REPORT ON ALARM CLOCK USING VERILOG

DSD Project Report

Submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF TECHNOLOGY in

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

SHARATH BALAJI T

(211EC249)

Submitted to

Dr. Rathnamala Rao

Assistant Professor

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING,

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA,

SURATHKAL, MANGALORE -575025

January 2023

Abstract

In this project, an alarm clock is designed and implemented using Verilog HDL. The clock includes features such as setting the time, setting an alarm, displaying the time and alarm, and stopping the alarm. The clock is controlled by a state machine, with different states for setting the time, setting the alarm, and displaying the time and alarm. The alarm can be loaded or stopped. The project also includes a testbench to verify the correct functioning of the clock. In this work, the design of the Alarm Clock is implemented in Verilog HDL and simulated using a hardware simulation tool. The functionality of the Alarm Clock is verified through simulation waveforms and test cases. The results show that the Alarm Clock can ring at loaded time perfectly. The goal of this project is to demonstrate the ability to design and implement a functional digital system using Verilog.

Contents

1.Introduction

Alarm Clock is a device basically used for reminding/alert something after a particular time. The introduction of this project aims to provide an overview of the design and implementation of an alarm clock using Verilog.

The alarm clock is a digital system that is controlled by a state machine and includes features such as setting the time, setting an alarm, and displaying the current time and alarm time on a display. The alarm can be set for a particular time and the alarm rings at the given time. The alarm can be turned on or turned off. The project also includes a testbench to verify the correct functioning of the clock.

The goal of this project is to demonstrate the ability to design and implement a functional digital system using Verilog, which is a hardware description language used to design digital systems. The project will cover the design, implementation and testing of the digital system i.e., Alarm Clock.

2.Design

The design of the Alarm clock is shown in this discussion.

- First, we will see the Block Diagram of functioning of the alarm clock.

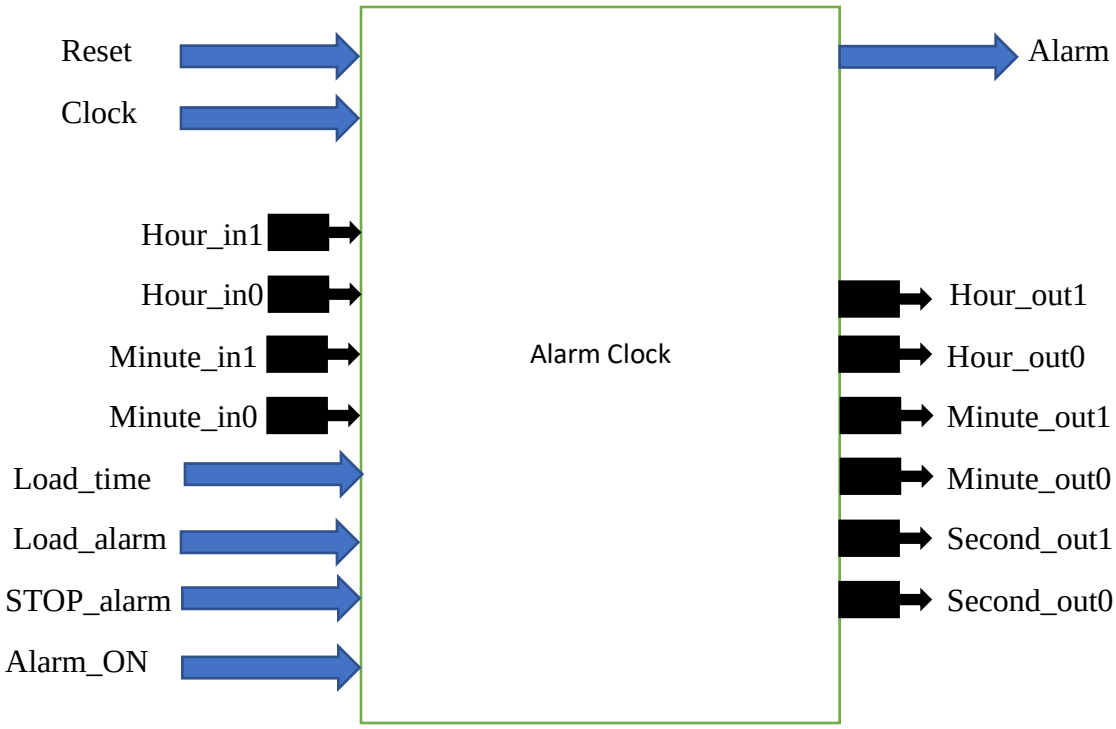


Fig.1: Block Diagram of Alarm Clock

The alarm clock outputs a real-time clock with a 24-hour format and provides an alarm feature. Users also can set the clock time through switches.

NOTE: the time format used in this clock is 24 hours,60 minutes and 60 seconds.

The format in which the time is taken to load time and set alarm is in the below format:

[Hour_MSB] [Hour_LSB]: [Minute_MSB] [Minute_LSB]

The format in which the time is counted, and alarm rings is in the below format:

[Hour_MSB] [Hour_LSB]: [Minute_MSB] [Minute_LSB]: [Second_MSB] [Second _LSB]

- Next, we will see the elaborated design of the Alarm Clock that is generated in the hardware simulation tool

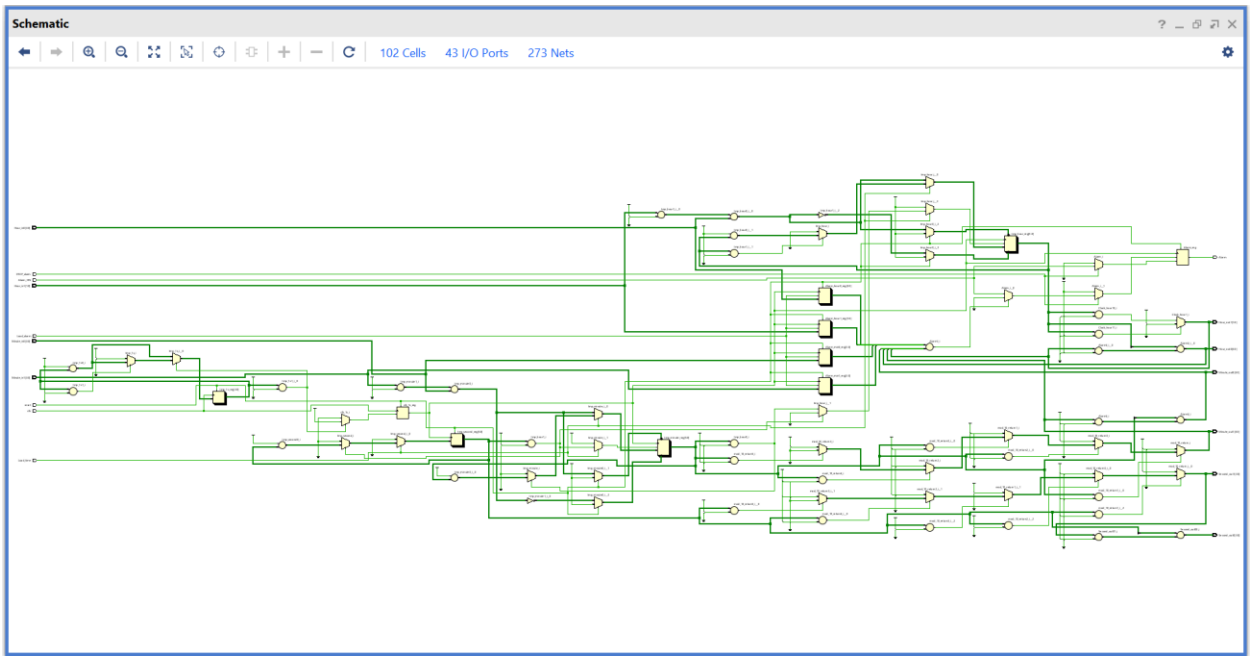


Fig.2: Elaborated Design of the Alarm Clock using Vivado

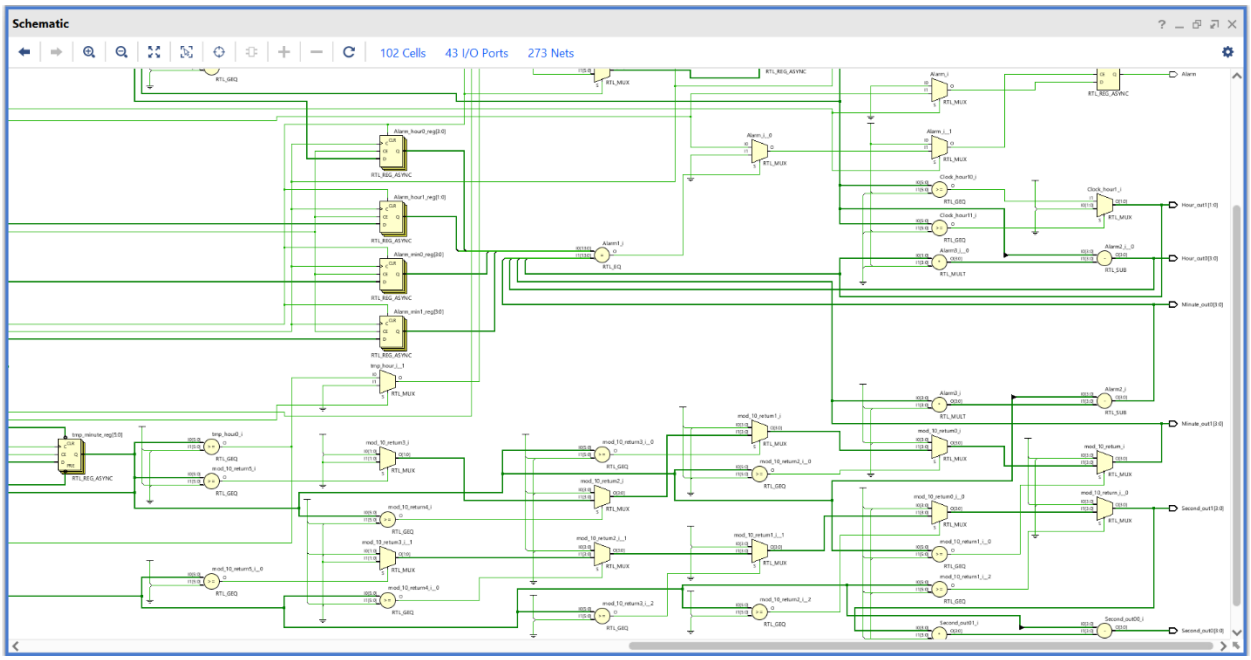


Fig.3: Zoomed-in picture of Elaborated Design of the Alarm Clock using Vivado

3.Implementation

First, we will see the flow chart of how the Verilog code works. According to this flowchart the alarm clock function is written in the code with keywords used in the flowchart, so whole flow of code can be understood by the flow chart.

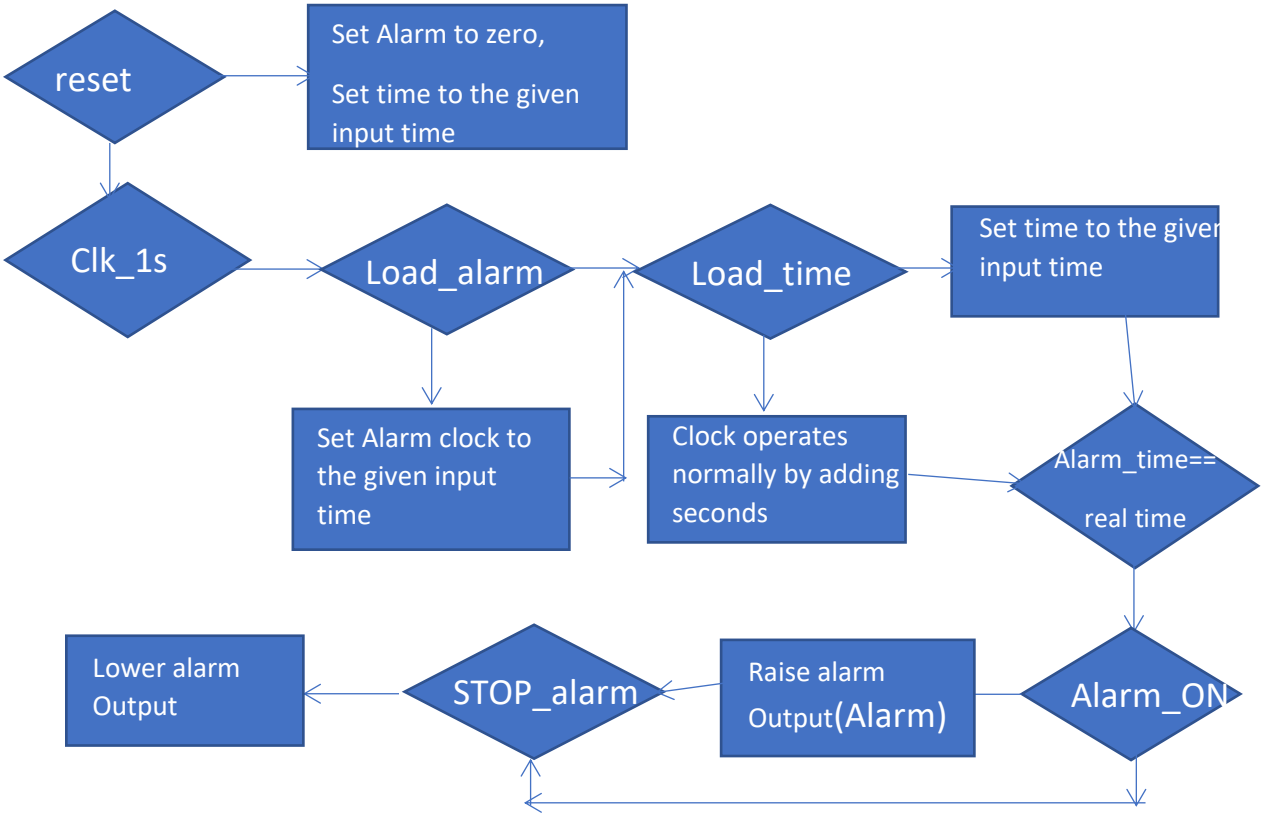


Chart: Flow Chart of the working of the Verilog of Alarm Clock

In this program we have used Clk_1s for generating the clock of 1 second time counting using the clock of the clk defined in the code.

We will see the Verilog module and testbench on which the Alarm Clock works.

- **Verilog Code for Alarm Clock**

```
27  module ClkAlarm(
28  input reset,
29  input clk,
30  input [1:0] Hour_in1,
31  input [3:0] Hour_in0,
32  input [3:0] Minute_in1,
33  input [3:0] Minute_in0,
34  input Load_time,
35  input   Load_alarm,
36  input   STOP_alarm,
37  input   Alarm_ON,
38  output reg Alarm,
39  output [1:0]  Hour_out1,
40  output [3:0]  Hour_out0,
41  output [3:0]  Minute_out1,
42  output [3:0]  Minute_out0,
43  output [3:0]  Second_out1,
44  output [3:0]  Second_out0);
45
46  reg clk_1s;
47  reg [3:0] tmp_1s;
48  reg [5:0] tmp_hour, tmp_minute, tmp_second;
49  reg [1:0] Clock_hour1,Alarm_hour1;
50  reg [3:0] Clock_hour0,Alarm_hour0;
51  reg [3:0] Clock_min1,Alarm_min1;
52  reg [3:0] Clock_min0,Alarm_min0;
53  reg [3:0] Clock_sec1,Alarm_sec1;
54  reg [3:0] Clock_sec0,Alarm_sec0;
55
56  function [3:0] mod_10;
57   input [5:0] number;
58   begin
59   mod_10 = (number >=50) ? 5 : ((number >= 40)? 4 :((number >= 30)? 3 :((number >= 20)? 2 :((number >= 10)? 1 :0))));
60   end
61  endfunction
```

Fig.4: Verilog Code written in Vivado for implementation

In Fig.4, Input, output is defined and there is a function mod_10 used for that takes input number and gives out the remainder after dividing by 10, this function is useful later on when the tmp_second and tmp_minute is restricted to 60 seconds count only in the Clock.

```
always @(posedge clk_1s or posedge reset )
 begin
 if(reset) begin
  Alarm_hour1 <= 2'b00;
  Alarm_hour0 <= 4'b0000;
  Alarm_min1 <= 4'b0000;
  Alarm_min0 <= 4'b0000;
  Alarm_sec1 <= 4'b0000;
  Alarm_sec0 <= 4'b0000;
  tmp_hour <= Hour_in1*10 + Hour_in0;
  tmp_minute <= Minute_in1*10 + Minute_in0;
  tmp_second <= 0;
 end
 else begin
 if(Load_alarm) begin
  Alarm_hour1 <= Hour_in1;
  Alarm_hour0 <= Hour_in0;
  Alarm_min1 <= Minute_in1;
  Alarm_min0 <= Minute_in0;
  Alarm_sec1 <= 4'b0000;
  Alarm_sec0 <= 4'b0000;
 end
 if(Load_time) begin
  tmp_hour <= Hour_in1*10 + Hour_in0;
  tmp_minute <= Minute_in1*10 + Minute_in0;
  tmp_second <= 0;
 end
```

Fig.5: Verilog Code written in Vivado for implementation

```
 else begin
  tmp_second <= tmp_second + 1;
  if(tmp_second >=59) begin
  tmp_minute <= tmp_minute + 1;
  tmp_second <= 0;
  if(tmp_minute >=59) begin
  tmp_minute <= 0;
  tmp_hour <= tmp_hour + 1;
  if(tmp_hour >= 24) begin
  tmp_hour <= 0;
  end
  end
  end

  end
  end
  end

  always @(posedge clk or posedge reset)
  begin
  if(reset)
  begin
  tmp_1s <= 0;
  clk_1s <= 0;
  end
  else begin
  tmp_1s <= tmp_1s + 1;
  if(tmp_1s <= 5)
  clk_1s <= 0;
  else if (tmp_1s >= 10) begin
  clk_1s <= 1;
  tmp_1s <= 1;
  end
```

Fig.6: Verilog Code written in Vivado for implementation

In Fig.5, As told in the flow chart the code, the code flows according to the path given in the flowchart.

In Fig.6, we have used Clk_1s for generating the clock of 1 second time counting using the clock of the clk defined in the code.

```
123     else
124     clk_1s <= 1;
125     end
126     end
127
128     always @(*) begin
129
130     if(tmp_hour>=20) begin
131     Clock_hour1 = 2;
132     end
133     else begin
134     if(tmp_hour >=10)
135     Clock_hour1  = 1;
136     else
137     Clock_hour1 = 0;
138     end
139     Clock_hour0 = tmp_hour - Clock_hour1*10;
140     Clock_min1 = mod_10(tmp_minute);
141     Clock_min0 = tmp_minute - Clock_min1*10;
142     Clock_sec1 = mod_10(tmp_second);
143     Clock_sec0 = tmp_second - Clock_sec1*10;
144     end
```

Fig.7: Verilog Code written in Vivado for implementation

In Fig.7, The clock starts counting from the time given when reset and you can observe that the format in which the time is counted, and alarm rings is in the below format:

[Hour_MSB] [Hour_LSB]:

[Minute_MSB] [Minute_LSB]:

[Second_MSB] [Second _LSB]

```
146
147 always @(posedge clk_1s or posedge reset)
148 begin
149   if(reset)
150   Alarm <=0;
151   else begin
152   if({Alarm_hour1,Alarm_hour0,Alarm_min1,Alarm_min0}=={Clock_hour1,Clock_hour0,Clock_min1,Clock_min0})
153   begin
154   if(Alarm_ON) Alarm <= 1;
155   end
156   if(STOP_alarm) Alarm <=0;
157
158   end
159   end
160
161
162   assign Hour_out1 = Clock_hour1;
163   assign Hour_out0 = Clock_hour0;
164   assign Minute_out1 = Clock_min1;
165   assign Minute_out0 = Clock_min0;
166   assign Second_out1 = Clock_sec1;
167   assign Second_out0 = Clock_sec0;
168
169 endmodule
170
```

Fig.8: Verilog Code written in Vivado for implementation

In Fig.8, Alarm matches the given alarm time with the clock time and rings if Alarm_ON is activated and continues till the Stop_alarm is activated.

- **Testbench for the Verilog code is shown below**

```
21  module Alarm_Testbench;
22  //Inputs
23   reg reset;
24   reg clk;
25   reg [1:0] Hour_in1;
26   reg [3:0] Hour_in0;
27   reg [3:0] Minute_in1;
28   reg [3:0] Minute_in0;
29   reg Load_time;
30   reg Load_alarm;
31   reg STOP_alarm;
32   reg Alarm_ON;
33
34   // Outputs
35   wire Alarm;
36   wire [1:0] Hour_out1;
37   wire [3:0] Hour_out0;
38   wire [3:0] Minute_out1;
39   wire [3:0] Minute_out0;
40   wire [3:0] Second_out1;
41   wire [3:0] Second_out0;
```

Fig.9: Testbench for the Verilog Code written in Vivado for implementation

```
44  ClkAlarm uut (
45  .reset(reset),
46  .clk(clk),
47  .Hour_in1(Hour_in1),
48  .Hour_in0(Hour_in0),
49  .Minute_in1(Minute_in1),
50  .Minute_in0(Minute_in0),
51  .Load_time(Load_time),
52  .Load_alarm(Load_alarm),
53  .STOP_alarm(STOP_alarm),
54  .Alarm_ON(Alarm_ON),
55  .Alarm(Alarm),
56  .Hour_out1(Hour_out1),
57  .Hour_out0(Hour_out0),
58  .Minute_out1(Minute_out1),
59  .Minute_out0(Minute_out0),
60  .Second_out1(Second_out1),
61  .Second_out0(Second_out0)
62  );
63
64  initial begin
65   clk = 0;
66   forever #50000000 clk = ~clk;
67  end
```

Fig.10: Testbench for the Verilog Code written in Vivado for implementation

In Fig.9, Testbench for Verilog is written.

In Fig.10, module ClkAlarm is called ,clk is given input.

```
68  initial begin
69  // Initialize Inputs
70  reset = 1;
71  Hour_in1 = 1;
72  Hour_in0 = 0;
73  Minute_in1 = 1;
74  Minute_in0 = 5;
75  Load_time = 0;
76  Load_alarm = 0;
77  STOP_alarm = 0;
78  Alarm_ON = 0;
79
80  #1000000000;
81  reset = 0;
82  Hour_in1 = 1;
83  Hour_in0 = 0;
84  Minute_in1 = 2;
85  Minute_in0 = 1;
86  Load_time = 0;
87  Load_alarm = 1;
88  STOP_alarm = 0;
89  Alarm_ON = 1;
```

Fig.11: Testbench for the Verilog Code written in Vivado for implementation

```
91
92  #1000000000;
93   reset = 0;
94   Hour_in1 = 1;
95   Hour_in0 = 0;
96   Minute_in1 = 2;
97   Minute_in0 = 1;
98   Load_time = 0;
99   Load_alarm = 0;
100  STOP_alarm = 0;
101  Alarm_ON = 1;
102
103 wait(Alarm);
104 #1000000000;
105 #1000000000;
106 #1000000000;
107 #1000000000;
108 #1000000000;
109 #1000000000;
110 STOP_alarm = 1;
111
112
113  end
114 endmodule
```

Fig.12: Testbench for the Verilog Code written in Vivado for implementation

Fig.11, shows that the clock starts from 10:15 which is given when reset and alarm is set to 10:21

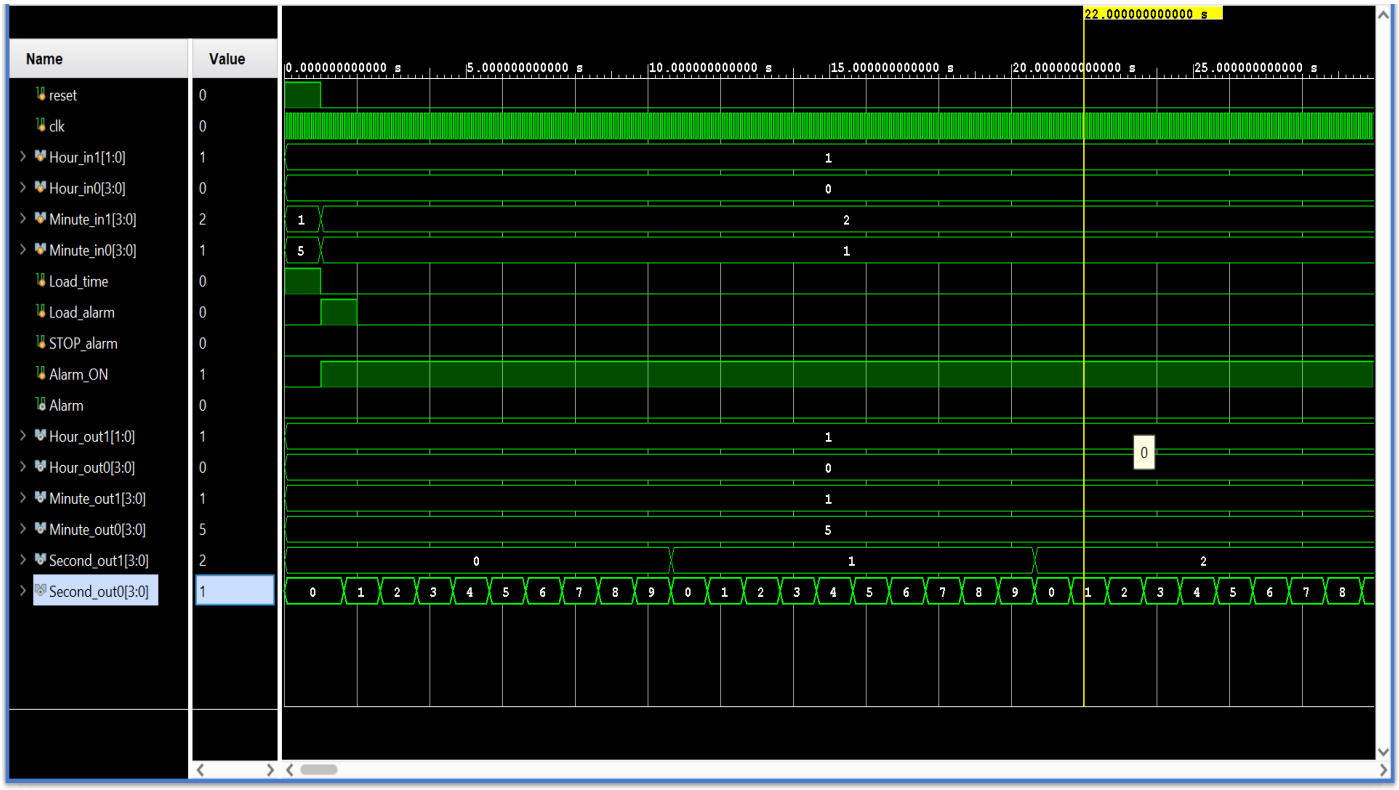
Fig.12, shows the alarm is on to 10:21 and stops after a certain time after the alarm rings

4.Results and Discussions

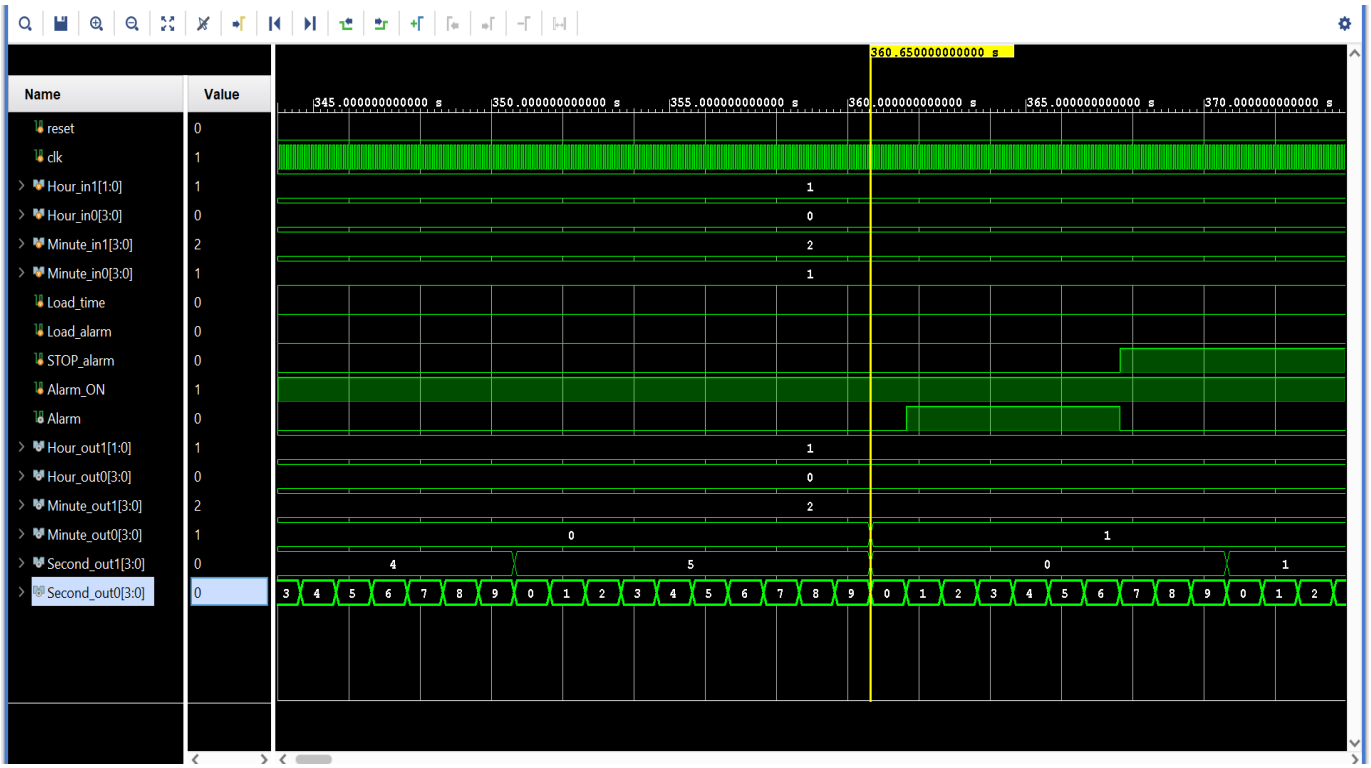
Here, we will see the output waveform generated by the simulation tool for the given test cases in the testbench

Here, the loading time is 10:15 that means the clock starts counting from 10:15, after that, the alarm is set to 10:21 that is the alarm should start alert after 6 minutes(360s) from the clock starts counting.

You can see the loading of time, loading of alarm, on alarm in the below picture. When alarm is set on, the clock starts searching for the set alarm time i.e.,10:21.



The clock starts searching for the set alarm time meanwhile it counts time. When input time and the clock time matches the alarm starts ringing and continues till the alarm is set off. When the time of the clock is 10:21:01 the alarm is high, depicting that alarm starts ringing. This can be verified by the below waveform.



In the picture you can see when the alarm is stopped the alarm stops the alert. Therefore, the functionality of the alarm is verified from above discussion and the output waveforms from the results.

5.Conclusion

This project has successfully demonstrated the ability to design and implement a functional alarm clock using Verilog. Alarm is one of the most important things in one's daily life. We have seen the flowchart model, elaborated design and implementation of the design using Verilog.

We have seen the Verilog code of the Alarm clock in the implementation section along with a testbench. We have confirmed the functionality of the Alarm clock by the output waveforms shown in the results and the discussions of the waveform done above.

As alarm is a basic need in one's life it can be used in various field of digital electronics, so knowing the design and implementation of the alarm using Verilog can make the further research and development easier.

6.References

The making of the report and the design of the Verilog code of alarm clock is supported by reference of the following:

1. https://en.wikipedia.org/wiki/Alarm_clock
2. https://www.fpga4student.com
3. https://www.slideshare.net/AbhishekSainkar1/digital-clock-using-verilog

Thank You