.

Block Diagram



The Temperature Sensor

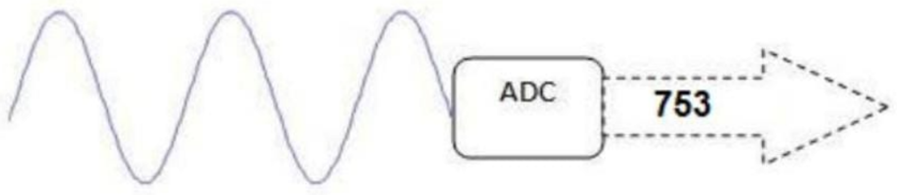
Firstly the temp sensor which is available , affordable , efficient and suitable is LM35.The resolution of LM35 IS 10mV per every degree where the input voltage is connected to Vs and also another terminal is grounded, now middle pin or the 3rd terminal generates Analog output equivalent to the temperature sensed by a sensor with resolution considered.Generally, operating voltages for this sensor are 4V to 30V, But this voltage is designed as per the suitability of the ADC converter which is also driven by the same input voltage. More rightly so the typical voltage of 5V is considered for lm35, which is also the same consideration for the system.

SOME KEY FEATURES OF LM35 ARE:

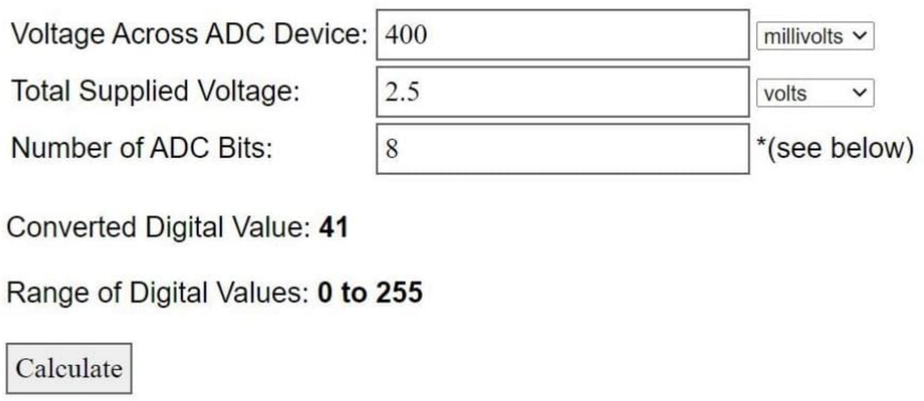
1. Linear 10mV/C scale factor
2. 0.5 to 1 C ensured accuracy
3. Rated from -55 C to 150 C RANGE
4. Less than 60uA current drain
5. Low self heating in air
6. Calibrated in Celsius

Analog to Digital Converter

Out of available ADC converters in the market, FLASH ADC converters or parallel ADC are fast output generators and are widely used for fast output requirements The ADC converter used in this case is Basically the TLC0820ACN CHIP FROM TEXAS INSTRUMENT, WHICH IS A SIMPLE 8 BIT KSPS ADC WITH PARALLEL OUTPUT FOR AN ADC DEVICE TO HAVE PROPER CONVERSION, NO OF BITS AND INPUT VOLTAGE PLAYS CRUCIAL ROLE, Below is a conversion algorithm to suit the system outcome since output of LM35 will be (temp*10mV), input voltage is to be selected in such a way that output has 8 bits digital output (for FPGA) and converted digital value(decimal) is exactly (temp).



$$\text{Analog value} = \frac{\text{Voltage drop/output x maximum ADC value}}{\text{total voltage supplied}}$$



Voltage Across ADC Device:	400	millivolts
Total Supplied Voltage:	2.5	volts
Number of ADC Bits:	8	*(see below)

Converted Digital Value: **41**

Range of Digital Values: **0 to 255**

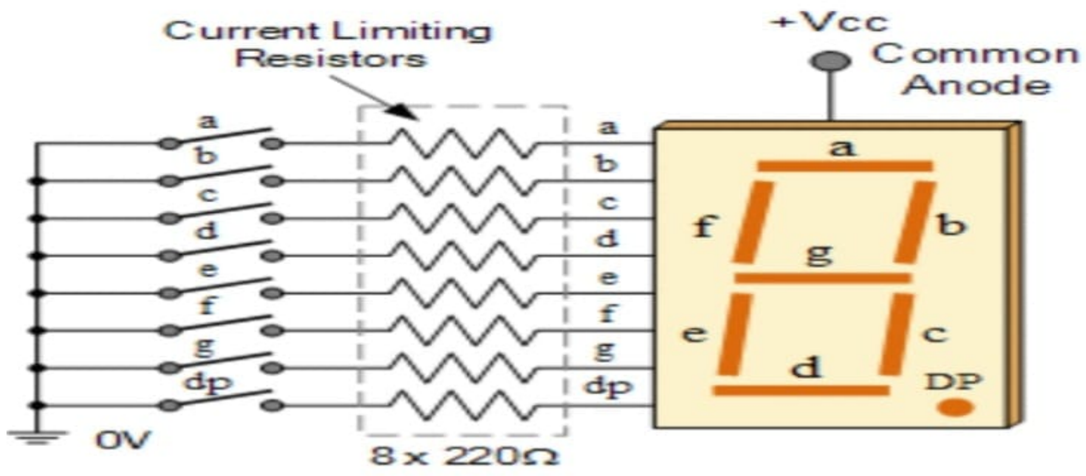
So a value of 2.5V as supply voltage for this adc converter resolves us to an error of 1 C which is acceptable. Range of digital values is 0 – 255 .

Seven segment display

The 8 bit digital output of adc converter is the input for the module and before that lim1(higher limit) and limit (lower limit) is pre set for the system to operate, whenever lm35 finds change in temperature , analog signal sent to adc is converted to digital and is received to fpga main module. Now the cross checking of the 8 bit binary to be falling in b/w lim1 and lim2 or not is done , and accordingly buzzer activation is triggered .

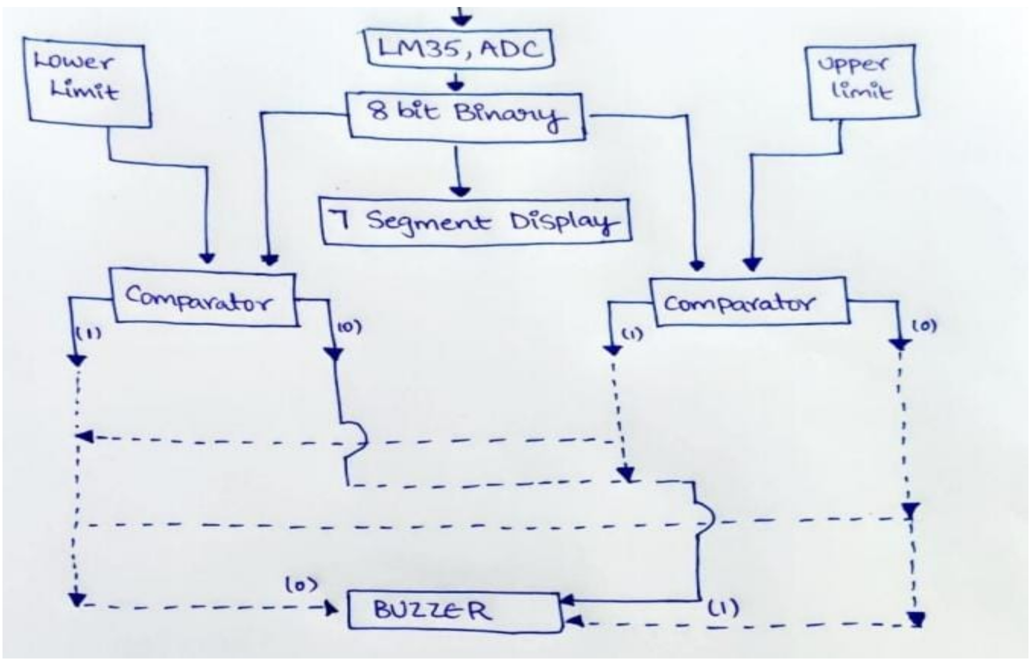
On the other hand the module is set to display the digital value in a 7 segment display.

The process carries by, where its activity is only to monitor temperature for as long as possible and no preset options are set because of its prime motive to just keep tracking temperatures and find out if temperature is troublesome to the particular environment it is fit into.Thereby it just activated once for activity and power off is only way to stop it from active state another thing is, the limits can be changed whenever required and not restricted to one time enter.



Since the core application of the above picture is done rightly so inside a fpga kit , our job is to just assign proper values to these a,b,c,d,e,f,g. since we considered a 2 digit 7 seg display. We need such 7 bit data, which is why dis1 and dis2 are considered and both are [6:0] bus where most significant bit equals a and continues, ending with least significant bit equals g.Hence input over 100 is displayed as ee. dis1 displays one digit of 7 segment display while dis2 displays tens digit of 7 seg display.

Flowchart



Verilog code

```
`timescale 1ns / 1ps
module temp_monsys(input [7:0] bin, //input 8bit binary
 output reg buz, // output light buzzer
// input [7:0] lim1, // highest limit for temperature
 //input [7:0] lim2, // lowest limit for temperature
 output reg [6:0]dis1, // ones digit 7seg display
 output reg [6:0]dis2 // tens digit 7seg display
 );
 reg g1,g2,l1,l2,e1,e2;
 reg [7:0]lim1,lim2;
 always @(bin)
 begin
 lim1=8'b00011110; lim2=8'b00010100;

 //two digit 7seg display

 case(bin)
 8'b00000000 : begin dis1 = 7'b1111110; dis2=7'b1111110; end
 8'b00000001 : begin dis1 = 7'b0110000; dis2=7'b1111110; end
 8'b00000010 : begin dis1 = 7'b1101101; dis2=7'b1111110; end
 8'b00000011 : begin dis1 = 7'b1111001; dis2=7'b1111110; end
 8'b00000100 : begin dis1 = 7'b0110011; dis2=7'b1111110; end
```

```
8'b00000101 : begin dis1 = 7'b1011011; dis2=7'b1111110; end
8'b00000110 : begin dis1 = 7'b1011111; dis2=7'b1111110; end
8'b00000111 : begin dis1 = 7'b1110000; dis2=7'b1111110; end
8'b00001000 : begin dis1 = 7'b1111111; dis2=7'b1111110; end
8'b00001001 : begin dis1 = 7'b1110011; dis2=7'b1111110; end
8'b00001010 : begin dis1 = 7'b1111110; dis2=7'b0110000; end
8'b00001011 : begin dis1 = 7'b0110000; dis2=7'b0110000; end
8'b00001100 : begin dis1 = 7'b1101101; dis2=7'b0110000; end
8'b00001101 : begin dis1 = 7'b1111001; dis2=7'b0110000; end
8'b00001110 : begin dis1 = 7'b0110011; dis2=7'b0110000; end
8'b00001111 : begin dis1 = 7'b1011011; dis2=7'b0110000; end
8'b00010000 : begin dis1 = 7'b1011111; dis2=7'b0110000; end
8'b00010001 : begin dis1 = 7'b1110000; dis2=7'b0110000; end
8'b00010010 : begin dis1 = 7'b1111111; dis2=7'b0110000; end
8'b00010011 : begin dis1 = 7'b1110011; dis2=7'b0110000; end
8'b00010100 : begin dis1 = 7'b1111110; dis2=7'b1101101; end
8'b00010101 : begin dis1 = 7'b0110000; dis2=7'b1101101; end
8'b00010110 : begin dis1 = 7'b1101101; dis2=7'b1101101; end
8'b00010111 : begin dis1 = 7'b1111001; dis2=7'b1101101; end
8'b00011000 : begin dis1 = 7'b0110011; dis2=7'b1101101; end
8'b00011001 : begin dis1 = 7'b1011011; dis2=7'b1101101; end
8'b00011010 : begin dis1 = 7'b1011111; dis2=7'b1101101; end
8'b00011011 : begin dis1 = 7'b1110000; dis2=7'b1101101; end
8'b00011100 : begin dis1 = 7'b1111111; dis2=7'b1101101; end
```

```
8'b00011101 : begin dis1 = 7'b1110011; dis2=7'b1101101; end

8'b00011110 : begin dis1 = 7'b1111110; dis2=7'b1111001; end

8'b00011111 : begin dis1 = 7'b0110000; dis2=7'b1111001; end

8'b00100000 : begin dis1 = 7'b1101101; dis2=7'b1111001; end

8'b00100001 : begin dis1 = 7'b1111001; dis2=7'b1111001; end

8'b00100010 : begin dis1 = 7'b0110011; dis2=7'b1111001; end

8'b00100011 : begin dis1 = 7'b1011011; dis2=7'b1111001; end

8'b00100100 : begin dis1 = 7'b1011111; dis2=7'b1111001; end

8'b00100101 : begin dis1 = 7'b1110000; dis2=7'b1111001; end

8'b00100110 : begin dis1 = 7'b1111111; dis2=7'b1111001; end

8'b00100111 : begin dis1 = 7'b1110011; dis2=7'b1111001; end

8'b00101000 : begin dis1 = 7'b1111110; dis2=7'b0110011; end

8'b00101001 : begin dis1 = 7'b0110000; dis2=7'b0110011; end

8'b00101010 : begin dis1 = 7'b1101101; dis2=7'b0110011; end

8'b00101011 : begin dis1 = 7'b1111001; dis2=7'b0110011; end

8'b00101100 : begin dis1 = 7'b0110011; dis2=7'b0110011; end

8'b00101101 : begin dis1 = 7'b1011011; dis2=7'b0110011; end

8'b00101110 : begin dis1 = 7'b1011111; dis2=7'b0110011; end

8'b00101111 : begin dis1 = 7'b1110000; dis2=7'b0110011; end

8'b00110000 : begin dis1 = 7'b1111111; dis2=7'b0110011; end

8'b00110001 : begin dis1 = 7'b1110011; dis2=7'b0110011; end

8'b00110010 : begin dis1 = 7'b1111110; dis2=7'b1011011; end

8'b00110011 : begin dis1 = 7'b0110000; dis2=7'b1011011; end

8'b00110100 : begin dis1 = 7'b1101101; dis2=7'b1011011; end
```

```
8'b00110101 : begin dis1 = 7'b1111001; dis2=7'b1011011; end
8'b00110110 : begin dis1 = 7'b0110011; dis2=7'b1011011; end
8'b00110111 : begin dis1 = 7'b1011011; dis2=7'b1011011; end
8'b00111000 : begin dis1 = 7'b1011111; dis2=7'b1011011; end
8'b00111001 : begin dis1 = 7'b1110000; dis2=7'b1011011; end
8'b00111010 : begin dis1 = 7'b1111111; dis2=7'b1011011; end
8'b00111011 : begin dis1 = 7'b1110011; dis2=7'b1011011; end
8'b00111100 : begin dis1 = 7'b1111110; dis2=7'b1011111; end
8'b00111101 : begin dis1 = 7'b0110000; dis2=7'b1011111; end
8'b00111110 : begin dis1 = 7'b1101101; dis2=7'b1011111; end
8'b00111111 : begin dis1 = 7'b1111001; dis2=7'b1011111; end
8'b01000000 : begin dis1 = 7'b0110011; dis2=7'b1011111; end
8'b01000001 : begin dis1 = 7'b1011011; dis2=7'b1011111; end
8'b01000010 : begin dis1 = 7'b1011111; dis2=7'b1011111; end
8'b01000011 : begin dis1 = 7'b1110000; dis2=7'b1011111; end
8'b01000100 : begin dis1 = 7'b1111111; dis2=7'b1011111; end
8'b01000101 : begin dis1 = 7'b1110011; dis2=7'b1011111; end
8'b01000110 : begin dis1 = 7'b1111110; dis2=7'b1110000; end
8'b01000111 : begin dis1 = 7'b0110000; dis2=7'b1110000; end
8'b01001000 : begin dis1 = 7'b1101101; dis2=7'b1110000; end
8'b01001001 : begin dis1 = 7'b1111001; dis2=7'b1110000; end
8'b01001010 : begin dis1 = 7'b0110011; dis2=7'b1110000; end
8'b01001011 : begin dis1 = 7'b1011011; dis2=7'b1110000; end
8'b01001100 : begin dis1 = 7'b1011111; dis2=7'b1110000; end
```

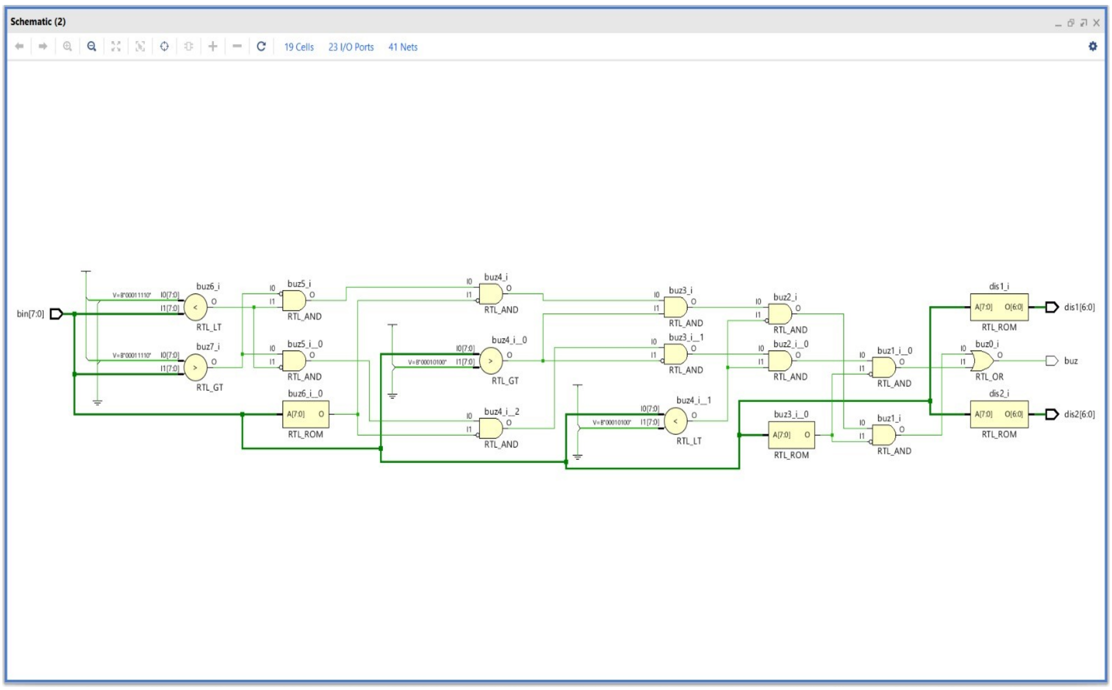
```
8'b01001101 : begin dis1 = 7'b1110000; dis2=7'b1110000; end

8'b01001110 : begin dis1 = 7'b1111111; dis2=7'b1110000; end

8'b01001111 : begin dis1 = 7'b1110011; dis2=7'b1110000; end

8'b01010000 : begin dis1 = 7'b1111110; dis2=7'b1111111; end

8'b01010001 : begin dis1 = 7'b0110000; dis2=7'b1111111; end

8'b01010010 : begin dis1 = 7'b1101101; dis2=7'b1111111; end

8'b01010011 : begin dis1 = 7'b1111001; dis2=7'b1111111; end

8'b01010100 : begin dis1 = 7'b0110011; dis2=7'b1111111; end

8'b01010101 : begin dis1 = 7'b1011011; dis2=7'b1111111; end

8'b01010110 : begin dis1 = 7'b1011111; dis2=7'b1111111; end

8'b01010111 : begin dis1 = 7'b1110000; dis2=7'b1111111; end

8'b01011000 : begin dis1 = 7'b1111111; dis2=7'b1111111; end

8'b01011001 : begin dis1 = 7'b1110011; dis2=7'b1111111; end

8'b01011010 : begin dis1 = 7'b1111110; dis2=7'b1110011; end

8'b01011011 : begin dis1 = 7'b0110000; dis2=7'b1110011; end

8'b01011100 : begin dis1 = 7'b1101101; dis2=7'b1110011; end

8'b01011101 : begin dis1 = 7'b1111001; dis2=7'b1110011; end

8'b01011110 : begin dis1 = 7'b0110011; dis2=7'b1110011; end

8'b01011111 : begin dis1 = 7'b1011011; dis2=7'b1110011; end

8'b01100000 : begin dis1 = 7'b1011111; dis2=7'b1110011; end

8'b01100001 : begin dis1 = 7'b1110000; dis2=7'b1110011; end

8'b01100010 : begin dis1 = 7'b1111111; dis2=7'b1110011; end

8'b01100011 : begin dis1 = 7'b1110011; dis2=7'b1110011; end

default : begin dis1 = 7'b1101111; dis2=7'b1101111; end
```

```
 endcase

// limit1 and binary input comparator

 begin
 g1 <= ( lim1 > bin )? 1'b1 : 1'b0;
 l1 <= ( lim1 < bin )? 1'b1 : 1'b0;
 e1 <= ( lim1 == bin)? 1'b1 : 1'b0;
 end
// limit2 and binary input comparator

 begin
 g2 <= ( bin > lim2 )? 1'b1 : 1'b0;
 l2 <= ( bin < lim2 )? 1'b1 : 1'b0;
 e2 <= ( lim2 == bin)? 1'b1 : 1'b0;
 end

// light buzzer when bin is below lim1 or above lim2

buz=( ((~g1)&(l1)&(~e1)&(g2)&(~l2)&(~e2)) |
((g1)&(~l1)&(~e1)&(~g2)&(l2)&(~e2)) );

end
endmodule
```

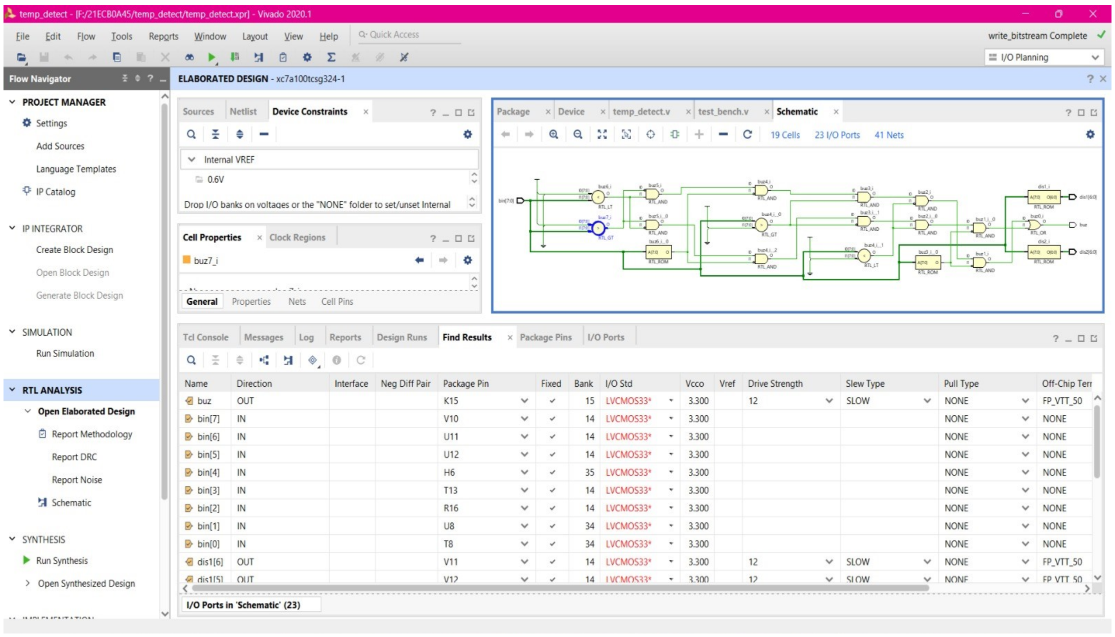
Testbench

```
`timescale 1ns / 1ps

module tb();

reg[7:0]bin;

wire buz;

//reg[7:0]lim1,lim2;

wire[6:0]dis1,dis2;

temp_monsys t1(bin,buz,dis1,dis2);

initial

begin

 #50 bin = 8'b00000000 ;

 #50 bin = 8'b01111000 ;

 #50 bin = 8'b00111110 ;

 #50 bin = 8'b00010100 ;

 #50 bin = 8'b00010101 ;

 #50 bin = 8'b00010110 ;

 #50 bin = 8'b00010111 ;

 #50 bin = 8'b00011000 ;

 #50 bin = 8'b00011001 ;

 #50 bin = 8'b00011010 ;

 #50 bin = 8'b00011011 ;

 #50 bin = 8'b00011100 ;

 #50 bin = 8'b00011101 ;
 #50 bin = 8'b00011110 ;
 #50 bin = 8'b00101001 ;
 #50 bin = 8'b01110111 ;
 #50 bin = 8'b01111001 ;
 end
endmodule
```

Schematic



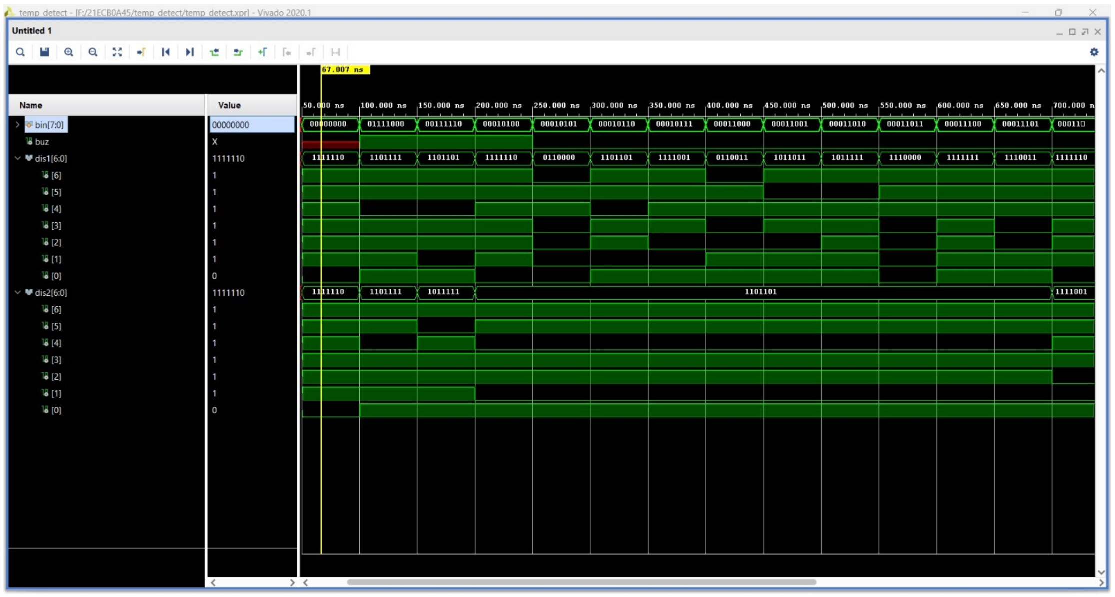
Bitstream

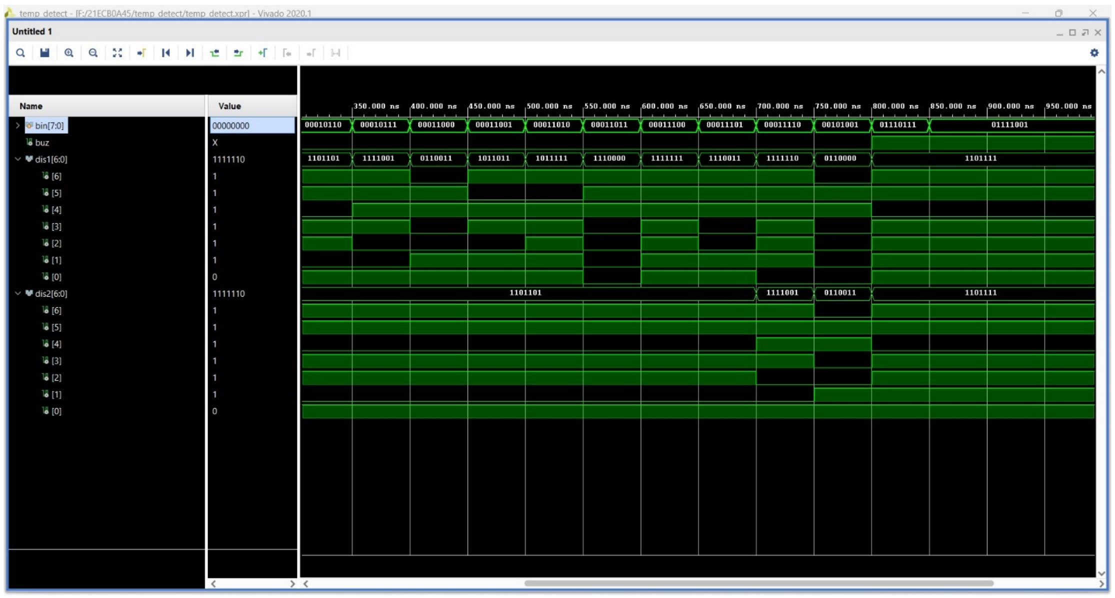


Name	Direction	Interface	Neg Diff Pair	Package Pin	Fixed	Bank	I/O Std	Vcco	Vref	Drive Strength	Slew Type	Pull Type	Off-Chip Terr
buz	OUT			K15	✓	15	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
bin[7]	IN			V10	✓	14	LVCMOS33*	3.300				NONE	NONE
bin[6]	IN			U11	✓	14	LVCMOS33*	3.300				NONE	NONE
bin[5]	IN			U12	✓	14	LVCMOS33*	3.300				NONE	NONE
bin[4]	IN			H6	✓	35	LVCMOS33*	3.300				NONE	NONE
bin[3]	IN			T13	✓	14	LVCMOS33*	3.300				NONE	NONE
bin[2]	IN			R16	✓	14	LVCMOS33*	3.300				NONE	NONE
bin[1]	IN			U8	✓	34	LVCMOS33*	3.300				NONE	NONE
bin[0]	IN			T8	✓	34	LVCMOS33*	3.300				NONE	NONE
dis1[6]	OUT			V11	✓	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50
dis1[5]	OUT			V12	✓	14	LVCMOS33*	3.300		12	SLOW	NONE	FP_VTT_50

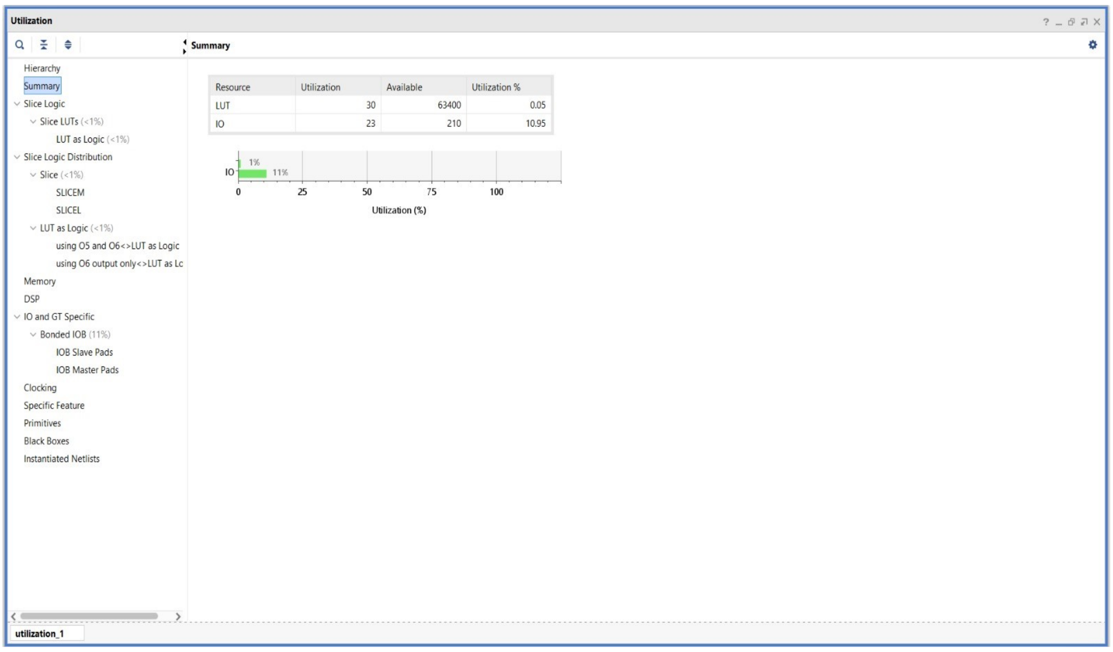
I/O Ports in 'Schematic' (23)

Simulations





Utilization report



Utilization

Summary

Resource	Utilization	Available	Utilization %
LUT	30	63400	0.05
IO	23	210	10.95

Future Scope

1)The project can be extended to 3 digit display as per the requirement with slightly more verilog statements in the verilog code.

2) It can be converted into one wire thermometer which can be used for daily purposes.

3) Temp sensor can be changed according to requirement. (because it is limited to certain temperatures only) or ADC converter can be changed to 8+ bits (but one needs to look at operating voltages otherwise resolution may vary and thereby Verilog code should be modified)

Conclusion

TEMP MONITORING SYSTEM is successfully implemented using Verilog HDL on FPGA board. The toughest part of the project is to detect and convert the Analog signal into digital form i.e in binary bit form. And later placement of comparators which gives the least delay is one more problem during the project considering the prime idea behind this project is from the accident that took place in an incubator, this design is enough and better suited and more likely to work efficiently.

Real-Life Applications

This system is not only for the purpose of incubators as mentioned, but it can serve in many more temperature fluctuations cases such as over load and high temperatures which cause electrical (instrumental) failures.

▪ The following is a stat which describes accidents related to temperature fluctuations.

▪ Few more thermally sensitive devices or devices working in high temperatures require close monitoring, for which this can be employed in.

▪ Industrial Systems includes all industries that work under high temperatures , especially the Reactors where huge man power is invested in unpredictable conditions.

References

1] Quitoartblogospot FPGA Verilog project on DS18B20

Thermometer .

http://quitoart.blogspot.com/2017/11/fpga-verilog-ds18b20-temperat ure-

sensor.html?m=1

2] Design and Analysis of a Wireless Temperature

Monitoring System

Nor Alina Khairi, Asral Bahari Jambek, and Teoh Wei Boon,

Uda Hashim

School of Microelectronic Engineering, Universiiy Malaysia Perlis, Malaysia.

Institute of Nanoelectronic Engineering, University Malaysia Perlis