MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Department of Electrical & Electronics Engineering

Digital System Design Lab

MINI PROJECT SYNOPSIS

FPGA BASED ALARM CLOCK

Submitted by

Name	*Roll Number*	*Reg No*	*Batch*
Soham Vashishtha	**59**	**230906528**	**C2**
Manan Deep Singh	**44**	**230906378**	**C2**
Vedant Ghongade	**55**	**230906470**	**C2**
Arjun R Pai	**53**	**230906458**	**C2**

ABSTRACT

This project implements a Alarm clock with alarm functionality using Verilog HDL. It describes a basic alarm clock. In addition to providing a 24-hour format real-time clock output, the alarm clock has an alarm feature. It can be used to initialize the clock time to a particular time for setting the alarm for ringing. The alarm can enabled or disabled and can be stopped according to our need. The alarm is triggered when the alarm time initialized equals the current time.

Switches can also be used by users to adjust the clock time. In this project, we have implemented a clock that has 7 output signals, including the hour, minute, and seconds signals. When reset signal=1 or by setting the signal LD_time=1, we can provide the system with a starting time value. By setting LD_alarm=1, you can adjust the alarm time even further.

The input AL_ON is used to enable or disable the alarm. Only when AL_ON is 1 the alarm will produce sound. The alarm can be silenced using the STOP_al signal. 10Hz is the input clock that is provided. Using this input clock, we created a clk with a time period of one second, which we then used to increment seconds, minutes, and hours.

This project provides a comprehensive approach to designing a digital alarm clock using Verilog HDL, combining theoretical knowledge with practical coding and simulation. By the end of the project, a fully functional digital alarm clock will be demonstrated, showcasing a solid understanding of digital design principles and Verilog programming.

FLOWCHART

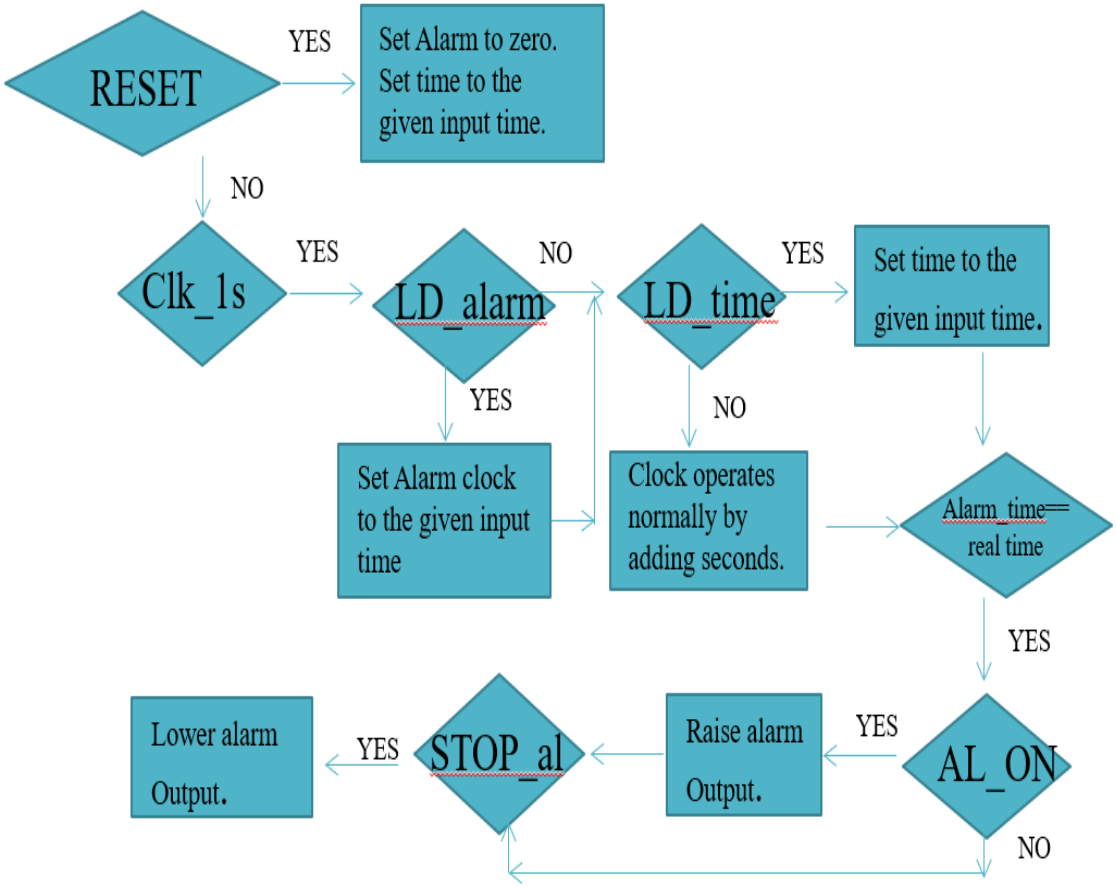


Fig 1: Flowchart of Operation performed in Alarm Clock

BLOCK DIAGRAM

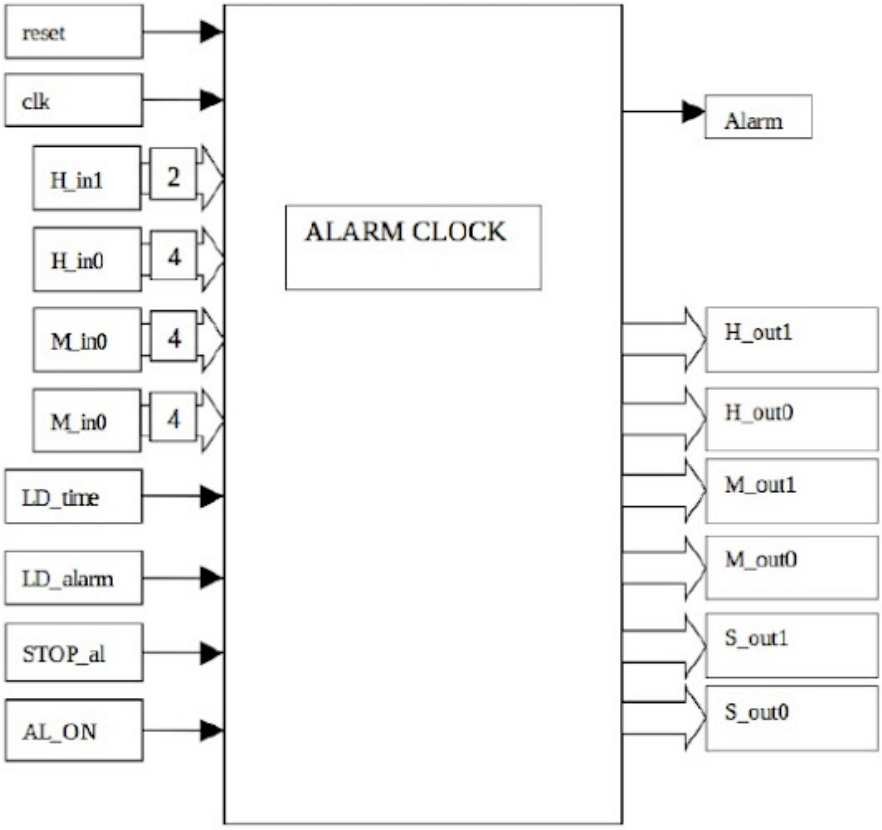


Fig 2: Block Diagram of Alarm Clock

APPLICATION OF ALARM CLOCK

- Going to work or school: This is the most common use for alarm clocks. People set alarms to wake up on time to do everyday household chores.

- **Scheduling appointments or meetings:** You can use alarms to remind people of important appointments or meetings.

- **Time management:** For people who struggle with time management, alarm clocks can help them stay organized and on schedule.

- **Mindfulness or meditation practice:** Some people use alarm clocks to set timers for meditation or mindfulness practice.

- **Cooking or baking:** For recipes that require precise timing, alarms can be useful to ensure that ingredients are cooked or baked at the right time.

- **Emergency preparedness:** When an emergency occurs, alarms can be used to alert people to potential danger or to signal the start of a workout routine.

- **Create a consistent sleep schedule:** Using an alarm to wake people up at the same time every day can help people establish a regular sleep pattern.

- **Modern applications:** Smartphones and wearables: Many smartphones and smartwatches have built-in alarms, and often come with additional features like snooze buttons, custom sounds, and vibration alerts.

- **Internet-connected devices:** Some alarm clocks connect to the Internet, allowing for remote control, integration with other smart home devices, and access to online content.

Verilog Code for Alarm Clock

```
module Aclock(
input reset,
input clk,
input [1:0] H_in1,
input [3:0] H_in0,
input [3:0] M_in1,
input [3:0] M_in0,
input LD_time,
input   LD_alarm,
input   STOP_al,
input   AL_ON,
output reg Alarm,
output [1:0]  H_out1,
output [3:0]  H_out0,
output [3:0]  M_out1,
output [3:0]  M_out0,
output [3:0]  S_out1,
output [3:0]  S_out0);

reg clk_1s;
reg [3:0] tmp_1s;
reg [5:0] tmp_hour, tmp_minute, tmp_second;
reg [1:0] c_hour1,a_hour1;
reg [3:0] c_hour0,a_hour0;
reg [3:0] c_min1,a_min1;
reg [3:0] c_min0,a_min0;
reg [3:0] c_sec1,a_sec1;
reg [3:0] c_sec0,a_sec0;

function [3:0] mod_10;
 input [5:0] number;
 begin
 mod_10 = (number >=50) ? 5 : ((number >= 40)? 4 :((number >= 30)? 3 :((number >= 20)? 2
:((number >= 10)? 1 :0))));
 end
endfunction

always @(posedge clk_1s or posedge reset )
 begin
 if(reset) begin
 a_hour1 <= 2'b00;
 a_hour0 <= 4'b0000;
 a_min1 <= 4'b0000;
```

```
a_min0 <= 4'b0000;
a_sec1 <= 4'b0000;
a_sec0 <= 4'b0000;
tmp_hour <= H_in1*10 + H_in0;
tmp_minute <= M_in1*10 + M_in0;
tmp_second <= 0;
end
else begin
if(LD_alarm) begin
a_hour1 <= H_in1;
a_hour0 <= H_in0;
a_min1 <= M_in1;
a_min0 <= M_in0;
a_sec1 <= 4'b0000;
a_sec0 <= 4'b0000;
end
if(LD_time) begin
tmp_hour <= H_in1*10 + H_in0;
tmp_minute <= M_in1*10 + M_in0;
tmp_second <= 0;
end
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
else
clk_1s <= 1;
end
end

always @(*) begin

if(tmp_hour>=20) begin
c_hour1 = 2;
end
else begin
if(tmp_hour >=10)
c_hour1  = 1;
else
c_hour1 = 0;
end
c_hour0 = tmp_hour - c_hour1*10;
c_min1 = mod_10(tmp_minute);
c_min0 = tmp_minute - c_min1*10;
c_sec1 = mod_10(tmp_second);
c_sec0 = tmp_second - c_sec1*10;
end


always @(posedge clk_1s or posedge reset)
begin
if(reset)
Alarm <=0;
else begin
if({a_hour1,a_hour0,a_min1,a_min0}=={c_hour1,c_hour0,c_min1,c_min0})
begin
if(AL_ON) Alarm <= 1;
end
if(STOP_al) Alarm <=0;

end
end
```

```
 assign H_out1 = c_hour1;
 assign H_out0 = c_hour0;
 assign M_out1 = c_min1;
 assign M_out0 = c_min0;
 assign S_out1 = c_sec1;
 assign S_out0 = c_sec0;

endmodule
```

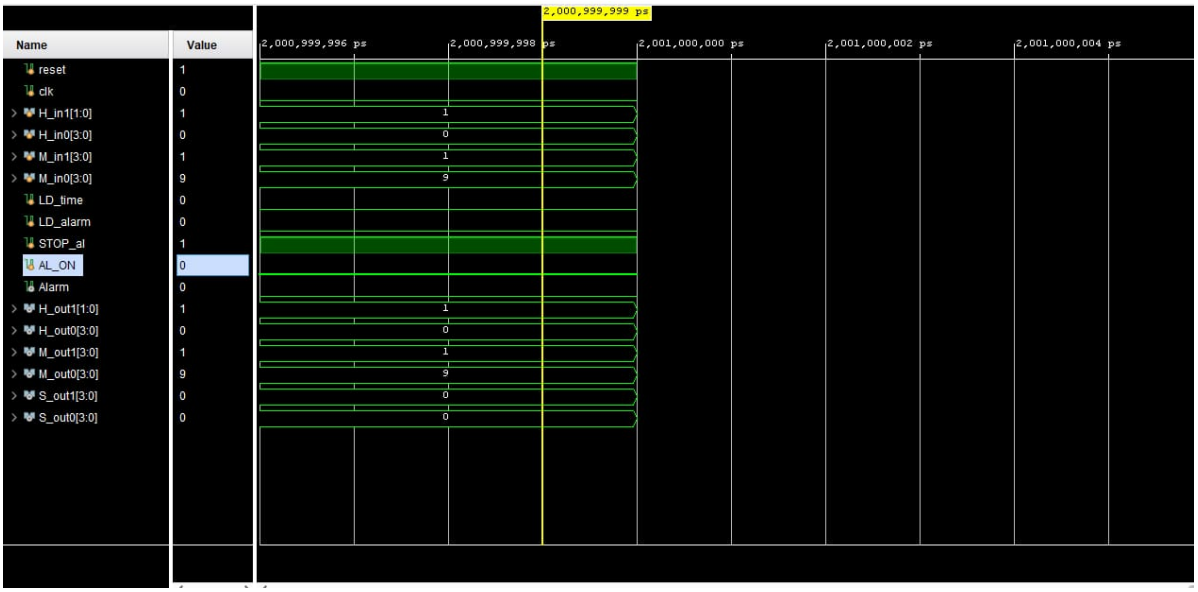
//Testbench for Alarm Clock**

```
module Testbench;

 reg reset;
 reg clk;
 reg [1:0] H_in1;
 reg [3:0] H_in0;
 reg [3:0] M_in1;
 reg [3:0] M_in0;
 reg LD_time;
 reg LD_alarm;
 reg STOP_al;
 reg AL_ON;

 // Outputs
 wire Alarm;
 wire [1:0] H_out1;
 wire [3:0] H_out0;
 wire [3:0] M_out1;
 wire [3:0] M_out0;
 wire [3:0] S_out1;
 wire [3:0] S_out0;


 Aclock uut (
 .reset(reset),
 .clk(clk),
 .H_in1(H_in1),
 .H_in0(H_in0),
 .M_in1(M_in1),
 .M_in0(M_in0),
 .LD_time(LD_time),
 .LD_alarm(LD_alarm),
 .STOP_al(STOP_al),
 .AL_ON(AL_ON),
```

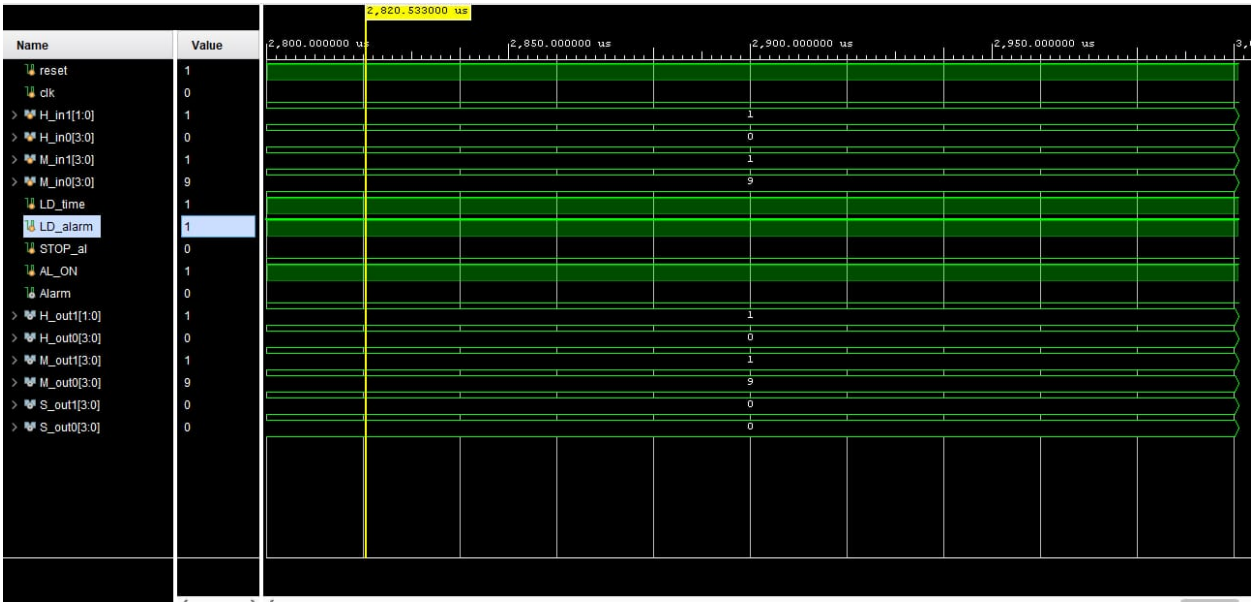
```
 .Alarm(Alarm),
 .H_out1(H_out1),
 .H_out0(H_out0),
 .M_out1(M_out1),
 .M_out0(M_out0),
 .S_out1(S_out1),
 .S_out0(S_out0)
 );

 initial begin
  clk = 0;
  forever #50000000 clk = ~clk;
 end
 initial begin
 // Initialize Inputs
 reset = 1;
 H_in1 = 1;
 H_in0 = 0;
 M_in1 = 1;
 M_in0 = 9;
 LD_time = 0;
 LD_alarm = 0;
 STOP_al = 0;
 AL_ON = 0;

 #1000000000;
 reset = 0;
 H_in1 = 1;
 H_in0 = 0;
 M_in1 = 2;
 M_in0 = 0;
 LD_time = 0;
 LD_alarm = 1;
 STOP_al = 0;
 AL_ON = 1;


#1000000000;
 reset = 0;
 H_in1 = 1;
 H_in0 = 0;
 M_in1 = 2;
 M_in0 = 0;
 LD_time = 0;
 LD_alarm = 0;
 STOP_al = 0;
```

```
 AL_ON = 1;

wait(Alarm);
#1000000000;
#1000000000;
#1000000000;
#1000000000;
#1000000000;
#1000000000;
STOP_al = 1;


 end
endmodule
```

EXPECTED OUTCOME



If alarm is ON (and Alarm will go high if the alarm time equals the real time). If low the the alarm function is OFF.



It will go high if the alarm time equals the current time, and AL_ON is high. This will remain high, until STOP_al goes high, which will bring Alarm back low.

At the end, this alarm clock will constantly display accurate time in both 12-hour and 24-hour forms, Effective Alarm System: Users can set, control, and customise alarms.